sherpa

Release 1.3

sherpa development team

Jul 22, 2024

CONTENTS

1 Introduction **1**

2 Download pdf **3**

3 Social groups **5**
3.1 WeChat . . . 5
3.2 QQ . . . 5
3.3 Bilibili (B) . . . 6
3.4 YouTube . . . 6

4 Run Next-gen Kaldi in your browser **7**
4.1 Visit our Huggingface space . . . 7
4.2 YouTube Video . . . 10
4.3 Other Huggingface spaces . . . 10

5 Pre-trained models **11**
5.1 Pre-trained models for different projects . . . 11
5.2 How to download . . . 11

6 sherpa **15**
6.1 Installation . . . 15
6.2 Pre-trained models . . . 19

7 sherpa-ncnn **49**
7.1 Tutorials . . . 49
7.2 Installation . . . 49
7.3 Python API . . . 64
7.4 WebAssembly . . . 73
7.5 C API . . . 78
7.6 Endpointing . . . 81
7.7 Android . . . 84
7.8 iOS . . . 99
7.9 Pre-trained models . . . 123
7.10 Examples . . . 154
7.11 FAQs . . . 165

8 sherpa-onnx **167**
8.1 Tutorials . . . 167
8.2 Installation . . . 167
8.3 Frequently Asked Question (FAQs) . . . 189
8.4 Python . . . 191

8.5 C API . . . 214
8.6 Java API . . . 218
8.7 Javascript API . . . 228
8.8 Kotlin API . . . 229
8.9 Swift API . . . 229
8.10 Go API . . . 230
8.11 C# API . . . 237
8.12 WebAssembly . . . 246
8.13 Android . . . 252
8.14 iOS . . . 265
8.15 Flutter . . . 273
8.16 WebSocket . . . 276
8.17 Hotwords (Contextual biasing) . . . 291
8.18 Keyword spotting . . . 307
8.19 Punctuation . . . 309
8.20 Audio tagging . . . 313
8.21 Spoken language identification . . . 321
8.22 VAD . . . 324
8.23 Pre-trained models . . . 324
8.24 SenseVoice . . . 508
8.25 Speaker Identification . . . 523
8.26 Text-to-speech (TTS) . . . 523

9 Triton **555**
9.1 Installation . . . 555
9.2 Triton-server . . . 556
9.3 Triton-client . . . 557
9.4 Perf Analyzer . . . 558
9.5 TensorRT acceleration . . . 559

CHAPTER

ONE

INTRODUCTION

sherpa is the deployment framework of the `Next-gen Kaldi` project.

sherpa does only one thing, using a pre-trained model to transcribe speech. If you are interested in how to train your own model or fine tune a pre-trained model, please refer to icefall.

At present, sherpa has the following sub-projects:

- k2-fsa/sherpa
- k2-fsa/sherpa-onnx
- k2-fsa/sherpa-ncnn

The differences are compared below:

	k2-fsa/sherpa	k2-fsa/sherpa-onnx	k2-fsa/sherpa-ncnn
Installation difficulty	**hard**	`easy`	`easy`
NN lib	PyTorch	onnxruntime	ncnn
CPU Support	x86, x86_64	x86, x86_64, `arm32`, `arm64`	x86, x86_64, `arm32`, `arm64`, `**RISC-V**`
GPU Support	Yes (with `CUDA` for NVIDIA GPUs)	Yes	Yes (with `Vulkan` for ARM GPUs)
OS Support	Linux, Windows, macOS	Linux, Windows, macOS, `iOS`, `Android`	Linux, Windows, macOS, `iOS`, `Android`
Support batch_size > 1	Yes	Yes	`No`
Provided APIs	C++, Python	C, C++, Python, C#, Java, Kotlin, Swift, Go, JavaScript, Dart	C, C++, Python, C#, Kotlin, Swift, Go
Supported functions	streaming speech recognition, non-streaming speech recognition	streaming speech recognition, non-streaming speech recognition, text-to-speech, speaker identification, speaker verification, spoken language identification, audio tagging, VAD, keyword spotting,	streaming speech recognition

We also support Triton. Please see *Triton*.

CHAPTER

TWO

DOWNLOAD PDF

We provide a single pdf file containing all the documentation.

Hint: All Chinese related content is not included in the pdf file.

Please download it from the following address:

https://k2-fsa.github.io/sherpa/sherpa.pdf

Note: For Chinese users, you can use the following mirror:

https://hub.nuaa.cf/k2-fsa/sherpa/releases/download/doc/sherpa.pdf

Please always download the latest version.

The pdf file is updated automagically whenever the doc is changed.

CHAPTER

THREE

SOCIAL GROUPS

3.1 WeChat

If you have a WeChat account, you can scan the following QR code to join the WeChat group of next-gen Kaldi to get help.

3.2 QQ

The QQ group is also given below:

Hint: The QQ group number is `744602236`.

3.3 Bilibili (B)

Please visit https://search.bilibili.com/all?keyword=%E6%96%B0%E4%B8%80%E4%BB%A3Kaldi for various demo videos of Next-gen Kaldi.

3.4 YouTube

To get the latest news of next-gen Kaldi, please subscribe the following YouTube channel by Nadira Povey:

https://www.youtube.com/channel/UC_VaumpkmINz1pNkFXAN9mw

CHAPTER

FOUR

RUN NEXT-GEN KALDI IN YOUR BROWSER

This page describes how to try Next-gen Kaldi in your browser.

Hint: You don't need to download or install anything. All you need is a browser.

The server is running on CPU within a docker container provided by Huggingface and you use a browser to interact with it. The browser can be run on Windows, macOS, Linux, or even on your phone or iPad.

You can upload a file for recognition, record your speech via a microphone from within the browser and submit it for recognition, or even provider an URL to an audio file for speech recognition.

Now let's get started.

4.1 Visit our Huggingface space

Start your browser and visit the following address:

https://huggingface.co/spaces/k2-fsa/automatic-speech-recognition

and you will see a page like the following screenshot:

Hint: If you don't have access to Huggingface, please visit the following mirror:

https://hf-mirror.com/spaces/k2-fsa/automatic-speech-recognition

You can:

1. Select a language for recognition. Currently, we provide pre-trained models from icefall for the following languages: `Chinese`, `English`, and `Chinese+English`.
2. After selecting the target language, you can select a pre-trained model corresponding to the language.
3. Select the decoding method. Currently, it provides `greedy search` and `modified_beam_search`.
4. If you selected `modified_beam_search`, you can choose the number of active paths during the search.
5. Either upload a file or record your speech for recognition.
6. Click the button `Submit for recognition`.
7. Wait for a moment and you will get the recognition results.

The following screenshot shows an example when selecting `Chinese+English`:

In the bottom part of the page, you can find a table of examples. You can click one of them and then click `Submit for recognition`.

Language	Select a model	Decoding method	Number of active paths for modified_beam_search	Upload from disk
English	csukuangfj/icefall-asr-librispeech-pruned-transducer-stateless3-2022-05-13	greedy_search	4	./test_wavs/librispeech/1089-134686-0001.wa
English	csukuangfj/icefall-asr-librispeech-pruned-transducer-stateless3-2022-05-13	greedy_search	4	./test_wavs/librispeech/1221-135766-0001.wa
English	csukuangfj/icefall-asr-librispeech-pruned-transducer-stateless3-2022-05-13	greedy_search	4	./test_wavs/librispeech/1221-135766-0002.wa
English	wgb14/icefall-asr-gigaspeech-pruned-transducer-stateless2	greedy_search	4	./test_wavs/gigaspeech/1-minute-audiobook.opus
English	wgb14/icefall-asr-gigaspeech-pruned-transducer-stateless2	greedy_search	4	./test_wavs/gigaspeech/100-seconds-podcast.opus
English	wgb14/icefall-asr-gigaspeech-pruned-transducer-stateless2	greedy_search	4	./test_wavs/gigaspeech/100-seconds-

4.2 YouTube Video

We provide the following YouTube video demonstrating how to use https://huggingface.co/spaces/k2-fsa/automatic-speech-recognition.

Note: To get the latest news of next-gen Kaldi, please subscribe the following YouTube channel by Nadira Povey:

https://www.youtube.com/channel/UC_VaumpkmINz1pNkFXAN9mw

https://youtu.be/ElN3r9dkKE4

4.3 Other Huggingface spaces

- ASR + WebAssembly + sherpa-ncnn: Please see *Huggingface Spaces (WebAssembly)*
- TTS: Please see: https://huggingface.co/spaces/k2-fsa/text-to-speech

CHAPTER

FIVE

PRE-TRAINED MODELS

5.1 Pre-trained models for different projects

Project	Pretrained models
k2-fsa/sherpa	*Click here*
k2-fsa/sherpa-onnx	*Click here*
k2-fsa/sherpa-ncnn	*Click here*

5.2 How to download

We are hosting our pre-trained models on Huggingface as git repositories managed by Git LFS.

There are at least two methods for downloading:

- Using `git lfs`
- Using `wget`

In the following, we use the pre-trained model *pkufool/icefall-asr-zipformer-streaming-wenetspeech-20230615 (Chinese)* as an example.

5.2.1 Using git lfs

Please first install `git-lfs` by following https://git-lfs.com/.

Linux

macOS

Windows

```
# apt/deb
sudo apt-get install git-lfs

# yum/rpm
sudo yum install git-lfs
```

Please see https://github.com/git-lfs/git-lfs/blob/main/INSTALLING.md for details.

```
brew install git-lfs
```

Please visit https://gitforwindows.org/ to download and install `git-lfs`.

Then use the following commands to download pre-trained models:

Linux/macOS

Windows (Powershell)

Windows (cmd)

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/pkufool/icefall-asr-zipformer-
↪streaming-wenetspeech-20230615
cd icefall-asr-zipformer-streaming-wenetspeech-20230615
git lfs pull --include "exp/*chunk-16-left-128.*onnx"
```

```
$env:GIT_LFS_SKIP_SMUDGE="1"
git clone https://huggingface.co/pkufool/icefall-asr-zipformer-streaming-wenetspeech-
↪20230615
cd icefall-asr-zipformer-streaming-wenetspeech-20230615
git lfs pull --include "exp/*chunk-16-left-128.*onnx"
```

```
set GIT_LFS_SKIP_SMUDGE="1"
git clone https://huggingface.co/pkufool/icefall-asr-zipformer-streaming-wenetspeech-
↪20230615
cd icefall-asr-zipformer-streaming-wenetspeech-20230615
git lfs pull --include "exp/*chunk-16-left-128.*onnx"
```

Note: It is very important to set the environment variable `GIT_LFS_SKIP_SMUDGE` to `1`. We don't recommend using `git lfs install` as it will download many large files that we don't need.

5.2.2 Using wget

First, let us visit the huggingface git repository of the pre-trained model:

Click `Files and versions` and navigate to the directory containing files for downloading:

Right click the arrow that indicates downloading and copy the link address. After that, you can use, for instance, `wget` to download the file with the following command:

```
wget https://huggingface.co/pkufool/icefall-asr-zipformer-streaming-wenetspeech-20230615/
↪resolve/main/exp/decoder-epoch-12-avg-4-chunk-16-left-128.int8.onnx
```

Repeat the process until you have downloaded all the required files.

CHAPTER

SIX

SHERPA

Hint: During speech recognition, it does not need to access the Internet. Everyting is processed locally on your device.

k2-fsa/sherpa use PyTorch for neural network computation.

https://k2-fsa.github.io/icefall/model-export/export-with-torch-jit-script.html>`_ for how to export models.

In the following, we describe how to use k2-fsa/sherpa.

6.1 Installation

6.1.1 From pre-compiled wheels

Note: This method supports only Linux and macOS for now. If you want to use Windows, please refer to *From source*.

You can find a list of pre-compiled wheels at the following URLs:

- CPU: https://k2-fsa.github.io/sherpa/cpu.html
- CUDA: https://k2-fsa.github.io/sherpa/cuda.html

In the following, we demonstrate how to install `k2-fsa/sherpa` from pre-compiled wheels.

Linux (CPU)

Suppose that we want to install the following wheel

```
https://huggingface.co/csukuangfj/kaldifeat/resolve/main/ubuntu-cpu/k2_sherpa-1.3.
↪dev20230725+cpu.torch2.0.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
```

we can use the following methods:

```
# Before installing k2-fsa/sherpa, we have to install the following dependencies:
#  torch, k2, and kaldifeat

pip install torch==2.0.1+cpu -f https://download.pytorch.org/whl/torch_stable.html
pip install k2==1.24.4.dev20231220+cpu.torch2.0.1 -f https://k2-fsa.github.io/k2/cpu.html
pip install kaldifeat==1.25.3.dev20231221+cpu.torch2.0.1 -f https://csukuangfj.github.io/
↪kaldifeat/cpu.html
```

(continues on next page)

(continued from previous page)

```
# Now we can install k2-fsa/sherpa
pip install k2_sherpa==1.3.dev20230725+cpu.torch2.0.1 -f https://k2-fsa.github.io/sherpa/
↪cpu.html
```

Please see *Check your installation*.

macOS (CPU)

Suppose that we want to install the following wheel

```
https://huggingface.co/csukuangfj/kaldifeat/resolve/main/macos/k2_sherpa-1.3.
↪dev20230725+cpu.torch2.0.1-cp311-cp311-macosx_10_9_x86_64.whl
```

we can use the following methods:

```
# Before installing k2-fsa/sherpa, we have to install the following dependencies:
#  torch, k2, and kaldifeat

pip install torch==2.0.1+cpu -f https://download.pytorch.org/whl/torch_stable.html
pip install k2==1.24.4.dev20231220+cpu.torch2.0.1 -f https://k2-fsa.github.io/k2/cpu.html
pip install kaldifeat==1.25.3.dev20231221+cpu.torch2.0.1 -f https://csukuangfj.github.io/
↪kaldifeat/cpu.html

# Now we can install k2-fsa/sherpa
pip install k2_sherpa==1.3.dev20230725+cpu.torch2.0.1 -f https://k2-fsa.github.io/sherpa/
↪cpu.html
```

Please see *Check your installation*.

Linux (CUDA)

Suppose that we want to install the following wheel

```
https://huggingface.co/csukuangfj/kaldifeat/resolve/main/ubuntu-cuda/k2_sherpa-1.3.
↪dev20230725+cuda11.7.torch2.0.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.
↪whl
```

we can use the following methods:

```
# Before installing k2-fsa/sherpa, we have to install the following dependencies:
#  torch, k2, and kaldifeat

pip install torch==2.0.1+cu117 -f https://download.pytorch.org/whl/torch_stable.html
pip install k2==1.24.4.dev20231220+cuda11.7.torch2.0.1 -f https://k2-fsa.github.io/k2/
↪cuda.html
pip install kaldifeat==1.25.3.dev20231221+cuda11.7.torch2.0.1  -f https://csukuangfj.
↪github.io/kaldifeat/cuda.html


# Now we can install k2-fsa/sherpa
pip install k2_sherpa==1.3.dev20230725+cuda11.7.torch2.0.1 -f https://k2-fsa.github.io/
↪sherpa/cuda.html
```

(continues on next page)

(continued from previous page)

Please see *Check your installation*.

6.1.2 From source

This section describe how to install `k2-fsa/sherpa` from source.

Install dependencies

Before installing `k2-fsa/sherpa` from source, we have to install the following dependencies.

- PyTorch
- k2
- kaldifeat

CPU

CUDA

Suppose that we select `torch==2.0.1`. We can use the following commands to install the dependencies:

Linux

macOS

Windows

```
pip install torch==2.0.1+cpu -f https://download.pytorch.org/whl/torch_stable.html
pip install k2==1.24.4.dev20231220+cpu.torch2.0.1 -f https://k2-fsa.github.io/k2/cpu.html
pip install kaldifeat==1.25.3.dev20231221+cpu.torch2.0.1 -f https://csukuangfj.github.io/
↪kaldifeat/cpu.html
```

```
pip install torch==2.0.1+cpu -f https://download.pytorch.org/whl/torch_stable.html
pip install k2==1.24.4.dev20231220+cpu.torch2.0.1 -f https://k2-fsa.github.io/k2/cpu.html
pip install kaldifeat==1.25.3.dev20231221+cpu.torch2.0.1 -f https://csukuangfj.github.io/
↪kaldifeat/cpu.html
```

To be done.

Suppose that we select `torch==2.0.1+cu117`. We can use the following commands to install the dependencies:

```
pip install torch==2.0.1+cu117 -f https://download.pytorch.org/whl/torch_
↪stable.html
pip install k2==1.24.4.dev20231220+cuda11.7.torch2.0.1 -f https://k2-fsa.
↪github.io/k2/cuda.html
pip install kaldifeat==1.25.3.dev20231221+cuda11.7.torch2.0.1  -f https://
↪csukuangfj.github.io/kaldifeat/cuda.html
```

Next, please follow https://k2-fsa.github.io/k2/installation/cuda-cudnn.html to install CUDA toolkit.

Now we can start to build `k2-fsa/sherpa` from source.

For general users

You can use the following commands to install k2-fsa/sherpa:

```
# Please make sure you have installed PyTorch, k2, and kaldifeat
# before you continue
#
git clone http://github.com/k2-fsa/sherpa
cd sherpa
python3 -m pip install --verbose .
```

To uninstall k2-fsa/sherpa, please use

```
# Please run it outside of the k2-fsa/sherpa repo
#
pip uninstall k2-sherpa
```

Please see *Check your installation*.

For developers and advanced users

You can also use the following commands to install k2-fsa/sherpa.

The advantage is that you can have several versions of k2-fsa/sherpa in a single environment.

```
git clone http://github.com/k2-fsa/sherpa
cd sherpa
mkdir build
cd build

# For torch >= 2.0, please use
#
#  cmake -DCMAKE_CXX_STANDARD=17 ..
#

cmake ..
make -j

export PATH=$PWD/bin:$PATH
export PYTHONPATH=$PWD/lib:$PWD/../sherpa/python:$PYTHONPATH
```

Please see *Check your installation*.

6.1.3 Check your installation

To check that you have installed `k2-fsa/sherpa` successfully, please run:

```
python3 -c "import sherpa; print(sherpa.__file__); print(sherpa.__version__)"

sherpa-online --help
sherpa-offline --help

sherpa-online-microphone --help
```

(continues on next page)

(continued from previous page)

```
sherpa-offline-microphone --help

sherpa-online-websocket-server --help
sherpa-online-websocket-client --help
sherpa-online-websocket-client-microphone --help

sherpa-offline-websocket-server --help
sherpa-offline-websocket-client --help
```

Congratulations! You have installed k2-fsa/sherpa successfully. Please refer to *Pre-trained models* to download pre-trained models.

Have fun with k2-fsa/sherpa!

We suggest that you install `k2-fsa/sherpa` by following *From pre-compiled wheels*.

6.1.4 Where to get help

If you have any issues about the installation, please create an issue at the following address:

https://github.com/k2-fsa/sherpa/issues

6.2 Pre-trained models

Two kinds of end-to-end (E2E) models are supported by k2-fsa/sherpa:

- CTC
- Transducer

Hint: For transducer-based models, we only support stateless transducers. To the best of our knowledge, only icefall supports that. In other words, only transducer models from icefall are currently supported.

For CTC-based models, we support any type of models trained using CTC loss as long as you can export the model via torchscript. Models from the following frameworks are currently supported: icefall, WeNet, and torchaudio (Wav2Vec 2.0). If you have a CTC model and want it to be supported in k2-fsa/sherpa, please create an issue at https://github.com/k2-fsa/sherpa/issues.

Hint: You can try the pre-trained models in your browser without installing anything. See https://huggingface.co/spaces/k2-fsa/automatic-speech-recognition.

This page lists all available pre-trained models that you can download.

Hint: We provide pre-trained models for the following languages:

- Arabic
- Chinese
- English
- German

- Tibetan

Hint: We provide a colab notebook for you to try offline recognition step by step.

It shows how to install sherpa and use it as offline recognizer, which supports the models from icefall, the WeNet framework and torchaudio.

6.2.1 Offline CTC models

This sections list pre-trained CTC models from the following frameworks:

icefall

Hint: We use the binary `sherpa-offline` below for demonstration. You can replace `sherpa-offline` with `sherpa-offline-websocket-server`.

In this section, we list all pre-trained CTC models from icefall.

icefall-asr-gigaspeech-conformer-ctc (English)

```
# This model is trained using GigaSpeech
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/wgb14/icefall-asr-gigaspeech-
↪conformer-ctc
cd icefall-asr-gigaspeech-conformer-ctc
git lfs pull --include "exp/cpu_jit.pt"
git lfs pull --include "data/lang_bpe_500/HLG.pt"
git lfs pull --include "data/lang_bpe_500/tokens.txt"
mkdir test_wavs
cd test_wavs
wget https://huggingface.co/csukuangfj/wav2vec2.0-torchaudio/resolve/main/test_wavs/1089-
↪134686-0001.wav
wget https://huggingface.co/csukuangfj/wav2vec2.0-torchaudio/resolve/main/test_wavs/1221-
↪135766-0001.wav
wget https://huggingface.co/csukuangfj/wav2vec2.0-torchaudio/resolve/main/test_wavs/1221-
↪135766-0002.wav
cd ..

# Decode with H
sherpa-offline \
  --nn-model=./exp/cpu_jit.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav

# Decode with HLG
sherpa-offline \
```

(continues on next page)

(continued from previous page)

```
  --nn-model=./exp/cpu_jit.pt \
  --hlg=./data/lang_bpe_500/HLG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall-asr-librispeech-conformer-ctc-jit-bpe-500-2021-11-09 (English)

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/icefall-asr-
↪librispeech-conformer-ctc-jit-bpe-500-2021-11-09
cd icefall-asr-librispeech-conformer-ctc-jit-bpe-500-2021-11-09

git lfs pull --include "exp/cpu_jit.pt"
git lfs pull --include "data/lang_bpe_500/tokens.txt"
git lfs pull --include "data/lang_bpe_500/HLG.pt"
git lfs pull --include "data/lang_bpe_500/HLG_modified.pt"

# Decode with H
sherpa-offline \
  --nn-model=./exp/cpu_jit.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  --use-gpu=false \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav

# Decode with HLG
sherpa-offline \
  --nn-model=./exp/cpu_jit.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  --hlg=./data/lang_bpe_500/HLG.pt \
  --use-gpu=false \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav

# Decode with HLG (modified)
sherpa-offline \
  --nn-model=./exp/cpu_jit.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  --hlg=./data/lang_bpe_500/HLG_modified.pt \
  --use-gpu=false \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall-asr-tedlium3-conformer-ctc2 (English)

```
# This model is trained using Tedlium3
#
# See https://github.com/k2-fsa/icefall/pull/696
#

GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/videodanchik/icefall-asr-tedlium3-
↪conformer-ctc2
cd icefall-asr-tedlium3-conformer-ctc2
git lfs pull --include "exp/cpu_jit.pt"

git lfs pull --include "data/lang_bpe/HLG.pt"
git lfs pull --include "data/lang_bpe/tokens.txt"

git lfs pull --include "test_wavs/DanBarber_2010-219.wav"
git lfs pull --include "test_wavs/DanielKahneman_2010-157.wav"
git lfs pull --include "test_wavs/RobertGupta_2010U-15.wav"

# Decode with H
sherpa-offline \
  --nn-model=./exp/cpu_jit.pt \
  --tokens=./data/lang_bpe/tokens.txt \
  ./test_wavs/DanBarber_2010-219.wav \
  ./test_wavs/DanielKahneman_2010-157.wav \
  ./test_wavs/RobertGupta_2010U-15.wav

# Decode with HLG
sherpa-offline \
  --nn-model=./exp/cpu_jit.pt \
  --hlg=./data/lang_bpe/HLG.pt \
  --tokens=./data/lang_bpe/tokens.txt \
  ./test_wavs/DanBarber_2010-219.wav \
  ./test_wavs/DanielKahneman_2010-157.wav \
  ./test_wavs/RobertGupta_2010U-15.wav
```

icefall_asr_librispeech_conformer_ctc (English)

```
# This model is trained using LibriSpeech
#
# See https://github.com/k2-fsa/icefall/pull/13
#

GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/pkufool/icefall_asr_librispeech_
↪conformer_ctc
cd icefall_asr_librispeech_conformer_ctc

git lfs pull --include "exp/cpu_jit.pt"
git lfs pull --include "data/lang_bpe/HLG.pt"

# Decode with H
```

(continues on next page)

(continued from previous page)

```
sherpa-offline \
  --nn-model=./exp/cpu_jit.pt \
  --tokens=./data/lang_bpe/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav

# Decode with HLG
sherpa-offline \
  --nn-model=./exp/cpu_jit.pt \
  --hlg=./data/lang_bpe/HLG.pt \
  --tokens=./data/lang_bpe/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall_asr_aishell_conformer_ctc (Chinese)

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/pkufool/icefall_asr_aishell_
↪conformer_ctc
cd icefall_asr_aishell_conformer_ctc
git lfs pull --include "exp/cpu_jit.pt"
git lfs pull --include "data/lang_char/HLG.pt"

# Decode with an H graph
sherpa-offline \
  --nn-model=./exp/cpu_jit.pt \
  --tokens=./data/lang_char/tokens.txt \
  ./test_waves/BAC009S0764W0121.wav \
  ./test_waves/BAC009S0764W0122.wav \
  ./test_waves/BAC009S0764W0123.wav

# Decode with an HLG graph
sherpa-offline \
  --nn-model=./exp/cpu_jit.pt \
  --tokens=./data/lang_char/tokens.txt \
  --hlg=./data/lang_char/HLG.pt \
  ./test_waves/BAC009S0764W0121.wav \
  ./test_waves/BAC009S0764W0122.wav \
  ./test_waves/BAC009S0764W0123.wav
```

icefall-asr-mgb2-conformer_ctc-2022-27-06 (Arabic)

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/AmirHussein/icefall-asr-mgb2-
↪conformer_ctc-2022-27-06
cd icefall-asr-mgb2-conformer_ctc-2022-27-06
git lfs pull --include "exp/cpu_jit.pt"
git lfs pull --include "data/lang_bpe_5000/HLG.pt"
git lfs pull --include "data/lang_bpe_5000/tokens.txt"

# Decode with an H graph
sherpa-offline \
  --nn-model=./exp/cpu_jit.pt \
  --tokens=./data/lang_bpe_5000/tokens.txt \
  ./test_wavs/94D37D38-B203-4FC0-9F3A-538F5C174920_spk-0001_seg-0053813:0054281.wav \
  ./test_wavs/94D37D38-B203-4FC0-9F3A-538F5C174920_spk-0001_seg-0051454:0052244.wav \
  ./test_wavs/94D37D38-B203-4FC0-9F3A-538F5C174920_spk-0001_seg-0052244:0053004.wav

# Decode with an HLG graph
sherpa-offline \
  --nn-model=./exp/cpu_jit.pt \
  --tokens=./data/lang_bpe_5000/tokens.txt \
  --hlg=./data/lang_bpe_5000/HLG.pt \
  ./test_wavs/94D37D38-B203-4FC0-9F3A-538F5C174920_spk-0001_seg-0053813:0054281.wav \
  ./test_wavs/94D37D38-B203-4FC0-9F3A-538F5C174920_spk-0001_seg-0051454:0052244.wav \
  ./test_wavs/94D37D38-B203-4FC0-9F3A-538F5C174920_spk-0001_seg-0052244:0053004.wav
```

WeNet

This section lists models from WeNet.

wenet-english-model (English)

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/wenet-english-model
cd wenet-english-model
git lfs pull --include "final.zip"

sherpa-offline \
 --normalize-samples=false \
 --modified=true \
 --nn-model=./final.zip \
 --tokens=./units.txt \
 --use-gpu=false \
 ./test_wavs/1089-134686-0001.wav \
 ./test_wavs/1221-135766-0001.wav \
 ./test_wavs/1221-135766-0002.wav
```

wenet-chinese-model (Chinese)

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/wenet-chinese-model
cd wenet-chinese-model
git lfs pull --include "final.zip"

sherpa-offline \
  --normalize-samples=false \
  --modified=true \
  --nn-model=./final.zip \
  --tokens=./units.txt \
  ./test_wavs/BAC009S0764W0121.wav \
  ./test_wavs/BAC009S0764W0122.wav \
  ./test_wavs/BAC009S0764W0123.wav \
  ./test_wavs/DEV_T0000000000.wav \
  ./test_wavs/DEV_T0000000001.wav \
  ./test_wavs/DEV_T0000000002.wav
```

torchaudio

This section lists models from torchaudio.

wav2vec2_asr_base (English)

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/wav2vec2.0-torchaudio
cd wav2vec2.0-torchaudio

# Note: There are other kinds of models fine-tuned with different
# amount of data. We use a model that is fine-tuned with 10 minutes of data.

git lfs pull --include "wav2vec2_asr_base_10m.pt"

sherpa-offline \
 --nn-model=wav2vec2_asr_base_10m.pt \
 --tokens=tokens.txt \
 --use-gpu=false \
 ./test_wavs/1089-134686-0001.wav \
 ./test_wavs/1221-135766-0001.wav \
 ./test_wavs/1221-135766-0002.wav
```

voxpopuli_asr_base (German)

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/wav2vec2.0-torchaudio
cd wav2vec2.0-torchaudio
git lfs pull --include "voxpopuli_asr_base_10k_de.pt"

sherpa-offline \
 --nn-model=voxpopuli_asr_base_10k_de.pt \
```

(continues on next page)

(continued from previous page)

```
 --tokens=tokens-de.txt \
 --use-gpu=false \
 ./test_wavs/20120315-0900-PLENARY-14-de_20120315.wav \
 ./test_wavs/20170517-0900-PLENARY-16-de_20170517.wav
```

NeMo

This section lists models from NeMo.

sherpa-nemo-ctc-en-citrinet-512 (English)

This model is converted from

https://catalog.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_en_citrinet_512

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/sherpa-nemo-ctc-en-
↪citrinet-512
cd sherpa-nemo-ctc-en-citrinet-512
git lfs pull --include "model.pt"

sherpa-offline \
   --nn-model=./model.pt \
   --tokens=./tokens.txt \
   --use-gpu=false \
   --modified=false \
   --nemo-normalize=per_feature \
   ./test_wavs/0.wav \
   ./test_wavs/1.wav \
   ./test_wavs/2.wav
```

```
ls -lh model.pt
-rw-r--r-- 1 kuangfangjun root 142M Mar  9 21:23 model.pt
```

Caution: It is of paramount importance to specify `--nemo-normalize=per_feature`.

sherpa-nemo-ctc-zh-citrinet-512 (Chinese)

This model is converted from

https://catalog.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_zh_citrinet_512

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/sherpa-nemo-ctc-zh-
↪citrinet-512
cd sherpa-nemo-ctc-zh-citrinet-512
git lfs pull --include "model.pt"

sherpa-offline \
   --nn-model=./model.pt \
```

(continues on next page)

(continued from previous page)

```
   --tokens=./tokens.txt \
   --use-gpu=false \
   --modified=true \
   --nemo-normalize=per_feature \
   ./test_wavs/0.wav \
   ./test_wavs/1.wav \
   ./test_wavs/2.wav
```

```
ls -lh model.pt
-rw-r--r-- 1 kuangfangjun root 153M Mar 10 15:07 model.pt
```

Hint: Since the vocabulary size of this model is very large, i.e, 5207, we use `--modified=true` to use a modified CTC topology

Caution: It is of paramount importance to specify `--nemo-normalize=per_feature`.

sherpa-nemo-ctc-zh-citrinet-1024-gamma-0-25 (Chinese)

This model is converted from

https://catalog.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_zh_citrinet_1024_gamma_0_25

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/sherpa-nemo-ctc-zh-
↪citrinet-1024-gamma-0-25
cd sherpa-nemo-ctc-zh-citrinet-1024-gamma-0-25
git lfs pull --include "model.pt"

sherpa-offline \
   --nn-model=./model.pt \
   --tokens=./tokens.txt \
   --use-gpu=false \
   --modified=true \
   --nemo-normalize=per_feature \
   ./test_wavs/0.wav \
   ./test_wavs/1.wav \
   ./test_wavs/2.wav
```

```
ls -lh model.pt
-rw-r--r-- 1 kuangfangjun root 557M Mar 10 16:29 model.pt
```

Hint: Since the vocabulary size of this model is very large, i.e, 5207, we use `--modified=true` to use a modified CTC topology

Caution: It is of paramount importance to specify `--nemo-normalize=per_feature`.

sherpa-nemo-ctc-de-citrinet-1024 (German)

This model is converted from

https://catalog.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_de_citrinet_1024

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/sherpa-nemo-ctc-de-
↪citrinet-1024
cd sherpa-nemo-ctc-de-citrinet-1024
git lfs pull --include "model.pt"

sherpa-offline \
   --nn-model=./model.pt \
   --tokens=./tokens.txt \
   --use-gpu=false \
   --modified=false \
   --nemo-normalize=per_feature \
   ./test_wavs/0.wav \
   ./test_wavs/1.wav \
   ./test_wavs/2.wav
```

```
ls -lh model.pt
-rw-r--r-- 1 kuangfangjun root 541M Mar 10 16:55 model.pt
```

Caution: It is of paramount importance to specify `--nemo-normalize=per_feature`.

sherpa-nemo-ctc-en-conformer-small (English)

This model is converted from

https://registry.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_en_conformer_ctc_small

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/sherpa-nemo-ctc-en-
↪conformer-small
cd sherpa-nemo-ctc-en-conformer-small
git lfs pull --include "model.pt"

sherpa-offline \
   --nn-model=./model.pt \
   --tokens=./tokens.txt \
   --use-gpu=false \
   --modified=false \
   --nemo-normalize=per_feature \
   ./test_wavs/0.wav \
   ./test_wavs/1.wav \
   ./test_wavs/2.wav
```

```
ls -lh model.pt
-rw-r--r--  1 fangjun  staff    82M Mar 10 19:55 model.pt
```

Caution: It is of paramount importance to specify `--nemo-normalize=per_feature`.

sherpa-nemo-ctc-en-conformer-medium (English)

This model is converted from

https://registry.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_en_conformer_ctc_medium

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/sherpa-nemo-ctc-en-
↪conformer-medium
cd sherpa-nemo-ctc-en-conformer-medium
git lfs pull --include "model.pt"

sherpa-offline \
   --nn-model=./model.pt \
   --tokens=./tokens.txt \
   --use-gpu=false \
   --modified=false \
   --nemo-normalize=per_feature \
   ./test_wavs/0.wav \
   ./test_wavs/1.wav \
   ./test_wavs/2.wav
```

```
ls -lh model.pt
-rw-r--r--  1 fangjun  staff   152M Mar 10 20:26 model.pt
```

Caution: It is of paramount importance to specify `--nemo-normalize=per_feature`.

sherpa-nemo-ctc-en-conformer-large (English)

This model is converted from

https://registry.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_en_conformer_ctc_large

Hint: The vocabulary size is 129

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/sherpa-nemo-ctc-en-
↪conformer-large
cd sherpa-nemo-ctc-en-conformer-large
git lfs pull --include "model.pt"

sherpa-offline \
   --nn-model=./model.pt \
   --tokens=./tokens.txt \
   --use-gpu=false \
   --modified=false \
   --nemo-normalize=per_feature \
```

(continues on next page)

(continued from previous page)

```
   ./test_wavs/0.wav \
   ./test_wavs/1.wav \
   ./test_wavs/2.wav
```

```
ls -lh model.pt
-rw-r--r--  1 fangjun  staff   508M Mar 10 20:44 model.pt
```

Caution: It is of paramount importance to specify `--nemo-normalize=per_feature`.

sherpa-nemo-ctc-de-conformer-large (German)

This model is converted from

https://registry.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_de_conformer_ctc_large

Hint: The vocabulary size is 129

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/sherpa-nemo-ctc-de-
↪conformer-large
cd sherpa-nemo-ctc-de-conformer-large
git lfs pull --include "model.pt"

sherpa-offline \
   --nn-model=./model.pt \
   --tokens=./tokens.txt \
   --use-gpu=false \
   --modified=false \
   --nemo-normalize=per_feature \
   ./test_wavs/0.wav \
   ./test_wavs/1.wav \
   ./test_wavs/2.wav
```

```
ls -lh model.pt
-rw-r--r--  1 fangjun  staff   508M Mar 10 21:34 model.pt
```

Caution: It is of paramount importance to specify `--nemo-normalize=per_feature`.

How to convert NeMo models to sherpa

This section describes how to export NeMo pre-trained CTC models to sherpa.

You can find a list of pre-trained models from NeMo by visiting:

> https://catalog.ngc.nvidia.com/orgs/nvidia/collections/nemo_asr.

Let us take `stt_en_conformer_ctc_small` as an example.

You can use the following code to obtain `model.pt` and `tokens.txt`:

```
import nemo.collections.asr as nemo_asr
m = nemo_asr.models.EncDecCTCModelBPE.from_pretrained('stt_en_conformer_ctc_small')
m.export("model.pt")

with open('tokens.txt', 'w', encoding='utf-8') as f:
  f.write("<blk> 0\n")
  for i, s in enumerate(m.decoder.vocabulary):
    f.write(f"{s} {i+1}\n")
```

One thing to note is that the blank token has the largest token ID in `NeMo`. However, it is always `0` in sherpa. During network computation, we shift the last column of the `log_prob` tensor to the first column so that it matches the convention about using 0 for the blank in sherpa.

You can find the exported `model.pt` and `tokens.txt` by visiting

> https://huggingface.co/csukuangfj/sherpa-nemo-ctc-en-conformer-small

6.2.2 Offline transducer models

Hint: We use the binary `sherpa-offline` below for demonstration. You can replace `sherpa-offline` with `sherpa-offline-websocket-server`.

Hint: Please visit https://huggingface.co/spaces/k2-fsa/automatic-speech-recognition to try the pre-trained models in your browser. You don't need to install anything.

icefall

This sections lists models trained using icefall.

English

icefall-asr-librispeech-zipformer-2023-05-15

```
# This model is trained using LibriSpeech with zipformer transducer
#
# See https://github.com/k2-fsa/icefall/pull/1058
#
```

(continues on next page)

(continued from previous page)

```
# normal-scaled model, number of model parameters: 65549011, i.e., 65.55 M
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/Zengwei/icefall-asr-librispeech-
↪zipformer-2023-05-15
cd icefall-asr-librispeech-zipformer-2023-05-15

git lfs pull --include "exp/jit_script.pt"
git lfs pull --include "data/lang_bpe_500/LG.pt"

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/jit_script.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/1089-134686-0001.wav \
    ./test_wavs/1221-135766-0001.wav \
    ./test_wavs/1221-135766-0002.wav
done

sherpa-offline \
  --decoding-method=fast_beam_search \
  --nn-model=./exp/jit_script.pt \
  --lg=./data/lang_bpe_500/LG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall-asr-librispeech-zipformer-small-2023-05-16

```
# This model is trained using LibriSpeech with zipformer transducer
#
# See https://github.com/k2-fsa/icefall/pull/1058
#
# small-scaled model, number of model parameters: 23285615, i.e., 23.3 M
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/Zengwei/icefall-asr-librispeech-
↪zipformer-small-2023-05-16
cd icefall-asr-librispeech-zipformer-small-2023-05-16

git lfs pull --include "exp/jit_script.pt"
git lfs pull --include "data/lang_bpe_500/LG.pt"

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/jit_script.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/1089-134686-0001.wav \
    ./test_wavs/1221-135766-0001.wav \
```

(continues on next page)

(continued from previous page)

```
    ./test_wavs/1221-135766-0002.wav
done

sherpa-offline \
  --decoding-method=fast_beam_search \
  --nn-model=./exp/jit_script.pt \
  --lg=./data/lang_bpe_500/LG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall-asr-librispeech-zipformer-large-2023-05-16

```
# This model is trained using LibriSpeech with zipformer transducer
#
# See https://github.com/k2-fsa/icefall/pull/1058
#
# large-scaled model, number of model parameters: 148439574, i.e., 148.4 M
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/Zengwei/icefall-asr-librispeech-
↪zipformer-large-2023-05-16
cd icefall-asr-librispeech-zipformer-large-2023-05-16

git lfs pull --include "exp/jit_script.pt"
git lfs pull --include "data/lang_bpe_500/LG.pt"

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/jit_script.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/1089-134686-0001.wav \
    ./test_wavs/1221-135766-0001.wav \
    ./test_wavs/1221-135766-0002.wav
done

sherpa-offline \
  --decoding-method=fast_beam_search \
  --nn-model=./exp/jit_script.pt \
  --lg=./data/lang_bpe_500/LG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04

```
# This model is trained using GigaSpeech + LibriSpeech + Common Voice 13.0 with zipformer
#
# See https://github.com/k2-fsa/icefall/pull/1010
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/yfyeung/icefall-asr-multidataset-
↪pruned_transducer_stateless7-2023-05-04
cd icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04
git lfs pull --include "exp/cpu_jit-epoch-30-avg-4.pt"
cd exp
ln -s cpu_jit-epoch-30-avg-4.pt cpu_jit.pt
cd ..

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/1089-134686-0001.wav \
    ./test_wavs/1221-135766-0001.wav \
    ./test_wavs/1221-135766-0002.wav
done
```

icefall-asr-librispeech-pruned-transducer-stateless8-2022-12-02

```
# This model is trained using GigaSpeech + LibriSpeech with zipformer
#
# See https://github.com/k2-fsa/icefall/pull/728
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/WeijiZhuang/icefall-asr-
↪librispeech-pruned-transducer-stateless8-2022-12-02
cd icefall-asr-librispeech-pruned-transducer-stateless8-2022-12-02
git lfs pull --include "exp/cpu_jit-torch-1.10.pt"
git lfs pull --include "data/lang_bpe_500/LG.pt"

cd exp
rm cpu_jit.pt
ln -sv cpu_jit-torch-1.10.pt cpu_jit.pt
cd ..

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/1089-134686-0001.wav \
    ./test_wavs/1221-135766-0001.wav \
    ./test_wavs/1221-135766-0002.wav
done
```

(continues on next page)

(continued from previous page)

```
sherpa-offline \
  --decoding-method=fast_beam_search \
  --nn-model=./exp/cpu_jit.pt \
  --lg=./data/lang_bpe_500/LG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall-asr-librispeech-pruned-transducer-stateless8-2022-11-14

```
# This model is trained using GigaSpeech + LibriSpeech with zipformer
#
# See https://github.com/k2-fsa/icefall/pull/675
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/icefall-asr-
↪librispeech-pruned-transducer-stateless8-2022-11-14
cd icefall-asr-librispeech-pruned-transducer-stateless8-2022-11-14
git lfs pull --include "exp/cpu_jit.pt"
git lfs pull --include "data/lang_bpe_500/LG.pt"

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/1089-134686-0001.wav \
    ./test_wavs/1221-135766-0001.wav \
    ./test_wavs/1221-135766-0002.wav
done

sherpa-offline \
  --decoding-method=fast_beam_search \
  --nn-model=./exp/cpu_jit.pt \
  --lg=./data/lang_bpe_500/LG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall-asr-librispeech-pruned-transducer-stateless7-2022-11-11

```
# This model is trained using LibriSpeech with zipformer
#
# See https://github.com/k2-fsa/icefall/pull/672
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/icefall-asr-
↪librispeech-pruned-transducer-stateless7-2022-11-11
```

(continues on next page)

(continued from previous page)

```
cd icefall-asr-librispeech-pruned-transducer-stateless7-2022-11-11
git lfs pull --include "exp/cpu_jit-torch-1.10.0.pt"
git lfs pull --include "data/lang_bpe_500/LG.pt"
cd exp
ln -s cpu_jit-torch-1.10.0.pt cpu_jit.pt
cd ..

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/1089-134686-0001.wav \
    ./test_wavs/1221-135766-0001.wav \
    ./test_wavs/1221-135766-0002.wav
done

sherpa-offline \
  --decoding-method=fast_beam_search \
  --nn-model=./exp/cpu_jit.pt \
  --lg=./data/lang_bpe_500/LG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall-asr-librispeech-pruned-transducer-stateless3-2022-05-13

```
# This model is trained using LibriSpeech + GigaSpeech
#
# See https://github.com/k2-fsa/icefall/pull/363
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/icefall-asr-
↪librispeech-pruned-transducer-stateless3-2022-05-13
cd icefall-asr-librispeech-pruned-transducer-stateless3-2022-05-13
git lfs pull --include "exp/cpu_jit.pt"
git lfs pull --include "data/lang_bpe_500/LG.pt"

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/1089-134686-0001.wav \
    ./test_wavs/1221-135766-0001.wav \
    ./test_wavs/1221-135766-0002.wav
done

sherpa-offline \
  --decoding-method=fast_beam_search \
```

(continues on next page)

(continued from previous page)

```
  --nn-model=./exp/cpu_jit.pt \
  --lg=./data/lang_bpe_500/LG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall-asr-gigaspeech-pruned-transducer-stateless2

```
# This model is trained using GigaSpeech
#
# See https://github.com/k2-fsa/icefall/pull/318
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/wgb14/icefall-asr-gigaspeech-
↪pruned-transducer-stateless2
cd icefall-asr-gigaspeech-pruned-transducer-stateless2
git lfs pull --include "exp/cpu_jit-iter-3488000-avg-15.pt"
git lfs pull --include "data/lang_bpe_500/bpe.model"

cd ../exp
ln -s cpu_jit-iter-3488000-avg-15.pt cpu_jit.pt
cd ..

# Since this repo does not provide tokens.txt, we generate it from bpe.model
# by ourselves
/path/to/sherpa/scripts/bpe_model_to_tokens.py ./data/lang_bpe_500/bpe.model > ./data/
↪lang_bpe_500/tokens.txt

mkdir test_wavs
cd test_wavs
wget https://huggingface.co/csukuangfj/wav2vec2.0-torchaudio/resolve/main/test_wavs/1089-
↪134686-0001.wav
wget https://huggingface.co/csukuangfj/wav2vec2.0-torchaudio/resolve/main/test_wavs/1221-
↪135766-0001.wav
wget https://huggingface.co/csukuangfj/wav2vec2.0-torchaudio/resolve/main/test_wavs/1221-
↪135766-0002.wav

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/1089-134686-0001.wav \
    ./test_wavs/1221-135766-0001.wav \
    ./test_wavs/1221-135766-0002.wav
done
```

Chinese

icefall_asr_wenetspeech_pruned_transducer_stateless2

```
# This models is trained using WenetSpeech
#
# See https://github.com/k2-fsa/icefall/pull/349
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/luomingshuang/icefall_asr_
↪wenetspeech_pruned_transducer_stateless2

cd icefall_asr_wenetspeech_pruned_transducer_stateless2
git lfs pull --include "exp/cpu_jit_epoch_10_avg_2_torch_1.7.1.pt"
git lfs pull --include "data/lang_char/LG.pt"
cd exp
ln -s cpu_jit_epoch_10_avg_2_torch_1.7.1.pt cpu_jit.pt
cd ..

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_char/tokens.txt \
    ./test_wavs/DEV_T0000000000.wav \
    ./test_wavs/DEV_T0000000001.wav \
    ./test_wavs/DEV_T0000000002.wav
done

sherpa-offline \
  --decoding-method=$m \
  --nn-model=./exp/cpu_jit.pt \
  --lg=./data/lang_char/LG.pt \
  --tokens=./data/lang_char/tokens.txt \
  ./test_wavs/DEV_T0000000000.wav \
  ./test_wavs/DEV_T0000000001.wav \
  ./test_wavs/DEV_T0000000002.wav
```

icefall_asr_aidatatang-200zh_pruned_transducer_stateless2

```
# This models is trained using aidatatang_200zh
#
# See https://github.com/k2-fsa/icefall/pull/355
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/luomingshuang/icefall_asr_
↪aidatatang-200zh_pruned_transducer_stateless2
cd icefall_asr_aidatatang-200zh_pruned_transducer_stateless2
git lfs pull --include "exp/cpu_jit_torch.1.7.1.pt"

cd exp
ln -sv cpu_jit_torch.1.7.1.pt cpu_jit.pt
```

(continues on next page)

(continued from previous page)

```
cd ..

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_char/tokens.txt \
    ./test_wavs/T0055G0036S0002.wav \
    ./test_wavs/T0055G0036S0003.wav \
    ./test_wavs/T0055G0036S0004.wav
done
```

icefall-asr-alimeeting-pruned-transducer-stateless7

```
# This models is trained using alimeeting (https://www.openslr.org/119/)
#
# See https://github.com/k2-fsa/icefall/pull/751
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/desh2608/icefall-asr-alimeeting-
↪pruned-transducer-stateless7
cd icefall-asr-alimeeting-pruned-transducer-stateless7

git lfs pull --include "exp/cpu_jit.pt"

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_char/tokens.txt \
    ./test_wavs/165.wav \
    ./test_wavs/74.wav \
    ./test_wavs/209.wav
done
```

Chinese + English

icefall_asr_tal-csasr_pruned_transducer_stateless5

```
# This models is trained using TAL_CSASR dataset from
# https://ai.100tal.com/dataset
# where each utterance contains both English and Chinese.
#
# See https://github.com/k2-fsa/icefall/pull/428
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/luomingshuang/icefall_asr_tal-
↪csasr_pruned_transducer_stateless5
cd icefall_asr_tal-csasr_pruned_transducer_stateless5
git lfs pull --include "exp/cpu_jit.pt"
```

(continues on next page)

(continued from previous page)

```
for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_char/tokens.txt \
    ./test_wavs/210_36476_210_8341_1_1533271973_7057520_132.wav \
    ./test_wavs/210_36476_210_8341_1_1533271973_7057520_138.wav \
    ./test_wavs/210_36476_210_8341_1_1533271973_7057520_145.wav \
    ./test_wavs/210_36476_210_8341_1_1533271973_7057520_148.wav
done
```

Tibetan

icefall-asr-xbmu-amdo31-pruned-transducer-stateless7-2022-12-02

```
# This model is trained using the XBMU-AMDO31 corpus
#
# See https://github.com/k2-fsa/icefall/pull/706
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/syzym/icefall-asr-xbmu-amdo31-
↪pruned-transducer-stateless7-2022-12-02
cd icefall-asr-xbmu-amdo31-pruned-transducer-stateless7-2022-12-02
git lfs pull --include "exp/cpu_jit.pt"
git lfs pull --include "data/lang_bpe_500/LG.pt"

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/a_0_cacm-A70_31116.wav \
    ./test_wavs/a_0_cacm-A70_31117.wav \
    ./test_wavs/a_0_cacm-A70_31118.wav
done

sherpa-offline \
  --decoding-method=fast_beam_search \
  --nn-model=./exp/cpu_jit.pt \
  --lg=./data/lang_bpe_500/LG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/a_0_cacm-A70_31116.wav \
  ./test_wavs/a_0_cacm-A70_31117.wav \
  ./test_wavs/a_0_cacm-A70_31118.wav
```

icefall-asr-xbmu-amdo31-pruned-transducer-stateless5-2022-11-29

```
# This model is trained using the XBMU-AMDO31 corpus
#
# See https://github.com/k2-fsa/icefall/pull/706
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/syzym/icefall-asr-xbmu-amdo31-
↪pruned-transducer-stateless5-2022-11-29
cd icefall-asr-xbmu-amdo31-pruned-transducer-stateless5-2022-11-29
git lfs pull --include "data/lang_bpe_500/LG.pt"
git lfs pull --include "data/lang_bpe_500/tokens.txt"
git lfs pull --include "exp/cpu_jit-epoch-28-avg-23-torch-1.10.0.pt"
git lfs pull --include "test_wavs/a_0_cacm-A70_31116.wav"
git lfs pull --include "test_wavs/a_0_cacm-A70_31117.wav"
git lfs pull --include "test_wavs/a_0_cacm-A70_31118.wav"

cd exp
rm cpu_jit.pt
ln -sv cpu_jit-epoch-28-avg-23-torch-1.10.0.pt cpu_jit.pt
cd ..

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-offline \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/a_0_cacm-A70_31116.wav \
    ./test_wavs/a_0_cacm-A70_31117.wav \
    ./test_wavs/a_0_cacm-A70_31118.wav
done

sherpa-offline \
  --decoding-method=fast_beam_search \
  --nn-model=./exp/cpu_jit.pt \
  --lg=./data/lang_bpe_500/LG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/a_0_cacm-A70_31116.wav \
  ./test_wavs/a_0_cacm-A70_31117.wav \
  ./test_wavs/a_0_cacm-A70_31118.wav
```

6.2.3 Online transducer models

Hint: We use the binary `sherpa-online` below for demonstration. You can replace `sherpa-online` with `sherpa-online-websocket-server` and `sherpa-online-microphone`.

Hint: At present, only streaming transducer models from icefall are supported.

icefall

This sections lists models trained using icefall.

English

icefall-asr-librispeech-streaming-zipformer-2023-05-17

```
# This model is trained using LibriSpeech with zipformer transducer
#
# See https://github.com/k2-fsa/icefall/pull/1058
#
# normal-scaled model, number of model parameters: 66110931, i.e., 66.11 M
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/Zengwei/icefall-asr-librispeech-
↪streaming-zipformer-2023-05-17
cd icefall-asr-librispeech-streaming-zipformer-2023-05-17

git lfs pull --include "exp/jit_script_chunk_16_left_128.pt"

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-online \
    --decoding-method=$m \
    --nn-model=./exp/jit_script_chunk_16_left_128.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/1089-134686-0001.wav \
    ./test_wavs/1221-135766-0001.wav \
    ./test_wavs/1221-135766-0002.wav
done

# For fast_beam_search with LG
sherpa-online \
  --decoding-method=fast_beam_search \
  --nn-model=./exp/jit_script_chunk_16_left_128.pt \
  --lg=./data/lang_bpe_500/LG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall-asr-librispeech-pruned-transducer-stateless7-streaming-2022-12-29

```
# This model is trained using LibriSpeech with streaming zipformer transducer
#
# See https://github.com/k2-fsa/icefall/pull/787
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/Zengwei/icefall-asr-librispeech-
↪pruned-transducer-stateless7-streaming-2022-12-29
cd icefall-asr-librispeech-pruned-transducer-stateless7-streaming-2022-12-29
```

(continues on next page)

(continued from previous page)

```
git lfs pull --include "exp/cpu_jit.pt"
git lfs pull --include "data/lang_bpe_500/LG.pt"

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-online \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/1089-134686-0001.wav \
    ./test_wavs/1221-135766-0001.wav \
    ./test_wavs/1221-135766-0002.wav
done

# For fast_beam_search with LG
sherpa-online \
  --decoding-method=fast_beam_search \
  --nn-model=./exp/cpu_jit.pt \
  --lg=./data/lang_bpe_500/LG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall-asr-librispeech-conv-emformer-transducer-stateless2-2022-07-05

```
# This model is trained using LibriSpeech with ConvEmformer transducer
#
# See https://github.com/k2-fsa/icefall/pull/440
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/Zengwei/icefall-asr-librispeech-
↪conv-emformer-transducer-stateless2-2022-07-05
cd icefall-asr-librispeech-conv-emformer-transducer-stateless2-2022-07-05

git lfs pull --include "exp/cpu-jit-epoch-30-avg-10-torch-1.10.0.pt"
git lfs pull --include "data/lang_bpe_500/LG.pt"
cd exp
ln -sv cpu-jit-epoch-30-avg-10-torch-1.10.0.pt cpu_jit.pt
cd ..

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-online \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/1089-134686-0001.wav \
    ./test_wavs/1221-135766-0001.wav \
    ./test_wavs/1221-135766-0002.wav
done

# For fast_beam_search with LG
```

(continues on next page)

(continued from previous page)

```
./build/bin/sherpa-online \
  --decoding-method=fast_beam_search \
  --nn-model=./exp/cpu_jit.pt \
  --lg=./data/lang_bpe_500/LG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall-asr-librispeech-lstm-transducer-stateless2-2022-09-03

```
# This model is trained using LibriSpeech with LSTM transducer
#
# See https://github.com/k2-fsa/icefall/pull/558
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/icefall-asr-
↪librispeech-lstm-transducer-stateless2-2022-09-03
cd icefall-asr-librispeech-lstm-transducer-stateless2-2022-09-03

git lfs pull --include "exp/encoder_jit_trace-iter-468000-avg-16.pt"
git lfs pull --include "exp/decoder_jit_trace-iter-468000-avg-16.pt"
git lfs pull --include "exp/joiner_jit_trace-iter-468000-avg-16.pt"
git lfs pull --include "data/lang_bpe_500/LG.pt"

cd exp
ln -sv encoder_jit_trace-iter-468000-avg-16.pt encoder_jit_trace.pt
ln -sv decoder_jit_trace-iter-468000-avg-16.pt decoder_jit_trace.pt
ln -sv joiner_jit_trace-iter-468000-avg-16.pt joiner_jit_trace.pt
cd ..

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-online \
    --decoding-method=$m \
    --encoder-model=./exp/encoder_jit_trace.pt \
    --decoder-model=./exp/decoder_jit_trace.pt \
    --joiner-model=./exp/joiner_jit_trace.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/1089-134686-0001.wav \
    ./test_wavs/1221-135766-0001.wav \
    ./test_wavs/1221-135766-0002.wav
done

# For fast_beam_search with LG
sherpa-online \
  --decoding-method=fast_beam_search \
  --encoder-model=./exp/encoder_jit_trace.pt \
  --decoder-model=./exp/decoder_jit_trace.pt \
  --joiner-model=./exp/joiner_jit_trace.pt \
  --lg=./data/lang_bpe_500/LG.pt \
```

(continues on next page)

(continued from previous page)

```
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall-asr-librispeech-pruned-stateless-emformer-rnnt2-2022-06-01

```
# This model is trained using LibriSpeech with Emformer transducer
#
# See https://github.com/k2-fsa/icefall/pull/390
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/icefall-asr-
↪librispeech-pruned-stateless-emformer-rnnt2-2022-06-01
cd icefall-asr-librispeech-pruned-stateless-emformer-rnnt2-2022-06-01

git lfs pull --include "exp/cpu_jit-epoch-39-avg-6-use-averaged-model-1.pt"
git lfs pull --include "data/lang_bpe_500/LG.pt"
cd exp
ln -sv cpu_jit-epoch-39-avg-6-use-averaged-model-1.pt cpu_jit.pt
cd ..

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-online \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_wavs/1089-134686-0001.wav \
    ./test_wavs/1221-135766-0001.wav \
    ./test_wavs/1221-135766-0002.wav
done

# For fast_beam_search with LG

sherpa-online \
  --decoding-method=fast_beam_search \
  --nn-model=./exp/cpu_jit.pt \
  --lg=./data/lang_bpe_500/LG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_wavs/1089-134686-0001.wav \
  ./test_wavs/1221-135766-0001.wav \
  ./test_wavs/1221-135766-0002.wav
```

icefall_librispeech_streaming_pruned_transducer_stateless4_20220625

```
# This model is trained using LibriSpeech with Conformer transducer
#
# See https://github.com/k2-fsa/icefall/pull/440
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/pkufool/icefall_librispeech_
↪streaming_pruned_transducer_stateless4_20220625
cd icefall_librispeech_streaming_pruned_transducer_stateless4_20220625

git lfs pull --include "exp/cpu_jit-epoch-25-avg-3.pt"
git lfs pull --include "data/lang_bpe_500/LG.pt"
cd exp
ln -sv cpu_jit-epoch-25-avg-3.pt cpu_jit.pt
cd ..

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-online \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_bpe_500/tokens.txt \
    ./test_waves/1089-134686-0001.wav \
    ./test_waves/1221-135766-0001.wav \
    ./test_waves/1221-135766-0002.wav
done

# For fast_beam_search with LG

sherpa-online \
  --decoding-method=fast_beam_search \
  --nn-model=./exp/cpu_jit.pt \
  --lg=./data/lang_bpe_500/LG.pt \
  --tokens=./data/lang_bpe_500/tokens.txt \
  ./test_waves/1089-134686-0001.wav \
  ./test_waves/1221-135766-0001.wav \
  ./test_waves/1221-135766-0002.wav
```

Chinese

icefall_asr_wenetspeech_pruned_transducer_stateless5_streaming

```
# This model is trained using WenetSpeech with Conformer transducer
#
# See https://github.com/k2-fsa/icefall/pull/447
#
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/luomingshuang/icefall_asr_
↪wenetspeech_pruned_transducer_stateless5_streaming
cd icefall_asr_wenetspeech_pruned_transducer_stateless5_streaming

git lfs pull --include "exp/cpu_jit_epoch_7_avg_1_torch.1.7.1.pt"
```

(continues on next page)

(continued from previous page)

```
git lfs pull --include "data/lang_char/LG.pt"
cd exp
ln -sv cpu_jit_epoch_7_avg_1_torch.1.7.1.pt cpu_jit.pt
cd ..

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-online \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_char/tokens.txt \
    ./test_wavs/DEV_T0000000000.wav \
    ./test_wavs/DEV_T0000000001.wav \
    ./test_wavs/DEV_T0000000002.wav
done

# For fast_beam_search with LG

sherpa-online \
  --decoding-method=fast_beam_search \
  --nn-model=./exp/cpu_jit.pt \
  --lg=./data/lang_char/LG.pt \
  --tokens=./data/lang_char/tokens.txt \
  ./test_wavs/DEV_T0000000000.wav \
  ./test_wavs/DEV_T0000000001.wav \
  ./test_wavs/DEV_T0000000002.wav
```

Chinese + English (all-in-one)

pfluo/k2fsa-zipformer-chinese-english-mixed

It is a streaming zipformer model

```
# This model supports both Chinese and English
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/pfluo/k2fsa-zipformer-chinese-
↪english-mixed
cd k2fsa-zipformer-chinese-english-mixed
git lfs pull --include "exp/cpu_jit.pt"

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-online \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_char_bpe/tokens.txt \
    ./test_wavs/0.wav \
    ./test_wavs/1.wav \
    ./test_wavs/2.wav \
    ./test_wavs/3.wav \
    ./test_wavs/4.wav
done
```

icefall-asr-conv-emformer-transducer-stateless2-zh

It is a ConvEmformer model

```
# This model supports both Chinese and English
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/ptrnull/icefall-asr-conv-emformer-
↪transducer-stateless2-zh
cd icefall-asr-conv-emformer-transducer-stateless2-zh
git lfs pull --include "exp/cpu_jit-epoch-11-avg-1.pt"
cd exp
ln -sv cpu_jit-epoch-11-avg-1.pt cpu_jit.pt
cd ..

for m in greedy_search modified_beam_search fast_beam_search; do
  sherpa-online \
    --decoding-method=$m \
    --nn-model=./exp/cpu_jit.pt \
    --tokens=./data/lang_char_bpe/tokens.txt \
    ./test_wavs/0.wav \
    ./test_wavs/1.wav \
    ./test_wavs/2.wav \
    ./test_wavs/3.wav \
    ./test_wavs/4.wav
done
```

CHAPTER

SEVEN

SHERPA-NCNN

Hint: During speech recognition, it does not need to access the Internet. Everyting is processed locally on your device.

We support using ncnn to replace PyTorch for neural network computation. The code is put in a separate repository sherpa-ncnn

sherpa-ncnn is self-contained and everything can be compiled from source.

Please refer to https://k2-fsa.github.io/icefall/model-export/export-ncnn.html for how to export models to ncnn format.

In the following, we describe how to build sherpa-ncnn for Linux, macOS, Windows, embedded systems, Android, and iOS.

Also, we show how to use it for speech recognition with pre-trained models.

7.1 Tutorials

This page contains links to tutorials written by our users.

Caution: The tutorials are not necessarily written in English.

7.2 Installation

Hint: Please refer to *Python API* for its usage with Python.

In this section, we describe how to install sherpa-ncnn for the following platforms:

7.2.1 Installation videos

This section presents some videos about how to install and use sherpa-ncnn.

Window (64-bit)

The following video shows how to install and use sherpa-ncnn on 64-bit Windows.

Thanks to https://space.bilibili.com/7990701 for his contribution.

Caution: It is in Chinese.

7.2.2 Linux

This page describes how to build sherpa-ncnn on Linux.

Hint: You can follow this section if you want to build sherpa-ncnn directly on your board.

Hint: For the Python API, please refer to *Python API*.

All you need is to run:

x86/x86_64

32-bit ARM

64-bit ARM

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make -j6
```

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
mkdir build
cd build
cmake \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_C_FLAGS="-march=armv7-a -mfloat-abi=hard -mfpu=neon" \
  -DCMAKE_CXX_FLAGS="-march=armv7-a -mfloat-abi=hard -mfpu=neon" \
  ..
make -j6
```

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
mkdir build
```

(continues on next page)

(continued from previous page)

```
cd build
cmake \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_C_FLAGS="-march=armv8-a" \
  -DCMAKE_CXX_FLAGS="-march=armv8-a" \
  ..
make -j6
```

After building, you will find two executables inside the `bin` directory:

```
$ ls -lh bin/
total 13M
-rwxr-xr-x 1 kuangfangjun root 6.5M Dec 18 11:31 sherpa-ncnn
-rwxr-xr-x 1 kuangfangjun root 6.5M Dec 18 11:31 sherpa-ncnn-microphone
```

That's it!

Please read *Pre-trained models* for usages about the generated binaries.

Read below if you want to learn more.

You can strip the binaries by

```
$ strip bin/sherpa-ncnn
$ strip bin/sherpa-ncnn-microphone
```

After stripping, the file size of each binary is:

```
$ ls -lh bin/
total 12M
-rwxr-xr-x 1 kuangfangjun root 5.8M Dec 18 11:35 sherpa-ncnn
-rwxr-xr-x 1 kuangfangjun root 5.8M Dec 18 11:36 sherpa-ncnn-microphone
```

Hint: By default, all external dependencies are statically linked. That means, the generated binaries are self-contained.

You can use the following commands to check that and you will find they depend only on system libraries.

```
$ readelf -d bin/sherpa-ncnn

Dynamic section at offset 0x5c0650 contains 34 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libgomp.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libpthread.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libstdc++.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libm.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libmvec.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libgcc_s.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
 0x000000000000001d (RUNPATH)            Library runpath: [$ORIGIN:]

$ readelf -d bin/sherpa-ncnn-microphone

Dynamic section at offset 0x5c45d0 contains 34 entries:
  Tag        Type                         Name/Value
```

(continues on next page)

(continued from previous page)

```
0x0000000000000001 (NEEDED)             Shared library: [libpthread.so.0]
0x0000000000000001 (NEEDED)             Shared library: [libgomp.so.1]
0x0000000000000001 (NEEDED)             Shared library: [libstdc++.so.6]
0x0000000000000001 (NEEDED)             Shared library: [libm.so.6]
0x0000000000000001 (NEEDED)             Shared library: [libmvec.so.1]
0x0000000000000001 (NEEDED)             Shared library: [libgcc_s.so.1]
0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
0x000000000000001d (RUNPATH)            Library runpath: [$ORIGIN:]
```

Please create an issue at https://github.com/k2-fsa/sherpa-ncnn/issues if you have any problems.

7.2.3 macOS

This page describes how to build sherpa-ncnn on macOS.

Hint: For the Python API, please refer to *Python API*.

All you need is to run:

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make -j6
```

After building, you will find two executables inside the `bin` directory:

```
$ ls -lh bin/
total 24232
-rwxr-xr-x  1 fangjun  staff   5.9M Dec 18 12:39 sherpa-ncnn
-rwxr-xr-x  1 fangjun  staff   6.0M Dec 18 12:39 sherpa-ncnn-microphone
```

That's it!

Please read *Pre-trained models* for usages about the generated binaries.

Read below if you want to learn more.

You can strip the binaries by

```
$ strip bin/sherpa-ncnn
$ strip bin/sherpa-ncnn-microphone
```

After stripping, the file size of each binary is:

```
$ ls -lh bin/
total 23000
-rwxr-xr-x  1 fangjun  staff   5.6M Dec 18 12:40 sherpa-ncnn
-rwxr-xr-x  1 fangjun  staff   5.6M Dec 18 12:40 sherpa-ncnn-microphone
```

Hint: By default, all external dependencies are statically linked. That means, the generated binaries are self-contained.

You can use the following commands to check that and you will find they depend only on system libraries.

```
$ otool -L bin/sherpa-ncnn
bin/sherpa-ncnn:
        /usr/local/opt/libomp/lib/libomp.dylib (compatibility version 5.0.0,␣
↪current version 5.0.0)
        /usr/lib/libc++.1.dylib (compatibility version 1.0.0, current version␣
↪902.1.0)
        /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current␣
↪version 1281.100.1)

$ otool -L bin/sherpa-ncnn-microphone
bin/sherpa-ncnn-microphone:
        /System/Library/Frameworks/CoreAudio.framework/Versions/A/CoreAudio␣
↪(compatibility version 1.0.0, current version 1.0.0)
        /System/Library/Frameworks/AudioToolbox.framework/Versions/A/
↪AudioToolbox (compatibility version 1.0.0, current version 1000.0.0)
        /System/Library/Frameworks/AudioUnit.framework/Versions/A/AudioUnit␣
↪(compatibility version 1.0.0, current version 1.0.0)
        /System/Library/Frameworks/CoreFoundation.framework/Versions/A/
↪CoreFoundation (compatibility version 150.0.0, current version 1677.104.0)
        /System/Library/Frameworks/CoreServices.framework/Versions/A/
↪CoreServices (compatibility version 1.0.0, current version 1069.24.0)
        /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current␣
↪version 1281.100.1)
        /usr/local/opt/libomp/lib/libomp.dylib (compatibility version 5.0.0,␣
↪current version 5.0.0)
        /usr/lib/libc++.1.dylib (compatibility version 1.0.0, current version␣
↪902.1.0)
```

Please create an issue at https://github.com/k2-fsa/sherpa-ncnn/issues if you have any problems.

7.2.4 Windows

This page describes how to build sherpa-ncnn on Windows.

Hint: For the Python API, please refer to *Python API*.

Hint: MinGW is known not to work. Please install `Visual Studio` before you continue.

64-bit Windows (x64)

All you need is to run:

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
cmake --build . --config Release
```

It will generate two executables inside `./bin/Release/`:

- `sherpa-ncnn.exe`: For decoding a single wave file.
- `sherpa-ncnn-microphone.exe`: For real-time speech recognition from a microphone

That's it!

Please read *Pre-trained models* for usages about the generated binaries.

Please create an issue at https://github.com/k2-fsa/sherpa-ncnn/issues if you have any problems.

32-bit Windows (x86)

All you need is to run:

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
mkdir build
cd build

# Please select one toolset among VS 2015, 2017, 2019, and 2022 below
# We use VS 2022 as an example.

# For Visual Studio 2015
# cmake -T v140,host=x64 -A Win32 -D CMAKE_BUILD_TYPE=Release ..

# For Visual Studio 2017
# cmake -T v141,host=x64 -A Win32 -D CMAKE_BUILD_TYPE=Release ..

# For Visual Studio 2019
# cmake -T v142,host=x64 -A Win32 -D CMAKE_BUILD_TYPE=Release ..

# For Visual Studio 2022
cmake -T v143,host=x64 -A Win32 -D CMAKE_BUILD_TYPE=Release ..

cmake --build . --config Release -- -m:6
```

It will generate two executables inside `./bin/Release/`:

- `sherpa-ncnn.exe`: For decoding a single wave file.
- `sherpa-ncnn-microphone.exe`: For real-time speech recognition from a microphone

That's it!

Please read *Pre-trained models* for usages about the generated binaries.

Please create an issue at https://github.com/k2-fsa/sherpa-ncnn/issues if you have any problems.

7.2.5 Embedded Linux (arm)

This page describes how to build sherpa-ncnn for embedded Linux (arm, 32-bit) with cross-compiling on an x86 machine with Ubuntu OS.

> **Caution:** If you want to build sherpa-ncnn directly on your board, please don't use this document. Refer to *Linux* instead.

> **Caution:** If you want to build sherpa-ncnn directly on your board, please don't use this document. Refer to *Linux* instead.

> **Caution:** If you want to build sherpa-ncnn directly on your board, please don't use this document. Refer to *Linux* instead.

Hint: This page is for cross-compiling.

Install toolchain

The first step is to install a toolchain for cross-compiling.

> **Warning:** You can use any toolchain that is suitable for your platform. The toolchain we use below is just an example.

Visit https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-a/downloads/8-3-2019-03 to download the toolchain:

We are going to download `gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabihf.tar.xz`, which has been uploaded to https://huggingface.co/csukuangfj/sherpa-ncnn-toolchains.

Assume you want to install it in the folder `$HOME/software`:

```
mkdir -p $HOME/software
cd $HOME/software
wget https://huggingface.co/csukuangfj/sherpa-ncnn-toolchains/resolve/main/gcc-arm-8.3-
↪2019.03-x86_64-arm-linux-gnueabihf.tar.xz
tar xvf gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabihf.tar.xz
```

Next, we need to set the following environment variable:

```
export PATH=$HOME/software/gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabihf/bin:$PATH
```

To check that we have installed the cross-compiling toolchain successfully, please run:

```
arm-linux-gnueabihf-gcc --version
```

which should print the following log:

```
arm-linux-gnueabihf-gcc (GNU Toolchain for the A-profile Architecture 8.3-2019.03 (arm-
↪rel-8.36)) 8.3.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Congratulations! You have successfully installed a toolchain for cross-compiling sherpa-ncnn.

Build sherpa-ncnn

Finally, let us build sherpa-ncnn.

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
./build-arm-linux-gnueabihf.sh
```

After building, you will get two binaries:

```
$ ls -lh  build-arm-linux-gnueabihf/install/bin/

total 6.6M
-rwxr-xr-x 1 kuangfangjun root 2.2M Jan 14 21:46 sherpa-ncnn
-rwxr-xr-x 1 kuangfangjun root 2.2M Jan 14 21:46 sherpa-ncnn-alsa
```

That's it!

Hint:

- `sherpa-ncnn` is for decoding a single file
- `sherpa-ncnn-alsa` is for real-time speech recongition by reading the microphone with ALSA

Caution: We recommend that you use `sherpa-ncnn-alsa` on embedded systems such as Raspberry pi.

You need to provide a `device_name` when invoking `sherpa-ncnn-alsa`. We describe below how to find the device name for your microphone.

Run the following command:

```
arecord -l
```

to list all avaliable microphones for recording. If it complains that `arecord: command not found`, please use `sudo apt-get install alsa-utils` to install it.

If the above command gives the following output:

```
**** List of CAPTURE Hardware Devices ****
card 0: Audio [Axera Audio], device 0: 49ac000.i2s_mst-es8328-hifi-analog␣
↪es8328-hifi-analog-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

In this case, I only have 1 microphone. It is `card 0` and that card has only `device 0`. To select `card 0` and `device 0` on that card, we need to pass `plughw:0,0` to `sherpa-ncnn-alsa`. (Note: It has the format `plughw:card_number,device_index`.)

For instance, you have to use

```
# Note: We use int8 models for encoder and joiner below.
./bin/sherpa-ncnn-alsa \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/tokens.txt \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/encoder_jit_trace-
↪pnnx.ncnn.int8.param \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/encoder_jit_trace-
↪pnnx.ncnn.int8.bin \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/decoder_jit_trace-
↪pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/decoder_jit_trace-
↪pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/joiner_jit_trace-
↪pnnx.ncnn.int8.param \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/joiner_jit_trace-
↪pnnx.ncnn.int8.bin \
  "plughw:0,0"
```

Please change the card number and also the device index on the selected card accordingly in your own situation. Otherwise, you won't be able to record with your microphone.

Please read *Pre-trained models* for usages about the generated binaries.

Read below if you want to learn more.

Hint: By default, all external dependencies are statically linked. That means, the generated binaries are self-contained.

You can use the following commands to check that and you will find they depend only on system libraries.

```
$ readelf -d build-arm-linux-gnueabihf/install/bin/sherpa-ncnn

Dynamic section at offset 0x1c7ee8 contains 30 entries:
  Tag        Type                         Name/Value
 0x00000001 (NEEDED)                     Shared library: [libstdc++.so.6]
 0x00000001 (NEEDED)                     Shared library: [libm.so.6]
 0x00000001 (NEEDED)                     Shared library: [libgcc_s.so.1]
 0x00000001 (NEEDED)                     Shared library: [libpthread.so.0]
 0x00000001 (NEEDED)                     Shared library: [libc.so.6]
 0x0000000f (RPATH)                      Library rpath: [$ORIGIN]

$ readelf -d build-arm-linux-gnueabihf/install/bin/sherpa-ncnn-alsa

Dynamic section at offset 0x22ded8 contains 32 entries:
  Tag        Type                         Name/Value
 0x00000001 (NEEDED)                     Shared library: [libasound.so.2]
 0x00000001 (NEEDED)                     Shared library: [libgomp.so.1]
 0x00000001 (NEEDED)                     Shared library: [libpthread.so.0]
 0x00000001 (NEEDED)                     Shared library: [libstdc++.so.6]
 0x00000001 (NEEDED)                     Shared library: [libm.so.6]
```

(continues on next page)

(continued from previous page)

```
0x00000001 (NEEDED)                     Shared library: [libgcc_s.so.1]
0x00000001 (NEEDED)                     Shared library: [libc.so.6]
0x0000000f (RPATH)                      Library rpath: [$ORIGIN]
```

Please create an issue at https://github.com/k2-fsa/sherpa-ncnn/issues if you have any problems.

7.2.6 Embedded Linux (aarch64)

This page describes how to build sherpa-ncnn for embedded Linux (aarch64, 64-bit) with cross-compiling on an x86 machine with Ubuntu OS.

> **Caution:** If you want to build sherpa-ncnn directly on your board, please don't use this document. Refer to *Linux* instead.

> **Caution:** If you want to build sherpa-ncnn directly on your board, please don't use this document. Refer to *Linux* instead.

> **Caution:** If you want to build sherpa-ncnn directly on your board, please don't use this document. Refer to *Linux* instead.

Hint: This page is for cross-compiling.

Install toolchain

The first step is to install a toolchain for cross-compiling.

> **Warning:** You can use any toolchain that is suitable for your platform. The toolchain we use below is just an example.

Visit https://releases.linaro.org/components/toolchain/binaries/latest-7/aarch64-linux-gnu/ to download the toolchain.

We are going to download `gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu.tar.xz`, which has been uploaded to https://huggingface.co/csukuangfj/sherpa-ncnn-toolchains.

Assume you want to install it in the folder `$HOME/software`:

```
mkdir -p $HOME/software
cd $HOME/software
wget https://huggingface.co/csukuangfj/sherpa-ncnn-toolchains/resolve/main/gcc-linaro-7.
↪5.0-2019.12-x86_64_aarch64-linux-gnu.tar.xz

# For users from China
#  huggingface,
```

(continues on next page)

(continued from previous page)

```
# wget https://hf-mirror.com/csukuangfj/sherpa-ncnn-toolchains/resolve/main/gcc-linaro-7.
↪5.0-2019.12-x86_64_aarch64-linux-gnu.tar.xz
tar xvf gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu.tar.xz
```

Next, we need to set the following environment variable:

```
export PATH=$HOME/software/gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu/bin:$PATH
```

To check that we have installed the cross-compiling toolchain successfully, please run:

```
aarch64-linux-gnu-gcc --version
```

which should print the following log:

```
aarch64-linux-gnu-gcc (Linaro GCC 7.5-2019.12) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Congratulations! You have successfully installed a toolchain for cross-compiling sherpa-ncnn.

Build sherpa-ncnn

Finally, let us build sherpa-ncnn.

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
./build-aarch64-linux-gnu.sh
```

After building, you will get two binaries:

```
$ ls -lh build-aarch64-linux-gnu/install/bin/
total 10M
-rwxr-xr-x 1 kuangfangjun root 3.4M Jan 13 21:16 sherpa-ncnn
-rwxr-xr-x 1 kuangfangjun root 3.4M Jan 13 21:16 sherpa-ncnn-alsa
```

That's it!

Hint:

- `sherpa-ncnn` is for decoding a single file
- `sherpa-ncnn-alsa` is for real-time speech recongition by reading the microphone with ALSA

sherpa-ncnn-alsa

Caution: We recommend that you use `sherpa-ncnn-alsa` on embedded systems such as Raspberry pi.

You need to provide a `device_name` when invoking `sherpa-ncnn-alsa`. We describe below how to find the device name for your microphone.

Run the following command:

```
arecord -l
```

to list all avaliable microphones for recording. If it complains that `arecord: command not found`, please use `sudo apt-get install alsa-utils` to install it.

If the above command gives the following output:

```
**** List of CAPTURE Hardware Devices ****
card 3: UACDemoV10 [UACDemoV1.0], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

In this case, I only have 1 microphone. It is `card 3` and that card has only `device 0`. To select `card 3` and `device 0` on that card, we need to pass `plughw:3,0` to `sherpa-ncnn-alsa`. (Note: It has the format `plughw:card_number,device_index`.)

For instance, you have to use

```
./bin/sherpa-ncnn-alsa \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/tokens.txt \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/encoder_jit_trace-
↪pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/encoder_jit_trace-
↪pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/decoder_jit_trace-
↪pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/decoder_jit_trace-
↪pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/joiner_jit_trace-
↪pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/joiner_jit_trace-
↪pnnx.ncnn.bin \
  "plughw:3,0"
```

Please change the card number and also the device index on the selected card accordingly in your own situation. Otherwise, you won't be able to record with your microphone.

Please read *Pre-trained models* for usages about the generated binaries.

Hint: If you want to select a pre-trained model for Raspberry that can be run on real-time, we recommend you to use *marcoyang/sherpa-ncnn-conv-emformer-transducer-small-2023-01-09 (English)*.

Read below if you want to learn more.

Hint: By default, all external dependencies are statically linked. That means, the generated binaries are self-contained.

You can use the following commands to check that and you will find they depend only on system libraries.

```
$ readelf -d build-aarch64-linux-gnu/install/bin/sherpa-ncnn

Dynamic section at offset 0x302a80 contains 30 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libgomp.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libpthread.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libstdc++.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libm.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libgcc_s.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
 0x000000000000000f (RPATH)              Library rpath: [$ORIGIN]

$ readelf -d build-aarch64-linux-gnu/install/bin/sherpa-ncnn-alsa

Dynamic section at offset 0x34ea48 contains 31 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libasound.so.2]
 0x0000000000000001 (NEEDED)             Shared library: [libgomp.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libpthread.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libstdc++.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libm.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libgcc_s.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
 0x000000000000000f (RPATH)              Library rpath: [$ORIGIN]
```

Please create an issue at https://github.com/k2-fsa/sherpa-ncnn/issues if you have any problems.

7.2.7 Embedded Linux (riscv64)

This page describes how to build sherpa-ncnn for embedded Linux (RISC-V, 64-bit) with cross-compiling on an x64 machine with Ubuntu OS.

Hint: We provide a colab notebook for you to try this section step by step.

If you are using Windows/macOS or you don't want to setup your local environment for cross-compiling, please use the above colab notebook.

Install toolchain

The first step is to install a toolchain for cross-compiling.

```
sudo apt-get install gcc-riscv64-linux-gnu
sudo apt-get install g++-riscv64-linux-gnu
```

To check that you have installed the toolchain successfully, please run

```
$ riscv64-linux-gnu-gcc --version
riscv64-linux-gnu-gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
```

(continues on next page)

(continued from previous page)

```
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

$ riscv64-linux-gnu-g++ --version
riscv64-linux-gnu-g++ (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Build sherpa-ncnn

Next, let us build sherpa-ncnn.

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
./build-riscv64-linux-gnu.sh
```

After building, you will get two binaries:

```
$ ls -lh build-riscv64-linux-gnu/install/bin/
total 3.8M
-rwxr-xr-x 1 kuangfangjun root 1.9M May 23 22:12 sherpa-ncnn
-rwxr-xr-x 1 kuangfangjun root 1.9M May 23 22:12 sherpa-ncnn-alsa
```

That's it!

Hint:

- `sherpa-ncnn` is for decoding a single file
- `sherpa-ncnn-alsa` is for real-time speech recongition by reading the microphone with ALSA

Please read *Pre-trained models* for usages about the generated binaries.

Hint: If you want to select a pre-trained model for VisionFive 2 that can be run on real-time, we recommend you to use *csukuangfj/sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16 (Bilingual, Chinese + English)*.

You can use the following command with the above model:

```
./sherpa-ncnn \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/tokens.
↪txt \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/
↪encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/
↪encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/
↪decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/
↪decoder_jit_trace-pnnx.ncnn.bin \
```

(continues on next page)

(continued from previous page)

```
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/joiner_
↪jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/joiner_
↪jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/test_wavs/
↪5.wav \
  4 \
  greedy_search
```

Read below if you want to learn more.

Hint: By default, all external dependencies are statically linked. That means, the generated binaries are self-contained.

You can use the following commands to check that and you will find they depend only on system libraries.

```
$ readelf -d build-riscv64-linux-gnu/install/bin/sherpa-ncnn

Dynamic section at offset 0x1d6dc0 contains 31 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libgomp.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libpthread.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libstdc++.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libm.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libgcc_s.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [ld-linux-riscv64-
↪lp64d.so.1]
 0x000000000000001d (RUNPATH)            Library runpath: [$ORIGIN]
 0x0000000000000020 (PREINIT_ARRAY)      0x1e18e0
 0x0000000000000021 (PREINIT_ARRAYSZ)    0x8

$ readelf -d build-riscv64-linux-gnu/install/bin/sherpa-ncnn-alsa

Dynamic section at offset 0x1d3db0 contains 32 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libasound.so.2]
 0x0000000000000001 (NEEDED)             Shared library: [libgomp.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libpthread.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libstdc++.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libm.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libgcc_s.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [ld-linux-riscv64-
↪lp64d.so.1]
 0x000000000000001d (RUNPATH)            Library runpath: [$ORIGIN]
 0x0000000000000020 (PREINIT_ARRAY)      0x1de8c8
 0x0000000000000021 (PREINIT_ARRAYSZ)    0x8
```

Please create an issue at https://github.com/k2-fsa/sherpa-ncnn/issues if you have any problems.

If you want to build an Android app, please refer to *Android*. If you want to build an iOS app, please refer to *iOS*.

7.3 Python API

Hint: It is known to work for `Python >= 3.6` on Linux, macOS, and Windows.

In this section, we describe

1. How to install the Python package sherpa-ncnn
2. How to use sherpa-ncnn Python API for real-time speech recognition with a microphone
3. How to use sherpa-ncnn Python API to recognize a single file

7.3.1 Installation

You can use `1` of the `4` methods below to install the Python package sherpa-ncnn:

Method 1

Hint: This method supports `x86_64`, `arm64` (e.g., Mac M1, 64-bit Raspberry Pi), and `arm32` (e.g., 32-bit Raspberry Pi).

```
pip install sherpa-ncnn
```

If you use `Method 1`, it will install pre-compiled libraries. The `disadvantage` is that it may `not be optimized` for your platform, while the `advantage` is that you don't need to install `cmake` or a C++ compiler.

For the following methods, you have to first install:

- `cmake`, which can be installed using `pip install cmake`
- A C++ compiler, e.g., GCC on Linux and macOS, Visual Studio on Windows

Method 2

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
python3 setup.py install
```

Method 3

```
pip install git+https://github.com/k2-fsa/sherpa-ncnn
```

Method 4 (For developers and embedded boards)

x86/x86_64

32-bit ARM

64-bit ARM

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
mkdir build
cd build

cmake \
  -D SHERPA_NCNN_ENABLE_PYTHON=ON \
  -D SHERPA_NCNN_ENABLE_PORTAUDIO=OFF \
  -D BUILD_SHARED_LIBS=ON \
  ..

make -j6

export PYTHONPATH=$PWD/lib:$PWD/../sherpa-ncnn/python:$PYTHONPATH
```

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
mkdir build
cd build

cmake \
  -D SHERPA_NCNN_ENABLE_PYTHON=ON \
  -D SHERPA_NCNN_ENABLE_PORTAUDIO=OFF \
  -D BUILD_SHARED_LIBS=ON \
  -DCMAKE_C_FLAGS="-march=armv7-a -mfloat-abi=hard -mfpu=neon" \
  -DCMAKE_CXX_FLAGS="-march=armv7-a -mfloat-abi=hard -mfpu=neon" \
  ..

make -j6

export PYTHONPATH=$PWD/lib:$PWD/../sherpa-ncnn/python:$PYTHONPATH
```

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
mkdir build
cd build

cmake \
  -D SHERPA_NCNN_ENABLE_PYTHON=ON \
  -D SHERPA_NCNN_ENABLE_PORTAUDIO=OFF \
  -D BUILD_SHARED_LIBS=ON \
  -DCMAKE_C_FLAGS="-march=armv8-a" \
  -DCMAKE_CXX_FLAGS="-march=armv8-a" \
  ..

make -j6
```

(continues on next page)

(continued from previous page)

```
export PYTHONPATH=$PWD/lib:$PWD/../sherpa-ncnn/python:$PYTHONPATH
```

Let us check whether sherpa-ncnn was installed successfully:

```
python3 -c "import sherpa_ncnn; print(sherpa_ncnn.__file__)"
python3 -c "import _sherpa_ncnn; print(_sherpa_ncnn.__file__)"
```

They should print the location of `sherpa_ncnn` and `_sherpa_ncnn`.

Hint: If you use `Method 1`, `Method 2`, and `Method 3`, you can also use

```
python3 -c "import sherpa_ncnn; print(sherpa_ncnn.__version__)"
```

It should print the version of sherpa-ncnn, e.g., `1.1`.

Next, we describe how to use sherpa-ncnn Python API for speech recognition:

- (1) Real-time speech recognition with a microphone
- (2) Recognize a file

7.3.2 Real-time recognition with a microphone

The following Python code shows how to use sherpa-ncnn Python API for real-time speech recognition with a microphone.

Hint: We use sounddevice for recording. Please run `pip install sounddevice` before you run the code below.

Note: You can download the code from

https://github.com/k2-fsa/sherpa-ncnn/blob/master/python-api-examples/speech-recognition-from-microphone.py

Listing 7.1: Real-time speech recognition with a microphone using sherpa-ncnn Python API

```
import sys

try:
    import sounddevice as sd
except ImportError as e:
    print("Please install sounddevice first. You can use")
    print()
    print("  pip install sounddevice")
    print()
    print("to install it")
    sys.exit(-1)
```

(continues on next page)

(continued from previous page)

```
import sherpa_ncnn


def create_recognizer():
    # Please replace the model files if needed.
    # See https://k2-fsa.github.io/sherpa/ncnn/pretrained_models/index.html
    # for download links.
    recognizer = sherpa_ncnn.Recognizer(
        tokens="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/tokens.txt",
        encoder_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/encoder_jit_
↪trace-pnnx.ncnn.param",
        encoder_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/encoder_jit_trace-
↪pnnx.ncnn.bin",
        decoder_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/decoder_jit_
↪trace-pnnx.ncnn.param",
        decoder_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/decoder_jit_trace-
↪pnnx.ncnn.bin",
        joiner_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/joiner_jit_trace-
↪pnnx.ncnn.param",
        joiner_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/joiner_jit_trace-
↪pnnx.ncnn.bin",
        num_threads=4,
    )
    return recognizer


def main():
    print("Started! Please speak")
    recognizer = create_recognizer()
    sample_rate = recognizer.sample_rate
    samples_per_read = int(0.1 * sample_rate)  # 0.1 second = 100 ms
    last_result = ""
    with sd.InputStream(
        channels=1, dtype="float32", samplerate=sample_rate
    ) as s:
        while True:
            samples, _ = s.read(samples_per_read)  # a blocking read
            samples = samples.reshape(-1)
            recognizer.accept_waveform(sample_rate, samples)
            result = recognizer.text
            if last_result != result:
                last_result = result
                print(result)


if __name__ == "__main__":
    devices = sd.query_devices()
    print(devices)
    default_input_device_idx = sd.default.device[0]
    print(f'Use default device: {devices[default_input_device_idx]["name"]}')

    try:
```

(continues on next page)

(continued from previous page)

```
        main()
```

Code explanation:

1. Import the required packages

```
try:
    import sounddevice as sd
except ImportError as e:
    print("Please install sounddevice first. You can use")
    print()
    print("  pip install sounddevice")
    print()
    print("to install it")
    sys.exit(-1)

import sherpa_ncnn
```

Two packages are imported:

- sounddevice, for recording with a microphone
- sherpa-ncnn, for real-time speech recognition

2. Create the recognizer

```
def create_recognizer():
    # Please replace the model files if needed.
    # See https://k2-fsa.github.io/sherpa/ncnn/pretrained_models/index.html
    # for download links.
    recognizer = sherpa_ncnn.Recognizer(
        tokens="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/tokens.txt",
        encoder_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/encoder_jit_
↪trace-pnnx.ncnn.param",
        encoder_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/encoder_jit_trace-
↪pnnx.ncnn.bin",
        decoder_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/decoder_jit_
↪trace-pnnx.ncnn.param",
        decoder_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/decoder_jit_trace-
↪pnnx.ncnn.bin",
        joiner_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/joiner_jit_trace-
↪pnnx.ncnn.param",
        joiner_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/joiner_jit_trace-
↪pnnx.ncnn.bin",
        num_threads=4,
    )
    return recognizer

def main():
    print("Started! Please speak")
    recognizer = create_recognizer()
```

We use the model *csukuangfj/sherpa-ncnn-conv-emformer-transducer-2022-12-06 (Chinese + English)* as an example, which is able to recognize both English and Chinese. You can replace it with other pre-trained models.

Please refer to *Pre-trained models* for more models.

Hint: The above example uses a `float16` encoder and joiner. You can also use the following code to switch to `8-bit` (i.e., `int8`) quantized encoder and joiner.

```
recognizer = sherpa_ncnn.Recognizer(
    tokens="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/tokens.txt",
    encoder_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/encoder_
↪jit_trace-pnnx.ncnn.int8.param",
    encoder_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/encoder_jit_
↪trace-pnnx.ncnn.int8.bin",
    decoder_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/decoder_
↪jit_trace-pnnx.ncnn.param",
    decoder_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/decoder_jit_
↪trace-pnnx.ncnn.bin",
    joiner_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/joiner_jit_
↪trace-pnnx.ncnn.int8.param",
    joiner_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/joiner_jit_
↪trace-pnnx.ncnn.int8.bin",
    num_threads=4,
)
```

3. Start recording

```
    sample_rate = recognizer.sample_rate
    with sd.InputStream(
```

Note that:

- We set channel to 1 since the model supports only a single channel
- We use dtype `float32` so that the resulting audio samples are normalized to the range `[-1, 1]`.
- The sampling rate has to be `recognizer.sample_rate`, which is 16 kHz for all models at present.

4. Read audio samples from the microphone

```
    samples_per_read = int(0.1 * sample_rate)  # 0.1 second = 100 ms
    ) as s:
        while True:
```

Note that:

- It reads `100 ms` of audio samples at a time. You can choose a larger value, e.g., `200 ms`.
- No queue or callback is used. Instead, we use a blocking read here.
- The `samples` array is reshaped to a `1-D` array

5. Invoke the recognizer with audio samples

```
            samples, _ = s.read(samples_per_read)  # a blocking read
```

Note that:

- `samples` has to be a 1-D tensor and should be normalized to the range `[-1, 1]`.
- Upon accepting the audio samples, the recognizer starts the decoding automatically. There is no separate call for decoding.

6. Get the recognition result

```
            samples = samples.reshape(-1)
            recognizer.accept_waveform(sample_rate, samples)
            result = recognizer.text
            if last_result != result:
```

We use `recognizer.text` to get the recognition result. To avoid unnecessary output, we compare whether there is new result in `recognizer.text` and don't print to the console if there is nothing new recognized.

That's it!

Summary

In summary, you need to:

1. Create the recognizer
2. Start recording
3. Read audio samples
4. Call `recognizer.accept_waveform(sample_rate, samples)`
5. Call `recognizer.text` to get the recognition result

The following is a YouTube video for demonstration.

https://youtu.be/74SxVueROok

Hint: If you don't have access to YouTube, please see the following video from bilibili:

Note: https://github.com/k2-fsa/sherpa-ncnn/blob/master/python-api-examples/speech-recognition-from-microphone-with-endpoint-detection.py supports endpoint detection.

Please see the following video for its usage:

7.3.3 Recognize a file

The following Python code shows how to use sherpa-ncnn Python API to recognize a wave file.

Caution: The sampling rate of the wave file has to be 16 kHz. Also, it should contain only a single channel and samples should be 16-bit (i.e., int16) encoded.

Note: You can download the code from

https://github.com/k2-fsa/sherpa-ncnn/blob/master/python-api-examples/decode-file.py

Listing 7.2: Decode a file with sherpa-ncnn Python API

```
import wave

import numpy as np
import sherpa_ncnn


def main():
    recognizer = sherpa_ncnn.Recognizer(
        tokens="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/tokens.txt",
        encoder_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/encoder_jit_
↪trace-pnnx.ncnn.param",
        encoder_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/encoder_jit_trace-
↪pnnx.ncnn.bin",
        decoder_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/decoder_jit_
↪trace-pnnx.ncnn.param",
        decoder_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/decoder_jit_trace-
↪pnnx.ncnn.bin",
        joiner_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/joiner_jit_trace-
↪pnnx.ncnn.param",
        joiner_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/joiner_jit_trace-
↪pnnx.ncnn.bin",
        num_threads=4,
    )

    filename = (
        "./sherpa-ncnn-conv-emformer-transducer-2022-12-06/test_wavs/1.wav"
    )
    with wave.open(filename) as f:
        assert f.getframerate() == recognizer.sample_rate, (
            f.getframerate(),
            recognizer.sample_rate,
        )
        assert f.getnchannels() == 1, f.getnchannels()
        assert f.getsampwidth() == 2, f.getsampwidth()  # it is in bytes
        num_samples = f.getnframes()
        samples = f.readframes(num_samples)
        samples_int16 = np.frombuffer(samples, dtype=np.int16)
        samples_float32 = samples_int16.astype(np.float32)
```

(continues on next page)

(continued from previous page)

```
        samples_float32 = samples_float32 / 32768

    recognizer.accept_waveform(recognizer.sample_rate, samples_float32)

    tail_paddings = np.zeros(
        int(recognizer.sample_rate * 0.5), dtype=np.float32
    )
    recognizer.accept_waveform(recognizer.sample_rate, tail_paddings)

    recognizer.input_finished()

    print(recognizer.text)


if __name__ == "__main__":
    main()
```

We use the model *csukuangfj/sherpa-ncnn-conv-emformer-transducer-2022-12-06 (Chinese + English)* as an example, which is able to recognize both English and Chinese. You can replace it with other pre-trained models.

Please refer to *Pre-trained models* for more models.

Hint: The above example uses a `float16` encoder and joiner. You can also use the following code to switch to `8-bit` (i.e., `int8`) quantized encoder and joiner.

```
recognizer = sherpa_ncnn.Recognizer(
    tokens="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/tokens.txt",
    encoder_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/encoder_
↪jit_trace-pnnx.ncnn.int8.param",
    encoder_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/encoder_jit_
↪trace-pnnx.ncnn.int8.bin",
    decoder_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/decoder_
↪jit_trace-pnnx.ncnn.param",
    decoder_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/decoder_jit_
↪trace-pnnx.ncnn.bin",
    joiner_param="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/joiner_jit_
↪trace-pnnx.ncnn.int8.param",
    joiner_bin="./sherpa-ncnn-conv-emformer-transducer-2022-12-06/joiner_jit_
↪trace-pnnx.ncnn.int8.bin",
    num_threads=4,
)
```

7.4 WebAssembly

In this section, we describe how to build sherpa-ncnn for WebAssembly so that you can run real-time speech recognition with WebAssembly.

Please follow the steps below to build and run sherpa-ncnn for WebAssembly.

7.4.1 Install Emscripten

We need to compile the C/C++ files in sherpa-ncnn with the help of emscripten.

Please refer to https://emscripten.org/docs/getting_started/downloads for detailed installation instructions.

The following is an example to show you how to install it on Linux/macOS.

```
git clone https://github.com/emscripten-core/emsdk.git
cd emsdk
git pull
./emsdk install latest
./emsdk activate latest
source ./emsdk_env.sh
```

To check that you have installed emscripten successfully, please run:

```
emcc -v
```

The above command should print something like below:

```
emcc (Emscripten gcc/clang-like replacement + linker emulating GNU ld) 3.1.48␣
↪(e967e20b4727956a30592165a3c1cde5c67fa0a8)
shared:INFO: (Emscripten: Running sanity checks)
(py38) fangjuns-MacBook-Pro:open-source fangjun$ emcc -v
emcc (Emscripten gcc/clang-like replacement + linker emulating GNU ld) 3.1.48␣
↪(e967e20b4727956a30592165a3c1cde5c67fa0a8)
clang version 18.0.0 (https://github.com/llvm/llvm-project␣
↪a54545ba6514802178cf7cf1c1dd9f7efbf3cde7)
Target: wasm32-unknown-emscripten
Thread model: posix
InstalledDir: /Users/fangjun/open-source/emsdk/upstream/bin
```

Congratulations! You have successfully installed emscripten.

7.4.2 Build

After installing emscripten, we can build sherpa-ncnn for WebAssembly now.

Please use the following command to build it:

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn

cd wasm/assets
wget -q https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-
↪streaming-zipformer-bilingual-zh-en-2023-02-13.tar.bz2
```

(continues on next page)

(continued from previous page)

```
tar xvf sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13.tar.bz2
mv -v sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/*pnnx.ncnn.param .
mv -v sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/*pnnx.ncnn.bin .
mv -v sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/tokens.txt .
rm -rf sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13
rm -v sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13.tar.bz2
cd ../..

./build-wasm-simd.sh
```

Hint: You can visit https://github.com/k2-fsa/sherpa-ncnn/releases/tag/models to download a different model.

After building, you should see the following output:

```
Install the project...
-- Install configuration: "Release"
-- Installing: /Users/fangjun/open-source/sherpa-ncnn/build-wasm-simd/install/lib/
↪libkaldi-native-fbank-core.a
-- Installing: /Users/fangjun/open-source/sherpa-ncnn/build-wasm-simd/install/lib/
↪libncnn.a
-- Installing: /Users/fangjun/open-source/sherpa-ncnn/build-wasm-simd/install/./sherpa-
↪ncnn.pc
-- Installing: /Users/fangjun/open-source/sherpa-ncnn/build-wasm-simd/install/lib/
↪libsherpa-ncnn-core.a
-- Installing: /Users/fangjun/open-source/sherpa-ncnn/build-wasm-simd/install/lib/
↪libsherpa-ncnn-c-api.a
-- Installing: /Users/fangjun/open-source/sherpa-ncnn/build-wasm-simd/install/include/
↪sherpa-ncnn/c-api/c-api.h
-- Installing: /Users/fangjun/open-source/sherpa-ncnn/build-wasm-simd/install/bin/wasm/
↪sherpa-ncnn-wasm-main.js
-- Installing: /Users/fangjun/open-source/sherpa-ncnn/build-wasm-simd/install/bin/wasm/
↪sherpa-ncnn.js
-- Installing: /Users/fangjun/open-source/sherpa-ncnn/build-wasm-simd/install/bin/wasm/
↪app.js
-- Installing: /Users/fangjun/open-source/sherpa-ncnn/build-wasm-simd/install/bin/wasm/
↪index.html
-- Up-to-date: /Users/fangjun/open-source/sherpa-ncnn/build-wasm-simd/install/bin/wasm/
↪sherpa-ncnn-wasm-main.js
-- Installing: /Users/fangjun/open-source/sherpa-ncnn/build-wasm-simd/install/bin/wasm/
↪sherpa-ncnn-wasm-main.wasm
-- Installing: /Users/fangjun/open-source/sherpa-ncnn/build-wasm-simd/install/bin/wasm/
↪sherpa-ncnn-wasm-main.data
+ ls -lh install/bin/wasm
total 280152
-rw-r--r--  1 fangjun  staff   9.0K Feb  6 15:42 app.js
-rw-r--r--  1 fangjun  staff   936B Feb  6 15:42 index.html
-rw-r--r--  1 fangjun  staff   135M Feb  6 17:06 sherpa-ncnn-wasm-main.data
-rw-r--r--  1 fangjun  staff    79K Feb  6 17:06 sherpa-ncnn-wasm-main.js
-rw-r--r--  1 fangjun  staff   1.7M Feb  6 17:06 sherpa-ncnn-wasm-main.wasm
-rw-r--r--  1 fangjun  staff   6.9K Feb  6 15:42 sherpa-ncnn.js
```

Now you can use the following command to run it:

```
cd build-wasm-simd/install/bin/wasm/
python3 -m http.server 6006
```

Start your browser and visit http://localhost:6006/; you should see the following page:

Now click start and speak! You should see the recognition results in the text box.

Warning: We are using a bilingual model (Chinese + English) in the above example, which means you can only speak Chinese or English in this case.

A screenshot is given below:

Congratulations! You have successfully run real-time speech recognition with WebAssembly in your browser.

7.4.3 Use pre-built WebAssembly library

In this section, we describe how to use the pre-built WebAssembly library of sherpa-ncnn for real-time speech recognition.

Note: Note that the pre-built library used in this section uses a bilingual model (Chinese + English), which is from *csukuangfj/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13 (Bilingual, Chinese + English)*.

Download

Please use the following command to download the pre-built library for version `v2.1.7`, which is the latest release as of 2024.02.06.

Hint: Please always use the latest release. You can visit https://github.com/k2-fsa/sherpa-ncnn/releases to find the latest release.

```
wget -q https://github.com/k2-fsa/sherpa-ncnn/releases/download/v2.1.7/sherpa-ncnn-wasm-
↪simd-v2.1.7.tar.bz2
tar xvf sherpa-ncnn-wasm-simd-v2.1.7.tar.bz2
rm sherpa-ncnn-wasm-simd-v2.1.7.tar.bz2
cd sherpa-ncnn-wasm-simd-v2.1.7

python3 -m http.server 6006
```

Start your browser and visit http://localhost:6006/; you should see the following page:

Now click start and speak! You should see the recognition results in the text box.

Warning: We are using a bilingual model (Chinese + English) in the above example, which means you can only speak Chinese or English in this case.

A screenshot is given below:

Congratulations! You have successfully run real-time speech recognition with WebAssembly in your browser.

7.4.4 Huggingface Spaces (WebAssembly)

We provide two Huggingface spaces so that you can try real-time speech recognition with WebAssembly in your browser.

English only

https://huggingface.co/spaces/k2-fsa/web-assembly-asr-sherpa-ncnn-en

Hint: If you don't have access to Huggingface, please visit the following mirror:

https://modelscope.cn/studios/k2-fsa/web-assembly-asr-sherpa-ncnn-en/summary

Note: The script for building this space can be found at https://github.com/k2-fsa/sherpa-ncnn/blob/master/.github/workflows/wasm-simd-hf-space-en.yaml

Chinese + English

https://huggingface.co/spaces/k2-fsa/web-assembly-asr-sherpa-ncnn-zh-en

Hint: If you don't have access to Huggingface, please visit the following mirror:

https://modelscope.cn/studios/k2-fsa/web-assembly-asr-sherpa-ncnn-zh-en/summary

Note: The script for building this space can be found at https://github.com/k2-fsa/sherpa-ncnn/blob/master/.github/workflows/wasm-simd-hf-space-zh-en.yaml

7.5 C API

In this section, we describe how to use the C API of sherpa-ncnn.

Specifically, we will describe:

- How to generate required files
- How to use `pkg-config` with sherpa-ncnn

7.5.1 Generate required files

Before using the C API of sherpa-ncnn, we need to first build required libraries. You can choose either to build static libraries or shared libraries.

Build shared libraries

Assume that we want to put library files and header files in the directory `/tmp/sherpa-ncnn/shared`:

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
mkdir build-shared
cd build-shared

cmake \
  -DSHERPA_NCNN_ENABLE_C_API=ON \
  -DCMAKE_BUILD_TYPE=Release \
  -DBUILD_SHARED_LIBS=ON \
  -DCMAKE_INSTALL_PREFIX=/tmp/sherpa-ncnn/shared \
  ..

make -j6
make install
```

You should find the following files inside `/tmp/sherpa-ncnn/shared`:

macOS

Linux

```
$ tree /tmp/sherpa-ncnn/shared/
/tmp/sherpa-ncnn/shared/
├── bin
│   ├── sherpa-ncnn
│   └── sherpa-ncnn-microphone
├── include
│   └── sherpa-ncnn
│       └── c-api
│           └── c-api.h
├── lib
│   ├── libkaldi-native-fbank-core.dylib
│   ├── libncnn.dylib
│   ├── libsherpa-ncnn-c-api.dylib
│   └── libsherpa-ncnn-core.dylib
└── sherpa-ncnn.pc

5 directories, 8 files
```

```
$ tree /tmp/sherpa-ncnn/shared/
/tmp/sherpa-ncnn/shared/
├── bin
│   ├── sherpa-ncnn
│   └── sherpa-ncnn-microphone
├── include
│   └── sherpa-ncnn
│       └── c-api
│           └── c-api.h
├── lib
│   ├── libkaldi-native-fbank-core.so
```

(continues on next page)

(continued from previous page)

```
│   ├── libncnn.so
│   ├── libsherpa-ncnn-c-api.so
│   └── libsherpa-ncnn-core.so
└── sherpa-ncnn.pc

5 directories, 8 files
```

Build static libraries

Assume that we want to put library files and header files in the directory `/tmp/sherpa-ncnn/static`:

```
git clone https://github.com/k2-fsa/sherpa-ncnn
cd sherpa-ncnn
mkdir build-static
cd build-static

cmake \
  -DSHERPA_NCNN_ENABLE_C_API=ON \
  -DCMAKE_BUILD_TYPE=Release \
  -DBUILD_SHARED_LIBS=OFF \
  -DCMAKE_INSTALL_PREFIX=/tmp/sherpa-ncnn/static \
  ..

make -j6
make install
```

You should find the following files in `/tmp/sherpa-ncnn/static`:

```
$ tree /tmp/sherpa-ncnn/static/
/tmp/sherpa-ncnn/static/
├── bin
│   ├── sherpa-ncnn
│   └── sherpa-ncnn-microphone
├── include
│   └── sherpa-ncnn
│       └── c-api
│           └── c-api.h
├── lib
│   ├── libkaldi-native-fbank-core.a
│   ├── libncnn.a
│   ├── libsherpa-ncnn-c-api.a
│   └── libsherpa-ncnn-core.a
└── sherpa-ncnn.pc

5 directories, 8 files
```

7.5.2 Build decode-file-c-api.c with generated files

To build the following file:

https://github.com/k2-fsa/sherpa-ncnn/blob/master/c-api-examples/decode-file-c-api.c

We can use:

static link

dynamic link

```
export PKG_CONFIG_PATH=/tmp/sherpa-ncnn/static:$PKG_CONFIG_PATH

cd ./c-api-examples
gcc -o decode-file-c-api $(pkg-config --cflags sherpa-ncnn) ./decode-file-c-api.c $(pkg-
↪config --libs sherpa-ncnn)
```

```
export PKG_CONFIG_PATH=/tmp/sherpa-ncnn/shared:$PKG_CONFIG_PATH

cd ./c-api-examples
gcc -o decode-file-c-api $(pkg-config --cflags sherpa-ncnn) ./decode-file-c-api.c $(pkg-
↪config --libs sherpa-ncnn)
```

7.6 Endpointing

We have three rules for endpoint detection. If any of them is activated, we assume an endpoint is detected.

Note: We borrow the implementation from

https://kaldi-asr.org/doc/structkaldi_1_1OnlineEndpointRule.html

7.6.1 Rule 1

In `Rule 1`, we count the duration of trailing silence. If it is larger than a user specified value, `Rule 1` is activated. The following is an example, which uses `2.4 seconds` as the threshold.

Two cases are given:

(1) In the first case, nothing has been decoded when the duration of trailing silence reaches 2.4 seconds.

(2) In the second case, we first decode something before the duration of trailing silence reaches 2.4 seconds.

In both cases, `Rule 1` is activated.

Hint: In the Python API, you can specify `rule1_min_trailing_silence` while constructing an instance of `sherpa_ncnn.Recognizer`.

In the C++ API, you can specify `rule1.min_trailing_silence` when creating `EndpointConfig`.

7.6.2 Rule 2

In `Rule 2`, we require that it has to first decode something before we count the trailing silence. In the following example, after decoding something, `Rule 2` is activated when the duration of trailing silence is larger than the user specified value `1.2` seconds.

Hint: In the Python API, you can specify `rule2_min_trailing_silence` while constructing an instance of `sherpa_ncnn.Recognizer`.

In the C++ API, you can specify `rule2.min_trailing_silence` when creating `EndpointConfig`.

7.6.3 Rule 3

`Rule 3` is activated when the utterance length in seconds is larger than a given value. In the following example, `Rule 3` is activated after the first segment reaches a given value, which is `20` seconds in this case.

Hint: In the Python API, you can specify `rule3_min_utterance_length` while constructing an instance of `sherpa_ncnn.Recognizer`.

In the C++ API, you can specify `rule3.min_utterance_length` when creating `EndpointConfig`.

Note: If you want to deactive this rule, please provide a very large value for `rule3_min_utterance_length` or `rule3.min_utterance_length`.

7.6.4 Demo

Multilingual (Chinese + English)

The following video demonstrates using the Python API of sherpa-ncnn for real-time speech recogntinion with endpointing.

Hint: The code is available at

https://github.com/k2-fsa/sherpa-ncnn/blob/master/python-api-examples/speech-recognition-from-microphone-with-endpoint-detectio
py

7.6.5 FAQs

How to compute duration of silence

For each frame to be decoded, we can output either a blank or a non-blank token. We record the number of contiguous blanks that has been decoded so far. In the current default setting, each frame is `10 ms`. Thus, we can get the duration of trailing silence by counting the number of contiguous trailing blanks.

Note: If a model uses a subsampling factor of 4, the time resolution becomes `10 * 4 = 40 ms`.

7.7 Android

In this section, we describe how to build an Android app for `real-time` speech recognition with sherpa-ncnn. We also provide real-time speech recognition video demos.

Hint: During speech recognition, it does not need to access the Internet. Everyting is processed locally on your phone.

7.7.1 Video demos

In this page, we list some videos about using sherpa-ncnn for real-time speech recognition on Android.

Hint: You can find pre-built `APK` packages used by the following videos at:

https://huggingface.co/csukuangfj/sherpa-ncnn-apk/tree/main

- CPU versions require `Android >= 5.0`
- GPU versions with `Vulkan` require `Android >= 7.0`

Note: You can also find the latest APK for each release at

https://github.com/k2-fsa/sherpa-ncnn/releases

Video 1: Chinese

Video 2: Chinese + English

Video 3: Chinese with background noise

Video 4: Chinese poem with background music

7.7.2 Build sherpa-ncnn for Android

Install Android Studio

The first step is to download and install Android Studio.

Please refer to https://developer.android.com/studio for how to install Android Studio.

Hint: Any recent version of Android Studio should work fine. Also, you can use the default settings of Android Studio during installation.

For reference, we post the version we are using below:

Download sherpa-ncnn

Next, download the source code of sherpa-ncnn:

```
git clone https://github.com/k2-fsa/sherpa-ncnn
```

Install NDK

Step 1, start Android Studio.

Step 2, Open `sherpa-ncnn/android/SherpaNcnn`.

Step 3, Select `Tools -> SDK Manager`.

Step 4, `Install NDK`.

In the following, we assume `Android SDK location` was set to `/Users/fangjun/software/my-android`. You can change it accordingly below.

After installing NDK, you can find it in

```
/Users/fangjun/software/my-android/ndk/22.1.7171670
```

Warning: If you selected a different version of NDK, please replace `22.1.7171670` accordingly.

Next, let us set the environment variable `ANDROID_NDK` for later use.

Fig. 7.1: Step 1: Click `Open` to select `sherpa-ncnn/android/SherpaNcnn`

Fig. 7.2: Step 2: Open `SherpaNcnn`.

Fig. 7.3: Step 3: Select `Tools -> SDK Manager`.

Fig. 7.4: Step 4: Install NDK.

```
export ANDROID_NDK=/Users/fangjun/software/my-android/ndk/22.1.7171670
```

Note: Note from https://github.com/Tencent/ncnn/wiki/how-to-build#build-for-android

(Important) remove the hardcoded debug flag in Android NDK to fix the android-ndk issue: https://github.com/android/ndk/issues/243

1. open `$ANDROID_NDK/build/cmake/android.toolchain.cmake` for ndk < r23 or `$ANDROID_NDK/build/cmake/android-legacy.toolchain.cmake` for ndk >= r23

2. delete the line containing "-g"

```
list(APPEND ANDROID_COMPILER_FLAGS
-g
-DANDROID
```

Caution: If you don't delete the line containin `-g` above, the generated library `libncnn.so` can be as large as `21 MB` or even larger!

Build sherpa-ncnn (C++ code)

After installing `NDK`, it is time to build the C++ code of sherpa-ncnn.

In the following, we show how to build sherpa-ncnn for the following Android ABIs:

- `arm64-v8a`
- `armeabi-v7a`
- `x86_64`
- `x86`

Caution: You only need to select one and only one ABI. `arm64-v8a` is probably the most common one.

If you want to test the app on an emulator, you probably need `x86_64`.

Hint: Building scripts for this section are for macOS and Linux. If you are using Windows or if you don't want to build the shared libraries by yourself, you can download pre-compiled shared libraries for this section by visiting

https://github.com/k2-fsa/sherpa-ncnn/releases

Hint: We provide a colab notebook for you to try this section step by step.

If you are using Windows or you don't want to setup your local environment to build the C++ libraries, please use the above colab notebook.

Build for arm64-v8a

```
cd sherpa-ncnn # Go to the root repo
./build-android-arm64-v8a.sh
```

After building, you will find the following shared libraries:

```
$ ls -lh build-android-arm64-v8a/install/lib/lib*.so
-rwxr-xr-x  1 fangjun  staff   848K Dec 18 16:49 build-android-arm64-v8a/install/lib/
↪libkaldi-native-fbank-core.so
-rwxr-xr-x  1 fangjun  staff   3.4M Dec 18 16:49 build-android-arm64-v8a/install/lib/
↪libncnn.so
-rwxr-xr-x  1 fangjun  staff   195K Dec 18 16:49 build-android-arm64-v8a/install/lib/
↪libsherpa-ncnn-core.so
-rwxr-xr-x  1 fangjun  staff    19K Dec 18 16:49 build-android-arm64-v8a/install/lib/
↪libsherpa-ncnn-jni.so
```

Please copy them to `android/SherpaNcnn/app/src/main/jniLibs/arm64-v8a/`:

```
$ cp build-android-arm64-v8a/install/lib/lib*.so  android/SherpaNcnn/app/src/main/
↪jniLibs/arm64-v8a/
```

You should see the following screen shot after running the above copy `cp` command.

Note: If you have `Android >= 7.0` and want to run sherpa-ncnn on GPU, please replace `./build-android-arm64-v8a.sh` with `./build-android-arm64-v8a-with-vulkan.sh` and replace `build-android-arm64-v8a/install/lib/lib*.so` with `./build-android-arm64-v8a-with-vulkan/install/lib/lib*.so`. That is all you need to do and you don't need to change any code.

Also, you need to install Vulkan sdk. Please see https://github.com/k2-fsa/sherpa-ncnn/blob/master/install-vulkan-macos.md for details.

Build for armeabi-v7a

```
cd sherpa-ncnn # Go to the root repo
./build-android-armv7-eabi.sh
```

After building, you will find the following shared libraries:

```
$ ls -lh build-android-armv7-eabi/install/lib/lib*.so
-rwxr-xr-x  1 fangjun  staff   513K Dec 18 17:04 build-android-armv7-eabi/install/lib/
↪libkaldi-native-fbank-core.so
-rwxr-xr-x  1 fangjun  staff   1.9M Dec 18 17:04 build-android-armv7-eabi/install/lib/
↪libncnn.so
-rwxr-xr-x  1 fangjun  staff   163K Dec 18 17:04 build-android-armv7-eabi/install/lib/
↪libsherpa-ncnn-core.so
-rwxr-xr-x  1 fangjun  staff    28K Dec 18 17:04 build-android-armv7-eabi/install/lib/
↪libsherpa-ncnn-jni.so
```

Please copy them to `android/SherpaNcnn/app/src/main/jniLibs/armeabi-v7a/`:

SherpaNcnn 〉 app 〉 src 〉 main 〉 jniLibs 〉 arm64-v8a
Android
Project
Commit
Pull Requests
Resource Manager
app
manifests
java
java (generated)
assets
jniLibs
arm64-v8a
.gitkeep
libkaldi-native-fbank-core.so
libncnn.so
libsherpa-ncnn-core.so
libsherpa-ncnn-jni.so
armeabi-v7a
x86_64
.gitignore
readme.md
res
res (generated)
Gradle Scripts
You should see thses
4 shared libraries
after building sherpa-
ncnn for arm64-v8a

```
cp build-android-armv7-eabi/install/lib/lib*.so android/SherpaNcnn/app/src/main/jniLibs/
↪armeabi-v7a/
```

You should see the following screen shot after running the above copy `cp` command.

Build for x86_64

```
cd sherpa-ncnn # Go to the root repo
./build-android-x86-64.sh
```

After building, you will find the following shared libraries:

```
$ ls -lh build-android-x86-64/install/lib/lib*.so
-rwxr-xr-x  1 fangjun  staff   901K Dec 18 17:14 build-android-x86-64/install/lib/
↪libkaldi-native-fbank-core.so
-rwxr-xr-x  1 fangjun  staff   6.9M Dec 18 17:14 build-android-x86-64/install/lib/
↪libncnn.so
```

(continues on next page)

(continued from previous page)

```
-rwxr-xr-x  1 fangjun  staff   208K Dec 18 17:14 build-android-x86-64/install/lib/
↪libsherpa-ncnn-core.so
-rwxr-xr-x  1 fangjun  staff    19K Dec 18 17:14 build-android-x86-64/install/lib/
↪libsherpa-ncnn-jni.so
```

Please copy them to `android/SherpaNcnn/app/src/main/jniLibs/x86_64/`:

```
cp build-android-x86-64/install/lib/lib*.so android/SherpaNcnn/app/src/main/jniLibs/x86_
↪64/
```

You should see the following screen shot after running the above copy `cp` command.

Build for x86

```
cd sherpa-ncnn # Go to the root repo
./build-android-x86.sh
```

Download pre-trained models

Please read *Pre-trained models* for all available pre-trained models.

In the following, we use a pre-trained model from https://huggingface.co/csukuangfj/sherpa-ncnn-conv-emformer-transducer-2022-12-06, which supports both Chinese and English.

Hint: The model is trained using icefall and the original torchscript model is from https://huggingface.co/ptrnull/icefall-asr-conv-emformer-transducer-stateless2-zh.

Use the following command to download the pre-trained model and place it into `android/SherpaNcnn/app/src/main/assets/`:

```
cd android/SherpaNcnn/app/src/main/assets/

sudo apt-get install git-lfs

GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/sherpa-ncnn-conv-
↪emformer-transducer-2022-12-06
cd sherpa-ncnn-conv-emformer-transducer-2022-12-06
git lfs pull --include "*.bin"

# Now, remove extra files to reduce the file size of the generated apk
rm -rf .git test_wavs scripts/
rm export-for-ncnn.sh *.png README.md
```

In the end, you should have the following files:

```
$ ls -lh
total 525224
-rw-r--r--  1 fangjun  staff   5.9M Dec 18 17:40 decoder_jit_trace-pnnx.ncnn.bin
-rw-r--r--  1 fangjun  staff   439B Dec 18 17:39 decoder_jit_trace-pnnx.ncnn.param
-rw-r--r--  1 fangjun  staff   141M Dec 18 17:40 encoder_jit_trace-pnnx.ncnn.bin
-rw-r--r--  1 fangjun  staff    99M Dec 18 17:40 encoder_jit_trace-pnnx.ncnn.int8.bin
-rw-r--r--  1 fangjun  staff    78K Dec 18 17:40 encoder_jit_trace-pnnx.ncnn.int8.param
-rw-r--r--  1 fangjun  staff    79K Dec 18 17:39 encoder_jit_trace-pnnx.ncnn.param
-rw-r--r--  1 fangjun  staff   6.9M Dec 18 17:40 joiner_jit_trace-pnnx.ncnn.bin
-rw-r--r--  1 fangjun  staff   3.5M Dec 18 17:40 joiner_jit_trace-pnnx.ncnn.int8.bin
-rw-r--r--  1 fangjun  staff   498B Dec 18 17:40 joiner_jit_trace-pnnx.ncnn.int8.param
-rw-r--r--  1 fangjun  staff   490B Dec 18 17:39 joiner_jit_trace-pnnx.ncnn.param
-rw-r--r--  1 fangjun  staff    53K Dec 18 17:39 tokens.txt

$ du -h -d1 .
256M    .
```

You should see the following screen shot after downloading the pre-trained model:

SherpaNcnn
app
Android
app
manifests
java
com.k2fsa.sherpa.ncnn
com.k2fsa.sherpa.ncnn (androidTest)
com.k2fsa.sherpa.ncnn (test)
java (generated)
assets
sherpa-ncnn-conv-emformer-transducer-2022-12-06
.gitattributes
decoder_jit_trace-pnnx.ncnn.bin
decoder_jit_trace-pnnx.ncnn.param
encoder_jit_trace-pnnx.ncnn.bin
encoder_jit_trace-pnnx.ncnn.int8.bin
encoder_jit_trace-pnnx.ncnn.int8.param
encoder_jit_trace-pnnx.ncnn.param
joiner_jit_trace-pnnx.ncnn.bin
joiner_jit_trace-pnnx.ncnn.int8.bin
joiner_jit_trace-pnnx.ncnn.int8.param
joiner_jit_trace-pnnx.ncnn.param
tokens.txt
.gitignore
jniLibs
res
res (generated)
Gradle Scripts
Project
Commit
Pull Requests
Resource Manager
Only these files are needed after
downloading the pre-trained model

Hint: If you select a different pre-trained model, make sure that you also change the corresponding code listed in the following screen shot:

Generate APK

Finally, it is time to build sherpa-ncnn to generate an APK package.

Select `Build -> Make Project`, as shown in the following screen shot.

You can find the generated APK in `android/SherpaNcnn/app/build/outputs/apk/debug/app-debug.apk`:

```
$ ls -lh android/SherpaNcnn/app/build/outputs/apk/debug/app-debug.apk
-rw-r--r--  1 fangjun  staff   152M Dec 18 17:53 android/SherpaNcnn/app/build/outputs/
↪apk/debug/app-debug.apk
```

Congratulations! You have successfully built an APK for Android.

Read below to learn more.

Analyze the APK

Select `Build -> Analyze APK ...` in the above screen shot, in the popped-up dialog select the generated APK `app-debug.apk`, and you will see the following screen shot:

You can see from the above screen shot that most part of the APK is occupied by the pre-trained model, while the runtime, including the shared libraries, is only `1.7 MB`.

Hint: We have pre-built APKs that can be downloaded from https://huggingface.co/csukuangfj/sherpa-ncnn-apk

Please refer to demo videos about using the above APKs: *Video demos*.

Code
Refactor
Build
Run
Tools
Git
Window
Help
Make Project
⌘F9
Make
Run Generate Sources Gradle Tasks
Recompile
Select Build Variant...
Build Bundle(s) / APK(s)
Generate Signed Bundle / APK...
Analyze APK...
Deploy Module to App Engine...
Edit Build Types...
Edit Flavors...
Edit Libraries and Dependencies...
Clean Project
Refresh Linked C++ Projects
Rebuild Project
ncnn
MainActivi
Click
Build -> Make
Project

Code
Refactor
Build
Run
Tools
Git
Window
Help
Make Project
⌘F9
Make
Run Generate Sources Gradle Tasks
Recompile
Select Build Variant...
Build Bundle(s) / APK(s)
Generate Signed Bundle / APK...
Analyze APK...
Deploy Module to App Engine...
Edit Build Types...
Edit Flavors...
Edit Libraries and Dependencies...
Clean Project
Refresh Linked C++ Projects
Rebuild Project
ncnn
MainActivi
Select
Build ->
Analyze APK...

7.8 iOS

In this section, we describe how to build an iOS app for `real-time` speech recognition with sherpa-ncnn and run it within a simulator on your Mac, run it on you iPhone or iPad.

We also provide video demos for real-time speech recognition.

Hint: During speech recognition, it does not need to access the Internet. Everyting is processed locally on your iPhone or iPad.

7.8.1 Video demos

In this page, we list some videos about using sherpa-ncnn for real-time speech recognition on `iPhone` and `iPad`.

Video 1: Chinese + English on iPhone 14 Pro (simulator)

Video 2: Chinese + English on iPad 11 Pro (simulator)

7.8.2 Build sherpa-ncnn for iOS

This section describes how to build sherpa-ncnn for `iPhone` and `iPad`.

Requirement

Warning: The minimum deployment requires the iOS version >= `13.0`.

Before we continue, please make sure the following requirements are satisfied:

- macOS. It won't work on Windows or Linux.
- Xcode. The version `14.2 (14C18)` is known to work. Other versions may also work.
- CMake. CMake 3.25.1 is known to work. Other versions may also work.
- (Optional) iPhone or iPad. This is for testing the app on your device. If you don't have a device, you can still run the app within a simulator on your Mac.

Caution:

If you get the following error:

```
CMake Error at toolchains/ios.toolchain.cmake:544 (get_filename_component):
  get_filename_component called with incorrect number of arguments
Call Stack (most recent call first):
  /usr/local/Cellar/cmake/3.29.0/share/cmake/Modules/CMakeDetermineSystem.
↪cmake:146 (include)
  CMakeLists.txt:2 (project)
```

please run:

```
sudo xcode-select --install
sudo xcodebuild -license
```

And then delete the build directory `./build-ios` and re-build.

Please see also https://github.com/k2-fsa/sherpa-onnx/issues/702.

Download sherpa-ncnn

First, let us download the source code of sherpa-ncnn.

Note: In the following, I will download sherpa-ncnn to `$HOME/open-source`, i.e., `/Users/fangjun/open-source`, on my Mac.

You can put it anywhere as you like.

```
mkdir -p $HOME/open-source
cd $HOME/open-source
git clone https://github.com/k2-fsa/sherpa-ncnn
```

Build sherpa-ncnn (in commandline, C++ Part)

After downloading sherpa-ncnn, let us build the C++ part of sherpa-ncnn.

```
cd $HOME/open-source/sherpa-ncnn/
./build-ios.sh
```

It will generate a directory `$HOME/open-source/sherpa-ncnn/build-ios`, which we have already pre-configured for you in Xcode.

Hint: You don't have to look at the generated files in `$HOME/open-source/sherpa-ncnn/build-ios` to build an app. We have pre-configured it for you.

If you are eager to learn more about the generated files or want to use sherpa-ncnn in your own iOS project, please have a look at *For the more curious*.

Build sherpa-ncnn (in Xcode)

Use the following command to open sherpa-ncnn in Xcode:

```
cd $HOME/open-source/sherpa-ncnn/ios-swift/SherpaNcnn
open SherpaNcnn.xcodeproj
```

It will start Xcode and you will see the following screenshot:

Fig. 7.5: Screenshot after running the command `open SherpaNcnn.xcodeproj`

Please select `Product -> Build` to build the project. See the screenshot below:

Fig. 7.6: Screenshot for selecting `Product -> Build`

After finishing the build, you should see the following screenshot:

Fig. 7.7: Screenshot after finishing the build.

Congratulations! You have successfully built the project. Let us run the project by selecting `Product -> Run`, which is shown in the following screenshot:

Please wait for a few seconds before Xcode starts the simulator.

Unfortunately, it will throw the following error:

The reason for the above error is that we have not provided the pre-trained model yet.

The file ViewController.swift pre-selects the pre-trained model to be *csukuangfj/sherpa-ncnn-conv-emformer-transducer-2022-12-06 (Chinese + English)*, shown in the screenshot below:

Fig. 7.8: Screenshot for `Product -> Run`.

Fig. 7.9: Screenshot for the error

Fig. 7.10: Screenshot for the pre-selected pre-trained model

Let us add the pre-trained model *csukuangfj/sherpa-ncnn-conv-emformer-transducer-2022-12-06 (Chinese + English)* to Xcode. Please follow *csukuangfj/sherpa-ncnn-conv-emformer-transducer-2022-12-06 (Chinese + English)* to download it from huggingface. You can download it to any directory as you like.

Please right click the project `SherpaNcnn` and select `Add Files to "SherpaNcnn"...` in the popup menu, as is shown in the screenshot below:

Fig. 7.11: Screenshot for adding files to SherpaNcnn

In the popup dialog, switch to the folder where you just downloaded the pre-trained model.

In the screenshot below, it is the folder `/Users/fangjun/open-source/icefall-models/sherpa-ncnn-conv-emformer-transducer-2022-12-06`:

Fig. 7.12: Screenshot for navigating to the folder containing the downloaded pre-trained

Select required files and click the button `Add`:

After adding pre-trained model files to Xcode, you should see the following screenshot:

At this point, you should be able to select the menu `Product -> Run` to run the project and you should finally see the following screenshot:

Click the button to start recording! A screenshot is given below:

Congratulations! You have finally succeeded in running sherpa-ncnn with iOS, though it is in a simulator.

Please read below if you want to run sherpa-ncnn on your iPhone or iPad.

Fig. 7.13: Screenshot for selecting required files

Fig. 7.14: Screenshot after add pre-trained model files

Fig. 7.15: Screenshot for a successful run.

Run sherpa-ncnn on your iPhone/iPad

First, please make sure the iOS version of your iPhone/iPad is >= `13.0`.

Click the menu `Xcode -> Settings...`, as is shown in the following screenshot:

In the popup dialog, please select `Account` and click + to add your Apple ID, as is shown in the following `screenshots`.

After adding your Apple ID, please connect your iPhone or iPad to your Mac and select your device in Xcode. The following screenshot is an example to select my iPhone.

Now your Xcode should look like below after selecting a device:

Please select `Product -> Run` again to run sherpa-ncnn on your selected device, as is shown in the following screenshot:

After a successful build, check your iPhone/iPad and you should see the following screenshot:

To fix that, please select `Settings -> General -> Device Management` on your device

Fig. 7.16: Screenshot for recording and recognition.

Please click `Apple Development: csukuangfj...` and click `Trust "Apple Development: csukuangfj@g..."` in the subsequent dialog, as is shown below:

At this point, you should be able to run the app on your device. The following is a screenshot about running it on my iPhone:

Congratulations! You have successfully run sherpa-ncnn on your device!

Fig. 7.17: Screenshot for `Xcode -> Settings...`

Fig. 7.18: Screenshot for selecting `Account` and click +.

7.8.3 For the more curious

This section is for those who want to learn more about how to use sherpa-ncnn in an iOS project.

Files generated by running ./build-ios.sh

After running:

```
./build-ios.sh
```

You may be curious about the generated files.

Hint: Please have a look at `./build-ios.sh` so that you know what it does for you.

The above command generates files inside the directory `./build-ios`:

```
sherpa-ncnn fangjun$ ls -lh build-ios
total 1912
drwxr-xr-x   6 fangjun  staff   192B Feb 26 16:48 build
drwxr-xr-x   4 fangjun  staff   128B Feb 26 16:46 install
drwxr-xr-x  15 fangjun  staff   480B Feb 14 18:09 openmp-11.0.0.src
-rw-r--r--   1 fangjun  staff   952K Dec  8  2021 openmp-11.0.0.src.tar.xz
drwxr-xr-x   6 fangjun  staff   192B Feb 26 16:44 openmp.xcframework
drwxr-xr-x   6 fangjun  staff   192B Feb 26 16:48 sherpa-ncnn.xcframework
```

Fig. 7.19: Screenshot for selecting `Apple ID` and click `Continue`

What is interesting here is the two framework folders `openmp.xcframework` and `sherpa-ncnn.xcframework`. All other folders can be safely removed. We only need the two framework folders.

In the following, we describe the content in these two framework folders.

openmp.xcframework

```
$ tree build-ios/openmp.xcframework/
build-ios/openmp.xcframework/
├── Headers
│   └── omp.h
├── Info.plist
├── ios-arm64
│   └── libomp.a
└── ios-arm64_x86_64-simulator
    └── libomp.a

3 directories, 4 files
```

Explanation:

- `omp.h`: The header file, which is used by ncnn
- `Info.plist`: A file that is dedicated for framework on macOS/iOS
- `ios-arm64/libopm.a`: A static library for iOS device, e.g., for iPhone

Fig. 7.20: Screenshot for adding your Apple ID and click `Next`

- `ios-arm64_x86_64-simulator/libomp.a`: A static library for iOS simulators, including simulators for Intel chips and Apple Silicon (e.g., M1)

```
sherpa-ncnn fangjun$ file build-ios/openmp.xcframework/ios-arm64_x86_64-simulator/libomp.
↪a
build-ios/openmp.xcframework/ios-arm64_x86_64-simulator/libomp.a: Mach-O universal␣
↪binary with 2 architectures: [x86_64:current ar archive random library] [arm64:current␣
↪ar archive random library]
build-ios/openmp.xcframework/ios-arm64_x86_64-simulator/libomp.a (for architecture x86_
↪64):     current ar archive random library
build-ios/openmp.xcframework/ios-arm64_x86_64-simulator/libomp.a (for architecture␣
↪arm64):      current ar archive random library

sherpa-ncnn fangjun$ lipo -info build-ios/openmp.xcframework/ios-arm64_x86_64-simulator/
↪libomp.a
Architectures in the fat file: build-ios/openmp.xcframework/ios-arm64_x86_64-simulator/
↪libomp.a are: x86_64 arm64

sherpa-ncnn fangjun$ file build-ios/openmp.xcframework/ios-arm64/libomp.a
build-ios/openmp.xcframework/ios-arm64/libomp.a: current ar archive random library

sherpa-ncnn fangjun$ lipo -info build-ios/openmp.xcframework/ios-arm64/libomp.a
Non-fat file: build-ios/openmp.xcframework/ios-arm64/libomp.a is architecture: arm64
```

Fig. 7.21: Screenshot for entering your password and click `Next`

sherpa-ncnn.xcframework

```
sherpa-ncnn fangjun$ tree build-ios/sherpa-ncnn.xcframework/
build-ios/sherpa-ncnn.xcframework/
├── Headers
│   └── sherpa-ncnn
│       └── c-api
│           └── c-api.h
├── Info.plist
├── ios-arm64
│   └── sherpa-ncnn.a
└── ios-arm64_x86_64-simulator
    └── sherpa-ncnn.a

5 directories, 4 files
```

Explanation:

- `c-api.h`: The header file, which is copied from https://github.com/k2-fsa/sherpa-ncnn/blob/master/sherpa-ncnn/c-api/c-api.h
- `Info.plist`: A file that is dedicated for framework on macOS/iOS
- `ios-arm64/sherpa-ncnn.a`: A static library for iOS, e.g., iPhone
- `ios-arm64_x86_64-simulator/sherpa-ncnn.a`: A static library for simulators, including simulators for Intel chips and Apple Silicon (e.g., M1)

Fig. 7.22: Screenshot after adding your Apple ID

Fig. 7.23: Screenshot for selecting your device

Fig. 7.24: Screenshot after selecting your device

Fig. 7.25: Screenshot for selecting `Product -> Run`

Fig. 7.26: Screenshot for running sherpa-ncnn on your device

Fig. 7.27: Screenshot for selecting *Settings* -> *General* -> *Device Management* on your device

Fig. 7.28: Screenshot for "Trust "Apple Development: csukuangfj@g…""

Fig. 7.29: Screenshot for running sherpa-ncnn on iPhone

```
sherpa-ncnn fangjun$ file build-ios/sherpa-ncnn.xcframework/ios-arm64_x86_64-simulator/
↪sherpa-ncnn.a
build-ios/sherpa-ncnn.xcframework/ios-arm64_x86_64-simulator/sherpa-ncnn.a: Mach-O␣
↪universal binary with 2 architectures: [x86_64:current ar archive] [arm64]
build-ios/sherpa-ncnn.xcframework/ios-arm64_x86_64-simulator/sherpa-ncnn.a (for␣
↪architecture x86_64):   current ar archive
build-ios/sherpa-ncnn.xcframework/ios-arm64_x86_64-simulator/sherpa-ncnn.a (for␣
↪architecture arm64):    current ar archive

sherpa-ncnn fangjun$ lipo -info build-ios/sherpa-ncnn.xcframework/ios-arm64_x86_64-
↪simulator/sherpa-ncnn.a
Architectures in the fat file: build-ios/sherpa-ncnn.xcframework/ios-arm64_x86_64-
↪simulator/sherpa-ncnn.a are: x86_64 arm64

sherpa-ncnn fangjun$ file build-ios/sherpa-ncnn.xcframework/ios-arm64/sherpa-ncnn.a
build-ios/sherpa-ncnn.xcframework/ios-arm64/sherpa-ncnn.a: current ar archive

sherpa-ncnn fangjun$ lipo -info build-ios/sherpa-ncnn.xcframework/ios-arm64/sherpa-ncnn.a
Non-fat file: build-ios/sherpa-ncnn.xcframework/ios-arm64/sherpa-ncnn.a is architecture:␣
↪arm64
```

How to use files generated by ./build-ios.sh in Xcode

In this section, we describe how to use `openmp.xcframework` and `sherpa-ncnn.xcframework` in Xcode.

The underlying implementation of sherpa-ncnn is in C++. It also provides C API.

To use `C API` in Xcode with Swift, we have to write a bridging header.

We provide a bridging header for you: SherpaNcnn-Bridging-Header.h. All you need is to add this file to your iOS project and click `Build Settings -> Swift Compiler - General` and set `Objective-C Bridging Header` to `${PROJECT_DIR}/../../swift-api-examples/SherpaNcnn-Bridging-Header.h`. See the screenshot below for reference:

Fig. 7.30: Screenshot for setting the bridging header

We list the content of the bridging header below for reference:

```
#ifndef SWIFT_API_EXAMPLES_SHERPANCNN_BRIDGING_HEADER_H_
#define SWIFT_API_EXAMPLES_SHERPANCNN_BRIDGING_HEADER_H_

#import "sherpa-ncnn/c-api/c-api.h"

#endif  // SWIFT_API_EXAMPLES_SHERPANCNN_BRIDGING_HEADER_H_
```

After adding the bridging header to your iOS project, Xcode will complain it cannot find `sherpa-ncnn/c-api/c-api.h`. The fix is to add the path `build-ios/sherpa-ncnn.xcframework/Headers` to `Header Search Paths` by changing `Build Settings -> Search Paths -> Header Search Paths`, as is shown in the following screenshot:

Fig. 7.31: Screenshot for setting the header search paths

Hint: Instead of using an absolute path, we use `${PROJECT_DIR}/../../build-ios/sherpa-ncnn.xcframework/Headers/`

For instance, my sherpa-ncnn is downloaded to `/Users/fangjun/open-source/sherpa-ncnn` and the path to `sherpa-ncnn.xcframework` is `/Users/fangjun/open-source/sherpa-ncnn/build-ios/sherpa-ncnn.xcframework`.

The value of `PROJECT_DIR` is `/Users/fangjun/open-source/sherpa-ncnn/ios-swift/SherpaNcnn`, so we can use `${PROJECT_DIR}/../../build-ios/sherpa-ncnn.xcframework/Headers/`.

Also note that `PROJECT_DIR` is a pre-defined variable in Xcode.

Please also add SherpaNcnn.swift to your iOS project, which is a utility to make it easier to access functions from the bridging header.

The next thing is to add `openmp.xcframework` and `sherpa-ncnn.xcframework` as dependencies to you iOS project. Select `Build Phases -> Link Binary with Libraries` and then click + to add `sherpa-ncnn.xcframework` and `openmp.xcframework`. See the screenshot below for reference:

Fig. 7.32: Screenshot for adding a framework to your project

Hint: After clicking +, please select `Add Other... -> Add Files..`, and then add the path to `sherpa-ncnn.xcframework`.

Repeat the step for `openmp.xcframework`.

See the screenshot below for reference:

Fig. 7.33: Screenshot for adding a framework

One more thing you need to do after adding the framework is to setup the framework search path. Click `Build Settings -> Search Paths -> Framework Search Paths` and add the path to `build-ios/`. See the screenshot below:

If you encounter link errors about the C++ standard library, please add `-lc++` to link against `libc++` by clicking `Build Settings -> Linking -> Other Linker Flags` and adding `-lc++`. See the screenshot below for reference:

That is all you need to add sherpa-ncnn to your iOS project.

7.9 Pre-trained models

In this section, we describe how to download and use all available pre-trained models.

Hint: Please install git-lfs before you continue.

Fig. 7.34: Screenshot for setting framework search paths

Fig. 7.35: Screenshot for adding `-lc++` to linker flags

7.9.1 Small models

In this section, we list models with fewer parameters that are suitable for resource constrained embedded systems.

Hint: If you are using Raspberry Pi 4, this section is not so helpful for you since all models in sherpa-ncnn are able to run in real-time on it.

This page is especially useful for systems with less resource than Raspberry Pi 4.

- *marcoyang/sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23 (Chinese)*
- *marcoyang/sherpa-ncnn-streaming-zipformer-20M-2023-02-17 (English)*
- *marcoyang/sherpa-ncnn-conv-emformer-transducer-small-2023-01-09 (English)*
- *csukuangfj/sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16 (Bilingual, Chinese + English)*
- *marcoyang/sherpa-ncnn-lstm-transducer-small-2023-02-13 (Bilingual, Chinese + English)*

7.9.2 Zipformer-transducer-based Models

Hint: Please refer to *Installation* to install sherpa-ncnn before you read this section.

marcoyang/sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23 (Chinese)

This model is a streaming Zipformer model which has around 14 millon parameters. It is trained on the WenetSpeech corpus so it supports only Chinese.

You can find the training code at https://github.com/k2-fsa/icefall/tree/master/egs/librispeech/ASR/pruned_transducer_stateless7_streaming

In the following, we describe how to download it and use it with sherpa-ncnn.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-ncnn

wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-
↪streaming-zipformer-zh-14M-2023-02-23.tar.bz2
tar xvf sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23.tar.bz2
```

Decode a single wave file

Hint: It supports decoding only wave files with a single channel and the sampling rate should be 16 kHz.

```
cd /path/to/sherpa-ncnn

for method in greedy_search modified_beam_search; do
  ./build/bin/sherpa-ncnn \
    ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/tokens.txt \
    ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/encoder_jit_trace-pnnx.ncnn.
↪param \
    ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/encoder_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/decoder_jit_trace-pnnx.ncnn.
↪param \
    ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/decoder_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/joiner_jit_trace-pnnx.ncnn.param␣
↪\
    ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/joiner_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/test_wavs/0.wav \
    2 \
    $method
done
```

You should see the following output:

```
Disable fp16 for Zipformer encoder
Don't Use GPU. has_gpu: 0, config.use_vulkan_compute: 1
ModelConfig(encoder_param="pruned_transducer_stateless7_streaming/exp-small-L/encoder_
↪jit_trace-pnnx.ncnn.param", encoder_bin="pruned_transducer_stateless7_streaming/exp-
↪small-L/encoder_jit_trace-pnnx.ncnn.bin", decoder_param="pruned_transducer_stateless7_
↪streaming/exp-small-L/decoder_jit_trace-pnnx.ncnn.param", decoder_bin="pruned_
↪transducer_stateless7_streaming/exp-small-L/decoder_jit_trace-pnnx.ncnn.bin", joiner_
↪param="pruned_transducer_stateless7_streaming/exp-small-L/joiner_jit_trace-pnnx.ncnn.
↪param", joiner_bin="pruned_transducer_stateless7_streaming/exp-small-L/joiner_jit_
↪trace-pnnx.ncnn.bin", tokens="data/lang_char/tokens.txt", encoder num_threads=4,␣
↪decoder num_threads=4, joiner num_threads=4)
DecoderConfig(method="modified_beam_search", num_active_paths=4, enable_endpoint=False,␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)))
wav filename: ./test_wavs_zh/0.wav
wav duration (s): 5.6115
Started!
Done!
Recognition result for ./test_wavs_zh/0.wav

Elapsed seconds: 0.678 s
Real time factor (RTF): 0.678 / 5.611 = 0.121
```

(continues on next page)

(continued from previous page)

```
Disable fp16 for Zipformer encoder
Don't Use GPU. has_gpu: 0, config.use_vulkan_compute: 1
ModelConfig(encoder_param="pruned_transducer_stateless7_streaming/exp-small-L/encoder_
↪jit_trace-pnnx.ncnn.param", encoder_bin="pruned_transducer_stateless7_streaming/exp-
↪small-L/encoder_jit_trace-pnnx.ncnn.bin", decoder_param="pruned_transducer_stateless7_
↪streaming/exp-small-L/decoder_jit_trace-pnnx.ncnn.param", decoder_bin="pruned_
↪transducer_stateless7_streaming/exp-small-L/decoder_jit_trace-pnnx.ncnn.bin", joiner_
↪param="pruned_transducer_stateless7_streaming/exp-small-L/joiner_jit_trace-pnnx.ncnn.
↪param", joiner_bin="pruned_transducer_stateless7_streaming/exp-small-L/joiner_jit_
↪trace-pnnx.ncnn.bin", tokens="data/lang_char/tokens.txt", encoder num_threads=4,␣
↪decoder num_threads=4, joiner num_threads=4)
DecoderConfig(method="modified_beam_search", num_active_paths=4, enable_endpoint=False,␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)))
wav filename: ./test_wavs_zh/1.wav
wav duration (s): 5.15306
Started!
Done!
Recognition result for ./test_wavs_zh/1.wav

Elapsed seconds: 0.676 s
Real time factor (RTF): 0.676 / 5.153 = 0.131

Disable fp16 for Zipformer encoder
Don't Use GPU. has_gpu: 0, config.use_vulkan_compute: 1
ModelConfig(encoder_param="pruned_transducer_stateless7_streaming/exp-small-L/encoder_
↪jit_trace-pnnx.ncnn.param", encoder_bin="pruned_transducer_stateless7_streaming/exp-
↪small-L/encoder_jit_trace-pnnx.ncnn.bin", decoder_param="pruned_transducer_stateless7_
↪streaming/exp-small-L/decoder_jit_trace-pnnx.ncnn.param", decoder_bin="pruned_
↪transducer_stateless7_streaming/exp-small-L/decoder_jit_trace-pnnx.ncnn.bin", joiner_
↪param="pruned_transducer_stateless7_streaming/exp-small-L/joiner_jit_trace-pnnx.ncnn.
↪param", joiner_bin="pruned_transducer_stateless7_streaming/exp-small-L/joiner_jit_
↪trace-pnnx.ncnn.bin", tokens="data/lang_char/tokens.txt", encoder num_threads=4,␣
↪decoder num_threads=4, joiner num_threads=4)
DecoderConfig(method="modified_beam_search", num_active_paths=4, enable_endpoint=False,␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)))
wav filename: ./test_wavs_zh/2.wav
wav duration (s): 4.52431
Started!
Done!
Recognition result for ./test_wavs_zh/2.wav

Elapsed seconds: 0.592 s
Real time factor (RTF): 0.592 / 4.524 = 0.131
```

Note: Please use `./build/bin/Release/sherpa-ncnn.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-ncnn

./build/bin/sherpa-ncnn-microphone \
  ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/tokens.txt \
  ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/decoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/joiner_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/joiner_jit_trace-pnnx.ncnn.bin \
  2 \
  greedy_search
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-ncnn-alsa* to do real-time speech recognition with your microphone if `sherpa-ncnn-microphone` does not work for you.

marcoyang/sherpa-ncnn-streaming-zipformer-20M-2023-02-17 (English)

This model is a streaming Zipformer model converted from

https://huggingface.co/desh2608/icefall-asr-librispeech-pruned-transducer-stateless7-streaming-small

which has around 20 millon parameters. It is trained on the LibriSpeech corpus so it supports only English. The word-error-rates(%) on `test-clean` is 3.88.

You can find the training code at https://github.com/k2-fsa/icefall/tree/master/egs/librispeech/ASR/pruned_transducer_stateless7_streaming

In the following, we describe how to download it and use it with sherpa-ncnn.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-ncnn

wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-
↪streaming-zipformer-20M-2023-02-17.tar.bz2
tar xvf sherpa-ncnn-streaming-zipformer-20M-2023-02-17.tar.bz2
```

Decode a single wave file

Hint: It supports decoding only wave files with a single channel and the sampling rate should be 16 kHz.

```
cd /path/to/sherpa-ncnn

for method in greedy_search modified_beam_search; do
  ./build/bin/sherpa-ncnn \
    ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/tokens.txt \
    ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/encoder_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/encoder_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/decoder_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/decoder_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/joiner_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/joiner_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/test_wavs/0.wav \
    2 \
    $method
done
```

You should see the following output:

```
Disable fp16 for Zipformer encoder
Don't Use GPU. has_gpu: 0, config.use_vulkan_compute: 1
ModelConfig(encoder_param="./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/encoder_jit_
↪trace-pnnx.ncnn.param", encoder_bin="./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/
↪encoder_jit_trace-pnnx.ncnn.bin", decoder_param="./sherpa-ncnn-streaming-zipformer-20M-
↪2023-02-17/decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./sherpa-ncnn-streaming-
↪zipformer-20M-2023-02-17/decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./sherpa-ncnn-
↪streaming-zipformer-20M-2023-02-17/joiner_jit_trace-pnnx.ncnn.param", joiner_bin="./
↪sherpa-ncnn-streaming-zipformer-20M-2023-02-17/joiner_jit_trace-pnnx.ncnn.bin", tokens=
↪"./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/tokens.txt", encoder num_threads=2,␣
↪decoder num_threads=2, joiner num_threads=2)
DecoderConfig(method="greedy_search", num_active_paths=4, enable_endpoint=False,␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)))
wav filename: ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/test_wavs/0.wav
```

(continues on next page)

(continued from previous page)

```
wav duration (s): 6.625
Started!
Done!
Recognition result for ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/test_wavs/0.wav
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BRAFFLELS
Elapsed seconds: 0.472 s
Real time factor (RTF): 0.472 / 6.625 = 0.071
```

Note: Please use `./build/bin/Release/sherpa-ncnn.exe` for Windows.

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-ncnn

./build/bin/sherpa-ncnn-microphone \
  ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/tokens.txt \
  ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/decoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/joiner_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-20M-2023-02-17/joiner_jit_trace-pnnx.ncnn.bin \
  2 \
  greedy_search
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-ncnn-alsa* to do real-time speech recognition with your microphone if `sherpa-ncnn-microphone` does not work for you.

csukuangfj/sherpa-ncnn-streaming-zipformer-en-2023-02-13 (English)

This model is converted from

https://huggingface.co/Zengwei/icefall-asr-librispeech-pruned-transducer-stateless7-streaming-2022-12-29

which supports only English as it is trained on the LibriSpeech corpus.

You can find the training code at

https://github.com/k2-fsa/icefall/tree/master/egs/librispeech/ASR/pruned_transducer_stateless7_streaming

In the following, we describe how to download it and use it with sherpa-ncnn.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-ncnn

wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-
↪streaming-zipformer-en-2023-02-13.tar.bz2
tar xvf sherpa-ncnn-streaming-zipformer-en-2023-02-13.tar.bz2
```

Decode a single wave file

Hint: It supports decoding only wave files with a single channel and the sampling rate should be 16 kHz.

```
cd /path/to/sherpa-ncnn

for method in greedy_search modified_beam_search; do
  ./build/bin/sherpa-ncnn \
    ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/tokens.txt \
    ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/encoder_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/encoder_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/test_wavs/1221-135766-0002.wav \
    2 \
    $method
done
```

You should see the following output:

```
ModelConfig(encoder_param="./sherpa-ncnn-streaming-zipformer-en-2023-02-13/encoder_jit_
↪trace-pnnx.ncnn.param", encoder_bin="./sherpa-ncnn-streaming-zipformer-en-2023-02-13/
↪encoder_jit_trace-pnnx.ncnn.bin", decoder_param="./sherpa-ncnn-streaming-zipformer-en-
↪2023-02-13/decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./sherpa-ncnn-streaming-
↪zipformer-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./sherpa-ncnn-
↪streaming-zipformer-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.param", joiner_bin="./
↪sherpa-ncnn-streaming-zipformer-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.bin", tokens=
↪"./sherpa-ncnn-streaming-zipformer-en-2023-02-13/tokens.txt", encoder num_threads=2,␣
↪decoder num_threads=2, joiner num_threads=2)
DecoderConfig(method="greedy_search", num_active_paths=4, enable_endpoint=False,␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)))
wav filename: ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/test_wavs/1221-135766-0002.
↪wav
wav duration (s): 4.825
```

(continues on next page)

(continued from previous page)

```
Started!
Done!
Recognition result for ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/test_wavs/1221-
↪135766-0002.wav
 YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION
Elapsed seconds: 0.569 s
Real time factor (RTF): 0.569 / 4.825 = 0.118
ModelConfig(encoder_param="./sherpa-ncnn-streaming-zipformer-en-2023-02-13/encoder_jit_
↪trace-pnnx.ncnn.param", encoder_bin="./sherpa-ncnn-streaming-zipformer-en-2023-02-13/
↪encoder_jit_trace-pnnx.ncnn.bin", decoder_param="./sherpa-ncnn-streaming-zipformer-en-
↪2023-02-13/decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./sherpa-ncnn-streaming-
↪zipformer-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./sherpa-ncnn-
↪streaming-zipformer-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.param", joiner_bin="./
↪sherpa-ncnn-streaming-zipformer-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.bin", tokens=
↪"./sherpa-ncnn-streaming-zipformer-en-2023-02-13/tokens.txt", encoder num_threads=2,␣
↪decoder num_threads=2, joiner num_threads=2)
DecoderConfig(method="modified_beam_search", num_active_paths=4, enable_endpoint=False,␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)))
wav filename: ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/test_wavs/1221-135766-0002.
↪wav
wav duration (s): 4.825
Started!
Done!
Recognition result for ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/test_wavs/1221-
↪135766-0002.wav
 YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION
Elapsed seconds: 0.554 s
Real time factor (RTF): 0.554 / 4.825 = 0.115
```

Note: Please use `./build/bin/Release/sherpa-ncnn.exe` for Windows.

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-ncnn

./build/bin/sherpa-ncnn-microphone \
  ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/tokens.txt \
  ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.bin \
  2 \
  greedy_search
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-ncnn-alsa* to do real-time speech recognition with your microphone if `sherpa-ncnn-microphone` does not work for you.

csukuangfj/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13 (Bilingual, Chinese + English)

This model is converted from

https://huggingface.co/pfluo/k2fsa-zipformer-chinese-english-mixed

which supports both Chinese and English. The model is contributed by the community and is trained on tens of thousands of some internal dataset.

In the following, we describe how to download it and use it with sherpa-ncnn.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-ncnn

wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-
↪streaming-zipformer-bilingual-zh-en-2023-02-13.tar.bz2
tar xvf sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13.tar.bz2
```

Decode a single wave file

Hint: It supports decoding only wave files with a single channel and the sampling rate should be 16 kHz.

```
cd /path/to/sherpa-ncnn

for method in greedy_search modified_beam_search; do
  ./build/bin/sherpa-ncnn \
    ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/tokens.txt \
    ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_trace-pnnx.
↪ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_trace-pnnx.
↪ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-pnnx.
↪ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-pnnx.
↪ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-pnnx.
↪ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-pnnx.
↪ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/test_wavs/1.wav \
    2 \
```

(continues on next page)

(continued from previous page)

```
    $method
done
```

You should see the following output:

```
ModelConfig(encoder_param="./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/
↪encoder_jit_trace-pnnx.ncnn.param", encoder_bin="./sherpa-ncnn-streaming-zipformer-
↪bilingual-zh-en-2023-02-13/encoder_jit_trace-pnnx.ncnn.bin", decoder_param="./sherpa-
↪ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.param",
↪ decoder_bin="./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_
↪trace-pnnx.ncnn.bin", joiner_param="./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-
↪2023-02-13/joiner_jit_trace-pnnx.ncnn.param", joiner_bin="./sherpa-ncnn-streaming-
↪zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.bin", tokens="./sherpa-
↪ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/tokens.txt", encoder num_threads=2,
↪ decoder num_threads=2, joiner num_threads=2)
DecoderConfig(method="greedy_search", num_active_paths=4, enable_endpoint=False,␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)))
wav filename: ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/test_wavs/1.
↪wav
wav duration (s): 5.1
Started!
Done!
Recognition result for ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/test_
↪wavs/1.wav
 ALWAYS ALWAYS
Elapsed seconds: 0.598 s
Real time factor (RTF): 0.598 / 5.100 = 0.117
ModelConfig(encoder_param="./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/
↪encoder_jit_trace-pnnx.ncnn.param", encoder_bin="./sherpa-ncnn-streaming-zipformer-
↪bilingual-zh-en-2023-02-13/encoder_jit_trace-pnnx.ncnn.bin", decoder_param="./sherpa-
↪ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-pnnx.ncnn.param",
↪ decoder_bin="./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_
↪trace-pnnx.ncnn.bin", joiner_param="./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-
↪2023-02-13/joiner_jit_trace-pnnx.ncnn.param", joiner_bin="./sherpa-ncnn-streaming-
↪zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-pnnx.ncnn.bin", tokens="./sherpa-
↪ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/tokens.txt", encoder num_threads=2,
↪ decoder num_threads=2, joiner num_threads=2)
DecoderConfig(method="modified_beam_search", num_active_paths=4, enable_endpoint=False,␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)))
wav filename: ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/test_wavs/1.
↪wav
wav duration (s): 5.1
Started!
Done!
```

(continues on next page)

(continued from previous page)

```
Recognition result for ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/test_
↪wavs/1.wav
 ALWAYS ALWAYS
Elapsed seconds: 0.943 s
Real time factor (RTF): 0.943 / 5.100 = 0.185
```

Note: Please use `./build/bin/Release/sherpa-ncnn.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-ncnn

./build/bin/sherpa-ncnn-microphone \
  ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/tokens.txt \
  ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_trace-pnnx.
↪ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/encoder_jit_trace-pnnx.
↪ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-pnnx.
↪ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/decoder_jit_trace-pnnx.
↪ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-pnnx.
↪ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13/joiner_jit_trace-pnnx.
↪ncnn.bin \
  2 \
  greedy_search
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-ncnn-alsa* to do real-time speech recognition with your microphone if `sherpa-ncnn-microphone` does not work for you.

csukuangfj/sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16 (Bilingual, Chinese + English)

This model is converted from

https://huggingface.co/csukuangfj/k2fsa-zipformer-bilingual-zh-en-t

which supports both Chinese and English. The model is contributed by the community and is trained on tens of thousands of some internal dataset.

In the following, we describe how to download it and use it with sherpa-ncnn.

Note: Unlike *csukuangfj/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13 (Bilingual, Chinese + English)*, this model is much smaller.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-ncnn

wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-
↪streaming-zipformer-small-bilingual-zh-en-2023-02-16.tar.bz2
tar xvf sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16.tar.bz2
```

Decode a single wave file

Hint: It supports decoding only wave files with a single channel and the sampling rate should be 16 kHz.

```
cd /path/to/sherpa-ncnn

for method in greedy_search modified_beam_search; do
  ./build/bin/sherpa-ncnn \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/tokens.txt \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/encoder_jit_trace-
↪pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/encoder_jit_trace-
↪pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/decoder_jit_trace-
↪pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/decoder_jit_trace-
↪pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/joiner_jit_trace-
↪pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/joiner_jit_trace-
↪pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/test_wavs/1.wav \
    2 \
```

(continues on next page)

(continued from previous page)

```
    $method
done
```

You should see the following output:

```
ModelConfig(encoder_param="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-
↪02-16/encoder_jit_trace-pnnx.ncnn.param", encoder_bin="./sherpa-ncnn-streaming-
↪zipformer-small-bilingual-zh-en-2023-02-16/encoder_jit_trace-pnnx.ncnn.bin", decoder_
↪param="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/decoder_jit_
↪trace-pnnx.ncnn.param", decoder_bin="./sherpa-ncnn-streaming-zipformer-small-bilingual-
↪zh-en-2023-02-16/decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./sherpa-ncnn-
↪streaming-zipformer-small-bilingual-zh-en-2023-02-16/joiner_jit_trace-pnnx.ncnn.param",
↪ joiner_bin="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/joiner_
↪jit_trace-pnnx.ncnn.bin", tokens="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-
↪en-2023-02-16/tokens.txt", encoder num_threads=2, decoder num_threads=2, joiner num_
↪threads=2)
DecoderConfig(method="greedy_search", num_active_paths=4, enable_endpoint=False,␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)))
wav filename: ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/test_
↪wavs/1.wav
wav duration (s): 5.1
Started!
Done!
Recognition result for ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-
↪16/test_wavs/1.wav
 ALWAYS
ModelConfig(encoder_param="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-
↪02-16/encoder_jit_trace-pnnx.ncnn.param", encoder_bin="./sherpa-ncnn-streaming-
↪zipformer-small-bilingual-zh-en-2023-02-16/encoder_jit_trace-pnnx.ncnn.bin", decoder_
↪param="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/decoder_jit_
↪trace-pnnx.ncnn.param", decoder_bin="./sherpa-ncnn-streaming-zipformer-small-bilingual-
↪zh-en-2023-02-16/decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./sherpa-ncnn-
↪streaming-zipformer-small-bilingual-zh-en-2023-02-16/joiner_jit_trace-pnnx.ncnn.param",
↪ joiner_bin="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/joiner_
↪jit_trace-pnnx.ncnn.bin", tokens="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-
↪en-2023-02-16/tokens.txt", encoder num_threads=2, decoder num_threads=2, joiner num_
↪threads=2)
DecoderConfig(method="modified_beam_search", num_active_paths=4, enable_endpoint=False,␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)))
wav filename: ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/test_
↪wavs/1.wav
wav duration (s): 5.1
Started!
Done!
```

(continues on next page)

(continued from previous page)

```
Recognition result for ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-
↪16/test_wavs/1.wav
 ALWAYS
```

Note: Please use `./build/bin/Release/sherpa-ncnn.exe` for Windows.

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-ncnn

./build/bin/sherpa-ncnn-microphone \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/tokens.txt \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/encoder_jit_trace-
↪pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/encoder_jit_trace-
↪pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/decoder_jit_trace-
↪pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/decoder_jit_trace-
↪pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/joiner_jit_trace-
↪pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/joiner_jit_trace-
↪pnnx.ncnn.bin \
  2 \
  greedy_search
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-ncnn-alsa* to do real-time speech recognition with your microphone if `sherpa-ncnn-microphone` does not work for you.

A faster model of sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16

We provide a second version of the model that is exported with `--decode-chunk-len=96` instead of `32`.

Hint: Please see the model export script at

https://huggingface.co/csukuangfj/sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/blob/main/96/export-for-ncnn-bilingual-small.sh

if you are interested.

Note: You can also find a third version with folder `64`.

The advantage of using this model is that it runs much faster, while the downside is that you will see some delay before you see the recognition result after you speak.

To decode a file, please use:

```
cd /path/to/sherpa-ncnn

for method in greedy_search modified_beam_search; do
  ./build/bin/sherpa-ncnn \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/96/tokens.txt \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/96/encoder_jit_
↪trace-pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/96/encoder_jit_
↪trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/96/decoder_jit_
↪trace-pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/96/decoder_jit_
↪trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/96/joiner_jit_
↪trace-pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/96/joiner_jit_
↪trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/test_wavs/1.wav \
    2 \
    $method
done
```

shaojieli/sherpa-ncnn-streaming-zipformer-fr-2023-04-14

This model is converted from

https://huggingface.co/shaojieli/icefall-asr-commonvoice-fr-pruned-transducer-stateless7-streaming-2023-04-02

which supports only French as it is trained on the CommonVoice corpus. In the following, we describe how to download it and use it with sherpa-ncnn.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-ncnn

wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-
↪streaming-zipformer-fr-2023-04-14.tar.bz2
tar xvf sherpa-ncnn-streaming-zipformer-fr-2023-04-14.tar.bz2
```

To decode a file, please use:

```
cd /path/to/sherpa-ncnn
for method in greedy_search modified_beam_search; do
  ./build/bin/sherpa-ncnn \
    ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/tokens.txt \
    ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/encoder_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/encoder_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/decoder_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/decoder_jit_trace-pnnx.ncnn.bin \
```

(continues on next page)

(continued from previous page)

```
    ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/joiner_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/joiner_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/test_wavs/common_voice_fr_19364697.
↪wav \
    2 \
    $method
done
```

You should see the following output:

```
RecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_dim=80),
↪ model_config=ModelConfig(encoder_param="./sherpa-ncnn-streaming-zipformer-fr-2023-04-
↪14/encoder_jit_trace-pnnx.ncnn.param", encoder_bin="./sherpa-ncnn-streaming-zipformer-
↪fr-2023-04-14/encoder_jit_trace-pnnx.ncnn.bin", decoder_param="./sherpa-ncnn-streaming-
↪zipformer-fr-2023-04-14/decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./sherpa-ncnn-
↪streaming-zipformer-fr-2023-04-14/decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./
↪sherpa-ncnn-streaming-zipformer-fr-2023-04-14/joiner_jit_trace-pnnx.ncnn.param",␣
↪joiner_bin="./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/joiner_jit_trace-pnnx.ncnn.
↪bin", tokens="./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/tokens.txt", encoder num_
↪threads=2, decoder num_threads=2, joiner num_threads=2), decoder_
↪config=DecoderConfig(method="greedy_search", num_active_paths=4), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.4, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=False)
wav filename: ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/test_wavs/common_voice_fr_
↪19364697.wav
wav duration (s): 7.128
Started!
Done!
Recognition result for ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/test_wavs/common_
↪voice_fr_19364697.wav
text:  CE SITE CONTIENT QUATRE TOMBEAUX DE LA DYNASTIE ASHÉMÉNIDE ET SEPT DES SASSANDIDES
timestamps: 0.96 1.44 1.52 1.76 1.96 2.08 2.28 2.56 2.64 2.76 2.8 2.96 3.04 3.2 3.28 3.4␣
↪3.48 3.72 3.8 4 4.16 4.24 4.32 4.44 4.6 4.68 4.92 5.2 5.52 5.84 6.04 6.12 6.24 6.56 6.
↪68
Elapsed seconds: 1.082 s
Real time factor (RTF): 1.082 / 7.128 = 0.152
RecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_dim=80),
↪ model_config=ModelConfig(encoder_param="./sherpa-ncnn-streaming-zipformer-fr-2023-04-
↪14/encoder_jit_trace-pnnx.ncnn.param", encoder_bin="./sherpa-ncnn-streaming-zipformer-
↪fr-2023-04-14/encoder_jit_trace-pnnx.ncnn.bin", decoder_param="./sherpa-ncnn-streaming-
↪zipformer-fr-2023-04-14/decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./sherpa-ncnn-
↪streaming-zipformer-fr-2023-04-14/decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./
↪sherpa-ncnn-streaming-zipformer-fr-2023-04-14/joiner_jit_trace-pnnx.ncnn.param",␣
↪joiner_bin="./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/joiner_jit_trace-pnnx.ncnn.
↪bin", tokens="./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/tokens.txt", encoder num_
↪threads=2, decoder num_threads=2, joiner num_threads=2), decoder_
↪config=DecoderConfig(method="modified_beam_search", num_active_paths=4), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.4, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=False)
```

(continues on next page)

(continued from previous page)

```
wav filename: ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/test_wavs/common_voice_fr_
↪19364697.wav
wav duration (s): 7.128
Started!
Done!
Recognition result for ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/test_wavs/common_
↪voice_fr_19364697.wav
text:  CE SITE CONTIENT QUATRE TOMBEAUX DE LA DYNASTIE ASHÉMÉNIDE ET SEPT DES SASSANDIDES
timestamps: 0.96 1.44 1.52 1.76 1.96 2.08 2.28 2.56 2.64 2.76 2.8 2.96 3.04 3.2 3.28 3.4␣
↪3.48 3.72 3.8 4 4.16 4.24 4.32 4.44 4.6 4.68 4.92 5.2 5.52 5.84 6.04 6.12 6.24 6.56 6.
↪68
Elapsed seconds: 0.812 s
Real time factor (RTF): 0.812 / 7.128 = 0.114
```

Note: Please use `./build/bin/Release/sherpa-ncnn.exe` for Windows.

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-ncnn
./build/bin/sherpa-ncnn-microphone \
  ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/tokens.txt \
  ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/decoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/joiner_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/joiner_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-fr-2023-04-14/test_wavs/common_voice_fr_19364697.wav␣
↪\
  2 \
  greedy_search
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-ncnn-alsa* to do real-time speech recognition with your microphone if `sherpa-ncnn-microphone` does not work for you.

7.9.3 LSTM-transducer-based Models

Hint: Please refer to *Installation* to install sherpa-ncnn before you read this section.

marcoyang/sherpa-ncnn-lstm-transducer-small-2023-02-13 (Bilingual, Chinese + English)

This model is a small version of lstm-transducer trained in icefall.

It only has `13.3 million parameters` and can be deployed on `embedded devices` for real-time speech recognition. You can find the models in `fp16` format at https://huggingface.co/marcoyang/sherpa-ncnn-lstm-transducer-small-2023-02-13.

The model is trained on a bi-lingual dataset `tal_csasr` (Chinese + English), so it can be used for both Chinese and English.

In the following, we show you how to download it and deploy it with sherpa-ncnn.

Please use the following commands to download it.

```
cd /path/to/sherpa-ncnn

wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-lstm-
↪transducer-small-2023-02-13.tar.bz2
tar xvf sherpa-ncnn-lstm-transducer-small-2023-02-13.tar.bz2
```

Note: Please refer to *Embedded Linux (arm)* for how to compile sherpa-ncnn for a 32-bit ARM platform.

Decode a single wave file with ./build/bin/sherpa-ncnn

Hint: It supports decoding only wave files with a single channel and the sampling rate should be 16 kHz.

```
cd /path/to/sherpa-ncnn

./build/bin/sherpa-ncnn \
  ./sherpa-ncnn-lstm-transducer-small-2023-02-13/tokens.txt \
  ./sherpa-ncnn-lstm-transducer-small-2023-02-13/encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-lstm-transducer-small-2023-02-13/encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-lstm-transducer-small-2023-02-13/decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-lstm-transducer-small-2023-02-13/decoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-lstm-transducer-small-2023-02-13/joiner_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-lstm-transducer-small-2023-02-13/joiner_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-lstm-transducer-small-2023-02-13/test_wavs/0.wav
```

Note: The default option uses 4 threads and `greedy_search` for decoding.

Note: Please use `./build/bin/Release/sherpa-ncnn.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

csukuangfj/sherpa-ncnn-2022-09-05 (English)

This is a model trained using the GigaSpeech and the LibriSpeech dataset.

Please see https://github.com/k2-fsa/icefall/pull/558 for how the model is trained.

You can find the training code at

https://github.com/k2-fsa/icefall/tree/master/egs/librispeech/ASR/lstm_transducer_stateless2

In the following, we describe how to download it and use it with sherpa-ncnn.

Please use the following commands to download it.

```
cd /path/to/sherpa-ncnn

wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-2022-09-
↪05.tar.bz2
tar xvf sherpa-ncnn-2022-09-05.tar.bz2
```

Hint: It supports decoding only wave files with a single channel and the sampling rate should be 16 kHz.

```
cd /path/to/sherpa-ncnn

for method in greedy_search modified_beam_search; do
  ./build/bin/sherpa-ncnn \
    ./sherpa-ncnn-2022-09-05/tokens.txt \
    ./sherpa-ncnn-2022-09-05/encoder_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-2022-09-05/encoder_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-2022-09-05/decoder_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-2022-09-05/decoder_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-2022-09-05/joiner_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-2022-09-05/joiner_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-2022-09-05/test_wavs/1089-134686-0001.wav \
    2 \
    $method
done
```

You should see the following output:

```
ModelConfig(encoder_param="./sherpa-ncnn-2022-09-05/encoder_jit_trace-pnnx.ncnn.param",␣
↪encoder_bin="./sherpa-ncnn-2022-09-05/encoder_jit_trace-pnnx.ncnn.bin", decoder_param=
↪"./sherpa-ncnn-2022-09-05/decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./sherpa-
↪ncnn-2022-09-05/decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./sherpa-ncnn-2022-09-
↪05/joiner_jit_trace-pnnx.ncnn.param", joiner_bin="./sherpa-ncnn-2022-09-05/joiner_jit_
↪trace-pnnx.ncnn.bin", tokens="./sherpa-ncnn-2022-09-05/tokens.txt", encoder num_
↪threads=2, decoder num_threads=2, joiner num_threads=2)
DecoderConfig(method="greedy_search", num_active_paths=4, enable_endpoint=False,␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)))
```

(continues on next page)

(continued from previous page)

```
wav filename: ./sherpa-ncnn-2022-09-05/test_wavs/1089-134686-0001.wav
wav duration (s): 6.625
Started!
Done!
Recognition result for ./sherpa-ncnn-2022-09-05/test_wavs/1089-134686-0001.wav
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
ModelConfig(encoder_param="./sherpa-ncnn-2022-09-05/encoder_jit_trace-pnnx.ncnn.param",␣
↪encoder_bin="./sherpa-ncnn-2022-09-05/encoder_jit_trace-pnnx.ncnn.bin", decoder_param=
↪"./sherpa-ncnn-2022-09-05/decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./sherpa-
↪ncnn-2022-09-05/decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./sherpa-ncnn-2022-09-
↪05/joiner_jit_trace-pnnx.ncnn.param", joiner_bin="./sherpa-ncnn-2022-09-05/joiner_jit_
↪trace-pnnx.ncnn.bin", tokens="./sherpa-ncnn-2022-09-05/tokens.txt", encoder num_
↪threads=2, decoder num_threads=2, joiner num_threads=2)
DecoderConfig(method="modified_beam_search", num_active_paths=4, enable_endpoint=False,␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)))
wav filename: ./sherpa-ncnn-2022-09-05/test_wavs/1089-134686-0001.wav
wav duration (s): 6.625
Started!
Done!
Recognition result for ./sherpa-ncnn-2022-09-05/test_wavs/1089-134686-0001.wav
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
```

Note: Please use `./build/bin/Release/sherpa-ncnn.exe` for Windows.

```
cd /path/to/sherpa-ncnn

./build/bin/sherpa-ncnn-microphone \
  ./sherpa-ncnn-2022-09-05/tokens.txt \
  ./sherpa-ncnn-2022-09-05/encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-2022-09-05/encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-2022-09-05/decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-2022-09-05/decoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-2022-09-05/joiner_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-2022-09-05/joiner_jit_trace-pnnx.ncnn.bin \
  2 \
  greedy_search
```

Note: Please use `./build/bin/Release/sherpa-ncnn-microphone.exe` for Windows.

It will print something like below:

```
Number of threads: 4
num devices: 4
```

(continues on next page)

(continued from previous page)

```
Use default device: 2
  Name: MacBook Pro Microphone
  Max input channels: 1
Started
```

Speak and it will show you the recognition result in real-time.

You can find a demo below:

https://youtu.be/m6ynSxycpX0

csukuangfj/sherpa-ncnn-2022-09-30 (Chinese)

This is a model trained using the WenetSpeech dataset.

Please see https://github.com/k2-fsa/icefall/pull/595 for how the model is trained.

In the following, we describe how to download it and use it with sherpa-ncnn.

Please use the following commands to download it.

```
cd /path/to/sherpa-ncnn

wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-2022-09-
↪30.tar.bz2
tar xvf sherpa-ncnn-2022-09-30.tar.bz2
```

Hint: It supports decoding only wave files with a single channel and the sampling rate should be 16 kHz.

```
cd /path/to/sherpa-ncnn

for method in greedy_search modified_beam_search; do
  ./build/bin/sherpa-ncnn \
    ./sherpa-ncnn-2022-09-30/tokens.txt \
    ./sherpa-ncnn-2022-09-30/encoder_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-2022-09-30/encoder_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-2022-09-30/decoder_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-2022-09-30/decoder_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-2022-09-30/joiner_jit_trace-pnnx.ncnn.param \
    ./sherpa-ncnn-2022-09-30/joiner_jit_trace-pnnx.ncnn.bin \
    ./sherpa-ncnn-2022-09-30/test_wavs/0.wav \
    2 \
    $method
done
```

You should see the following output:

```
ModelConfig(encoder_param="./sherpa-ncnn-2022-09-30/encoder_jit_trace-pnnx.ncnn.param",␣
↪encoder_bin="./sherpa-ncnn-2022-09-30/encoder_jit_trace-pnnx.ncnn.bin", decoder_param=
↪"./sherpa-ncnn-2022-09-30/decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./sherpa-
↪ncnn-2022-09-30/decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./sherpa-ncnn-2022-09-
↪30/joiner_jit_trace-pnnx.ncnn.param", joiner_bin="./sherpa-ncnn-2022-09-30/joiner_jit_
↪trace-pnnx.ncnn.bin", tokens="./sherpa-ncnn-2022-09-30/tokens.txt", encoder_num_
↪threads=2, decoder_num_threads=2, joiner_num_threads=2)
```

(continues on next page)

(continued from previous page)

```
DecoderConfig(method="greedy_search", num_active_paths=4, enable_endpoint=False,␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)))
wav filename: ./sherpa-ncnn-2022-09-30/test_wavs/0.wav
wav duration (s): 5.61462
Started!
Done!
Recognition result for ./sherpa-ncnn-2022-09-30/test_wavs/0.wav

ModelConfig(encoder_param="./sherpa-ncnn-2022-09-30/encoder_jit_trace-pnnx.ncnn.param",␣
↪encoder_bin="./sherpa-ncnn-2022-09-30/encoder_jit_trace-pnnx.ncnn.bin", decoder_param=
↪"./sherpa-ncnn-2022-09-30/decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./sherpa-
↪ncnn-2022-09-30/decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./sherpa-ncnn-2022-09-
↪30/joiner_jit_trace-pnnx.ncnn.param", joiner_bin="./sherpa-ncnn-2022-09-30/joiner_jit_
↪trace-pnnx.ncnn.bin", tokens="./sherpa-ncnn-2022-09-30/tokens.txt", encoder num_
↪threads=2, decoder num_threads=2, joiner num_threads=2)
DecoderConfig(method="modified_beam_search", num_active_paths=4, enable_endpoint=False,␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)))
wav filename: ./sherpa-ncnn-2022-09-30/test_wavs/0.wav
wav duration (s): 5.61462
Started!
Done!
Recognition result for ./sherpa-ncnn-2022-09-30/test_wavs/0.wav
```

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

```
cd /path/to/sherpa-ncnn

./build/bin/sherpa-ncnn-microphone \
  ./sherpa-ncnn-2022-09-30/tokens.txt \
  ./sherpa-ncnn-2022-09-30/encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-2022-09-30/encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-2022-09-30/decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-2022-09-30/decoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-2022-09-30/joiner_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-2022-09-30/joiner_jit_trace-pnnx.ncnn.bin \
  2 \
  greedy_search
```

Note: Please use `./build/bin/Release/sherpa-ncnn-microphone.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You can find a demo below:

https://youtu.be/bbQfoRT75oM

7.9.4 Conv-Emformer-transducer-based Models

Hint: Please refer to *Installation* to install sherpa-ncnn before you read this section.

marcoyang/sherpa-ncnn-conv-emformer-transducer-small-2023-01-09 (English)

This model is a small version of conv-emformer-transducer trained in icefall.

It only has `8.8 million parameters` and can be deployed on `embedded devices` for real-time speech recognition. You can find the models in `fp16` and `int8` format at https://huggingface.co/marcoyang/sherpa-ncnn-conv-emformer-transducer-small-2023-01-09.

This model is trained using LibriSpeech and thus it supports only English.

In the following, we show you how to download it and deploy it with sherpa-ncnn on an embedded device, whose CPU is RV1126 (Quad core ARM Cortex-A7)

Please use the following commands to download it.

```
cd /path/to/sherpa-ncnn

wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-conv-
↪emformer-transducer-small-2023-01-09.tar.bz2
tar xvf sherpa-ncnn-conv-emformer-transducer-small-2023-01-09.tar.bz2
```

Note: Please refer to *Embedded Linux (arm)* for how to compile sherpa-ncnn for a 32-bit ARM platform. In the following, we test the pre-trained model on an embedded device, whose CPU is RV1126 (Quad core ARM Cortex-A7).

Decode a single wave file with ./build/bin/sherpa-ncnn

Hint: It supports decoding only wave files with a single channel and the sampling rate should be 16 kHz.

```
cd /path/to/sherpa-ncnn

./build/bin/sherpa-ncnn \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/tokens.txt \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/encoder_jit_trace-pnnx.ncnn.
↪param \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/encoder_jit_trace-pnnx.ncnn.
↪bin \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/decoder_jit_trace-pnnx.ncnn.
↪param \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/decoder_jit_trace-pnnx.ncnn.
↪bin \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/joiner_jit_trace-pnnx.ncnn.
↪param \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/joiner_jit_trace-pnnx.ncnn.bin␣
↪\
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/test_wavs/1089-134686-0001.wav␣
↪\
```

The outputs are shown below. The CPU used for decoding is RV1126 (Quad core ARM Cortex-A7).

```
[root@RV1126_RV1109:/asr/small]# time ./sherpa-ncnn tokens.txt encoder_jit_trace
-pnnx.ncnn.param encoder_jit_trace-pnnx.ncnn.bin decoder_jit_trace-pnnx.ncnn.par
am decoder_jit_trace-pnnx.ncnn.bin joiner_jit_trace-pnnx.ncnn.param joiner_jit_t
race-pnnx.ncnn.bin ./test_wavs/1089-134686-0001.wav 4 greedy_search
Don't Use GPU. has_gpu: 0, config.use_vulkan_compute: 1
ModelConfig(encoder_param="encoder_jit_trace-pnnx.ncnn.param", encoder_bin="encoder_jit_trace-pnnx.ncnn.bin", decoder_param="decoder_jit
_trace-pnnx.ncnn.param", decoder_bin="decoder_jit_trace-pnnx.ncnn.bin", joiner_param="joiner_jit_trace-pnnx.ncnn.param", joiner_bin="joi
ner_jit_trace-pnnx.ncnn.bin", tokens="tokens.txt", encoder num_threads=4, decoder num_threads=4, joiner num_threads=4)
wav filename: ./test_wavs/1089-134686-0001.wav
wav duration (s): 6.625
Recognition result for ./test_wavs/1089-134686-0001.wav
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUARTER OF THE BRAFFLES
real    0m 3.26s
user    0m 10.75s
sys     0m 0.23s
```

Note: The default option uses 4 threads and `greedy_search` for decoding.

Note: Please use `./build/bin/Release/sherpa-ncnn.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

Decode a single wave file with ./build/bin/sherpa-ncnn (with int8 quantization)

Note: We also support int8 quantization to compresss the model and speed up inference. Currently, only encoder and joiner are quantized.

To decode the int8-quantized model, use the following command:

```
cd /path/to/sherpa-ncnn

./build/bin/sherpa-ncnn \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/tokens.txt \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/encoder_jit_trace-pnnx.ncnn.
↪int8.param \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/encoder_jit_trace-pnnx.ncnn.
↪int8.bin \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/decoder_jit_trace-pnnx.ncnn.
↪param \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/decoder_jit_trace-pnnx.ncnn.
↪bin \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/joiner_jit_trace-pnnx.ncnn.
↪int8.param \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/joiner_jit_trace-pnnx.ncnn.
↪int8.bin \
  ./sherpa-ncnn-conv-emformer-transducer-small-2023-01-09/test_wavs/1089-134686-0001.wav␣
↪\
```

The outputs are shown below. The CPU used for decoding is RV1126 (Quad core ARM Cortex-A7).

```
[root@RV1126_RV1109:/asr/small]# time ./sherpa-ncnn tokens.txt encoder_jit_trace
-pnnx.ncnn.int8.param encoder_jit_trace-pnnx.ncnn.int8.bin decoder_jit_trace-pnn
x.ncnn.param decoder_jit_trace-pnnx.ncnn.bin joiner_jit_trace-pnnx.ncnn.int8.par
am joiner_jit_trace-pnnx.ncnn.int8.bin ./test_wavs/1089-134686-0001.wav 4 greedy
_search
Don't Use GPU. has_gpu: 0, config.use_vulkan_compute: 1
ModelConfig(encoder_param="encoder_jit_trace-pnnx.ncnn.int8.param", encoder_bin="encoder_jit_trace-pnnx.ncnn.int8.bin", decoder_param="d
ecoder_jit_trace-pnnx.ncnn.param", decoder_bin="decoder_jit_trace-pnnx.ncnn.bin", joiner_param="joiner_jit_trace-pnnx.ncnn.int8.param",
joiner_bin="joiner_jit_trace-pnnx.ncnn.int8.bin", tokens="tokens.txt", encoder num_threads=4, decoder num_threads=4, joiner num_threads=
4)
wav filename: ./test_wavs/1089-134686-0001.wav
wav duration (s): 6.625
Recognition result for ./test_wavs/1089-134686-0001.wav
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUARTER OF THE BRAFFLES
real    0m 2.44s
user    0m 7.89s
sys     0m 0.23s
```

Compared to the original model in `fp16` format, the decoding speed is significantly improved. The decoding time is changed from `3.26 s` to `2.44 s`.

Note: When the model's weights are quantized to `float16`, it is converted to `float32` during computation.

When the model's weights are quantized to `int8`, it is using `int8` during computation.

Hint: Even if we use only 1 thread for the `int8` model, the resulting real time factor (RTF) is still less than 1.

csukuangfj/sherpa-ncnn-conv-emformer-transducer-2022-12-06 (Chinese + English)

This model is converted from https://huggingface.co/ptrnull/icefall-asr-conv-emformer-transducer-stateless2-zh, which supports both Chinese and English.

Hint: If you want to train your own model that is able to support both Chinese and English, please refer to our training code:

https://github.com/k2-fsa/icefall/tree/master/egs/tal_csasr/ASR

You can also try the pre-trained models in your browser without installing anything by visiting:

https://huggingface.co/spaces/k2-fsa/automatic-speech-recognition

In the following, we describe how to download and use it with sherpa-ncnn.

Please use the following commands to download it.

```
cd /path/to/sherpa-ncnn

wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-conv-
↪emformer-transducer-2022-12-06.tar.bz2
tar xvf sherpa-ncnn-conv-emformer-transducer-2022-12-06.tar.bz2
```

Decode a single wave file with ./build/bin/sherpa-ncnn

Hint: It supports decoding only wave files with a single channel and the sampling rate should be 16 kHz.

```
cd /path/to/sherpa-ncnn

./build/bin/sherpa-ncnn \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/tokens.txt \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/decoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/joiner_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/joiner_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/test_wavs/0.wav \
```

Note: Please use `./build/bin/Release/sherpa-ncnn.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

Real-time speech recognition from a microphone with build/bin/sherpa-ncnn-microphone

```
cd /path/to/sherpa-ncnn
./build/bin/sherpa-ncnn-microphone \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/tokens.txt \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/decoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/joiner_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-06/joiner_jit_trace-pnnx.ncnn.bin
```

Note: Please use `./build/bin/Release/sherpa-ncnn-microphone.exe` for Windows.

It will print something like below:

```
Number of threads: 4
num devices: 4
Use default device: 2
  Name: MacBook Pro Microphone
  Max input channels: 1
Started
```

Speak and it will show you the recognition result in real-time.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

csukuangfj/sherpa-ncnn-conv-emformer-transducer-2022-12-08 (Chinese)

Hint: This is a very small model that can be run in real-time on embedded sytems.

This model is trained using WenetSpeech dataset and it supports only Chinese.

In the following, we describe how to download and use it with sherpa-ncnn.

Please use the following commands to download it.

```
cd /path/to/sherpa-ncnn

wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-conv-
↪emformer-transducer-2022-12-08.tar.bz2
tar xvf sherpa-ncnn-conv-emformer-transducer-2022-12-08.tar.bz2
```

Decode a single wave file with ./build/bin/sherpa-ncnn

Hint: It supports decoding only wave files with a single channel and the sampling rate should be 16 kHz.

```
cd /path/to/sherpa-ncnn

./build/bin/sherpa-ncnn \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/tokens.txt \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/decoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/joiner_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/joiner_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-ncnn.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

Real-time speech recognition from a microphone with build/bin/sherpa-ncnn-microphone

```
cd /path/to/sherpa-ncnn
./build/bin/sherpa-ncnn-microphone \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/tokens.txt \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/decoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/joiner_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-08/joiner_jit_trace-pnnx.ncnn.bin
```

Note: Please use `./build/bin/Release/sherpa-ncnn-microphone.exe` for Windows.

It will print something like below:

```
Number of threads: 4
num devices: 4
Use default device: 2
  Name: MacBook Pro Microphone
  Max input channels: 1
Started
```

Speak and it will show you the recognition result in real-time.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

csukuangfj/sherpa-ncnn-conv-emformer-transducer-2022-12-04 (English)

This model is trained using GigaSpeech and LibriSpeech. It supports only English.

In the following, we describe how to download and use it with sherpa-ncnn.

Please use the following commands to download it.

```
cd /path/to/sherpa-ncnn

wget https://github.com/k2-fsa/sherpa-ncnn/releases/download/models/sherpa-ncnn-conv-
↪emformer-transducer-2022-12-04.tar.bz2
tar xvf sherpa-ncnn-conv-emformer-transducer-2022-12-04.tar.bz2
```

Decode a single wave file with ./build/bin/sherpa-ncnn

Hint: It supports decoding only wave files with a single channel and the sampling rate should be 16 kHz.

```
cd /path/to/sherpa-ncnn

./build/bin/sherpa-ncnn \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/tokens.txt \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/decoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/joiner_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/joiner_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/test_wavs/1089-134686-0001.wav
```

Note: Please use `./build/bin/Release/sherpa-ncnn.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

Real-time speech recognition from a microphone with build/bin/sherpa-ncnn-microphone

```
cd /path/to/sherpa-ncnn
./build/bin/sherpa-ncnn-microphone \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/tokens.txt \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/decoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/joiner_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-conv-emformer-transducer-2022-12-04/joiner_jit_trace-pnnx.ncnn.param
```

Note: Please use `./build/bin/Release/sherpa-ncnn-microphone.exe` for Windows.

It will print something like below:

```
Number of threads: 4
num devices: 4
Use default device: 2
  Name: MacBook Pro Microphone
  Max input channels: 1
Started
```

Speak and it will show you the recognition result in real-time.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

7.10 Examples

In this section, we describe some usage examples of sherpa-ncnn on various boards.

7.10.1 Raspberry Pi 3B E14

This page posts some screenshots of running sherpa-ncnn on Raspberry Pi 3B E14.

Hint: You can find pre-compiled binaries used in this example at

https://huggingface.co/csukuangfj/sherpa-ncnn-pre-compiled-binaries/tree/main/arm32

Board info

OS release

lscpu

```
pi@raspberrypi:~ $ lscpu
Architecture:        armv7l
Byte Order:          Little Endian
CPU(s):              4
On-line CPU(s) list: 0-3
Thread(s) per core:  1
Core(s) per socket:  4
Socket(s):           1
Vendor ID:           ARM
Model:               4
Model name:          Cortex-A53
Stepping:            r0p4
CPU max MHz:         1200.0000
CPU min MHz:         600.0000
BogoMIPS:            57.60
Flags:               half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idi
                     vt vfpd32 lpae evtstrm crc32
```

cpuinfo

```
pi@raspberrypi:~ $ cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev 4 (v7l)
BogoMIPS        : 38.40
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 1
model name      : ARMv7 Processor rev 4 (v7l)
BogoMIPS        : 38.40
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 2
model name      : ARMv7 Processor rev 4 (v7l)
BogoMIPS        : 38.40
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 3
model name      : ARMv7 Processor rev 4 (v7l)
BogoMIPS        : 38.40
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

Hardware        : BCM2835
Revision        : a02082
Serial          : 00000000e3495009
pi@raspberrypi:~ $
```

RTF (1 thread)

RTF (2 threads)

7.10.2 Jetson Nano

This page posts some screenshots of running sherpa-ncnn on Jetson Nano.

Hint: You can find pre-compiled binaries used in this example at

https://huggingface.co/csukuangfj/sherpa-ncnn-pre-compiled-binaries/tree/main/aarch64

Board info

```
nikki@nikki-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23$ lscpu
Architecture:        aarch64
Byte Order:          Little Endian
CPU(s):              4
On-line CPU(s) list: 0-3
Thread(s) per core:  1
Core(s) per socket:  4
Socket(s):           1
Vendor ID:           ARM
Model:               1
Model name:          Cortex-A57
Stepping:            r1p1
CPU max MHz:         1479.0000
CPU min MHz:         102.0000
BogoMIPS:            38.40
L1d cache:           32K
L1i cache:           48K
L2 cache:            2048K
Flags:               fp asimd evtstrm aes pmull sha1 sha2 crc32
nikki@nikki-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.6 LTS
Release:        18.04
Codename:       bionic
nikki@nikki-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23$ uname -a
Linux nikki-desktop 4.9.253-tegra #1 SMP PREEMPT Sat Feb 19 08:59:22 PST 2022 aarch64 aarch64 aarch64 GNU/Linux
nikki@nikki-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23$
```

RTF (4 threads)

```
nikki@nikki-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23$ ./sherpa-ncnn   ./tokens.
txt   ./encoder_jit_trace-pnnx.ncnn.param   ./encoder_jit_trace-pnnx.ncnn.bin   ./decoder_jit_trace-pnnx.ncnn.param   ./
decoder_jit_trace-pnnx.ncnn.bin   ./joiner_jit_trace-pnnx.ncnn.param   ./joiner_jit_trace-pnnx.ncnn.bin   ./test_wavs/0.
wav   4
Disable fp16 for Zipformer encoder
Don't Use GPU. has_gpu: 0, config.use_vulkan_compute: 1
RecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_dim=80, max_feature_vectors=-1), model_
config=ModelConfig(encoder_param="./encoder_jit_trace-pnnx.ncnn.param", encoder_bin="./encoder_jit_trace-pnnx.ncnn.bin",
 decoder_param="./decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./j
oiner_jit_trace-pnnx.ncnn.param", joiner_bin="./joiner_jit_trace-pnnx.ncnn.bin", tokens="./tokens.txt", encoder num_thre
ads=4, decoder num_threads=4, joiner num_threads=4), decoder_config=DecoderConfig(method="greedy_search", num_active_pat
hs=4), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=2.4, min_ut
terance_length=0), rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0), r
ule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_endpoint=Fals
e)
wav filename: ./test_wavs/0.wav
wav duration (s): 5.6115
Started!
Done!
Recognition result for ./test_wavs/0.wav
text: 对我做了介绍那么我想说的是大家如果对我的研究感兴趣
timestamps:
Elapsed seconds: 0.845 s
Real time factor (RTF): 0.845 / 5.611 = 0.151
nikki@nikki-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23$
```

7.10.3 Jetson NX

This page posts some screenshots of running sherpa-ncnn on `Jetson NX`.

Hint: You can find pre-compiled binaries used in this example at

https://huggingface.co/csukuangfj/sherpa-ncnn-pre-compiled-binaries/tree/main/aarch64

Board info

```
e2ai@e2ai-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13$ lscpu
Architecture:        aarch64
Byte Order:          Little Endian
CPU(s):              6
On-line CPU(s) list: 0-5
Thread(s) per core:  1
Core(s) per socket:  2
Socket(s):           3
Vendor ID:           Nvidia
Model:               0
Model name:          ARMv8 Processor rev 0 (v8l)
Stepping:            0x0
CPU max MHz:         1907.2000
CPU min MHz:         115.2000
BogoMIPS:            62.50
L1d cache:           64K
L1i cache:           128K
L2 cache:            2048K
L3 cache:            4096K
Flags:               fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp
e2ai@e2ai-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.6 LTS
Release:        18.04
Codename:       bionic
e2ai@e2ai-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13$ uname -a
Linux e2ai-desktop 4.9.253-tegra #1 SMP PREEMPT Sat Feb 19 08:58:27 PST 2022 aarch64 aarch64 aarch64 GNU/Linux
e2ai@e2ai-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13$
```

RTF (2 threads)

```
e2ai@e2ai-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13$ ./sherpa-ncnn \
>   ./tokens.txt \
>   ./encoder_jit_trace-pnnx.ncnn.param \
>   ./encoder_jit_trace-pnnx.ncnn.bin \
  ./decoder_jit_trace-pnnx.ncnn.param \
  ./decoder_jit_trace-pnnx.ncnn.bin \
  ./joiner_jit_trace-pnnx.ncnn.param \
  ./joiner_jit_trace-pnnx.ncnn.bin \
  ./test_wavs/0.wav \
  2>   ./decoder_jit_trace-pnnx.ncnn.param \
>   ./decoder_jit_trace-pnnx.ncnn.bin \
>   ./joiner_jit_trace-pnnx.ncnn.param \
>   ./joiner_jit_trace-pnnx.ncnn.bin \
>   ./test_wavs/0.wav \
>   2
Disable fp16 for Zipformer encoder
Don't Use GPU. has_gpu: 0, config.use_vulkan_compute: 1
RecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_dim=80, max_feature_vectors=-1), model_
config=ModelConfig(encoder_param="./encoder_jit_trace-pnnx.ncnn.param", encoder_bin="./encoder_jit_trace-pnnx.ncnn.bin",
 decoder_param="./decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./j
oiner_jit_trace-pnnx.ncnn.param", joiner_bin="./joiner_jit_trace-pnnx.ncnn.bin", tokens="./tokens.txt", encoder num_thre
ads=2, decoder num_threads=2, joiner num_threads=2), decoder_config=DecoderConfig(method="greedy_search", num_active_pat
hs=4), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=2.4, min_ut
terance_length=0), rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0), r
ule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_endpoint=Fals
e)
wav filename: ./test_wavs/0.wav
wav duration (s): 10.0531
Started!
Done!
Recognition result for ./test_wavs/0.wav
text: 昨天是 MONDAY TODAY IS LIBR THE DAY AFTER TOMORROW是星期三
timestamps:
Elapsed seconds: 3.661 s
Real time factor (RTF): 3.661 / 10.053 = 0.364
e2ai@e2ai-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13$
```

RTF (4 threads)

```
e2ai@e2ai-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13$ ./sherpa-ncnn   ./
tokens.txt   ./encoder_jit_trace-pnnx.ncnn.param   ./encoder_jit_trace-pnnx.ncnn.bin   ./decoder_jit_trace-pnnx.ncnn.par
am   ./decoder_jit_trace-pnnx.ncnn.bin   ./joiner_jit_trace-pnnx.ncnn.param   ./joiner_jit_trace-pnnx.ncnn.bin   ./test_
wavs/0.wav 4
Disable fp16 for Zipformer encoder
Don't Use GPU. has_gpu: 0, config.use_vulkan_compute: 1
RecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_dim=80, max_feature_vectors=-1), model_
config=ModelConfig(encoder_param="./encoder_jit_trace-pnnx.ncnn.param", encoder_bin="./encoder_jit_trace-pnnx.ncnn.bin",
 decoder_param="./decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./j
oiner_jit_trace-pnnx.ncnn.param", joiner_bin="./joiner_jit_trace-pnnx.ncnn.bin", tokens="./tokens.txt", encoder num_thre
ads=4, decoder num_threads=4, joiner num_threads=4), decoder_config=DecoderConfig(method="greedy_search", num_active_pat
hs=4), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=2.4, min_ut
terance_length=0), rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0), r
ule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_endpoint=Fals
e)
wav filename: ./test_wavs/0.wav
wav duration (s): 10.0531
Started!
Done!
Recognition result for ./test_wavs/0.wav
text: 昨天是 MONDAY TODAY IS LIBR THE DAY AFTER TOMORROW是星期三
timestamps:
Elapsed seconds: 3.163 s
Real time factor (RTF): 3.163 / 10.053 = 0.315
e2ai@e2ai-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13$
```

RTF (6 threads)

```
e2ai@e2ai-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13$ ./sherpa-ncnn   ./
tokens.txt   ./encoder_jit_trace-pnnx.ncnn.param   ./encoder_jit_trace-pnnx.ncnn.bin   ./decoder_jit_trace-pnnx.ncnn.par
am   ./decoder_jit_trace-pnnx.ncnn.bin   ./joiner_jit_trace-pnnx.ncnn.param   ./joiner_jit_trace-pnnx.ncnn.bin   ./test_
wavs/0.wav 6
Disable fp16 for Zipformer encoder
Don't Use GPU. has_gpu: 0, config.use_vulkan_compute: 1
RecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_dim=80, max_feature_vectors=-1), model_
config=ModelConfig(encoder_param="./encoder_jit_trace-pnnx.ncnn.param", encoder_bin="./encoder_jit_trace-pnnx.ncnn.bin",
 decoder_param="./decoder_jit_trace-pnnx.ncnn.param", decoder_bin="./decoder_jit_trace-pnnx.ncnn.bin", joiner_param="./j
oiner_jit_trace-pnnx.ncnn.param", joiner_bin="./joiner_jit_trace-pnnx.ncnn.bin", tokens="./tokens.txt", encoder num_thre
ads=6, decoder num_threads=6, joiner num_threads=6), decoder_config=DecoderConfig(method="greedy_search", num_active_pat
hs=4), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=2.4, min_ut
terance_length=0), rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0), r
ule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_endpoint=Fals
e)
wav filename: ./test_wavs/0.wav
wav duration (s): 10.0531
Started!
Done!
Recognition result for ./test_wavs/0.wav
text: 昨天是 MONDAY TODAY IS LIBR THE DAY AFTER TOMORROW是星期三
timestamps:
Elapsed seconds: 3.854 s
Real time factor (RTF): 3.854 / 10.053 = 0.383
e2ai@e2ai-desktop:~/Downloads/sherpa-ncnn/sherpa-ncnn-streaming-zipformer-bilingual-zh-en-2023-02-13$
```

7.10.4 VisionFive 2

This page describes how to run sherpa-ncnn on VisionFive2, which is a 64-bit RISC-V board with 4 CPUs.

Hint: You can find pre-compiled binaries used in this example at

https://huggingface.co/csukuangfj/sherpa-ncnn-pre-compiled-binaries/tree/main/riscv64

Caution: The latest debian image from https://doc-en.rvspace.org/VisionFive2/Quick_Start_Guide/VisionFive2_QSG/flashing_with_mac_linux.html does not work since it does not support USB devices.

That is, you cannot use USB microphones on the board with the above debian image.

Note: We have compiled https://github.com/starfive-tech/VisionFive2 and the resulting `sdcard.img` is available at https://huggingface.co/csukuangfj/visionfive2-sd-card-img.

Please use this image for testing. It supports USB microphones.

The username for this image is `root` and the password is `starfive`.

Board info

```
root@buildroot:~# lscpu
Architecture:          riscv64
  Byte Order:          Little Endian
CPU(s):                4
  On-line CPU(s) list: 0-3
root@buildroot:~# cat /proc/cpuinfo
processor       : 0
hart            : 1
isa             : rv64imafdc
mmu             : sv39
isa-ext         :
uarch           : sifive,u74-mc

processor       : 1
hart            : 2
isa             : rv64imafdc
mmu             : sv39
isa-ext         :
uarch           : sifive,u74-mc

processor       : 2
hart            : 3
isa             : rv64imafdc
mmu             : sv39
isa-ext         :
uarch           : sifive,u74-mc

processor       : 3
hart            : 4
isa             : rv64imafdc
mmu             : sv39
isa-ext         :
uarch           : sifive,u74-mc
```

RTF (4 threads)

We use *csukuangfj/sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16 (Bilingual, Chinese + English)* for testing. The RTF is given below:

```
root@buildroot:next-gen-kaldi# cat run-sherpa-ncnn.sh
#!/usr/bin/env bash

./sherpa-ncnn \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/tokens.txt \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/encoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/encoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/decoder_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/decoder_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/joiner_jit_trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/joiner_jit_trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/test_wavs/0.wav \
  4 \
  greedy_search

root@buildroot:next-gen-kaldi# ./run-sherpa-ncnn.sh
Don't Use GPU. has_gpu: 0, config.use_vulkan_compute: 1
RecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_dim=80), model_config=ModelConfig(encoder_param="./sherpa-ncnn-strea
ming-zipformer-small-bilingual-zh-en-2023-02-16/64/encoder_jit_trace-pnnx.ncnn.param", encoder_bin="./sherpa-ncnn-streaming-zipformer-small-bilingual
-zh-en-2023-02-16/64/encoder_jit_trace-pnnx.ncnn.bin", decoder_param="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/decoder_j
it_trace-pnnx.ncnn.param", decoder_bin="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/decoder_jit_trace-pnnx.ncnn.bin", joine
r_param="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/joiner_jit_trace-pnnx.ncnn.param", joiner_bin="./sherpa-ncnn-streaming
-zipformer-small-bilingual-zh-en-2023-02-16/64/joiner_jit_trace-pnnx.ncnn.bin", tokens="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-
02-16/tokens.txt", encoder num_threads=4, decoder num_threads=4, joiner num_threads=4), decoder_config=DecoderConfig(method="greedy_search", num_acti
ve_paths=4), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0), rule
2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.4, min_utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min
_trailing_silence=0, min_utterance_length=20)), enable_endpoint=False)
wav filename: ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/test_wavs/0.wav
wav duration (s): 10.0531
Started!
Done!
Recognition result for ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/test_wavs/0.wav
text: 昨天是 MONDAY TODAY IS李八二 TODAY AFTER TOMORROW是星期三
timestamps: 0.64 1.08 1.6 2.08 2.24 2.36 4.04 4.28 5.12 5.48 5.68 6.12 6.8 6.96 7.28 7.92 8.04 8.2 8.28 9.04 9.36 9.6 10.04
Elapsed seconds: 9.334 s
Real time factor (RTF): 9.334 / 10.053 = 0.928
```

You can see that the RTF is less than 1, which means it is able to perform streaming (i.e., real-time) speech recognition.

The following posts the commands used for testing so that you can copy and paste them if you want to test it by yourself.

```
./sherpa-ncnn \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/tokens.txt \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/encoder_jit_
↪trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/encoder_jit_
↪trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/decoder_jit_
↪trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/decoder_jit_
↪trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/joiner_jit_trace-
↪pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/joiner_jit_trace-
↪pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/test_wavs/0.wav \
  4 \
  greedy_search
```

Real-time speech recognition with a microphone

Since the board does not have microphones, we use a USB microphone for testing.

> **Caution:** We use the image from https://huggingface.co/csukuangfj/visionfive2-sd-card-img/tree/main, which provides support for USB microphones.

After connecting a USB microphone to the board, use the following command to check it:

```
root@buildroot:next-gen-kaldi# arecord -l
**** List of CAPTURE Hardware Devices ****
card 2: UACDemoV10 [UACDemoV1.0], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

The output shows `Card 2` and `device 0`, so the device name is `hw:2,0`.

The command to start the program for real-time speech recognition is

```
./sherpa-ncnn-alsa \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/tokens.txt \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/encoder_jit_
↪trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/encoder_jit_
↪trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/decoder_jit_
↪trace-pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/decoder_jit_
↪trace-pnnx.ncnn.bin \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/joiner_jit_trace-
↪pnnx.ncnn.param \
  ./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/joiner_jit_trace-
↪pnnx.ncnn.bin \
  hw:2,0 \
  4 \
  greedy_search
```

A screenshot is given below:

```
root@buildroot:next-gen-kaldi# ./run-sherpa-ncnn-alsa.sh
./sherpa-ncnn-alsa: /usr/lib/libasound.so.2: no version information available (required by ./sherpa-ncnn-alsa)
./sherpa-ncnn-alsa: /usr/lib/libasound.so.2: no version information available (required by ./sherpa-ncnn-alsa)
RecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_dim=80), model_config=ModelConfig(encoder_param="./sherpa-ncnn-strea
ming-zipformer-small-bilingual-zh-en-2023-02-16/64/encoder_jit_trace-pnnx.ncnn.param", encoder_bin="./sherpa-ncnn-streaming-zipformer-small-bilingual
-zh-en-2023-02-16/64/encoder_jit_trace-pnnx.ncnn.bin", decoder_param="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/decoder_j
it_trace-pnnx.ncnn.param", decoder_bin="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/decoder_jit_trace-pnnx.ncnn.bin", joine
r_param="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-02-16/64/joiner_jit_trace-pnnx.ncnn.param", joiner_bin="./sherpa-ncnn-streaming
-zipformer-small-bilingual-zh-en-2023-02-16/64/joiner_jit_trace-pnnx.ncnn.bin", tokens="./sherpa-ncnn-streaming-zipformer-small-bilingual-zh-en-2023-
02-16/tokens.txt", encoder num_threads=4, decoder num_threads=4, joiner num_threads=4), decoder_config=DecoderConfig(method="greedy_search", num_acti
ve_paths=4), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0), rule
2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min
_trailing_silence=0, min_utterance_length=300)), enable_endpoint=True)
Don't Use GPU. has_gpu: 0, config.use_vulkan_compute: 1
Failed to set sample rate to 16000
Current sample rate is 48000
Creating a resampler:
   in_sample_rate: 48000
   output_sample_rate: 16000
Recording started!
Use recording device: hw:2,0
0:星光二上的语音识别
1:这是一块基于 risk five
2:的开发版
```

7.11 FAQs

7.11.1 Where to get help

If you have any questions, please create an issue at https://github.com/k2-fsa/sherpa-ncnn

We also have active social groups:

- : Kaldi
- Kaldi, ,
- QQ 744602236

7.11.2 No default input device found

If you are using Linux and if `sherpa-ncnn-microphone` throws the following error:

```
Num device: 0
No default input device found.
```

Please consider using `sherpa-ncnn-alsa` to replace `sherpa-ncnn-microphone`. If you cannot find `sherpa-ncnn-alsa` in `./build/bin`, please run the following commands:

```
cd /path/to/sherpa-ncnn
sudo apt-get install alsa-utils libasound2-dev
cd build
rm CMakeCache.txt # Important, remove the cmake cache file
make -j
```

After the above commands, you should see a binary file `./build/bin/sherpa-ncnn-alsa`.

Please follow *sherpa-ncnn-alsa* to use `sherpa-ncnn-alsa`.

CHAPTER

EIGHT

SHERPA-ONNX

Hint: During speech recognition, it does not need to access the Internet. Everyting is processed locally on your device.

We support using onnx with onnxruntime to replace PyTorch for neural network computation. The code is put in a separate repository sherpa-onnx.

sherpa-onnx is self-contained and everything can be compiled from source.

Please refer to https://k2-fsa.github.io/icefall/model-export/export-onnx.html for how to export models to onnx format.

In the following, we describe how to build sherpa-onnx for Linux, macOS, Windows, embedded systems, Android, and iOS.

Also, we show how to use it for speech recognition with pre-trained models.

8.1 Tutorials

This page contains links to tutorials written by our users.

Caution: The tutorials are not necessarily written in English.

8.2 Installation

In this section, we describe how to install sherpa-onnx on various platforms.

Requirements:

- `CMake >= 3.13`
- A compiler that supports at least `C++14`

Hint: You can use `pip install cmake` to install the latest cmake.

8.2.1 Linux

This page describes how to build sherpa-onnx on Linux.

All you need is to run:

CPU

Nvidia GPU (CUDA)

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make -j6
```

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release -DBUILD_SHARED_LIBS=ON -DSHERPA_ONNX_ENABLE_GPU=ON ..
make -j6
```

Hint: You need to install CUDA toolkit. Otherwise, you would get errors at runtime.

You can refer to https://k2-fsa.github.io/k2/installation/cuda-cudnn.html to install CUDA toolkit.

After building, you will find an executable `sherpa-onnx` inside the `bin` directory.

That's it!

Please refer to *Pre-trained models* for a list of pre-trained models.

8.2.2 macOS

This page describes how to build sherpa-onnx on macOS.

Hint: It supports both Intel and Apple Silicon (e.g., M1).

All you need is to run:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make -j6
```

After building, you will find an executable `sherpa-onnx` inside the `bin` directory.

That's it!

Please refer to *Pre-trained models* for a list of pre-trained models.

8.2.3 Windows

This page describes how to build sherpa-onnx on Windows.

Hint: MinGW is known not to work. Please install `Visual Studio` before you continue.

Note: You can download pre-compiled binaries for both 32-bit and 64-bit Windows from the following URL https://huggingface.co/csukuangfj/sherpa-onnx-libs/tree/main.

Please always download the latest version.

URLs to download the version `1.9.12` is given below.

64-bit Windows (static lib)	https://huggingface.co/csukuangfj/sherpa-onnx-libs/resolve/main/win64/sherpa-onnx-v1.9.12-win-x64-static.tar.bz2
64-bit Windows (shared lib)	https://huggingface.co/csukuangfj/sherpa-onnx-libs/resolve/main/win64/sherpa-onnx-v1.9.12-win-x64-shared.tar.bz2
32-bit Windows (static lib)	https://huggingface.co/csukuangfj/sherpa-onnx-libs/resolve/main/win32/sherpa-onnx-v1.9.12-win-x86-static.tar.bz2
32-bit Windows (shared lib)	https://huggingface.co/csukuangfj/sherpa-onnx-libs/resolve/main/win32/sherpa-onnx-v1.9.12-win-x86-shared.tar.bz2

If you cannot access `huggingface.co`, then please replace `huggingface.co` with `hf-mirror.com`.

64-bit Windows (x64)

All you need is to run:

CPU

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
cmake --build . --config Release
```

Nvidia GPU (CUDA)

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release -DBUILD_SHARED_LIBS=ON -DSHERPA_ONNX_ENABLE_GPU=ON ..
cmake --build . --config Release
```

Hint: You need to install CUDA toolkit. Otherwise, you would get errors at runtime.

After building, you will find an executable `sherpa-onnx.exe` inside the `bin/Release` directory.

That's it!

Please refer to *Pre-trained models* for a list of pre-trained models.

32-bit Windows (x86)

Hint: It does not support NVIDIA GPU for `Win32/x86`.

All you need is to run:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
mkdir build
cd build

# Please select one toolset among VS 2015, 2017, 2019, and 2022 below
# We use VS 2022 as an example.

# For Visual Studio 2015
# cmake -T v140,host=x64 -A Win32 -D CMAKE_BUILD_TYPE=Release ..

# For Visual Studio 2017
# cmake -T v141,host=x64 -A Win32 -D CMAKE_BUILD_TYPE=Release ..

# For Visual Studio 2019
# cmake -T v142,host=x64 -A Win32 -D CMAKE_BUILD_TYPE=Release ..

# For Visual Studio 2022
cmake -T v143,host=x64 -A Win32 -D CMAKE_BUILD_TYPE=Release ..

cmake --build . --config Release
```

After building, you will find an executable `sherpa-onnx.exe` inside the `bin/Release` directory.

That's it!

Please refer to *Pre-trained models* for a list of pre-trained models.

Hint: By default, it builds static libraries of sherpa-onnx. To get dynamic/shared libraries, please pass `-DBUILD_SHARED_LIBS=ON` to `cmake`. That is, use

```
cmake -DCMAKE_BUILD_TYPE=Release -DBUILD_SHARED_LIBS=ON ..
```

8.2.4 Embedded Linux (aarch64)

This page describes how to build sherpa-onnx for embedded Linux (aarch64, 64-bit) with cross-compiling on an x64 machine with Ubuntu OS.

Warning: By cross-compiling we mean that you do the compilation on a `x86_64` machine. And you copy the generated binaries from a `x86_64` machine and run them on an `aarch64` machine.

If you want to compile sherpa-onnx on an `aarch64` machine directly, please see *Linux*.

Note: You can download pre-compiled binaries for `aarch64` from the following URL https://huggingface.co/csukuangfj/sherpa-onnx-libs/tree/main/aarch64

Please always download the latest version.

Example command to download the version `1.9.12`:

```
# binaries built with shared libraries
wget https://huggingface.co/csukuangfj/sherpa-onnx-libs/resolve/main/aarch64/
↪sherpa-onnx-v1.9.12-linux-aarch64-shared.tar.bz2

# binaries built with static link
wget https://huggingface.co/csukuangfj/sherpa-onnx-libs/resolve/main/aarch64/
↪sherpa-onnx-v1.9.12-linux-aarch64-static.tar.bz2

# For users from China
#  huggingface,

# binaries built with shared libraries
wget https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/resolve/main/aarch64/
↪sherpa-onnx-v1.9.12-linux-aarch64-shared.tar.bz2

# binaries built with static link
wget https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/resolve/main/aarch64/
↪sherpa-onnx-v1.9.12-linux-aarch64-static.tar.bz2
```

Hint: We provide two colab notebooks for you to try this section step by step.

Build with `shared` libraries	Build with `static` libraries

If you are using Windows/macOS or you don't want to setup your local environment for cross-compiling, please use the above colab notebooks.

Install toolchain

The first step is to install a toolchain for cross-compiling.

> **Warning:** You can use any toolchain that is suitable for your platform. The toolchain we use below is just an example.

Visit https://releases.linaro.org/components/toolchain/binaries/latest-7/aarch64-linux-gnu/

We are going to download `gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu.tar.xz`, which has been uploaded to https://huggingface.co/csukuangfj/sherpa-ncnn-toolchains.

Assume you want to install it in the folder `$HOME/software`:

```
mkdir -p $HOME/software
cd $HOME/software
wget https://huggingface.co/csukuangfj/sherpa-ncnn-toolchains/resolve/main/gcc-linaro-7.
↪5.0-2019.12-x86_64_aarch64-linux-gnu.tar.xz

# For users from China
#  huggingface,
# wget https://hf-mirror.com/csukuangfj/sherpa-ncnn-toolchains/resolve/main/gcc-linaro-7.
↪5.0-2019.12-x86_64_aarch64-linux-gnu.tar.xz

tar xvf gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu.tar.xz
```

Next, we need to set the following environment variable:

```
export PATH=$HOME/software/gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu/bin:$PATH
```

To check that we have installed the cross-compiling toolchain successfully, please run:

```
aarch64-linux-gnu-gcc --version
```

which should print the following log:

```
aarch64-linux-gnu-gcc (Linaro GCC 7.5-2019.12) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Congratulations! You have successfully installed a toolchain for cross-compiling sherpa-onnx.

Build sherpa-onnx

Finally, let us build sherpa-onnx.

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
export BUILD_SHARED_LIBS=ON
./build-aarch64-linux-gnu.sh
```

After building, you will get two binaries:

```
sherpa-onnx$ ls -lh build-aarch64-linux-gnu/install/bin/
total 378K
-rwxr-xr-x 1 kuangfangjun root 187K Feb 21 21:55 sherpa-onnx
-rwxr-xr-x 1 kuangfangjun root 191K Feb 21 21:55 sherpa-onnx-alsa
```

Note: Please also copy the `onnxruntime` lib to your embedded systems and put it into the same directory as `sherpa-onnx` and `sherpa-onnx-alsa`.

```
sherpa-onnx$ ls -lh build-aarch64-linux-gnu/install/lib/*onnxruntime*
lrw-r--r-- 1 kuangfangjun root  24 Feb 21 21:38 build-aarch64-linux-gnu/install/lib/
↪libonnxruntime.so -> libonnxruntime.so.1.14.0
-rw-r--r-- 1 kuangfangjun root 15M Feb 21 21:38 build-aarch64-linux-gnu/install/lib/
↪libonnxruntime.so.1.14.0
```

That's it!

Hint:

- `sherpa-onnx` is for decoding a single file
- `sherpa-onnx-alsa` is for real-time speech recongition by reading the microphone with ALSA

sherpa-onnx-alsa

Caution: We recommend that you use `sherpa-onnx-alsa` on embedded systems such as Raspberry pi.

You need to provide a `device_name` when invoking `sherpa-onnx-alsa`. We describe below how to find the device name for your microphone.

Run the following command:

```
arecord -l
```

to list all avaliable microphones for recording. If it complains that `arecord: command not found`, please use `sudo apt-get install alsa-utils` to install it.

If the above command gives the following output:

```
**** List of CAPTURE Hardware Devices ****
card 3: UACDemoV10 [UACDemoV1.0], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

In this case, I only have 1 microphone. It is `card 3` and that card has only `device 0`. To select `card 3` and `device 0` on that card, we need to pass `plughw:3,0` to `sherpa-onnx-alsa`. (Note: It has the format `plughw:card_number,device_index`.)

For instance, you have to use

```
./sherpa-onnx-alsa \
  --encoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/
↪encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/
↪decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/
↪joiner-epoch-99-avg-1.onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/
↪tokens.txt \
  plughw:3,0
```

Please change the card number and also the device index on the selected card accordingly in your own situation. Otherwise, you won't be able to record with your microphone.

Please read *Pre-trained models* for usages about the generated binaries.

Hint: If you want to select a pre-trained model for Raspberry that can be run on real-time, we recommend you to use *Zipformer-transducer-based Models*.

Please create an issue at https://github.com/k2-fsa/sherpa-onnx/issues if you have any problems.

How to build static libraries and static linked binaries

If you want to build static libraries and static linked binaries, please first download a cross compile toolchain with GCC >= 9.0. The following is an example:

```
mkdir -p $HOME/software
cd $HOME/software
wget -q https://huggingface.co/csukuangfj/sherpa-ncnn-toolchains/resolve/main/gcc-arm-10.
↪3-2021.07-x86_64-aarch64-none-linux-gnu.tar.xz

# For users from China
#  huggingface,
# wget -q https://hf-mirror.com/csukuangfj/sherpa-ncnn-toolchains/resolve/main/gcc-arm-
↪10.3-2021.07-x86_64-aarch64-none-linux-gnu.tar.xz

tar xf gcc-arm-10.3-2021.07-x86_64-aarch64-none-linux-gnu.tar.xz
```

Next, we need to set the following environment variable:

```
export PATH=$HOME/software/gcc-arm-10.3-2021.07-x86_64-aarch64-none-linux-gnu/bin:$PATH
```

To check that we have installed the cross-compiling toolchain successfully, please run:

```
aarch64-none-linux-gnu-gcc --version
```

which should print the following log:

```
aarch64-none-linux-gnu-gcc (GNU Toolchain for the A-profile Architecture 10.3-2021.07␣
↪(arm-10.29)) 10.3.1 20210621
Copyright (C) 2020 Free Software Foundation, Inc.
```

(continues on next page)

(continued from previous page)

```
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Now you can build static libraries and static linked binaries with the following commands:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
export BUILD_SHARED_LIBS=OFF
./build-aarch64-linux-gnu.sh
```

You can use the following commands to check that the generated binaries are indeed static linked:

```
$ cd build-aarch64-linux-gnu/bin

$ ldd sherpa-onnx-alsa
    not a dynamic executable

$ readelf -d sherpa-onnx-alsa

Dynamic section at offset 0xed9950 contains 30 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libasound.so.2]
 0x0000000000000001 (NEEDED)             Shared library: [libdl.so.2]
 0x0000000000000001 (NEEDED)             Shared library: [libm.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libpthread.so.0]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
 0x000000000000000f (RPATH)              Library rpath: [$ORIGIN:/star-fj/fangjun/open-
↪source/sherpa-onnx/build-aarch64-linux-gnu/_deps/onnxruntime-sr
c/lib:]
 0x000000000000000c (INIT)               0x404218
```

8.2.5 Embedded Linux (arm)

This page describes how to build sherpa-onnx for embedded Linux (arm, 32-bit) with `cross-compiling` on an x86 machine with Ubuntu OS.

> **Caution:** If you want to build sherpa-onnx directly on your board, please don't use this document. Refer to *Linux* instead.

> **Caution:** If you want to build sherpa-onnx directly on your board, please don't use this document. Refer to *Linux* instead.

> **Caution:** If you want to build sherpa-onnx directly on your board, please don't use this document. Refer to *Linux* instead.

Hint: This page is for cross-compiling.

Note: You can download pre-compiled binaries for 32-bit `ARM` from the following URL https://huggingface.co/csukuangfj/sherpa-onnx-libs/tree/main/arm32

Please always download the latest version.

Example command to download the version `1.9.12`:

```
# binaries built with shared libraries
wget https://huggingface.co/csukuangfj/sherpa-onnx-libs/resolve/main/arm32/
↪sherpa-onnx-v1.9.12-linux-arm-gnueabihf-shared.tar.bz2

# binaries built with static link
wget https://huggingface.co/csukuangfj/sherpa-onnx-libs/resolve/main/arm32/
↪sherpa-onnx-v1.9.12-linux-arm-gnueabihf-static.tar.bz2

# For users from China
#  huggingface,

# binaries built with shared libraries
wget https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/resolve/main/arm32/
↪sherpa-onnx-v1.9.12-linux-arm-gnueabihf-shared.tar.bz2

# binaries built with static link
wget https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/resolve/main/arm32/
↪sherpa-onnx-v1.9.12-linux-arm-gnueabihf-static.tar.bz2
```

Hint: We provide two colab notebooks for you to try this section step by step.

Build with `shared` libraries	Build with `static` libraries

If you are using Windows/macOS or you don't want to setup your local environment for cross-compiling, please use the above colab notebooks.

Install toolchain

The first step is to install a toolchain for cross-compiling.

Warning: You can use any toolchain that is suitable for your platform. The toolchain we use below is just an example.

Visit https://developer.arm.com/downloads/-/arm-gnu-toolchain-downloads to download the toolchain:

We are going to download `gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabihf.tar.xz`, which has been uploaded to https://huggingface.co/csukuangfj/sherpa-ncnn-toolchains.

Assume you want to install it in the folder `$HOME/software`:

```
mkdir -p $HOME/software
cd $HOME/software
wget -q https://huggingface.co/csukuangfj/sherpa-ncnn-toolchains/resolve/main/gcc-arm-10.
↪3-2021.07-x86_64-arm-none-linux-gnueabihf.tar.xz

# For users from China
#  huggingface,
# wget -q https://hf-mirror.com/csukuangfj/sherpa-ncnn-toolchains/resolve/main/gcc-arm-
↪10.3-2021.07-x86_64-arm-none-linux-gnueabihf.tar.xz

tar xf gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabihf.tar.xz
```

Next, we need to set the following environment variable:

```
export PATH=$HOME/software/gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabihf/bin:$PATH
```

To check that we have installed the cross-compiling toolchain successfully, please run:

```
arm-none-linux-gnueabihf-gcc --version
```

which should print the following log:

```
arm-none-linux-gnueabihf-gcc (GNU Toolchain for the A-profile Architecture 10.3-2021.07␣
↪(arm-10.29)) 10.3.1 20210621
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Congratulations! You have successfully installed a toolchain for cross-compiling sherpa-onnx.

Build sherpa-onnx

Finally, let us build sherpa-onnx.

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
export BUILD_SHARED_LIBS=ON
./build-arm-linux-gnueabihf.sh
```

After building, you will get the following binaries:

```
$ ls -lh  build-arm-linux-gnueabihf/install/bin/

total 1.2M
-rwxr-xr-x 1 kuangfangjun root 395K Jul  7 16:28 sherpa-onnx
-rwxr-xr-x 1 kuangfangjun root 391K Jul  7 16:28 sherpa-onnx-alsa
-rwxr-xr-x 1 kuangfangjun root 351K Jul  7 16:28 sherpa-onnx-offline
```

That's it!

Hint:

- `sherpa-onnx` is for decoding a single file using a streaming model

- `sherpa-onnx-offline` is for decoding a single file using a non-streaming model
- `sherpa-onnx-alsa` is for real-time speech recongition using a streaming model by reading the microphone with ALSA

Caution: We recommend that you use `sherpa-onnx-alsa` on embedded systems such as Raspberry pi.

You need to provide a `device_name` when invoking `sherpa-onnx-alsa`. We describe below how to find the device name for your microphone.

Run the following command:

```
arecord -l
```

to list all avaliable microphones for recording. If it complains that `arecord: command not found`, please use `sudo apt-get install alsa-utils` to install it.

If the above command gives the following output:

```
**** List of CAPTURE Hardware Devices ****
card 0: Audio [Axera Audio], device 0: 49ac000.i2s_mst-es8328-hifi-analog␣
↪es8328-hifi-analog-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

In this case, I only have 1 microphone. It is `card 0` and that card has only `device 0`. To select `card 0` and `device 0` on that card, we need to pass `plughw:0,0` to `sherpa-onnx-alsa`. (Note: It has the format `plughw:card_number,device_index`.)

For instance, you have to use

```
# Note: We use int8 models below.
./bin/sherpa-onnx-alsa \
  ./sherpa-onnx-streaming-zipformer-en-2023-06-26/tokens.txt \
  ./sherpa-onnx-streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-
↪chunk-16-left-64.int8.onnx \
  ./sherpa-onnx-streaming-zipformer-en-2023-06-26/decoder-epoch-99-avg-1-
↪chunk-16-left-64.int8.onnx \
  ./sherpa-onnx-streaming-zipformer-en-2023-06-26/joiner-epoch-99-avg-1-
↪chunk-16-left-64.int8.onnx \
  "plughw:0,0"
```

Please change the card number and also the device index on the selected card accordingly in your own situation. Otherwise, you won't be able to record with your microphone.

Please read *Pre-trained models* for usages about the generated binaries.

Read below if you want to learn more.

Hint: By default, all external dependencies are statically linked. That means, the generated binaries are self-contained (except that it requires the onnxruntime shared library at runtime).

You can use the following commands to check that and you will find they depend only on system libraries.

```
$ readelf -d build-arm-linux-gnueabihf/install/bin/sherpa-onnx
```

(continues on next page)

(continued from previous page)

```
  Dynamic section at offset 0x61ee8 contains 30 entries:
    Tag        Type                         Name/Value
   0x00000001 (NEEDED)                     Shared library: [libonnxruntime.so.
↪1.14.0]
   0x00000001 (NEEDED)                     Shared library: [libstdc++.so.6]
   0x00000001 (NEEDED)                     Shared library: [libm.so.6]
   0x00000001 (NEEDED)                     Shared library: [libgcc_s.so.1]
   0x00000001 (NEEDED)                     Shared library: [libc.so.6]
   0x0000000f (RPATH)                      Library rpath: [$ORIGIN:$ORIGIN/../
↪lib:$ORIGIN/../../../sherpa_onnx/lib]

$ readelf -d build-arm-linux-gnueabihf/install/bin/sherpa-onnx-alsa

  Dynamic section at offset 0x60ee0 contains 31 entries:
    Tag        Type                         Name/Value
   0x00000001 (NEEDED)                     Shared library: [libasound.so.2]
   0x00000001 (NEEDED)                     Shared library: [libonnxruntime.so.
↪1.14.0]
   0x00000001 (NEEDED)                     Shared library: [libstdc++.so.6]
   0x00000001 (NEEDED)                     Shared library: [libm.so.6]
   0x00000001 (NEEDED)                     Shared library: [libgcc_s.so.1]
   0x00000001 (NEEDED)                     Shared library: [libc.so.6]
   0x0000000f (RPATH)                      Library rpath: [$ORIGIN]
```

Please create an issue at https://github.com/k2-fsa/sherpa-onnx/issues if you have any problems.

How to build static libraries and static linked binaries

If you want to build static libraries and static linked binaries, please first download a cross compile toolchain with GCC >= 9.0. The following is an example:

```
mkdir -p $HOME/software
cd $HOME/software
wget -q https://huggingface.co/csukuangfj/sherpa-ncnn-toolchains/resolve/main/gcc-arm-10.
↪3-2021.07-x86_64-arm-none-linux-gnueabihf.tar.xz

# For users from China
#  huggingface,
wget -q https://hf-mirror.com/csukuangfj/sherpa-ncnn-toolchains/resolve/main/gcc-arm-10.
↪3-2021.07-x86_64-arm-none-linux-gnueabihf.tar.xz

tar xf gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabihf.tar.xz
```

Next, we need to set the following environment variable:

```
export PATH=$HOME/software/gcc-arm-10.3-2021.07-x86_64-arm-none-linux-gnueabihf/bin:$PATH
```

To check that we have installed the cross-compiling toolchain successfully, please run:

```
arm-none-linux-gnueabihf-gcc --version
```

which should print the following log:

```
arm-none-linux-gnueabihf-gcc (GNU Toolchain for the A-profile Architecture 10.3-2021.07␣
↪(arm-10.29)) 10.3.1 20210621
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Now you can build static libraries and static linked binaries with the following commands:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
export BUILD_SHARED_LIBS=OFF
./build-arm-linux-gnueabihf.sh
```

You can use the following commands to check that the generated binaries are indeed static linked:

```
$ cd build-arm-linux-gnueabihf/bin

$ ldd sherpa-onnx-alsa
    not a dynamic executable

$ readelf -d sherpa-onnx-alsa

Dynamic section at offset 0xa68eb4 contains 31 entries:
  Tag        Type                         Name/Value
 0x00000001 (NEEDED)                     Shared library: [libasound.so.2]
 0x00000001 (NEEDED)                     Shared library: [libdl.so.2]
 0x00000001 (NEEDED)                     Shared library: [libm.so.6]
 0x00000001 (NEEDED)                     Shared library: [libpthread.so.0]
 0x00000001 (NEEDED)                     Shared library: [libc.so.6]
 0x00000001 (NEEDED)                     Shared library: [ld-linux-armhf.so.3]
 0x0000000f (RPATH)                      Library rpath: [$ORIGIN:/star-fj/fangjun/open-
↪source/sherpa-onnx/build-arm-linux-gnueabihf/_deps/espeak_ng-src/lib:/star-fj/fangjun/
↪open-source/sherpa-onnx/build-arm-linux-gnueabihf/_deps/onnxruntime-src/lib:]
 0x0000000c (INIT)                       0x13550
```

8.2.6 Embedded Linux (riscv64)

This page describes how to build sherpa-onnx for embedded Linux (RISC-V, 64-bit) with cross-compiling on an x64 machine with Ubuntu OS. It also demonstrates how to use `qemu` to run the compiled binaries.

Hint: We provide a colab notebook for you to try this section step by step.

If you are using Windows/macOS or you don't want to setup your local environment for cross-compiling, please use the above colab notebook.

Note: You can download pre-compiled binaries for `riscv64` from the following URL https://huggingface.co/csukuangfj/sherpa-onnx-libs/tree/main/riscv64

Please always download the latest version.

Example command to download the version `1.9.12`:

```
# binaries built with shared libraries
wget https://huggingface.co/csukuangfj/sherpa-onnx-libs/resolve/main/riscv64/
↪sherpa-onnx-v1.9.12-linux-riscv64-shared.tar.bz2

# For users from China
#  huggingface,

# binaries built with shared libraries
# wget https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/resolve/main/riscv64/
↪sherpa-onnx-v1.9.12-linux-riscv64-shared.tar.bz2
```

Install toolchain

The first step is to install a toolchain for cross-compiling.

```
mkdir -p $HOME/toolchain

wget -q https://occ-oss-prod.oss-cn-hangzhou.aliyuncs.com/resource//1663142514282/
↪Xuantie-900-gcc-linux-5.10.4-glibc-x86_64-V2.6.1-20220906.tar.gz

tar xf ./Xuantie-900-gcc-linux-5.10.4-glibc-x86_64-V2.6.1-20220906.tar.gz --strip-
↪components 1 -C $HOME/toolchain
```

Next, we need to set the following environment variable:

```
export PATH=$HOME/toolchain/bin:$PATH
```

To check that you have installed the toolchain successfully, please run

```
$ riscv64-unknown-linux-gnu-gcc --version

  riscv64-unknown-linux-gnu-gcc (Xuantie-900 linux-5.10.4 glibc gcc Toolchain V2.6.1 B-
↪20220906) 10.2.0
  Copyright (C) 2020 Free Software Foundation, Inc.
  This is free software; see the source for copying conditions.  There is NO
  warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

$ riscv64-unknown-linux-gnu-g++ --version

  riscv64-unknown-linux-gnu-g++ (Xuantie-900 linux-5.10.4 glibc gcc Toolchain V2.6.1 B-
↪20220906) 10.2.0
  Copyright (C) 2020 Free Software Foundation, Inc.
  This is free software; see the source for copying conditions.  There is NO
  warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Build sherpa-onnx

Next, let us build sherpa-onnx.

Hint: Currently, only shared libraries are supported. We will support static linking in the future.

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
./build-riscv64-linux-gnu.sh
```

After building, you will get the following files

```
$ ls -lh build-riscv64-linux-gnu/install/bin
$ echo "---"
$ ls -lh build-riscv64-linux-gnu/install/lib

  total 292K
  -rwxr-xr-x 1 root root 23K Mar 20 09:41 sherpa-onnx
  -rwxr-xr-x 1 root root 27K Mar 20 09:41 sherpa-onnx-alsa
  -rwxr-xr-x 1 root root 31K Mar 20 09:41 sherpa-onnx-alsa-offline
  -rwxr-xr-x 1 root root 40K Mar 20 09:41 sherpa-onnx-alsa-offline-speaker-identification
  -rwxr-xr-x 1 root root 23K Mar 20 09:41 sherpa-onnx-keyword-spotter
  -rwxr-xr-x 1 root root 27K Mar 20 09:41 sherpa-onnx-keyword-spotter-alsa
  -rwxr-xr-x 1 root root 23K Mar 20 09:41 sherpa-onnx-offline
  -rwxr-xr-x 1 root root 39K Mar 20 09:41 sherpa-onnx-offline-parallel
  -rwxr-xr-x 1 root root 19K Mar 20 09:41 sherpa-onnx-offline-tts
  -rwxr-xr-x 1 root root 31K Mar 20 09:41 sherpa-onnx-offline-tts-play-alsa
  ---
  total 30M
  -rw-r--r-- 1 root root 256K Mar 20 09:41 libespeak-ng.so
  -rw-r--r-- 1 root root  71K Mar 20 09:41 libkaldi-decoder-core.so
  -rw-r--r-- 1 root root  67K Mar 20 09:41 libkaldi-native-fbank-core.so
  -rw-r--r-- 1 root root  13M Mar 20 09:35 libonnxruntime.so
  -rw-r--r-- 1 root root  13M Mar 20 09:35 libonnxruntime.so.1.14.1
  lrwxrwxrwx 1 root root   23 Mar 20 09:41 libpiper_phonemize.so -> libpiper_phonemize.
↪so.1
  lrwxrwxrwx 1 root root   27 Mar 20 09:41 libpiper_phonemize.so.1 -> libpiper_phonemize.
↪so.1.2.0
  -rw-r--r-- 1 root root 395K Mar 20 09:41 libpiper_phonemize.so.1.2.0
  -rw-r--r-- 1 root root 1.3M Mar 20 09:41 libsherpa-onnx-core.so
  lrwxrwxrwx 1 root root   23 Mar 20 09:41 libsherpa-onnx-fst.so -> libsherpa-onnx-fst.
↪so.6
  -rw-r--r-- 1 root root 1.4M Mar 20 09:41 libsherpa-onnx-fst.so.6
  -rw-r--r-- 1 root root 752K Mar 20 09:41 libsherpa-onnx-kaldifst-core.so
  -rw-r--r-- 1 root root 202K Mar 20 09:41 libucd.so
  drwxr-xr-x 2 root root 4.0K Mar 20 09:41 pkgconfig
```

```
$ file build-riscv64-linux-gnu/install/bin/sherpa-onnx

build-riscv64-linux-gnu/install/bin/sherpa-onnx: ELF 64-bit LSB executable, UCB RISC-V,␣
↪RVC, double-float ABI, version 1 (GNU/Linux), dynamically linked, interpreter /lib/ld-
↪linux-riscv64-lp64d.so.1, for GNU/Linux 4.15.0, stripped
```

```
$ readelf -d build-riscv64-linux-gnu/install/bin/sherpa-onnx
```

```
$ find $HOME/toolchain/ -name ld-linux-riscv64-lp64d.so.1

  Dynamic section at offset 0x4d40 contains 39 entries:
    Tag        Type                         Name/Value
   0x0000000000000001 (NEEDED)             Shared library: [libsherpa-onnx-core.so]
   0x0000000000000001 (NEEDED)             Shared library: [libkaldi-native-fbank-core.
↪so]
   0x0000000000000001 (NEEDED)             Shared library: [libkaldi-decoder-core.so]
   0x0000000000000001 (NEEDED)             Shared library: [libsherpa-onnx-kaldifst-
↪core.so]
   0x0000000000000001 (NEEDED)             Shared library: [libsherpa-onnx-fst.so.6]
   0x0000000000000001 (NEEDED)             Shared library: [libpiper_phonemize.so.1]
   0x0000000000000001 (NEEDED)             Shared library: [libonnxruntime.so.1.14.1]
   0x0000000000000001 (NEEDED)             Shared library: [libespeak-ng.so]
   0x0000000000000001 (NEEDED)             Shared library: [libucd.so]
   0x0000000000000001 (NEEDED)             Shared library: [libstdc++.so.6]
   0x0000000000000001 (NEEDED)             Shared library: [libm.so.6]
   0x0000000000000001 (NEEDED)             Shared library: [libgcc_s.so.1]
   0x0000000000000001 (NEEDED)             Shared library: [libpthread.so.0]
   0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
   0x000000000000000f (RPATH)              Library rpath: [$ORIGIN:$ORIGIN/../lib:
↪$ORIGIN/../../../sherpa_onnx/lib]
   0x0000000000000020 (PREINIT_ARRAY)      0x15d20
   0x0000000000000021 (PREINIT_ARRAYSZ)    8 (bytes)
   0x0000000000000019 (INIT_ARRAY)         0x15d28
   0x000000000000001b (INIT_ARRAYSZ)       16 (bytes)
   0x000000000000001a (FINI_ARRAY)         0x15d38
   0x000000000000001c (FINI_ARRAYSZ)       8 (bytes)
   0x0000000000000004 (HASH)               0x10280
   0x000000006ffffef5 (GNU_HASH)           0x10418
   0x0000000000000005 (STRTAB)             0x10bd8
   0x0000000000000006 (SYMTAB)             0x105f0
   0x000000000000000a (STRSZ)              3652 (bytes)
   0x000000000000000b (SYMENT)             24 (bytes)
   0x0000000000000015 (DEBUG)              0x0
   0x0000000000000003 (PLTGOT)             0x16000
   0x0000000000000002 (PLTRELSZ)           1056 (bytes)
   0x0000000000000014 (PLTREL)             RELA
   0x0000000000000017 (JMPREL)             0x11bb0
   0x0000000000000007 (RELA)               0x11b80
   0x0000000000000008 (RELASZ)             1104 (bytes)
   0x0000000000000009 (RELAENT)            24 (bytes)
   0x000000006ffffffe (VERNEED)            0x11aa0
   0x000000006fffffff (VERNEEDNUM)         4
   0x000000006ffffff0 (VERSYM)             0x11a1c
   0x0000000000000000 (NULL)               0x0

 /root/toolchain/sysroot/lib/ld-linux-riscv64-lp64d.so.1
```

That's it!

Please create an issue at https://github.com/k2-fsa/sherpa-onnx/issues if you have any problems.

Read more if you want to run the binaries with `qemu`.

qemu

Hint: This subsection works only on x64 Linux.

Caution: Please don't use any other methods to install `qemu-riscv64`. Only the method listed in this subsection is known to work.

Please use the following command to download the `qemu-riscv64` binary.

```
mkdir -p $HOME/qemu

mkdir -p /tmp
cd /tmp
wget -q https://files.pythonhosted.org/packages/21/f4/
↪733f29c435987e8bb264a6504c7a4ea4c04d0d431b38a818ab63eef082b9/xuantie_qemu-20230825-py3-
↪none-manylinux1_x86_64.whl

unzip xuantie_qemu-20230825-py3-none-manylinux1_x86_64.whl
cp -v ./qemu/qemu-riscv64 $HOME/qemu

export PATH=$HOME/qemu:$PATH
```

To check that we have installed `qemu-riscv64` successfully, please run:

```
qemu-riscv64 -h
```

which should give the following output:

```
usage: qemu-riscv64 [options] program [arguments...]
Linux CPU emulator (compiled for riscv64 emulation)

Options and associated environment variables:

Argument             Env-variable      Description
-h                                     print this help
-help
-g port              QEMU_GDB          wait gdb connection to 'port'
-L path              QEMU_LD_PREFIX    set the elf interpreter prefix to 'path'
-s size              QEMU_STACK_SIZE   set the stack size to 'size' bytes
-cpu model           QEMU_CPU          select CPU (-cpu help for list)
-E var=value         QEMU_SET_ENV      sets targets environment variable (see below)
-U var               QEMU_UNSET_ENV    unsets targets environment variable (see below)
-0 argv0             QEMU_ARGV0        forces target process argv[0] to be 'argv0'
-r uname             QEMU_UNAME        set qemu uname release string to 'uname'
-B address           QEMU_GUEST_BASE   set guest_base address to 'address'
-R size              QEMU_RESERVED_VA  reserve 'size' bytes for guest virtual address␣
↪space
-d item[,...]        QEMU_LOG          enable logging of specified items (use '-d help'␣
↪for a list of items)
```

(continues on next page)

(continued from previous page)

```
-dfilter range[,...] QEMU_DFILTER      filter logging based on address range
-D logfile           QEMU_LOG_FILENAME write logs to 'logfile' (default stderr)
-p pagesize          QEMU_PAGESIZE     set the host page size to 'pagesize'
-singlestep          QEMU_SINGLESTEP   run in singlestep mode
-strace              QEMU_STRACE       log system calls
-pctrace             QEMU_PCTRACE      log pctrace
-seed                QEMU_RAND_SEED    Seed for pseudo-random number generator
-trace               QEMU_TRACE        [[enable=]<pattern>][,events=<file>][,file=<file>]
-csky-extend         CSKY_EXTEND       [tb_trace=<on|off>][,jcount_start=<addr>][,jcount_
↪end=<addr>][vdsp=<vdsp>][exit_addr=<addr>][denormal=<on|off>]
-CPF                 CSKY_PROFILING
-csky-trace          CSKY_TRACE        [port=<port>][,tb_trace=<on|off>][,mem_trace=
↪<on|off>][,auto_trace=<on|off>][,start=addr][,exit=addr]
-plugin              QEMU_PLUGIN       [file=]<file>[,arg=<string>]
-version             QEMU_VERSION      display version information and exit

Defaults:
QEMU_LD_PREFIX  = /usr/gnemul/qemu-riscv64
QEMU_STACK_SIZE = 8388608 byte

You can use -E and -U options or the QEMU_SET_ENV and
QEMU_UNSET_ENV environment variables to set and unset
environment variables for the target process.
It is possible to provide several variables by separating them
by commas in getsubopt(3) style. Additionally it is possible to
provide the -E and -U options multiple times.
The following lines are equivalent:
    -E var1=val2 -E var2=val2 -U LD_PRELOAD -U LD_DEBUG
    -E var1=val2,var2=val2 -U LD_PRELOAD,LD_DEBUG
    QEMU_SET_ENV=var1=val2,var2=val2 QEMU_UNSET_ENV=LD_PRELOAD,LD_DEBUG
Note that if you provide several changes to a single variable
the last change will stay in effect.

See <https://qemu.org/contribute/report-a-bug> for how to report bugs.
More information on the QEMU project at <https://qemu.org>.
```

We describe below how to use `qemu-riscv64` to run speech-to-text and text-to-speech.

Run speech-to-text with qemu

We use *csukuangfj/sherpa-onnx-streaming-zipformer-en-20M-2023-02-17 (English)* as the test model.

Note: You can select any model from *Pre-trained models*.

Please use the following command to download the model:

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-en-20M-2023-02-17.tar.bz2
```

(continues on next page)

(continued from previous page)

```
tar xvf sherpa-onnx-streaming-zipformer-en-20M-2023-02-17.tar.bz2
rm sherpa-onnx-streaming-zipformer-en-20M-2023-02-17.tar.bz2
```

Now you can use the following command to run it with `qemu-riscv64`:

```
cd /path/to/sherpa-onnx

export PATH=$HOME/qemu:$PATH

qemu-riscv64 build-riscv64-linux-gnu/install/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/encoder-epoch-99-avg-1.
↪onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/decoder-epoch-99-avg-1.
↪onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/joiner-epoch-99-avg-1.
↪onnx \
  ./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/test_wavs/0.wav
```

It will throw the following error:

```
qemu-riscv64: Could not open '/lib/ld-linux-riscv64-lp64d.so.1': No such file or␣
↪directory
```

Please use the following command instead:

```
cd /path/to/sherpa-onnx

export PATH=$HOME/qemu:$PATH
export QEMU_LD_PREFIX=$HOME/toolchain/sysroot

qemu-riscv64 build-riscv64-linux-gnu/install/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/encoder-epoch-99-avg-1.
↪onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/decoder-epoch-99-avg-1.
↪onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/joiner-epoch-99-avg-1.
↪onnx \
  ./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/test_wavs/0.wav
```

It will throw a second error:

```
build-riscv64-linux-gnu/install/bin/sherpa-onnx: error while loading shared libraries:␣
↪ld-linux-riscv64xthead-lp64d.so.1: cannot open shared object file: No such file or␣
↪directory
```

Please use the following command instead:

```
cd /path/to/sherpa-onnx

export PATH=$HOME/qemu:$PATH
export QEMU_LD_PREFIX=$HOME/toolchain/sysroot
```

(continues on next page)

(continued from previous page)

```
export LD_LIBRARY_PATH=$HOME/toolchain/sysroot/lib:$LD_LIBRARY_PATH

qemu-riscv64 build-riscv64-linux-gnu/install/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/encoder-epoch-99-avg-1.
↪onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/decoder-epoch-99-avg-1.
↪onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/joiner-epoch-99-avg-1.
↪onnx \
  ./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/test_wavs/0.wav
```

Finally, it prints the following output:

```
/content/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 build-riscv64-linux-gnu/
↪install/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/
↪tokens.txt --encoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/encoder-epoch-
↪99-avg-1.onnx --decoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/decoder-
↪epoch-99-avg-1.onnx --joiner=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/
↪joiner-epoch-99-avg-1.onnx ./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/test_
↪wavs/0.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/encoder-epoch-99-avg-1.onnx",␣
↪decoder="./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/decoder-epoch-99-avg-1.
↪onnx", joiner="./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/joiner-epoch-99-avg-
↪1.onnx"), paraformer=OnlineParaformerModelConfig(encoder="", decoder=""), wenet_
↪ctc=OnlineWenetCtcModelConfig(model="", chunk_size=16, num_left_chunks=4), zipformer2_
↪ctc=OnlineZipformer2CtcModelConfig(model=""), tokens="./sherpa-onnx-streaming-
↪zipformer-en-20M-2023-02-17/tokens.txt", num_threads=1, debug=False, provider="cpu",␣
↪model_type=""), lm_config=OnlineLMConfig(model="", scale=0.5), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=True, max_active_paths=4, hotwords_score=1.5, hotwords_file="", decoding_
↪method="greedy_search", blank_penalty=0)
./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/test_wavs/0.wav
Elapsed seconds: 70, Real time factor (RTF): 11
 THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUARTER OF THE BRAFFLELS
{ "text": " THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUARTER OF THE␣
↪BRAFFLELS", "tokens": [ " THE", " YE", "LL", "OW", " LA", "M", "P", "S", " WOULD", "␣
↪LIGHT", " UP", " HE", "RE", " AND", " THERE", " THE", " S", "QUA", "LI", "D", " ", "QUA
↪", "R", "TER", " OF", " THE", " B", "RA", "FF", "L", "EL", "S" ], "timestamps": [ 2.04,
↪ 2.16, 2.28, 2.36, 2.52, 2.64, 2.68, 2.76, 2.92, 3.08, 3.40, 3.60, 3.72, 3.88, 4.12, 4.
↪48, 4.64, 4.68, 4.84, 4.96, 5.16, 5.20, 5.32, 5.36, 5.60, 5.72, 5.92, 5.96, 6.08, 6.24,
↪ 6.36, 6.60 ], "ys_probs": [ -0.454799, -0.521409, -0.345871, -0.001244, -0.240359, -0.
↪013972, -0.010445, -0.051701, -0.000371, -0.171570, -0.002205, -0.026703, -0.006903, -
↪0.021168, -0.011662, -0.001059, -0.005089, -0.000273, -0.575480, -0.024973, -0.159344,␣
↪-0.000042, -0.011082, -0.187136, -0.004002, -0.292751, -0.084873, -0.241302, -0.543844,
↪ -0.428164, -0.853198, -0.093776 ], "lm_probs": [  ], "context_scores": [  ], "segment
↪": 0, "start_time": 0.00, "is_final": false}
```

(continues on next page)

(continued from previous page)

Hint: As you can see, the RTF is 11, indicating that it is very slow to run the model with the `qemu` simulator. Running on a real RISC-V board should be much faster.

Run text-to-speech with qemu

Please visit https://github.com/k2-fsa/sherpa-onnx/releases/tag/tts-models to download a text-to-speech model. We use the following model `vits-piper-en_US-amy-low.tar.bz2`:

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/tts-models/vits-piper-en_US-
↪amy-low.tar.bz2
tar xf vits-piper-en_US-amy-low.tar.bz2
rm vits-piper-en_US-amy-low.tar.bz2
```

After downloading the model, we can use the following command to run it:

```
cd /path/to/sherpa-onnx

export PATH=$HOME/qemu:$PATH
export QEMU_LD_PREFIX=$HOME/toolchain/sysroot
export LD_LIBRARY_PATH=$HOME/toolchain/sysroot/lib:$LD_LIBRARY_PATH

qemu-riscv64 build-riscv64-linux-gnu/install/bin/sherpa-onnx-offline-tts \
  --vits-model=./vits-piper-en_US-amy-low/en_US-amy-low.onnx \
  --vits-tokens=./vits-piper-en_US-amy-low/tokens.txt \
  --vits-data-dir=./vits-piper-en_US-amy-low/espeak-ng-data \
  --output-filename=./a-test.wav \
  "Friends fell out often because life was changing so fast. The easiest thing in the␣
↪world was to lose touch with someone."
```

The log of the above command is given below:

```
/content/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 build-riscv64-linux-gnu/
↪install/bin/sherpa-onnx-offline-tts --vits-model=./vits-piper-en_US-amy-low/en_US-amy-
↪low.onnx --vits-tokens=./vits-piper-en_US-amy-low/tokens.txt --vits-data-dir=./vits-
↪piper-en_US-amy-low/espeak-ng-data --output-filename=./a-test.wav 'Friends fell out␣
↪often because life was changing so fast. The easiest thing in the world was to lose␣
↪touch with someone.'

Elapsed seconds: 270.745 s
Audio duration: 7.904 s
Real-time factor (RTF): 270.745/7.904 = 34.254
The text is: Friends fell out often because life was changing so fast. The easiest thing␣
↪in the world was to lose touch with someone.. Speaker ID: 0
Saved to ./a-test.wav successfully!
```

If you want to build an Android app, please refer to *Android*. If you want to build an iOS app, please refer to *iOS*.

8.3 Frequently Asked Question (FAQs)

This page contains frequently asked questions for sherpa-onnx.

8.3.1

,

,

8.3.2 Cannot open shared library libasound_module_conf_pulse.so

The detailed errors are given below:

```
Cannot open shared library libasound_module_conf_pulse.so
(/usr/lib64/alsa-lib/libasound_module_conf_pulse.so: cannot open shared object file: No␣
↪such file or directory)
ALSA lib pcm.c:2722:(snd_pcm_open_noupdate) Unknown PCM
```

If you use Linux and get the above error when trying to use the microphone, please do the following:

1. Locate where is the file `libasound_module_conf_pulse.so` on your system

   ```
   find / -name libasound_module_conf_pulse.so 2>/dev/null
   ```

2. If the above search command prints:

   ```
   /usr/lib/x86_64-linux-gnu/alsa-lib/libasound_module_conf_pulse.so
   /usr/lib/i386-linux-gnu/alsa-lib/libasound_module_conf_pulse.so
   ```

3. Please run:

   ```
   sudo mkdir -p /usr/lib64/alsa-lib
   sudo ln -s /usr/lib/x86_64-linux-gnu/alsa-lib/libasound_module_conf_pulse.
   ↪so /usr/lib64/alsa-lib
   ```

4. Now your issue should be fixed.

8.3.3 TTS

Please see *How to enable UTF-8 on Windows*. You need to use `UTF-8` encoding for your system.

8.3.4 ./gitcompile: line 89: libtoolize: command not found

If you are using Linux and get the following error:

```
./gitcompile: line 89: libtoolize: command not found
```

Please run:

```
sudo apt-get install libtool
```

8.3.5 OSError: PortAudio library not found

If you have the following error on Linux (Ubuntu),

```
Traceback (most recent call last):
  File "/mnt/sdb/shared/sherpa-onnx/./python-api-examples/vad-microphone.py", line 8, in
↪<module>
    import sounddevice as sd
  File "/mnt/sdb/shared/py311/lib/python3.11/site-packages/sounddevice.py", line 71, in
↪<module>
    raise OSError('PortAudio library not found')
OSError: PortAudio library not found
```

Then please run:

```
sudo apt-get install libportaudio2
```

and then re-try.

8.3.6 imports github.com/k2-fsa/sherpa-onnx-go-linux: build constraints exclude all Go files

If you have the following output when running `go build`:

```
[root@VM-0-3-centos non-streaming-decode-files]# go build
package non-streaming-decode-files
 imports github.com/k2-fsa/sherpa-onnx-go/sherpa_onnx
 imports github.com/k2-fsa/sherpa-onnx-go-linux: build constraints exclude all Go files␣
↪in /root/go/pkg/mod/github.com/k2-fsa/sherpa-onnx-go-linux@v1.9.21
```

Please first run:

```
go env -w CGO_ENABLED=1
```

And then re-run `go build`.

8.4 Python

In this section, we describe how to install the Python package sherpa-onnx.

8.4.1 Install the Python Package

You can select one of the following methods to install the Python package.

Method 1 (From pre-compiled wheels, CPU only)

Hint: This method supports the following platfroms:

- Linux (`x64`, `aarch64`, `armv7l`),
- macOS (`x64`, `arm64`)
- Windows (`x64`, `x86`)

Note that this method installs a CPU-only version of sherpa-onnx.

```
pip install sherpa-onnx
```

To check you have installed sherpa-onnx successfully, please run

```
python3 -c "import sherpa_onnx; print(sherpa_onnx.__file__)"

which sherpa-onnx
sherpa-onnx --help

ls -lh $(dirname $(which sherpa-onnx))/sherpa-onnx*
```

Hint: You can find previous releases at https://k2-fsa.github.io/sherpa/onnx/cpu.html

For Chinese users and users who have no access to huggingface, please visit https://k2-fsa.github.io/sherpa/onnx/cpu-cn.html.

You can use:

```
pip install sherpa-onnx -f https://k2-fsa.github.io/sherpa/onnx/cpu.html
```

or:

```
# For Chinese uers
pip install sherpa-onnx -f https://k2-fsa.github.io/sherpa/onnx/cpu-cn.html
```

Method 2 (From pre-compiled wheels, CPU + CUDA)

Note: This method installs a version of sherpa-onnx supporting both CUDA and CPU. You need to pass the argument provider=cuda to use NVIDIA GPU, which always uses GPU 0. Otherwise, it uses CPU by default.

Please use the environment variable CUDA_VISIBLE_DEVICES to control which GPU is mapped to GPU 0.

By default, provider is set to cpu.

Remeber to follow https://onnxruntime.ai/docs/execution-providers/CUDA-ExecutionProvider.html#requirements to install CUDA 11.8.

If you have issues about installing CUDA 11.8, please have a look at https://k2-fsa.github.io/k2/installation/cuda-cudnn.html#cuda-11-8.

Note that you don't need to have sudo permission to install CUDA 11.8

This approach supports only Linux x64 and Windows x64.

Please use the following command to install CUDA-enabled sherpa-onnx:

```
# We use 1.10.16 here for demonstration.
#
# Please visit https://k2-fsa.github.io/sherpa/onnx/cuda.html
# to find available versions

pip install sherpa-onnx==1.10.16+cuda -f https://k2-fsa.github.io/sherpa/onnx/cuda.html

# For Chinese users, please use
# pip install sherpa-onnx==1.10.16+cuda -f https://k2-fsa.github.io/sherpa/onnx/cuda-cn.
↪html
```

The installation logs are given below:

```
Looking in links: https://k2-fsa.github.io/sherpa/onnx/cuda.html
Collecting sherpa-onnx==1.10.16+cuda
  Downloading https://huggingface.co/csukuangfj/sherpa-onnx-wheels/resolve/main/cuda/1.
↪10.16/sherpa_onnx-1.10.16%2Bcuda-cp310-cp310-linux_x86_64.whl (183.3 MB)
     183.3/183.3 MB 4.4 MB/s eta 0:00:00
Installing collected packages: sherpa-onnx
Successfully installed sherpa-onnx-1.10.16+cuda
```

To check that you have installed sherpa-onnx successfully, please run:

```
python3 -c "import sherpa_onnx; print(sherpa_onnx.__version__)"
```

which should print something like below:

```
1.10.16+cuda
```

Method 3 (From source)

CPU

Nvidia GPU (CUDA)

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
python3 setup.py install
```

```
git clone https://github.com/k2-fsa/sherpa-onnx
export SHERPA_ONNX_CMAKE_ARGS="-DSHERPA_ONNX_ENABLE_GPU=ON"
cd sherpa-onnx
python3 setup.py install
```

Method 4 (For developers)

CPU

Nvidia GPU (CUDA)

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
mkdir build
cd build

cmake \
  -DSHERPA_ONNX_ENABLE_PYTHON=ON \
  -DBUILD_SHARED_LIBS=ON \
  -DSHERPA_ONNX_ENABLE_CHECK=OFF \
  -DSHERPA_ONNX_ENABLE_PORTAUDIO=OFF \
  -DSHERPA_ONNX_ENABLE_C_API=OFF \
  -DSHERPA_ONNX_ENABLE_WEBSOCKET=OFF \
  ..

make -j
export PYTHONPATH=$PWD/../sherpa-onnx/python/:$PWD/lib:$PYTHONPATH
```

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
mkdir build
cd build

cmake \
  -DSHERPA_ONNX_ENABLE_PYTHON=ON \
  -DBUILD_SHARED_LIBS=ON \
  -DSHERPA_ONNX_ENABLE_CHECK=OFF \
  -DSHERPA_ONNX_ENABLE_PORTAUDIO=OFF \
  -DSHERPA_ONNX_ENABLE_C_API=OFF \
  -DSHERPA_ONNX_ENABLE_WEBSOCKET=OFF \
  -DSHERPA_ONNX_ENABLE_GPU=ON \
  ..
```

(continues on next page)

(continued from previous page)

```
make -j
export PYTHONPATH=$PWD/../sherpa-onnx/python/:$PWD/lib:$PYTHONPATH
```

Hint: You need to install CUDA toolkit. Otherwise, you would get errors at runtime.

You can refer to https://k2-fsa.github.io/k2/installation/cuda-cudnn.html to install CUDA toolkit.

Check your installation

To check that sherpa-onnx has been successfully installed, please use:

```
python3 -c "import sherpa_onnx; print(sherpa_onnx.__file__)"
```

It should print some output like below:

```
/Users/fangjun/py38/lib/python3.8/site-packages/sherpa_onnx/__init__.py
```

Please refer to:

https://github.com/k2-fsa/sherpa-onnx/tree/master/python-api-examples

for usages.

Please refer to *Pre-trained models* for a list of pre-trained models.

8.4.2 Decode files

In this section, we demonstrate how to use the Python API of sherpa-onnx to decode files.

Hint: We only support WAVE files of single channel and each sample should have 16-bit, while the sample rate of the file can be arbitrary and it does not need to be 16 kHz

Streaming zipformer

We use *csukuangfj/sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20 (Bilingual, Chinese + English)* as an example below.

```
cd /path/to/sherpa-onnx

python3 ./python-api-examples/online-decode-files.py \
  --tokens=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/encoder-epoch-
↪99-avg-1.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/decoder-epoch-
↪99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/joiner-epoch-99-
↪avg-1.onnx \
  ./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/0.wav \
  ./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/1.wav \
```

(continues on next page)

(continued from previous page)

```
  ./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/2.wav \
  ./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/3.wav \
  ./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/8k.wav
```

Hint: `online-decode-files.py` is from https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/online-decode-files.py

Note: You can replace `encoder-epoch-99-avg-1.onnx` with `encoder-epoch-99-avg-1.int8.onnx` to use `int8` models for decoding.

The output is given below:

```
Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Started!
Done!
./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/0.wav
 MONDAY TODAY IS LIBR THE DAY AFTER TOMORROW
----------
./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/1.wav
 ALWAYS ALWAYS
----------
./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/2.wav
 FREQUENTLY
----------
./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/3.wav
 ES
----------
./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/8k.wav
 IN TIME
----------
num_threads: 1
decoding_method: greedy_search
Wave duration: 17.640 s
Elapsed time: 3.907 s
Real time factor (RTF): 3.907/17.640 = 0.221
```

Non-streaming zipformer

We use *csukuangfj/sherpa-onnx-zipformer-en-2023-04-01 (English)* as an example below.

```
cd /path/to/sherpa-onnx

python3 ./python-api-examples/offline-decode-files.py \
   --tokens=./sherpa-onnx-zipformer-en-2023-04-01/tokens.txt \
   --encoder=./sherpa-onnx-zipformer-en-2023-04-01/encoder-epoch-99-avg-1.onnx \
   --decoder=./sherpa-onnx-zipformer-en-2023-04-01/decoder-epoch-99-avg-1.onnx \
   --joiner=./sherpa-onnx-zipformer-en-2023-04-01/joiner-epoch-99-avg-1.onnx \
   ./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/0.wav \
   ./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/1.wav \
   ./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/8k.wav
```

Hint: `offline-decode-files.py` is from https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/offline-decode-files.py

Note: You can replace `encoder-epoch-99-avg-1.onnx` with `encoder-epoch-99-avg-1.int8.onnx` to use `int8` models for decoding.

The output is given below:

```
Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Started!
Done!
./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/0.wav
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
----------
./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/1.wav
 GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY␣
↪CHILD WHOSE PLACE WAS ON THAT SAME DISHONOURED BOSOM TO CONNECT HER PARENT FOR EVER␣
↪WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
----------
./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/8k.wav
 YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION
----------
num_threads: 1
decoding_method: greedy_search
Wave duration: 4.825 s
Elapsed time: 2.567 s
Real time factor (RTF): 2.567/4.825 = 0.532
```

Non-streaming paraformer

We use *csukuangfj/sherpa-onnx-paraformer-zh-2023-03-28 (Chinese + English)* as an example below.

```
cd /path/to/sherpa-onnx

python3 ./python-api-examples/offline-decode-files.py \
 --tokens=./sherpa-onnx-paraformer-zh-2023-03-28/tokens.txt \
 --paraformer=./sherpa-onnx-paraformer-zh-2023-03-28/model.onnx \
 ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav \
 ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav \
 ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav \
 ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav
```

Note: You can replace `model.onnx` with `model.int8.onnx` to use `int8` models for decoding.

The output is given below:

```
Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Started!
Done!
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav

----------
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav

----------
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav

----------
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav

----------
num_threads: 1
decoding_method: greedy_search
Wave duration: 4.204 s
Elapsed time: 1.663 s
Real time factor (RTF): 1.663/4.204 = 0.396
```

8.4.3 Real-time speech recognition from a microphone

In this section, we demonstrate how to use the Python API of sherpa-onnx for real-time speech recognition with a microphone.

With endpoint detection

```
cd /path/to/sherpa-onnx

python3 ./python-api-examples/speech-recognition-from-microphone-with-endpoint-detection.
↪py \
  --tokens=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/encoder-epoch-
↪99-avg-1.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/decoder-epoch-
↪99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/joiner-epoch-99-
↪avg-1.onnx
```

Hint: `speech-recognition-from-microphone-with-endpoint-detection.py` is from https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/speech-recognition-from-microphone-with-endpoint-detection.py

In the above demo, the model files are from *csukuangfj/sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20 (Bilingual, Chinese + English)*.

Without endpoint detection

```
cd /path/to/sherpa-onnx

python3 ./python-api-examples/speech-recognition-from-microphone.py \
  --tokens=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/encoder-epoch-
↪99-avg-1.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/decoder-epoch-
↪99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/joiner-epoch-99-
↪avg-1.onnx
```

Hint: `speech-recognition-from-microphone.py` is from https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/speech-recognition-from-microphone.py

In the above demo, the model files are from *csukuangfj/sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20 (Bilingual, Chinese + English)*.

8.4.4 Speech recognition from URLs

sherpa-onnx also supports decoding from URLs.

Hint: Only streaming models are currently supported. Please modify the code for non-streaming models on need.

All types of URLs supported by `ffmpeg` are supported.

The following table lists some example URLs.

Type	Example
RTMP	`rtmp://localhost/live/livestream`
OPUS file	https://huggingface.co/spaces/k2-fsa/automatic-speech-recognition/resolve/main/test_wavs/wenetspeech/DEV_T0000000000.opus
WAVE file	https://huggingface.co/spaces/k2-fsa/automatic-speech-recognition/resolve/main/test_wavs/aishell2/ID0012W0030.wav
Local WAVE file	`file:///Users/fangjun/open-source/sherpa-onnx/a.wav`

Before you continue, please install `ffmpeg` first.

For instance, you can use `sudo apt-get install ffmpeg` for Ubuntu and `brew install ffmpeg` for macOS.

We use the model *csukuangfj/sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20 (Bilingual, Chinese + English)* for demonstration in the following examples.

Decode a URL

This example shows you how to decode a URL pointing to a file.

Hint: The file does not need to be a WAVE file. It can be a file of any format supported by `ffmpeg`.

```
python3 ./python-api-examples/speech-recognition-from-url.py \
  --encoder ./sherpa-onnx-streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --decoder ./sherpa-onnx-streaming-zipformer-en-2023-06-26/decoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --joiner ./sherpa-onnx-streaming-zipformer-en-2023-06-26/joiner-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --tokens ./sherpa-onnx-streaming-zipformer-en-2023-06-26/tokens.txt \
  --url https://huggingface.co/spaces/k2-fsa/automatic-speech-recognition/resolve/main/
↪test_wavs/librispeech/1089-134686-0001.wav
```

RTMP

In this example, we use `ffmpeg` to capture a microphone and push the audio stream to a server using RTMP, and then we start sherpa-onnx to pull the audio stream from the server for recognition.

Install the server

We will use srs as the server. Let us first install srs from source:

```
git clone -b develop https://github.com/ossrs/srs.git
cd srs/trunk
./configure
make

# Check that we have compiled srs successfully
./objs/srs --help

# Note: ./objs/srs is statically linked and depends only on system libraries.
```

Start the server

```
ulimit -HSn 10000

# switch to the directory srs/trunk
./objs/srs -c conf/srs.conf
```

The above command gives the following output:

```
srs(51047,0x7ff8451198c0) malloc: nano zone abandoned due to inability to preallocate␣
↪reserved vm space.
Asan: Please setup the env MallocNanoZone=0 to disable the warning, see https://
↪stackoverflow.com/a/70209891/17679565
[2023-07-05 12:19:23.017][INFO][51047][78gw8v44] XCORE-SRS/6.0.55(Bee)
[2023-07-05 12:19:23.021][INFO][51047][78gw8v44] config parse complete
[2023-07-05 12:19:23.021][INFO][51047][78gw8v44] you can check log by: tail -n 30 -f ./
↪objs/srs.log
[2023-07-05 12:19:23.021][INFO][51047][78gw8v44] please check SRS by: ./etc/init.d/srs␣
↪status
```

To check the status of srs, use

```
./etc/init.d/srs status
```

which gives the following output:

```
SRS(pid 51548) is running.                                  [  OK  ]
```

Hint: If you fail to start the srs server, please check the log file `./objs/srs.log` for a fix.

Start ffmpeg to push audio stream

First, let us list available recording devices on the current computer with the following command:

```
ffmpeg -hide_banner -f avfoundation -list_devices true -i ""
```

It gives the following output on my computer:

```
[AVFoundation indev @ 0x7f9f41904840] AVFoundation video devices:
[AVFoundation indev @ 0x7f9f41904840] [0] FaceTime HD Camera (Built-in)
[AVFoundation indev @ 0x7f9f41904840] [1] Capture screen 0
[AVFoundation indev @ 0x7f9f41904840] AVFoundation audio devices:
[AVFoundation indev @ 0x7f9f41904840] [0] Background Music
[AVFoundation indev @ 0x7f9f41904840] [1] MacBook Pro Microphone
[AVFoundation indev @ 0x7f9f41904840] [2] Background Music (UI Sounds)
[AVFoundation indev @ 0x7f9f41904840] [3] WeMeet Audio Device
: Input/output error
```

We will use the device `[1] MacBook Pro Microphone`. Note that its index is `1`, so we will use `-i ":1"` in the following command to start recording and push the recorded audio stream to the server under the address `rtmp://localhost/live/livestream`.

Hint: The default TCP port for RTMP is `1935`.

```
ffmpeg -hide_banner -f avfoundation -i ":1" -acodec aac -ab 64k -ar 16000 -ac 1 -f flv␣
↪rtmp://localhost/live/livestream
```

The above command gives the following output:

```
Input #0, avfoundation, from ':1':
  Duration: N/A, start: 830938.803938, bitrate: 1536 kb/s
  Stream #0:0: Audio: pcm_f32le, 48000 Hz, mono, flt, 1536 kb/s
Stream mapping:
  Stream #0:0 -> #0:0 (pcm_f32le (native) -> aac (native))
Press [q] to stop, [?] for help
Output #0, flv, to 'rtmp://localhost/live/livestream':
  Metadata:
    encoder         : Lavf60.3.100
  Stream #0:0: Audio: aac (LC) ([10][0][0][0] / 0x000A), 16000 Hz, mono, fltp, 64 kb/s
    Metadata:
      encoder         : Lavc60.3.100 aac
size=      64kB time=00:00:08.39 bitrate=  62.3kbits/s speed=0.977x
```

Start sherpa-onnx to pull audio stream

Now we can start sherpa-onnx to pull audio stream from `rtmp://localhost/live/livestream` for speech recognition.

```
python3 ./python-api-examples/speech-recognition-from-url.py \
  --encoder ./sherpa-onnx-streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --decoder ./sherpa-onnx-streaming-zipformer-en-2023-06-26/decoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --joiner ./sherpa-onnx-streaming-zipformer-en-2023-06-26/joiner-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --tokens ./sherpa-onnx-streaming-zipformer-en-2023-06-26/tokens.txt \
  --url rtmp://localhost/live/livestream
```

You should see the recognition result printed to the console as you speak.

Hint: You can replace `localhost` with your server IP and start sherpa-onnx on many computers at the same time to pull audio stream from the address rtmp://your_server_ip/live/livestream.

8.4.5 Streaming WebSocket Server

This section describes how to use the Python streaming WebSocket server of sherpa-onnx for speech recognition.

Hint: The server supports multiple clients connecting at the same time.

The code for the streaming server can be found at

https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/streaming_server.py

Start the server

Hint:

If you don't use a X.509 certificate, due to security reasons imposed by the browser, you are only allowed to use the domain `localhost` to access the server if you want to access the microphone in the browser. That is, you can only use

http://localhost:port

to access the server. You cannot use http://0.0.0.0:port, or http://127.0.0.1:port, or http://public_ip:port.

You can use the following command to generate a self-signed certificate:

```
cd python-api-examples/web
./generate-certificate.py
```

The above commands will generate 3 files. You only need to use the file `cert.pem`. When starting the server, you pass the following argument:

```
--certificate=./python-api-examples/web/cert.pem
```

Please refer to *Pre-trained models* to download a streaming model before you continue.

We will use *csukuangfj/sherpa-onnx-streaming-zipformer-en-2023-06-26 (English)* as an example.

First, let us download it:

```
cd /path/to/sherpa-onnx/

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-en-2023-06-26.tar.bz2
tar xvf sherpa-onnx-streaming-zipformer-en-2023-06-26.tar.bz2
rm sherpa-onnx-streaming-zipformer-en-2023-06-26.tar.bz2
```

Now we can use:

```
cd /path/to/sherpa-onnx/

 python3 ./python-api-examples/streaming_server.py \
  --encoder ./sherpa-onnx-streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --decoder ./sherpa-onnx-streaming-zipformer-en-2023-06-26/decoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --joiner ./sherpa-onnx-streaming-zipformer-en-2023-06-26/joiner-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --tokens ./sherpa-onnx-streaming-zipformer-en-2023-06-26/tokens.txt \
  --port 6006
```

It will print the following logs:

```
2023-08-11 16:29:51,522 INFO [streaming_server.py:678] {'encoder': './sherpa-onnx-
↪streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-chunk-16-left-128.onnx',
↪'decoder': './sherpa-onnx-streaming-zipformer-en-2023-06-26/decoder-epoch-99-avg-1-
↪chunk-16-left-128.onnx', 'joiner': './sherpa-onnx-streaming-zipformer-en-2023-06-26/
↪joiner-epoch-99-avg-1-chunk-16-left-128.onnx', 'tokens': './sherpa-onnx-streaming-
↪zipformer-en-2023-06-26/tokens.txt', 'sample_rate': 16000, 'feat_dim': 80, 'provider':
↪'cpu', 'decoding_method': 'greedy_search', 'num_active_paths': 4, 'use_endpoint': 1,
↪'rule1_min_trailing_silence': 2.4, 'rule2_min_trailing_silence': 1.2, 'rule3_min_
↪utterance_length': 20, 'port': 6006, 'nn_pool_size': 1, 'max_batch_size': 50, 'max_
↪wait_ms': 10, 'max_message_size': 1048576, 'max_queue_size': 32, 'max_active_
↪connections': 500, 'num_threads': 2, 'certificate': None, 'doc_root': './python-api-
↪examples/web'}
2023-08-11 16:29:57,476 INFO [streaming_server.py:520] No certificate provided
2023-08-11 16:29:57,480 INFO [server.py:707] server listening on 0.0.0.0:6006
2023-08-11 16:29:57,480 INFO [server.py:707] server listening on [::]:6006
2023-08-11 16:29:57,480 INFO [streaming_server.py:546] Please visit one of the following␣
↪addresses:

  http://localhost:6006

Since you are not providing a certificate, you cannot use your microphone from within␣
↪the browser using public IP addresses. Only localhost can be used.You also cannot use␣
↪0.0.0.0 or 127.0.0.1
```

We can use the following two methods to interact with the server:

- Use Python API
- Use a browser by accessing http://localhost:6006

We describe each method below in details.

Use Python API

We provide two Python example files:

Description	URL
Send a file for decoding	https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/online-websocket-client-decode-file.py
Send audio samples from a microphone for decoding	https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/speech-recognition-from-microphone.py

Send a file for decoding

Hint: The example file supports only `*.wav` files with a single channel and the each sample should be of type `int16_t`. The sample rate does not need to be 16000 Hz, e.g., it can be 48000 Hz, 8000 Hz or some other value.

We use the following command to send a file for decoding:

```
cd /path/to/sherpa-onnx

python3 ./python-api-examples/online-websocket-client-decode-file.py \
  --server-addr localhost \
  --server-port 6006 \
  ./sherpa-onnx-streaming-zipformer-en-2023-06-26/test_wavs/0.wav
```

It should give the following output:

```
2023-08-11 16:37:03,877 INFO [online-websocket-client-decode-file.py:133] Sending ./
↪sherpa-onnx-streaming-zipformer-en-2023-06-26/test_wavs/0.wav
2023-08-11 16:37:03,931 INFO [online-websocket-client-decode-file.py:115] {"text": "",
↪"segment": 0}
2023-08-11 16:37:04,012 INFO [online-websocket-client-decode-file.py:115] {"text": "",
↪"segment": 0}
2023-08-11 16:37:04,128 INFO [online-websocket-client-decode-file.py:115] {"text": "AFTER
↪", "segment": 0}
2023-08-11 16:37:04,170 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY", "segment": 0}
2023-08-11 16:37:04,228 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY", "segment": 0}
2023-08-11 16:37:04,331 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFA", "segment": 0}
2023-08-11 16:37:04,373 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE", "segment": 0}
2023-08-11 16:37:04,433 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE YELLOW LA", "segment": 0}
2023-08-11 16:37:04,535 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE YELLOW LAMPS", "segment": 0}
```

(continues on next page)

(continued from previous page)

```
2023-08-11 16:37:04,576 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT", "segment": 0}
2023-08-11 16:37:04,645 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP", "segment": 0}
2023-08-11 16:37:04,685 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE", "segment": 0}
2023-08-11 16:37:04,755 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE", "segment": 0}
2023-08-11 16:37:04,847 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE", "segment": 0}
2023-08-11 16:37:04,887 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUA",
↪"segment": 0}
2023-08-11 16:37:04,958 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID",
↪"segment": 0}
2023-08-11 16:37:05,057 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUAR
↪", "segment": 0}
2023-08-11 16:37:05,095 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF", "segment": 0}
2023-08-11 16:37:05,164 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BRO", "segment": 0}
2023-08-11 16:37:05,268 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHEL", "segment": 0}
2023-08-11 16:37:05,369 INFO [online-websocket-client-decode-file.py:115] {"text":
↪"AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS", "segment": 0}
2023-08-11 16:37:05,370 INFO [online-websocket-client-decode-file.py:154]
Final result is:
{"text": "AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE␣
↪SQUALID QUARTER OF THE BROTHELS", "segment": 0}
```

Send audio samples from a microphone for decoding

We use the following command to run the script:

```
cd /path/to/sherpa-onnx

python3 ./python-api-examples/online-websocket-client-microphone.py \
  --server-addr localhost \
  --server-port 6006
```

It should give you the following output:

```
{'server_addr': 'localhost', 'server_port': 6006}
Started! Please Speak
  0 Background Music, Core Audio (2 in, 2 out)
```

(continues on next page)

(continued from previous page)

```
  1 Background Music (UI Sounds), Core Audio (2 in, 2 out)
> 2 MacBook Pro Microphone, Core Audio (1 in, 0 out)
< 3 MacBook Pro Speakers, Core Audio (0 in, 2 out)
  4 WeMeet Audio Device, Core Audio (2 in, 2 out)
Use default device: MacBook Pro Microphone

Started! Please speak
```

If you speak, you will see the recognition result returned by the server.

Use a browser

Start your browser and visit the following address:

http://localhost:6006/

You should see a page like below:

Click `Streaming-Record` and you will see the following page:

Click the button `Click me to connect` to connect to the server and then you can click the `Streaming-Record` button to start recording. You should see the decoded results as you speak.

colab

We provide a colab notebook for you to try the Python streaming websocket server example of sherpa-onnx.

8.4.6 Non-Streaming WebSocket Server

This section describes how to use the Python non-streaming WebSocket server of sherpa-onnx for speech recognition.

Hint: The server supports multiple clients connecting at the same time.

The code for the non-streaming server can be found at

https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/non_streaming_server.py

Please refer to *Pre-trained models* to download a non-streaming model before you continue.

We use the following types of models for demonstration.

Description	URL
Non-streaming transducer	*csukuangfj/sherpa-onnx-zipformer-en-2023-06-26 (English)*
Non-streaming paraformer	*csukuangfj/sherpa-onnx-paraformer-zh-2023-03-28 (Chinese + English)*
Non-streaming CTC model from NeMo	*stt_en_conformer_ctc_medium*
Non-streaming Whisper tiny.en	*tiny.en*

Non-streaming transducer

Start the server

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-en-2023-06-26.tar.bz2
tar xvf sherpa-onnx-zipformer-en-2023-06-26.tar.bz2
rm sherpa-onnx-zipformer-en-2023-06-26.tar.bz2

python3 ./python-api-examples/non_streaming_server.py \
  --encoder ./sherpa-onnx-zipformer-en-2023-06-26/encoder-epoch-99-avg-1.onnx \
  --decoder ./sherpa-onnx-zipformer-en-2023-06-26/decoder-epoch-99-avg-1.onnx \
  --joiner ./sherpa-onnx-zipformer-en-2023-06-26/joiner-epoch-99-avg-1.onnx \
  --tokens ./sherpa-onnx-zipformer-en-2023-06-26/tokens.txt \
  --port 6006
```

Start the client

Decode multiple files in parallel

```
python3 ./python-api-examples/offline-websocket-client-decode-files-paralell.py \
  --server-addr localhost \
  --server-port 6006 \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/8k.wav
```

You should see the following output:

```
2023-08-11 18:19:26,000 INFO [offline-websocket-client-decode-files-paralell.py:139] {
↪'server_addr': 'localhost', 'server_port': 6006, 'sound_files': ['./sherpa-onnx-
↪zipformer-en-2023-06-26/test_wavs/0.wav', './sherpa-onnx-zipformer-en-2023-06-26/test_
↪wavs/1.wav', './sherpa-onnx-zipformer-en-2023-06-26/test_wavs/8k.wav']}
2023-08-11 18:19:26,034 INFO [offline-websocket-client-decode-files-paralell.py:113]␣
↪Sending ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/8k.wav
2023-08-11 18:19:26,058 INFO [offline-websocket-client-decode-files-paralell.py:113]␣
↪Sending ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/1.wav
2023-08-11 18:19:26,205 INFO [offline-websocket-client-decode-files-paralell.py:113]␣
↪Sending ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/0.wav
2023-08-11 18:19:26,262 INFO [offline-websocket-client-decode-files-paralell.py:131] ./
↪sherpa-onnx-zipformer-en-2023-06-26/test_wavs/8k.wav
 YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION
2023-08-11 18:19:26,609 INFO [offline-websocket-client-decode-files-paralell.py:131] ./
↪sherpa-onnx-zipformer-en-2023-06-26/test_wavs/1.wav
 GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY␣
↪CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT FOREVER WITH␣
↪THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
2023-08-11 18:19:26,773 INFO [offline-websocket-client-decode-files-paralell.py:131] ./
↪sherpa-onnx-zipformer-en-2023-06-26/test_wavs/0.wav
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
```

(continues on next page)

(continued from previous page)

Decode multiple files sequentially

```
python3 ./python-api-examples/offline-websocket-client-decode-files-sequential.py \
  --server-addr localhost \
  --server-port 6006 \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/8k.wav
```

You should see the following output:

```
2023-08-11 18:20:36,677 INFO [offline-websocket-client-decode-files-sequential.py:114]␣
↪Sending ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/0.wav
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
2023-08-11 18:20:36,861 INFO [offline-websocket-client-decode-files-sequential.py:114]␣
↪Sending ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/1.wav
 GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY␣
↪CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT FOREVER WITH␣
↪THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
2023-08-11 18:20:37,375 INFO [offline-websocket-client-decode-files-sequential.py:114]␣
↪Sending ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/8k.wav
 YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION
```

Non-streaming paraformer

Start the server

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-zh-2023-03-28.tar.bz2
tar xvf sherpa-onnx-paraformer-zh-2023-03-28.tar.bz2
rm sherpa-onnx-paraformer-zh-2023-03-28.tar.bz2

python3 ./python-api-examples/non_streaming_server.py \
  --paraformer ./sherpa-onnx-paraformer-zh-2023-03-28/model.int8.onnx \
  --tokens ./sherpa-onnx-paraformer-zh-2023-03-28/tokens.txt \
  --port 6006
```

Start the client

Decode multiple files in parallel

```
python3 ./python-api-examples/offline-websocket-client-decode-files-paralell.py \
  --server-addr localhost \
  --server-port 6006 \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav
```

You should see the following output:

```
2023-08-11 18:22:54,189 INFO [offline-websocket-client-decode-files-paralell.py:113]␣
↪Sending ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav
2023-08-11 18:22:54,233 INFO [offline-websocket-client-decode-files-paralell.py:113]␣
↪Sending ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav
2023-08-11 18:22:54,275 INFO [offline-websocket-client-decode-files-paralell.py:113]␣
↪Sending ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav
2023-08-11 18:22:54,295 INFO [offline-websocket-client-decode-files-paralell.py:113]␣
↪Sending ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav
2023-08-11 18:22:54,380 INFO [offline-websocket-client-decode-files-paralell.py:131] ./
↪sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav

2023-08-11 18:22:54,673 INFO [offline-websocket-client-decode-files-paralell.py:131] ./
↪sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav

2023-08-11 18:22:54,673 INFO [offline-websocket-client-decode-files-paralell.py:131] ./
↪sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav

2023-08-11 18:22:54,674 INFO [offline-websocket-client-decode-files-paralell.py:131] ./
↪sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav
```

Decode multiple files sequentially

```
python3 ./python-api-examples/offline-websocket-client-decode-files-sequential.py \
  --server-addr localhost \
  --server-port 6006 \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav
```

You should see the following output:

```
2023-08-11 18:24:32,678 INFO [offline-websocket-client-decode-files-sequential.py:141] {
↪'server_addr': 'localhost', 'server_port': 6006, 'sound_files': ['./sherpa-onnx-
↪paraformer-zh-2023-03-28/test_wavs/0.wav', './sherpa-onnx-paraformer-zh-2023-03-28/
↪test_wavs/1.wav', './sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav', './sherpa-
↪onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav']}
2023-08-11 18:24:32,709 INFO [offline-websocket-client-decode-files-sequential.py:114]␣
↪Sending ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav
```

(continues on next page)

(continued from previous page)

```
2023-08-11 18:24:32,883 INFO [offline-websocket-client-decode-files-sequential.py:114]␣
↪Sending ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav

2023-08-11 18:24:33,042 INFO [offline-websocket-client-decode-files-sequential.py:114]␣
↪Sending ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav

2023-08-11 18:24:33,175 INFO [offline-websocket-client-decode-files-sequential.py:114]␣
↪Sending ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav
```

Non-streaming CTC model from NeMo

Start the server

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-nemo-
↪ctc-en-conformer-medium.tar.bz2
tar xvf sherpa-onnx-nemo-ctc-en-conformer-medium.tar.bz2
rm sherpa-onnx-nemo-ctc-en-conformer-medium.tar.bz2

python3 ./python-api-examples/non_streaming_server.py \
  --nemo-ctc ./sherpa-onnx-nemo-ctc-en-conformer-medium/model.onnx \
  --tokens ./sherpa-onnx-nemo-ctc-en-conformer-medium/tokens.txt \
  --port 6006
```

Start the client

Decode multiple files in parallel

```
python3 ./python-api-examples/offline-websocket-client-decode-files-paralell.py \
  --server-addr localhost \
  --server-port 6006 \
  ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/0.wav \
  ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/1.wav \
  ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/8k.wav
```

You should see the following output:

```
2023-08-11 18:31:32,432 INFO [offline-websocket-client-decode-files-paralell.py:139] {
↪'server_addr': 'localhost', 'server_port': 6006, 'sound_files': ['./sherpa-onnx-nemo-
↪ctc-en-conformer-medium/test_wavs/0.wav', './sherpa-onnx-nemo-ctc-en-conformer-medium/
↪test_wavs/1.wav', './sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/8k.wav']}
2023-08-11 18:31:32,462 INFO [offline-websocket-client-decode-files-paralell.py:113]␣
↪Sending ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/0.wav
2023-08-11 18:31:32,513 INFO [offline-websocket-client-decode-files-paralell.py:113]␣
↪Sending ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/8k.wav
2023-08-11 18:31:32,533 INFO [offline-websocket-client-decode-files-paralell.py:113]␣
↪Sending ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/1.wav
```

(continues on next page)

(continued from previous page)

```
2023-08-11 18:31:32,670 INFO [offline-websocket-client-decode-files-paralell.py:131] ./
↪sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/0.wav
 after early nightfall the yellow lamps would light up here and there the squalid␣
↪quarter of the brothels
2023-08-11 18:31:32,741 INFO [offline-websocket-client-decode-files-paralell.py:131] ./
↪sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/8k.wav
 yet these thoughts affected hester pryne less with hope than apprehension
2023-08-11 18:31:33,117 INFO [offline-websocket-client-decode-files-paralell.py:131] ./
↪sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/1.wav
 god as a direct consequence of the sin which man thus punished had given her a lovely␣
↪child whose place was on that same dishonored bosom to connect her parent for ever␣
↪with the race and descent of mortals and to be finally a blessed soul in heaven
```

Decode multiple files sequentially

```
python3 ./python-api-examples/offline-websocket-client-decode-files-sequential.py \
  --server-addr localhost \
  --server-port 6006 \
  ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/0.wav \
  ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/1.wav \
  ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/8k.wav
```

You should see the following output:

```
2023-08-11 18:33:14,520 INFO [offline-websocket-client-decode-files-sequential.py:141] {
↪'server_addr': 'localhost', 'server_port': 6006, 'sound_files': ['./sherpa-onnx-nemo-
↪ctc-en-conformer-medium/test_wavs/0.wav', './sherpa-onnx-nemo-ctc-en-conformer-medium/
↪test_wavs/1.wav', './sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/8k.wav']}
2023-08-11 18:33:14,547 INFO [offline-websocket-client-decode-files-sequential.py:114]␣
↪Sending ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/0.wav
 after early nightfall the yellow lamps would light up here and there the squalid␣
↪quarter of the brothels
2023-08-11 18:33:14,716 INFO [offline-websocket-client-decode-files-sequential.py:114]␣
↪Sending ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/1.wav
 god as a direct consequence of the sin which man thus punished had given her a lovely␣
↪child whose place was on that same dishonored bosom to connect her parent for ever␣
↪with the race and descent of mortals and to be finally a blessed soul in heaven
2023-08-11 18:33:15,218 INFO [offline-websocket-client-decode-files-sequential.py:114]␣
↪Sending ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/8k.wav
 yet these thoughts affected hester pryne less with hope than apprehension
```

Non-streaming Whisper tiny.en

Start the server

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪whisper-tiny.en.tar.bz2
tar xvf sherpa-onnx-whisper-tiny.en.tar.bz2
rm sherpa-onnx-whisper-tiny.en.tar.bz2
```

(continues on next page)

(continued from previous page)

```
python3 ./python-api-examples/non_streaming_server.py \
  --whisper-encoder=./sherpa-onnx-whisper-tiny.en/tiny.en-encoder.onnx \
  --whisper-decoder=./sherpa-onnx-whisper-tiny.en/tiny.en-decoder.onnx \
  --tokens=./sherpa-onnx-whisper-tiny.en/tiny.en-tokens.txt \
  --port 6006
```

Start the client

Decode multiple files in parallel

```
python3 ./python-api-examples/offline-websocket-client-decode-files-paralell.py \
  --server-addr localhost \
  --server-port 6006 \
  ./sherpa-onnx-whisper-tiny.en/test_wavs/0.wav \
  ./sherpa-onnx-whisper-tiny.en/test_wavs/1.wav \
  ./sherpa-onnx-whisper-tiny.en/test_wavs/8k.wav
```

You should see the following output:

```
2023-08-11 18:35:28,866 INFO [offline-websocket-client-decode-files-paralell.py:139] {
↪'server_addr': 'localhost', 'server_port': 6006, 'sound_files': ['./sherpa-onnx-
↪whisper-tiny.en/test_wavs/0.wav', './sherpa-onnx-whisper-tiny.en/test_wavs/1.wav', './
↪sherpa-onnx-whisper-tiny.en/test_wavs/8k.wav']}
2023-08-11 18:35:28,894 INFO [offline-websocket-client-decode-files-paralell.py:113]␣
↪Sending ./sherpa-onnx-whisper-tiny.en/test_wavs/0.wav
2023-08-11 18:35:28,947 INFO [offline-websocket-client-decode-files-paralell.py:113]␣
↪Sending ./sherpa-onnx-whisper-tiny.en/test_wavs/1.wav
2023-08-11 18:35:29,082 INFO [offline-websocket-client-decode-files-paralell.py:113]␣
↪Sending ./sherpa-onnx-whisper-tiny.en/test_wavs/8k.wav
2023-08-11 18:35:29,754 INFO [offline-websocket-client-decode-files-paralell.py:131] ./
↪sherpa-onnx-whisper-tiny.en/test_wavs/0.wav
 After early nightfall, the yellow lamps would light up here and there, the squalid␣
↪quarter of the brothels.
2023-08-11 18:35:30,276 INFO [offline-websocket-client-decode-files-paralell.py:131] ./
↪sherpa-onnx-whisper-tiny.en/test_wavs/8k.wav
 Yet these thoughts affected Hester Prin less with hope than apprehension.
2023-08-11 18:35:31,592 INFO [offline-websocket-client-decode-files-paralell.py:131] ./
↪sherpa-onnx-whisper-tiny.en/test_wavs/1.wav
 God, as a direct consequence of the sin which man thus punished, had given her a lovely␣
↪child, whose place was on that same dishonored bosom to connect her parent forever␣
↪with the race and descent of mortals, and to be finally a blessed soul in heaven.
```

Decode multiple files sequentially

```
python3 ./python-api-examples/offline-websocket-client-decode-files-sequential.py \
  --server-addr localhost \
  --server-port 6006 \
  ./sherpa-onnx-whisper-tiny.en/test_wavs/0.wav \
  ./sherpa-onnx-whisper-tiny.en/test_wavs/1.wav \
  ./sherpa-onnx-whisper-tiny.en/test_wavs/8k.wav
```

You should see the following output:

```
2023-08-11 18:36:42,148 INFO [offline-websocket-client-decode-files-sequential.py:141] {
↪'server_addr': 'localhost', 'server_port': 6006, 'sound_files': ['./sherpa-onnx-
↪whisper-tiny.en/test_wavs/0.wav', './sherpa-onnx-whisper-tiny.en/test_wavs/1.wav', './
↪sherpa-onnx-whisper-tiny.en/test_wavs/8k.wav']}
2023-08-11 18:36:42,176 INFO [offline-websocket-client-decode-files-sequential.py:114]␣
↪Sending ./sherpa-onnx-whisper-tiny.en/test_wavs/0.wav
 After early nightfall, the yellow lamps would light up here and there, the squalid␣
↪quarter of the brothels.
2023-08-11 18:36:42,926 INFO [offline-websocket-client-decode-files-sequential.py:114]␣
↪Sending ./sherpa-onnx-whisper-tiny.en/test_wavs/1.wav
 God, as a direct consequence of the sin which man thus punished, had given her a lovely␣
↪child, whose place was on that same dishonored bosom to connect her parent forever␣
↪with the race and descent of mortals, and to be finally a blessed soul in heaven.
2023-08-11 18:36:44,314 INFO [offline-websocket-client-decode-files-sequential.py:114]␣
↪Sending ./sherpa-onnx-whisper-tiny.en/test_wavs/8k.wav
 Yet these thoughts affected Hester Prin less with hope than apprehension.
```

colab

We provide a colab notebook for you to try the Python non-streaming websocket server example of sherpa-onnx.

8.5 C API

In this section, we describe how to use the C API of sherpa-onnx.

Specifically, we will describe:

- How to generate required files
- How to use `pkg-config` with sherpa-onnx

You can find the implementation at

- https://github.com/k2-fsa/sherpa-onnx/blob/master/sherpa-onnx/c-api/c-api.h
- https://github.com/k2-fsa/sherpa-onnx/blob/master/sherpa-onnx/c-api/c-api.cc

8.5.1 Generate required files

Before using the C API of sherpa-onnx, we need to first build required libraries. You can choose either to build static libraries or shared libraries.

Build shared libraries

Assume that we want to put library files and header files in the directory `/tmp/sherpa-onnx/shared`:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
mkdir build-shared
cd build-shared

cmake \
  -DSHERPA_ONNX_ENABLE_C_API=ON \
  -DCMAKE_BUILD_TYPE=Release \
  -DBUILD_SHARED_LIBS=ON \
  -DCMAKE_INSTALL_PREFIX=/tmp/sherpa-onnx/shared \
  ..

make -j6
make install
```

You should find the following files inside `/tmp/sherpa-onnx/shared`:

macOS

Linux

```
$ tree /tmp/sherpa-onnx/shared/

/tmp/sherpa-onnx/shared/
├── bin
│   ├── sherpa-onnx
│   ├── sherpa-onnx-microphone
│   ├── sherpa-onnx-microphone-offline
│   ├── sherpa-onnx-offline
│   ├── sherpa-onnx-offline-websocket-server
│   ├── sherpa-onnx-online-websocket-client
│   └── sherpa-onnx-online-websocket-server
├── include
│   ├── cargs.h
│   └── sherpa-onnx
│       └── c-api
│           └── c-api.h
├── lib
│   ├── cargs.h
│   ├── libcargs.dylib
│   ├── libkaldi-native-fbank-core.dylib
│   ├── libonnxruntime.1.15.1.dylib
│   ├── libonnxruntime.dylib -> libonnxruntime.1.15.1.dylib
│   ├── libsherpa-onnx-c-api.dylib
│   ├── libsherpa-onnx-core.dylib
│   └── libsherpa-onnx-portaudio.dylib
└── sherpa-onnx.pc

5 directories, 18 files
```

```
$ tree /tmp/sherpa-onnx/shared/

/tmp/sherpa-onnx/shared/
|-- bin
|   |-- sherpa-onnx
|   |-- sherpa-onnx-alsa
|   |-- sherpa-onnx-microphone
|   |-- sherpa-onnx-microphone-offline
|   |-- sherpa-onnx-offline
|   |-- sherpa-onnx-offline-websocket-server
|   |-- sherpa-onnx-online-websocket-client
|   `-- sherpa-onnx-online-websocket-server
|-- include
|   |-- cargs.h
|   `-- sherpa-onnx
|       `-- c-api
|           `-- c-api.h
|-- lib
|   |-- cargs.h
|   |-- libcargs.so
|   |-- libkaldi-native-fbank-core.so
|   |-- libonnxruntime.so -> libonnxruntime.so.1.15.1
|   |-- libonnxruntime.so.1.15.1
|   |-- libsherpa-onnx-c-api.so
|   |-- libsherpa-onnx-core.so
|   `-- libsherpa-onnx-portaudio.so
`-- sherpa-onnx.pc

5 directories, 19 files
```

Build static libraries

Assume that we want to put library files and header files in the directory `/tmp/sherpa-onnx/static`:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
mkdir build-static
cd build-static

cmake \
  -DSHERPA_ONNX_ENABLE_C_API=ON \
  -DCMAKE_BUILD_TYPE=Release \
  -DBUILD_SHARED_LIBS=OFF \
  -DCMAKE_INSTALL_PREFIX=/tmp/sherpa-onnx/static \
  ..

make -j6
make install
```

You should find the following files in `/tmp/sherpa-onnx/static`:

macOS

Linux

```
$ tree /tmp/sherpa-onnx/static/

/tmp/sherpa-onnx//static/
├── bin
│   ├── sherpa-onnx
│   ├── sherpa-onnx-microphone
│   ├── sherpa-onnx-microphone-offline
│   ├── sherpa-onnx-offline
│   ├── sherpa-onnx-offline-websocket-server
│   ├── sherpa-onnx-online-websocket-client
│   └── sherpa-onnx-online-websocket-server
├── include
│   ├── cargs.h
│   └── sherpa-onnx
│       └── c-api
│           └── c-api.h
├── lib
│   ├── cargs.h
│   ├── libcargs.a
│   ├── libkaldi-native-fbank-core.a
│   ├── libonnxruntime.1.15.1.dylib
│   ├── libonnxruntime.dylib -> libonnxruntime.1.15.1.dylib
│   ├── libsherpa-onnx-c-api.a
│   ├── libsherpa-onnx-core.a
│   └── libsherpa-onnx-portaudio_static.a
└── sherpa-onnx.pc

5 directories, 18 files
```

```
$ tree /tmp/sherpa-onnx/static/

/tmp/sherpa-onnx/static/
|-- bin
|   |-- sherpa-onnx
|   |-- sherpa-onnx-alsa
|   |-- sherpa-onnx-microphone
|   |-- sherpa-onnx-microphone-offline
|   |-- sherpa-onnx-offline
|   |-- sherpa-onnx-offline-websocket-server
|   |-- sherpa-onnx-online-websocket-client
|   `-- sherpa-onnx-online-websocket-server
|-- include
|   |-- cargs.h
|   `-- sherpa-onnx
|       `-- c-api
|           `-- c-api.h
|-- lib
|   |-- cargs.h
|   |-- libcargs.a
|   |-- libkaldi-native-fbank-core.a
|   |-- libonnxruntime.so -> libonnxruntime.so.1.15.1
```

(continues on next page)

(continued from previous page)

```
|   |-- libonnxruntime.so.1.15.1
|   |-- libsherpa-onnx-c-api.a
|   |-- libsherpa-onnx-core.a
|   `-- libsherpa-onnx-portaudio_static.a
`-- sherpa-onnx.pc

5 directories, 19 files
```

8.5.2 Build decode-file-c-api.c with generated files

To build the following file:

https://github.com/k2-fsa/sherpa-onnx/blob/master/c-api-examples/decode-file-c-api.c

We can use:

static link

dynamic link

```
export PKG_CONFIG_PATH=/tmp/sherpa-onnx/static:$PKG_CONFIG_PATH

cd ./c-api-examples
gcc -o decode-file-c-api $(pkg-config --cflags sherpa-onnx) ./decode-file-c-api.c $(pkg-
↪config --libs sherpa-onnx)

./decode-file-c-api --help
```

```
export PKG_CONFIG_PATH=/tmp/sherpa-onnx/shared:$PKG_CONFIG_PATH

cd ./c-api-examples
gcc -o decode-file-c-api $(pkg-config --cflags sherpa-onnx) ./decode-file-c-api.c $(pkg-
↪config --libs sherpa-onnx)

./decode-file-c-api --help
```

8.5.3 colab

We provide a colab notebook for you to try the C API of sherpa-onnx.

8.6 Java API

In this section, we describe how to use the `Java` API of sherpa-onnx.

The core part of sherpa-onnx is written in C++. We have provided JNI interface for sherpa-onnx so that you can use it in Java.

Before using the Java API of sherpa-onnx, you have to build the `JNI` interface.

Hint: We provide pre-built `JNI` libs. Please see

https://huggingface.co/csukuangfj/sherpa-onnx-libs/tree/main/jni

For Chinese users, please use

https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/tree/main/jni

8.6.1 Build JNI interface (macOS)

In the following, we describe how to build the JNI interface for macOS. It is applicable for both macOS x64 and arm64.

For Linux users, please refer to *Build JNI interface (Linux)*

Hint: For Windows users, you have to modify the commands by yourself.

Setup the environment

Make sure you have the following two items ready:

- a working C/C++ compiler that supports C++17
- you are able to run `java` and `javac` commands in your terminal.

To check your environment, please run:

```
gcc --version
java -version
javac -version
```

The above three commands print the following output on my computer. You don't need to use the exact versions as I am using.

```
# output of gcc --version

Apple clang version 14.0.0 (clang-1400.0.29.202)
Target: x86_64-apple-darwin22.2.0
Thread model: posix
InstalledDir: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.
↪xctoolchain/usr/bin

# output of java -version

openjdk version "19.0.1" 2022-10-18
OpenJDK Runtime Environment (build 19.0.1+10-21)
OpenJDK 64-Bit Server VM (build 19.0.1+10-21, mixed mode, sharing)

# output of javac -version

javac 19.0.1
```

Build sherpa-onnx

Please use the following commands to build sherpa-onnx:

```
git clone https://github.com/k2-fsa/sherpa-onnx

cd sherpa-onnx

mkdir build

cd build

cmake \
  -DSHERPA_ONNX_ENABLE_PYTHON=OFF \
  -DSHERPA_ONNX_ENABLE_TESTS=OFF \
  -DSHERPA_ONNX_ENABLE_CHECK=OFF \
  -DBUILD_SHARED_LIBS=ON \
  -DSHERPA_ONNX_ENABLE_PORTAUDIO=OFF \
  -DSHERPA_ONNX_ENABLE_JNI=ON \
  ..

make -j4

ls -lh lib
```

You should see the following output for `ls -lh lib`:

```
total 11576
-rwxr-xr-x  1 fangjun  staff    33K May 15 11:24 libcargs.dylib
-rwxr-xr-x  1 fangjun  staff   350K May 15 11:24 libespeak-ng.dylib
-rwxr-xr-x  1 fangjun  staff   471K May 15 11:24 libkaldi-decoder-core.dylib
-rwxr-xr-x  1 fangjun  staff   113K May 15 11:24 libkaldi-native-fbank-core.dylib
-rwxr-xr-x  1 fangjun  staff   423K May 15 11:24 libpiper_phonemize.1.2.0.dylib
lrwxr-xr-x  1 fangjun  staff    30B May 15 11:24 libpiper_phonemize.1.dylib -> libpiper_
↪phonemize.1.2.0.dylib
lrwxr-xr-x  1 fangjun  staff    26B May 15 11:24 libpiper_phonemize.dylib -> libpiper_
↪phonemize.1.dylib
-rwxr-xr-x  1 fangjun  staff   113K May 15 11:25 libsherpa-onnx-c-api.dylib
-rwxr-xr-x  1 fangjun  staff   1.8M May 15 11:24 libsherpa-onnx-core.dylib
-rwxr-xr-x  1 fangjun  staff   1.5M May 15 11:24 libsherpa-onnx-fst.6.dylib
lrwxr-xr-x  1 fangjun  staff    26B May 15 11:24 libsherpa-onnx-fst.dylib -> libsherpa-
↪onnx-fst.6.dylib
-rwxr-xr-x  1 fangjun  staff    34K May 15 11:24 libsherpa-onnx-fstfar.7.dylib
lrwxr-xr-x  1 fangjun  staff    29B May 15 11:24 libsherpa-onnx-fstfar.dylib ->␣
↪libsherpa-onnx-fstfar.7.dylib
-rwxr-xr-x  1 fangjun  staff   127K May 15 11:25 libsherpa-onnx-jni.dylib
-rwxr-xr-x  1 fangjun  staff   933K May 15 11:24 libsherpa-onnx-kaldifst-core.dylib
-rwxr-xr-x  1 fangjun  staff   187K May 15 11:24 libucd.dylib
```

Note that all these `*.dylib` files are required.

`libsherpa-onnx-jni.dylib` contains the JNI interface for sherpa-onnx.

Download pre-built JNI libs

If you don't want to build `JNI` libs by yourself, please download pre-built `JNI` libs from

https://huggingface.co/csukuangfj/sherpa-onnx-libs/tree/main/jni

For Chinese users, please use

https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/tree/main/jni

Please always use the latest version. In the following, we describe how to download the version `1.10.2`.

Intel CPU (x86_64)

Apple Silicon (arm64)

```
wget https://huggingface.co/csukuangfj/sherpa-onnx-libs/resolve/main/jni/sherpa-onnx-v1.
↪10.2-osx-x86_64-jni.tar.bz2

# For Chinese users
# wget https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/resolve/main/jni/sherpa-onnx-v1.
↪10.2-osx-x86_64-jni.tar.bz2

tar xf sherpa-onnx-v1.10.2-osx-x86_64-jni.tar.bz2
rm sherpa-onnx-v1.10.2-osx-x86_64-jni.tar.bz2
```

```
wget https://huggingface.co/csukuangfj/sherpa-onnx-libs/resolve/main/jni/sherpa-onnx-v1.
↪10.2-osx-arm64-jni.tar.bz2

# For Chinese users
# wget https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/resolve/main/jni/sherpa-onnx-v1.
↪10.2-osx-arm64-jni.tar.bz2

tar xf sherpa-onnx-v1.10.2-osx-arm64-jni.tar.bz2
rm sherpa-onnx-v1.10.2-osx-arm64-jni.tar.bz2
```

After downloading, you should see the following files:

```
# For x86_64
ls -lh sherpa-onnx-v1.10.2-osx-x86_64-jni/lib/

-rwxr-xr-x  1 fangjun  staff    33K Jun 25 11:39 libcargs.dylib
-rwxr-xr-x  1 fangjun  staff   335K Jun 25 11:39 libespeak-ng.dylib
-rwxr-xr-x  1 fangjun  staff   484K Jun 25 11:39 libkaldi-decoder-core.dylib
-rwxr-xr-x  1 fangjun  staff   121K Jun 25 11:39 libkaldi-native-fbank-core.dylib
-rw-r--r--  1 fangjun  staff    25M Jun 25 11:39 libonnxruntime.1.17.1.dylib
lrwxr-xr-x  1 fangjun  staff    27B Jun 25 11:39 libonnxruntime.dylib -> libonnxruntime.
↪1.17.1.dylib
-rwxr-xr-x  1 fangjun  staff   427K Jun 25 11:39 libpiper_phonemize.1.2.0.dylib
lrwxr-xr-x  1 fangjun  staff    30B Jun 25 11:39 libpiper_phonemize.1.dylib -> libpiper_
↪phonemize.1.2.0.dylib
lrwxr-xr-x  1 fangjun  staff    26B Jun 25 11:39 libpiper_phonemize.dylib -> libpiper_
↪phonemize.1.dylib
-rwxr-xr-x  1 fangjun  staff   115K Jun 25 11:39 libsherpa-onnx-c-api.dylib
-rwxr-xr-x  1 fangjun  staff   1.9M Jun 25 11:39 libsherpa-onnx-core.dylib
-rwxr-xr-x  1 fangjun  staff   1.5M Jun 25 11:39 libsherpa-onnx-fst.dylib
```

(continues on next page)

(continued from previous page)

```
-rwxr-xr-x  1 fangjun  staff    40K Jun 25 11:39 libsherpa-onnx-fstfar.dylib
-rwxr-xr-x  1 fangjun  staff   134K Jun 25 11:39 libsherpa-onnx-jni.dylib
-rwxr-xr-x  1 fangjun  staff   963K Jun 25 11:39 libsherpa-onnx-kaldifst-core.dylib
-rwxr-xr-x  1 fangjun  staff   117K Jun 25 11:39 libsherpa-onnx-portaudio.dylib
-rwxr-xr-x  1 fangjun  staff   150K Jun 25 11:39 libssentencepiece_core.dylib
-rwxr-xr-x  1 fangjun  staff   187K Jun 25 11:39 libucd.dylib

# For arm64
ls -lh sherpa-onnx-v1.10.2-osx-arm64-jni/lib/

-rwxr-xr-x  1 fangjun  staff    49K Jun 25 11:39 libcargs.dylib
-rwxr-xr-x  1 fangjun  staff   318K Jun 25 11:39 libespeak-ng.dylib
-rwxr-xr-x  1 fangjun  staff   436K Jun 25 11:39 libkaldi-decoder-core.dylib
-rwxr-xr-x  1 fangjun  staff   122K Jun 25 11:39 libkaldi-native-fbank-core.dylib
-rw-r--r--  1 fangjun  staff    23M Jun 25 11:39 libonnxruntime.1.17.1.dylib
lrwxr-xr-x  1 fangjun  staff    27B Jun 25 11:39 libonnxruntime.dylib -> libonnxruntime.
↪1.17.1.dylib
-rwxr-xr-x  1 fangjun  staff   430K Jun 25 11:39 libpiper_phonemize.1.2.0.dylib
lrwxr-xr-x  1 fangjun  staff    30B Jun 25 11:39 libpiper_phonemize.1.dylib -> libpiper_
↪phonemize.1.2.0.dylib
lrwxr-xr-x  1 fangjun  staff    26B Jun 25 11:39 libpiper_phonemize.dylib -> libpiper_
↪phonemize.1.dylib
-rwxr-xr-x  1 fangjun  staff   132K Jun 25 11:39 libsherpa-onnx-c-api.dylib
-rwxr-xr-x  1 fangjun  staff   1.7M Jun 25 11:39 libsherpa-onnx-core.dylib
-rwxr-xr-x  1 fangjun  staff   1.4M Jun 25 11:39 libsherpa-onnx-fst.dylib
-rwxr-xr-x  1 fangjun  staff    56K Jun 25 11:39 libsherpa-onnx-fstfar.dylib
-rwxr-xr-x  1 fangjun  staff   151K Jun 25 11:39 libsherpa-onnx-jni.dylib
-rwxr-xr-x  1 fangjun  staff   859K Jun 25 11:39 libsherpa-onnx-kaldifst-core.dylib
-rwxr-xr-x  1 fangjun  staff   118K Jun 25 11:39 libsherpa-onnx-portaudio.dylib
-rwxr-xr-x  1 fangjun  staff   149K Jun 25 11:39 libssentencepiece_core.dylib
-rwxr-xr-x  1 fangjun  staff   188K Jun 25 11:39 libucd.dylib
```

8.6.2 Build JNI interface (Linux)

In the following, we describe how to build the JNI interface for Linux. It is applicable for both Linux x64 and arm64.

For macOS users, please refer to *Build JNI interface (macOS)*

Hint: For Windows users, you have to modify the commands by yourself.

Setup the environment

Make sure you have the following two items ready:

- a working C/C++ compiler that supports C++17
- you are able to run `java` and `javac` commands in your terminal.

To check your environment, please run:

```
gcc --version
java -version
javac -version
```

The above three commands print the following output on my computer. You don't need to use the exact versions as I am using.

```
# output of gcc --version

gcc (Ubuntu 12.3.0-1ubuntu1~23.04) 12.3.0
Copyright (C) 2022 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

# output of java -version

java version "17.0.11" 2024-04-16 LTS
Java(TM) SE Runtime Environment (build 17.0.11+7-LTS-207)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.11+7-LTS-207, mixed mode, sharing)

# output of javac -version

javac 17.0.11
```

Build sherpa-onnx

Please use the following commands to build sherpa-onnx:

```
git clone https://github.com/k2-fsa/sherpa-onnx

cd sherpa-onnx

mkdir build

cd build

cmake \
  -DSHERPA_ONNX_ENABLE_PYTHON=OFF \
  -DSHERPA_ONNX_ENABLE_TESTS=OFF \
  -DSHERPA_ONNX_ENABLE_CHECK=OFF \
  -DBUILD_SHARED_LIBS=ON \
  -DSHERPA_ONNX_ENABLE_PORTAUDIO=OFF \
  -DSHERPA_ONNX_ENABLE_JNI=ON \
  ..
```

(continues on next page)

(continued from previous page)

```
make -j4

ls -lh lib
```

You should see the following output for `ls -lh lib`:

```
total 7.7M
-rwxrwxr-x 1 fangjun fangjun  16K May 15 03:53 libcargs.so
-rwxrwxr-x 1 fangjun fangjun 396K May 15 03:53 libespeak-ng.so
-rwxrwxr-x 1 fangjun fangjun 652K May 15 03:53 libkaldi-decoder-core.so
-rwxrwxr-x 1 fangjun fangjun 108K May 15 03:53 libkaldi-native-fbank-core.so
lrwxrwxrwx 1 fangjun fangjun   23 May 15 03:53 libpiper_phonemize.so -> libpiper_
↪phonemize.so.1
lrwxrwxrwx 1 fangjun fangjun   27 May 15 03:53 libpiper_phonemize.so.1 -> libpiper_
↪phonemize.so.1.2.0
-rwxrwxr-x 1 fangjun fangjun 450K May 15 03:53 libpiper_phonemize.so.1.2.0
-rwxrwxr-x 1 fangjun fangjun 107K May 15 03:55 libsherpa-onnx-c-api.so
-rwxrwxr-x 1 fangjun fangjun 2.4M May 15 03:54 libsherpa-onnx-core.so
lrwxrwxrwx 1 fangjun fangjun   26 May 15 03:53 libsherpa-onnx-fstfar.so -> libsherpa-
↪onnx-fstfar.so.7
-rwxrwxr-x 1 fangjun fangjun  18K May 15 03:53 libsherpa-onnx-fstfar.so.7
lrwxrwxrwx 1 fangjun fangjun   23 May 15 03:53 libsherpa-onnx-fst.so -> libsherpa-onnx-
↪fst.so.6
-rwxrwxr-x 1 fangjun fangjun 2.1M May 15 03:53 libsherpa-onnx-fst.so.6
-rwxrwxr-x 1 fangjun fangjun 134K May 15 03:55 libsherpa-onnx-jni.so
-rwxrwxr-x 1 fangjun fangjun 1.2M May 15 03:53 libsherpa-onnx-kaldifst-core.so
-rwxrwxr-x 1 fangjun fangjun 229K May 15 03:53 libucd.so
```

Note that all these `*.so` files are required.

`libsherpa-onnx-jni.so` contains the JNI interface for sherpa-onnx.

Download pre-built JNI libs

If you don't want to build `JNI` libs by yourself, please download pre-built `JNI` libs from

https://huggingface.co/csukuangfj/sherpa-onnx-libs/tree/main/jni

For Chinese users, please use

https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/tree/main/jni

Please always use the latest version. In the following, we describe how to download the version `1.10.2`.

```
wget https://huggingface.co/csukuangfj/sherpa-onnx-libs/resolve/main/jni/sherpa-onnx-v1.
↪10.2-linux-x64-jni.tar.bz2

# For Chinese users
# wget https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/resolve/main/jni/sherpa-onnx-v1.
↪10.2-linux-x64-jni.tar.bz2

tar xf sherpa-onnx-v1.10.2-linux-x64-jni.tar.bz2
rm sherpa-onnx-v1.10.2-linux-x64-jni.tar.bz2
```

You should find the following files:

```
ls -lh sherpa-onnx-v1.10.2-linux-x64-jni/lib/

-rwxr-xr-x  1 fangjun  staff    16K Jun 25 11:41 libcargs.so
-rwxr-xr-x  1 fangjun  staff   383K Jun 25 11:41 libespeak-ng.so
-rwxr-xr-x  1 fangjun  staff   648K Jun 25 11:41 libkaldi-decoder-core.so
-rwxr-xr-x  1 fangjun  staff   214K Jun 25 11:41 libkaldi-native-fbank-core.so
-rw-r--r--  1 fangjun  staff    15M Jun 25 11:40 libonnxruntime.so
-rw-r--r--  1 fangjun  staff    15M Jun 25 11:40 libonnxruntime.so.1.17.1
lrwxr-xr-x  1 fangjun  staff    23B Jun 25 11:44 libpiper_phonemize.so -> libpiper_
↪phonemize.so.1
lrwxr-xr-x  1 fangjun  staff    27B Jun 25 11:44 libpiper_phonemize.so.1 -> libpiper_
↪phonemize.so.1.2.0
-rwxr-xr-x  1 fangjun  staff   535K Jun 25 11:41 libpiper_phonemize.so.1.2.0
-rwxr-xr-x  1 fangjun  staff   128K Jun 25 11:44 libsherpa-onnx-c-api.so
-rwxr-xr-x  1 fangjun  staff   2.5M Jun 25 11:43 libsherpa-onnx-core.so
-rwxr-xr-x  1 fangjun  staff   2.1M Jun 25 11:41 libsherpa-onnx-fst.so
-rwxr-xr-x  1 fangjun  staff    18K Jun 25 11:41 libsherpa-onnx-fstfar.so
-rwxr-xr-x  1 fangjun  staff   155K Jun 25 11:44 libsherpa-onnx-jni.so
-rwxr-xr-x  1 fangjun  staff   1.1M Jun 25 11:41 libsherpa-onnx-kaldifst-core.so
-rwxr-xr-x  1 fangjun  staff    81K Jun 25 11:41 libsherpa-onnx-portaudio.so
-rwxr-xr-x  1 fangjun  staff   275K Jun 25 11:41 libssentencepiece_core.so
-rwxr-xr-x  1 fangjun  staff   224K Jun 25 11:40 libucd.so
```

8.6.3 Build JNI interface (Windows)

If you want to build `JNI` libs by yourself, please see https://github.com/k2-fsa/sherpa-onnx/issues/882.

Hint: The PDFs in the above link are in Chinese.

If you want to download pre-built `JNI` libs, please see below.

Download pre-built JNI libs

If you don't want to build `JNI` libs by yourself, please download pre-built `JNI` libs from

https://huggingface.co/csukuangfj/sherpa-onnx-libs/tree/main/jni

For Chinese users, please use

https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/tree/main/jni

Please always use the latest version. In the following, we describe how to download the version `1.10.2`.

```
wget https://huggingface.co/csukuangfj/sherpa-onnx-libs/resolve/main/jni/sherpa-onnx-v1.
↪10.2-win-x64-jni.tar.bz2

# For Chinese users
# wget https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/resolve/main/jni/sherpa-onnx-v1.
↪10.2-win-x64-jni.tar.bz2
```

(continues on next page)

(continued from previous page)

```
tar xf sherpa-onnx-v1.10.2-win-x64-jni.tar.bz2
rm sherpa-onnx-v1.10.2-win-x64-jni.tar.bz2
```

You should find the following files:

```
ls -lh sherpa-onnx-v1.10.2-win-x64-jni/lib/

-rwxr-xr-x  1 fangjun  staff    12K Jun 25 11:43 cargs.dll
-rw-r--r--  1 fangjun  staff   3.1K Jun 25 11:43 cargs.lib
-rwxr-xr-x  1 fangjun  staff   271K Jun 25 11:43 espeak-ng.dll
-rw-r--r--  1 fangjun  staff    49K Jun 25 11:43 espeak-ng.lib
-rwxr-xr-x  1 fangjun  staff   533K Jun 25 11:44 kaldi-decoder-core.dll
-rw-r--r--  1 fangjun  staff   1.1M Jun 25 11:44 kaldi-decoder-core.lib
-rwxr-xr-x  1 fangjun  staff   109K Jun 25 11:43 kaldi-native-fbank-core.dll
-rw-r--r--  1 fangjun  staff   103K Jun 25 11:43 kaldi-native-fbank-core.lib
-rwxr-xr-x  1 fangjun  staff    10M Jun 25 11:43 onnxruntime.dll
-rwxr-xr-x  1 fangjun  staff    22K Jun 25 11:43 onnxruntime_providers_shared.dll
-rwxr-xr-x  1 fangjun  staff   439K Jun 25 11:43 piper_phonemize.dll
-rw-r--r--  1 fangjun  staff   212K Jun 25 11:43 piper_phonemize.lib
-rwxr-xr-x  1 fangjun  staff    76K Jun 25 11:47 sherpa-onnx-c-api.dll
-rw-r--r--  1 fangjun  staff    63K Jun 25 11:47 sherpa-onnx-c-api.lib
-rwxr-xr-x  1 fangjun  staff   1.7M Jun 25 11:47 sherpa-onnx-core.dll
-rw-r--r--  1 fangjun  staff   2.5M Jun 25 11:47 sherpa-onnx-core.lib
-rwxr-xr-x  1 fangjun  staff   1.4M Jun 25 11:43 sherpa-onnx-fst.dll
-rw-r--r--  1 fangjun  staff   3.8M Jun 25 11:43 sherpa-onnx-fst.lib
-rwxr-xr-x  1 fangjun  staff    30K Jun 25 11:43 sherpa-onnx-fstfar.dll
-rw-r--r--  1 fangjun  staff    24K Jun 25 11:43 sherpa-onnx-fstfar.lib
-rwxr-xr-x  1 fangjun  staff   109K Jun 25 11:47 sherpa-onnx-jni.dll
-rw-r--r--  1 fangjun  staff    92K Jun 25 11:47 sherpa-onnx-jni.lib
-rwxr-xr-x  1 fangjun  staff   1.1M Jun 25 11:44 sherpa-onnx-kaldifst-core.dll
-rw-r--r--  1 fangjun  staff   2.8M Jun 25 11:44 sherpa-onnx-kaldifst-core.lib
-rwxr-xr-x  1 fangjun  staff   173K Jun 25 11:43 sherpa-onnx-portaudio.dll
-rw-r--r--  1 fangjun  staff    43K Jun 25 11:43 sherpa-onnx-portaudio.lib
-rwxr-xr-x  1 fangjun  staff   135K Jun 25 11:43 ssentencepiece_core.dll
-rw-r--r--  1 fangjun  staff   174K Jun 25 11:43 ssentencepiece_core.lib
-rwxr-xr-x  1 fangjun  staff   168K Jun 25 11:43 ucd.dll
-rw-r--r--  1 fangjun  staff   5.9K Jun 25 11:43 ucd.lib
```

Hint: Only `*.dll` files are needed during runtime.

8.6.4 Build the jar package

Note: Please see the end of this page for how to download pre-built `jar`.

```
cd sherpa-onnx/sherpa-onnx/java-api
ls -lh
```

You should see the following output:

```
(py311) fangjun@ubuntu23-04:/mnt/sdb/shared/sherpa-onnx/sherpa-onnx/java-api$ ls -lh

total 8.0K
-rw-rw-r-- 1 fangjun fangjun 2.5K May  8 06:17 Makefile
drwxrwxr-x 3 fangjun fangjun 4.0K Mar  1 04:29 src
```

Please run the following command in the directory `sherpa-onnx/java-api`:

```
make
```

You should see the following output after running `make`:

```
(py311) fangjun@ubuntu23-04:/mnt/sdb/shared/sherpa-onnx/sherpa-onnx/java-api$ ls -lh
total 12K
drwxrwxr-x 3 fangjun fangjun 4.0K May 15 03:59 build
-rw-rw-r-- 1 fangjun fangjun 2.5K May  8 06:17 Makefile
drwxrwxr-x 3 fangjun fangjun 4.0K Mar  1 04:29 src
(py311) fangjun@ubuntu23-04:/mnt/sdb/shared/sherpa-onnx/sherpa-onnx/java-api$ ls -lh␣
↪build/
total 60K
drwxrwxr-x 3 fangjun fangjun 4.0K May 15 03:58 com
-rw-rw-r-- 1 fangjun fangjun  53K May 15 03:59 sherpa-onnx.jar
```

Congratulations! You have generated `sherpa-onnx.jar` successfully.

Hint: You can find the Java API source files at

https://github.com/k2-fsa/sherpa-onnx/tree/master/sherpa-onnx/java-api/src/com/k2fsa/sherpa/onnx

Download pre-built jar

If you don't want to build `jar` by yourself, you can download pre-built `jar` from from

https://huggingface.co/csukuangfj/sherpa-onnx-libs/tree/main/jni

For Chinese users, please use

https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/tree/main/jni

Please always use the latest version. In the following, we describe how to download the version `1.10.2`.

```
wget https://huggingface.co/csukuangfj/sherpa-onnx-libs/resolve/main/jni/sherpa-onnx-v1.
↪10.2.jar

# For Chinese users
# wget https://hf-mirror.com/csukuangfj/sherpa-onnx-libs/resolve/main/jni/sherpa-onnx-v1.
↪10.2.jar
```

8.6.5 Examples

Please see https://github.com/k2-fsa/sherpa-onnx/tree/master/java-api-examples

You can find detailed instructions at

https://github.com/k2-fsa/sherpa-onnx/blob/master/java-api-examples/README.md

8.7 Javascript API

For using Javascript in the browser, please see our *WebAssembly* doc.

This section describes how to use sherpa-onnx in Node with Javascript API.

8.7.1 Install

We provide npm packages for sherpa-onnx.

It can be found at

https://www.npmjs.com/package/sherpa-onnx-node

Hint: It requires `Node>=v16`.

Please always use the latest version.

To install it, please run:

```
npm install sherpa-onnx-node
```

It supports the following platforms:

- Linux x64
- Linux arm64
- macOS x64
- macOS arm64
- Windows x64

Hint: You don't need to pre-install anything in order to install `sherpa-onnx-node`.

That is, you don't need to install a C/C++ compiler. You don't need to install Python. You don't need to install CMake, etc.

8.7.2 Examples

Please see https://github.com/k2-fsa/sherpa-onnx/tree/master/nodejs-addon-examples

You can find detailed instructions at

https://github.com/k2-fsa/sherpa-onnx/tree/master/nodejs-addon-examples/README.md

8.8 Kotlin API

In this section, we describe how to use the `Kotlin` API of sherpa-onnx.

The core part of sherpa-onnx is written in C++. We have provided JNI interface for sherpa-onnx so that you can use it in Kotlin.

Before using the Kotlin API of sherpa-onnx, you have to build the `JNI` interface.

8.8.1 Build JNI interface

For macOS users, please refer to *Build JNI interface (macOS)*.

For Linux users, please refer to *Build JNI interface (Linux)*.

Hint: For Windows users, you have to modify the commands by yourself.

8.8.2 Examples

Please see https://github.com/k2-fsa/sherpa-onnx/tree/master/kotlin-api-examples

You can find detailed instructions at

https://github.com/k2-fsa/sherpa-onnx/blob/master/kotlin-api-examples/run.sh

8.9 Swift API

8.9.1 Build

Please use the following script to build sherpa-onnx for Swift API:

https://github.com/k2-fsa/sherpa-onnx/blob/master/build-swift-macos.sh

The following is an example command:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
./build-swift-macos.sh
```

8.9.2 Examples

Please see https://github.com/k2-fsa/sherpa-onnx/tree/master/swift-api-examples

Each example has a corresponding shell script. Please use the shell script to run it.

For instance,

- to run the text-to-speech example, please use https://github.com/k2-fsa/sherpa-onnx/blob/master/swift-api-examples/run-tts.sh
- to run the speech-to-text example, please use https://github.com/k2-fsa/sherpa-onnx/blob/master/swift-api-examples/run-decode-file.sh

8.10 Go API

In this section, we describe how to use the Go API of sherpa-onnx.

The Go API of sherpa-onnx supports both streaming and non-streaming speech recognition.

The following table lists some Go API examples:

Description	URL
Decode a file with **non-streaming** models	https://github.com/k2-fsa/sherpa-onnx/tree/master/go-api-examples/non-streaming-decode-files
Decode a file with **streaming** models	https://github.com/k2-fsa/sherpa-onnx/tree/master/go-api-examples/streaming-decode-files
Real-time speech recognition from a `microphone`	https://github.com/k2-fsa/sherpa-onnx/tree/master/go-api-examples/real-time-speech-recognition-from-microphone

One thing to note is that we have provided pre-built libraries for Go so that you don't need to build sherpa-onnx by yourself when using the Go API.

To make supporting multiple platforms easier, we split the Go API of sherpa-onnx into multiple packages, as listed in the following table:

OS	Package name	Supported Arch	Doc
Linux	sherpa-onnx-go-linux	`x86_64`, `aarch64`, `arm`	https://pkg.go.dev/github.com/k2-fsa/sherpa-onnx-go-linux
macOS	sherpa-onnx-go-macos	`x86_64`, `aarch64`	https://pkg.go.dev/github.com/k2-fsa/sherpa-onnx-go-macos
Windows	sherpa-onnx-go-windows	`x86_64`, `x86`	https://pkg.go.dev/github.com/k2-fsa/sherpa-onnx-go-windows

To simplify the usage, we have provided a single Go package for sherpa-onnx that supports multiple operating systems. It can be found at

https://github.com/k2-fsa/sherpa-onnx-go

Hint: Such a design is insipred by the following article:

Cross platform Go modules for giants.

You can use the following `import` to import sherpa-onnx-go into your Go project:

```
import (
  sherpa "github.com/k2-fsa/sherpa-onnx-go/sherpa_onnx"
)
```

In the following, we describe how to run our provided Go API examples.

Note: Before you continue, please make sure you have installed Go. If not, please follow https://go.dev/doc/install to install Go.

Hint: You need to enable cgo to build sherpa-onnx-go.

8.10.1 Decode files with non-streaming models

First, let us build the example:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx/go-api-examples/non-streaming-decode-files
go mod tidy
go build
./non-streaming-decode-files --help
```

You will find the following output:

```
Usage of ./non-streaming-decode-files:
      --debug int                Whether to show debug message
      --decoder string           Path to the decoder model
      --decoding-method string   Decoding method. Possible values: greedy_search,␣
↪modified_beam_search (default "greedy_search")
      --encoder string           Path to the encoder model
      --joiner string            Path to the joiner model
      --lm-model string          Optional. Path to the LM model
      --lm-scale float32         Optional. Scale for the LM model (default 1)
      --max-active-paths int     Used only when --decoding-method is modified_beam_
↪search (default 4)
      --model-type string        Optional. Used for loading the model in a faster way
      --nemo-ctc string          Path to the NeMo CTC model
      --num-threads int          Number of threads for computing (default 1)
      --paraformer string        Path to the paraformer model
      --provider string          Provider to use (default "cpu")
      --tokens string            Path to the tokens file
pflag: help requested
```

Congratulations! You have successfully built your first Go API example for speech recognition.

Note: If you are using Windows and don't see any output after running `./non-streaming-decode-files --help`, please copy `*.dll` from https://github.com/k2-fsa/sherpa-onnx-go-windows/tree/master/lib/x86_64-pc-windows-gnu (for Win64) or https://github.com/k2-fsa/sherpa-onnx-go-windows/tree/master/lib/i686-pc-windows-gnu (for Win32) to the directory `sherpa-onnx/go-api-examples/non-streaming-decode-files`.

Now let us refer to *Pre-trained models* to download a non-streaming model.

We give several examples below for demonstration.

Non-streaming transducer

We will use *csukuangfj/sherpa-onnx-zipformer-en-2023-06-26 (English)* as an example.

First, let us download it:

```
cd sherpa-onnx/go-api-examples/non-streaming-decode-files
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-en-2023-06-26.tar.bz2
tar xvf sherpa-onnx-zipformer-en-2023-06-26.tar.bz2
rm sherpa-onnx-zipformer-en-2023-06-26.tar.bz2
```

Now we can use:

```
./non-streaming-decode-files \
  --encoder ./sherpa-onnx-zipformer-en-2023-06-26/encoder-epoch-99-avg-1.onnx \
  --decoder ./sherpa-onnx-zipformer-en-2023-06-26/decoder-epoch-99-avg-1.onnx \
  --joiner ./sherpa-onnx-zipformer-en-2023-06-26/joiner-epoch-99-avg-1.onnx \
  --tokens ./sherpa-onnx-zipformer-en-2023-06-26/tokens.txt \
  --model-type transducer \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/0.wav
```

It should give you the following output:

```
2023/08/10 14:52:48.723098 Reading ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/0.wav
2023/08/10 14:52:48.741042 Initializing recognizer (may take several seconds)
2023/08/10 14:52:51.998848 Recognizer created!
2023/08/10 14:52:51.998870 Start decoding!
2023/08/10 14:52:52.258818 Decoding done!
2023/08/10 14:52:52.258847  after early nightfall the yellow lamps would light up here␣
↪and there the squalid quarter of the brothels
2023/08/10 14:52:52.258952 Wave duration: 6.625 seconds
```

Non-streaming paraformer

We will use *csukuangfj/sherpa-onnx-paraformer-zh-2023-03-28 (Chinese + English)* as an example.

First, let us download it:

```
cd sherpa-onnx/go-api-examples/non-streaming-decode-files
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-zh-2023-03-28.tar.bz2
tar xvf sherpa-onnx-paraformer-zh-2023-03-28.tar.bz2
rm sherpa-onnx-paraformer-zh-2023-03-28.tar.bz2
```

Now we can use:

```
./non-streaming-decode-files \
  --paraformer ./sherpa-onnx-paraformer-zh-2023-03-28/model.int8.onnx \
  --tokens ./sherpa-onnx-paraformer-zh-2023-03-28/tokens.txt \
```

(continues on next page)

(continued from previous page)

```
  --model-type paraformer \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav
```

It should give you the following output:

```
2023/08/10 15:07:10.745412 Reading ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav
2023/08/10 15:07:10.758414 Initializing recognizer (may take several seconds)
2023/08/10 15:07:13.992424 Recognizer created!
2023/08/10 15:07:13.992441 Start decoding!
2023/08/10 15:07:14.382157 Decoding done!
2023/08/10 15:07:14.382847
2023/08/10 15:07:14.382898 Wave duration: 5.614625 seconds
```

Non-streaming CTC model from NeMo

We will use *stt_en_conformer_ctc_medium* as an example.

First, let us download it:

```
cd sherpa-onnx/go-api-examples/non-streaming-decode-files
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-nemo-
↪ctc-en-conformer-medium.tar.bz2
tar xvf sherpa-onnx-nemo-ctc-en-conformer-medium.tar.bz2
rm sherpa-onnx-nemo-ctc-en-conformer-medium.tar.bz2
```

Now we can use:

```
./non-streaming-decode-files \
  --nemo-ctc ./sherpa-onnx-nemo-ctc-en-conformer-medium/model.onnx \
  --tokens ./sherpa-onnx-nemo-ctc-en-conformer-medium/tokens.txt \
  --model-type nemo_ctc \
  ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/0.wav
```

It should give you the following output:

```
2023/08/10 15:11:48.667693 Reading ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/
↪0.wav
2023/08/10 15:11:48.680855 Initializing recognizer (may take several seconds)
2023/08/10 15:11:51.900852 Recognizer created!
2023/08/10 15:11:51.900869 Start decoding!
2023/08/10 15:11:52.125605 Decoding done!
2023/08/10 15:11:52.125630  after early nightfall the yellow lamps would light up here␣
↪and there the squalid quarter of the brothels
2023/08/10 15:11:52.125645 Wave duration: 6.625 seconds
```

8.10.2 Decode files with streaming models

First, let us build the example:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx/go-api-examples/streaming-decode-files
go mod tidy
go build
./streaming-decode-files --help
```

You will find the following output:

```
Usage of ./streaming-decode-files:
      --debug int                Whether to show debug message
      --decoder string           Path to the decoder model
      --decoding-method string   Decoding method. Possible values: greedy_search,␣
↪modified_beam_search (default "greedy_search")
      --encoder string           Path to the encoder model
      --joiner string            Path to the joiner model
      --max-active-paths int     Used only when --decoding-method is modified_beam_
↪search (default 4)
      --model-type string        Optional. Used for loading the model in a faster way
      --num-threads int          Number of threads for computing (default 1)
      --provider string          Provider to use (default "cpu")
      --tokens string            Path to the tokens file
pflag: help requested
```

Note: If you are using Windows and don't see any output after running `./streaming-decode-files --help`, please copy `*.dll` from https://github.com/k2-fsa/sherpa-onnx-go-windows/tree/master/lib/x86_64-pc-windows-gnu (for Win64) or https://github.com/k2-fsa/sherpa-onnx-go-windows/tree/master/lib/i686-pc-windows-gnu (for Win32) to the directory `sherpa-onnx/go-api-examples/streaming-decode-files`.

Now let us refer to *Pre-trained models* to download a streaming model.

We give one example below for demonstration.

Streaming transducer

We will use *csukuangfj/sherpa-onnx-streaming-zipformer-en-2023-06-26 (English)* as an example.

First, let us download it:

```
cd sherpa-onnx/go-api-examples/streaming-decode-files
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-en-2023-06-26.tar.bz2
tar xvf sherpa-onnx-streaming-zipformer-en-2023-06-26.tar.bz2
rm sherpa-onnx-streaming-zipformer-en-2023-06-26.tar.bz2
```

Now we can use:

```
./streaming-decode-files \
  --encoder ./sherpa-onnx-streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
```

(continues on next page)

(continued from previous page)

```
  --decoder ./sherpa-onnx-streaming-zipformer-en-2023-06-26/decoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --joiner ./sherpa-onnx-streaming-zipformer-en-2023-06-26/joiner-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --tokens ./sherpa-onnx-streaming-zipformer-en-2023-06-26/tokens.txt \
  --model-type zipformer2 \
  ./sherpa-onnx-streaming-zipformer-en-2023-06-26/test_wavs/0.wav
```

It should give you the following output:

```
2023/08/10 15:17:00.226228 Reading ./sherpa-onnx-streaming-zipformer-en-2023-06-26/test_
↪wavs/0.wav
2023/08/10 15:17:00.241024 Initializing recognizer (may take several seconds)
2023/08/10 15:17:03.352697 Recognizer created!
2023/08/10 15:17:03.352711 Start decoding!
2023/08/10 15:17:04.057130 Decoding done!
2023/08/10 15:17:04.057215  after early nightfall the yellow lamps would light up here␣
↪and there the squalid quarter of the brothels
2023/08/10 15:17:04.057235 Wave duration: 6.625 seconds
```

8.10.3 Real-time speech recognition from microphone

Hint: You need to install `portaudio` for this example.

```
# for macOS
brew install portaudio
export PKG_CONFIG_PATH=/usr/local/Cellar/portaudio/19.7.0

# for Ubuntu
sudo apt-get install libasound-dev portaudio19-dev libportaudio2 libportaudiocpp0
```

To check that you have installed `portaudio` successfully, please run:

```
pkg-config --cflags --libs portaudio-2.0
```

It should give you something like below:

```
# for macOS
-I/usr/local/Cellar/portaudio/19.7.0/include -L/usr/local/Cellar/portaudio/19.
↪7.0/lib -lportaudio -framework CoreAudio -framework AudioToolbox -framework␣
↪AudioUnit -framework CoreFoundation -framework CoreServices

# for Ubuntu
-pthread -lportaudio -lasound -lm -lpthread
```

First, let us build the example:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx/go-api-examples/real-time-speech-recognition-from-microphone
go mod tidy
```

(continues on next page)

(continued from previous page)

```
go build
./real-time-speech-recognition-from-microphone --help
```

You will find the following output:

```
Select default input device: MacBook Pro Microphone
Usage of ./real-time-speech-recognition-from-microphone:
      --debug int                            Whether to show debug message
      --decoder string                       Path to the decoder model
      --decoding-method string               Decoding method. Possible values: greedy_
↪search, modified_beam_search (default "greedy_search")
      --enable-endpoint int                  Whether to enable endpoint (default 1)
      --encoder string                       Path to the encoder model
      --joiner string                        Path to the joiner model
      --max-active-paths int                 Used only when --decoding-method is␣
↪modified_beam_search (default 4)
      --model-type string                    Optional. Used for loading the model in a␣
↪faster way
      --num-threads int                      Number of threads for computing (default 1)
      --provider string                      Provider to use (default "cpu")
      --rule1-min-trailing-silence float32   Threshold for rule1 (default 2.4)
      --rule2-min-trailing-silence float32   Threshold for rule2 (default 1.2)
      --rule3-min-utterance-length float32   Threshold for rule3 (default 20)
      --tokens string                        Path to the tokens file
pflag: help requested
```

Now let us refer to *Pre-trained models* to download a streaming model.

We give one example below for demonstration.

Streaming transducer

We will use *csukuangfj/sherpa-onnx-streaming-zipformer-en-2023-06-26 (English)* as an example.

First, let us download it:

```
cd sherpa-onnx/go-api-examples/real-time-speech-recognition-from-microphone
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-en-2023-06-26.tar.bz2
tar xvf sherpa-onnx-streaming-zipformer-en-2023-06-26.tar.bz2
rm sherpa-onnx-streaming-zipformer-en-2023-06-26.tar.bz2
```

Now we can use:

```
./real-time-speech-recognition-from-microphone \
  --encoder ./sherpa-onnx-streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --decoder ./sherpa-onnx-streaming-zipformer-en-2023-06-26/decoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --joiner ./sherpa-onnx-streaming-zipformer-en-2023-06-26/joiner-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --tokens ./sherpa-onnx-streaming-zipformer-en-2023-06-26/tokens.txt \
  --model-type zipformer2
```

It should give you the following output:

```
Select default input device: MacBook Pro Microphone
2023/08/10 15:22:00 Initializing recognizer (may take several seconds)
2023/08/10 15:22:03 Recognizer created!
Started! Please speak
0:  this is the first test
1:  this is the second
```

8.10.4 colab

We provide a colab notebook for you to try the Go API examples of sherpa-onnx.

8.11 C# API

In this section, we describe how to use the C# API examples of sherpa-onnx.

The C# API of sherpa-onnx supports both streaming and non-streaming speech recognition.

The following table lists some C# API examples:

Description	URL
Decode a file with **non-streaming** models	https://github.com/k2-fsa/sherpa-onnx/tree/master/dotnet-examples/offline-decode-files
Decode a file with **streaming** models	https://github.com/k2-fsa/sherpa-onnx/tree/master/dotnet-examples/online-decode-files
Real-time speech recognition from a `microphone`	https://github.com/k2-fsa/sherpa-onnx/tree/master/dotnet-examples/speech-recognition-from-microphone

You can find the implementation in the following files:

- API for **streaming** speech recognition

  https://github.com/k2-fsa/sherpa-onnx/blob/master/scripts/dotnet/online.cs

- API for **non-streaming** speech recognition

  https://github.com/k2-fsa/sherpa-onnx/blob/master/scripts/dotnet/offline.cs

We also provide a nuget package for sherpa-onnx:

https://www.nuget.org/packages/org.k2fsa.sherpa.onnx

You can use the following statement in your `csproj` file to introduce the dependency on sherpa-onnx:

```
<PackageReference Include="org.k2fsa.sherpa.onnx" Version="*" />
```

One thing to note is that we have provided pre-built libraries for C# so that you don't need to build sherpa-onnx by yourself when using the C# API.

In the following, we describe how to run our provided C# API examples.

Note: Before you continue, please make sure you have installed .Net. If not, please follow https://dotnet.microsoft.com/en-us/download to install `.Net`.

Hint: `.Net` supports Windows, macOS, and Linux.

8.11.1 Decode files with non-streaming models

First, let us build the example:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx/dotnet-examples/offline-decode-files/
dotnet build -c Release
./bin/Release/net6.0/offline-decode-files --help
```

You will find the following output:

```
# Zipformer

dotnet run \
  --tokens=./sherpa-onnx-zipformer-en-2023-04-01/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-en-2023-04-01/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-en-2023-04-01/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-en-2023-04-01/joiner-epoch-99-avg-1.onnx \
  --files ./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/8k.wav

Please refer to
https://k2-fsa.github.io/sherpa/onnx/pretrained_models/offline-transducer/index.html
to download pre-trained non-streaming zipformer models.

# Paraformer

dotnet run \
  --tokens=./sherpa-onnx-paraformer-zh-2023-03-28/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-zh-2023-03-28/model.onnx \
  --files ./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/0.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav
Please refer to
https://k2-fsa.github.io/sherpa/onnx/pretrained_models/offline-paraformer/index.html
to download pre-trained paraformer models

# NeMo CTC

dotnet run \
  --tokens=./sherpa-onnx-nemo-ctc-en-conformer-medium/tokens.txt \
  --nemo-ctc=./sherpa-onnx-nemo-ctc-en-conformer-medium/model.onnx \
  --num-threads=1 \
  --files ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/0.wav \
  ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/1.wav \
  ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/8k.wav
```

(continues on next page)

(continued from previous page)

```
Please refer to
https://k2-fsa.github.io/sherpa/onnx/pretrained_models/offline-ctc/index.html
to download pre-trained paraformer models

Copyright (c) 2023 Xiaomi Corporation

  --tokens              Path to tokens.txt
  --encoder             Path to encoder.onnx. Used only for transducer models
  --decoder             Path to decoder.onnx. Used only for transducer models
  --joiner              Path to joiner.onnx. Used only for transducer models
  --paraformer          Path to model.onnx. Used only for paraformer models
  --nemo-ctc            Path to model.onnx. Used only for NeMo CTC models
  --num-threads         (Default: 1) Number of threads for computation
  --decoding-method     (Default: greedy_search) Valid decoding methods are:
                        greedy_search, modified_beam_search
  --max-active-paths    (Default: 4) Used only when --decoding--method is
                        modified_beam_search.
                        It specifies number of active paths to keep during the
                        search
  --files               Required. Audio files for decoding
  --help                Display this help screen.
  --version             Display version information.
```

Now let us refer to *Pre-trained models* to download a non-streaming model.

We give several examples below for demonstration.

Non-streaming transducer

We will use *csukuangfj/sherpa-onnx-zipformer-en-2023-06-26 (English)* as an example.

First, let us download it:

```
cd sherpa-onnx/dotnet-examples/offline-decode-files
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-en-2023-06-26.tar.bz2
tar xvf sherpa-onnx-zipformer-en-2023-06-26.tar.bz2
rm sherpa-onnx-zipformer-en-2023-06-26.tar.bz2
```

Now we can use:

```
dotnet run -c Release \
  --encoder ./sherpa-onnx-zipformer-en-2023-06-26/encoder-epoch-99-avg-1.onnx \
  --decoder ./sherpa-onnx-zipformer-en-2023-06-26/decoder-epoch-99-avg-1.onnx \
  --joiner ./sherpa-onnx-zipformer-en-2023-06-26/joiner-epoch-99-avg-1.onnx \
  --tokens ./sherpa-onnx-zipformer-en-2023-06-26/tokens.txt \
  --files ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/8k.wav
```

It should give you the following output:

```
/Users/runner/work/sherpa-onnx/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:117 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

--------------------
./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/0.wav
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
--------------------
./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/1.wav
 GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY␣
↪CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT FOREVER WITH␣
↪THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
--------------------
./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/8k.wav
 YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION
--------------------
```

Non-streaming paraformer

We will use *csukuangfj/sherpa-onnx-paraformer-zh-2023-03-28 (Chinese + English)* as an example.

First, let us download it:

```
cd sherpa-onnx/dotnet-examples/offline-decode-files
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-zh-2023-03-28.tar.bz2
tar xvf sherpa-onnx-paraformer-zh-2023-03-28.tar.bz2
rm sherpa-onnx-paraformer-zh-2023-03-28.tar.bz2
```

Now we can use:

```
dotnet run -c Release \
  --paraformer ./sherpa-onnx-paraformer-zh-2023-03-28/model.int8.onnx \
  --tokens ./sherpa-onnx-paraformer-zh-2023-03-28/tokens.txt \
  --files ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav
```

It should give you the following output:

```
/Users/runner/work/sherpa-onnx/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:117 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

--------------------
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav

--------------------
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav
```

(continues on next page)

(continued from previous page)

```
--------------------
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav

--------------------
```

Non-streaming CTC model from NeMo

We will use *stt_en_conformer_ctc_medium* as an example.

First, let us download it:

```
cd sherpa-onnx/dotnet-examples/offline-decode-files
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-nemo-
↪ctc-en-conformer-medium.tar.bz2
tar xvf sherpa-onnx-nemo-ctc-en-conformer-medium.tar.bz2
rm sherpa-onnx-nemo-ctc-en-conformer-medium.tar.bz2
```

Now we can use:

```
dotnet run -c Release \
  --nemo-ctc ./sherpa-onnx-nemo-ctc-en-conformer-medium/model.onnx \
  --tokens ./sherpa-onnx-nemo-ctc-en-conformer-medium/tokens.txt \
  --files ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/0.wav \
  ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/1.wav \
  ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/8k.wav
```

It should give you the following output:

```
/Users/runner/work/sherpa-onnx/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:117 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

--------------------
./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/0.wav
 after early nightfall the yellow lamps would light up here and there the squalid␣
↪quarter of the brothels
--------------------
./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/1.wav
 god as a direct consequence of the sin which man thus punished had given her a lovely␣
↪child whose place was on that same dishonored bosom to connect her parent for ever␣
↪with the race and descent of mortals and to be finally a blessed soul in heaven
--------------------
./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/8k.wav
 yet these thoughts affected hester pryne less with hope than apprehension
--------------------
```

8.11.2 Decode files with streaming models

First, let us build the example:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx/dotnet-examples/online-decode-files
dotnet build -c Release
./bin/Release/net6.0/online-decode-files --help
```

You will find the following output:

```
dotnet run \
  --tokens=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/encoder-epoch-
↪99-avg-1.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/decoder-epoch-
↪99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/joiner-epoch-99-
↪avg-1.onnx \
  --num-threads=2 \
  --decoding-method=modified_beam_search \
  --debug=false \
  --files ./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/0.wav \
  ./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/1.wav

Please refer to
https://k2-fsa.github.io/sherpa/onnx/pretrained_models/online-transducer/index.html
to download pre-trained streaming models.

Copyright (c) 2023 Xiaomi Corporation

  --tokens                        Required. Path to tokens.txt
  --provider                      (Default: cpu) Provider, e.g., cpu, coreml
  --encoder                       Required. Path to encoder.onnx
  --decoder                       Required. Path to decoder.onnx
  --joiner                        Required. Path to joiner.onnx
  --num-threads                   (Default: 1) Number of threads for computation
  --decoding-method               (Default: greedy_search) Valid decoding
                                  methods are: greedy_search,
                                  modified_beam_search
  --debug                         (Default: false) True to show model info
                                  during loading
  --sample-rate                   (Default: 16000) Sample rate of the data used
                                  to train the model
  --max-active-paths              (Default: 4) Used only when --decoding--method
                                  is modified_beam_search.
                                  It specifies number of active paths to keep
                                  during the search
  --enable-endpoint               (Default: false) True to enable endpoint
                                  detection.
  --rule1-min-trailing-silence    (Default: 2.4) An endpoint is detected if
                                  trailing silence in seconds is
                                  larger than this value even if nothing has
                                  been decoded. Used only when --enable-endpoint
```

(continues on next page)

(continued from previous page)

```
                                  is true.
  --rule2-min-trailing-silence    (Default: 1.2) An endpoint is detected if
                                  trailing silence in seconds is
                                  larger than this value after something that is
                                  not blank has been decoded. Used
                                  only when --enable-endpoint is true.
  --rule3-min-utterance-length    (Default: 20) An endpoint is detected if the
                                  utterance in seconds is
                                  larger than this value. Used only when
                                  --enable-endpoint is true.
  --files                         Required. Audio files for decoding
  --help                          Display this help screen.
  --version                       Display version information.
```

Now let us refer to *Pre-trained models* to download a streaming model.

We give one example below for demonstration.

Streaming transducer

We will use *csukuangfj/sherpa-onnx-streaming-zipformer-en-2023-06-26 (English)* as an example.

First, let us download it:

```
cd sherpa-onnx/dotnet-examples/online-decode-files/
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-en-2023-06-26.tar.bz2
tar xvf sherpa-onnx-streaming-zipformer-en-2023-06-26.tar.bz2
rm sherpa-onnx-streaming-zipformer-en-2023-06-26.tar.bz2
```

Now we can use:

```
dotnet run -c Release \
  --encoder ./sherpa-onnx-streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --decoder ./sherpa-onnx-streaming-zipformer-en-2023-06-26/decoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --joiner ./sherpa-onnx-streaming-zipformer-en-2023-06-26/joiner-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --tokens ./sherpa-onnx-streaming-zipformer-en-2023-06-26/tokens.txt \
  --files ./sherpa-onnx-streaming-zipformer-en-2023-06-26/test_wavs/0.wav \
  ./sherpa-onnx-streaming-zipformer-en-2023-06-26/test_wavs/1.wav \
  ./sherpa-onnx-streaming-zipformer-en-2023-06-26/test_wavs/8k.wav
```

You will find the following output:

```
/Users/runner/work/sherpa-onnx/sherpa-onnx/sherpa-onnx/csrc/features.
↪cc:AcceptWaveform:76 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

--------------------
./sherpa-onnx-streaming-zipformer-en-2023-06-26/test_wavs/0.wav
```

(continues on next page)

(continued from previous page)

```
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
--------------------
./sherpa-onnx-streaming-zipformer-en-2023-06-26/test_wavs/1.wav
 GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY␣
↪CHILD WHOSE PLACE WAS ON THAT SAME DISHONOURED BOSOM TO CONNECT HER PARENT FOR EVER␣
↪WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
--------------------
./sherpa-onnx-streaming-zipformer-en-2023-06-26/test_wavs/8k.wav
 YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION
--------------------
```

8.11.3 Real-time speech recognition from microphone

First, let us build the example:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx/dotnet-examples/speech-recognition-from-microphone
dotnet build -c Release
./bin/Release/net6.0/speech-recognition-from-microphone --help
```

You will find the following output:

```
dotnet run -c Release \
  --tokens ./icefall-asr-zipformer-streaming-wenetspeech-20230615/data/lang_char/tokens.
↪txt \
  --encoder ./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/encoder-epoch-12-
↪avg-4-chunk-16-left-128.onnx \
  --decoder ./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/decoder-epoch-12-
↪avg-4-chunk-16-left-128.onnx \
  --joiner ./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/joiner-epoch-12-
↪avg-4-chunk-16-left-128.onnx \

Please refer to
https://k2-fsa.github.io/sherpa/onnx/pretrained_models/online-transducer/index.html
to download pre-trained streaming models.

Copyright (c) 2023 Xiaomi Corporation

  --tokens                        Required. Path to tokens.txt
  --provider                      (Default: cpu) Provider, e.g., cpu, coreml
  --encoder                       Required. Path to encoder.onnx
  --decoder                       Required. Path to decoder.onnx
  --joiner                        Required. Path to joiner.onnx
  --num-threads                   (Default: 1) Number of threads for computation
  --decoding-method               (Default: greedy_search) Valid decoding
                                  methods are: greedy_search,
                                  modified_beam_search
  --debug                         (Default: false) True to show model info
                                  during loading
  --sample-rate                   (Default: 16000) Sample rate of the data used
```

(continues on next page)

(continued from previous page)

```
                                  to train the model
  --max-active-paths              (Default: 4) Used only when --decoding--method
                                  is modified_beam_search.
                                  It specifies number of active paths to keep
                                  during the search
  --enable-endpoint               (Default: true) True to enable endpoint
                                  detection.
  --rule1-min-trailing-silence    (Default: 2.4) An endpoint is detected if
                                  trailing silence in seconds is
                                  larger than this value even if nothing has
                                  been decoded. Used only when --enable-endpoint
                                  is true.
  --rule2-min-trailing-silence    (Default: 0.8) An endpoint is detected if
                                  trailing silence in seconds is
                                  larger than this value after something that is
                                  not blank has been decoded. Used
                                  only when --enable-endpoint is true.
  --rule3-min-utterance-length    (Default: 20) An endpoint is detected if the
                                  utterance in seconds is
                                  larger than this value. Used only when
                                  --enable-endpoint is true.
  --help                          Display this help screen.
  --version                       Display version information.
```

Now let us refer to *Pre-trained models* to download a streaming model.

We give one example below for demonstration.

Streaming transducer

We will use *csukuangfj/sherpa-onnx-streaming-zipformer-en-2023-06-26 (English)* as an example.

First, let us download it:

```
cd sherpa-onnx/dotnet-examples/speech-recognition-from-microphone
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-en-2023-06-26.tar.bz2
tar xvf sherpa-onnx-streaming-zipformer-en-2023-06-26.tar.bz2
rm sherpa-onnx-streaming-zipformer-en-2023-06-26.tar.bz2
```

Now we can use:

```
dotnet run -c Release \
  --encoder ./sherpa-onnx-streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --decoder ./sherpa-onnx-streaming-zipformer-en-2023-06-26/decoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --joiner ./sherpa-onnx-streaming-zipformer-en-2023-06-26/joiner-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --tokens ./sherpa-onnx-streaming-zipformer-en-2023-06-26/tokens.txt
```

You will find the following output:

```
PortAudio V19.7.0-devel, revision 147dd722548358763a8b649b3e4b41dfffbcfbb6
Number of devices: 5
 Device 0
   Name: Background Music
   Max input channels: 2
   Default sample rate: 44100
 Device 1
   Name: Background Music (UI Sounds)
   Max input channels: 2
   Default sample rate: 44100
 Device 2
   Name: MacBook Pro Microphone
   Max input channels: 1
   Default sample rate: 48000
 Device 3
   Name: MacBook Pro Speakers
   Max input channels: 0
   Default sample rate: 48000
 Device 4
   Name: WeMeet Audio Device
   Max input channels: 2
   Default sample rate: 48000

Use default device 2 (MacBook Pro Microphone)
StreamParameters [
  device=2
  channelCount=1
  sampleFormat=Float32
  suggestedLatency=0.034520833333333334
  hostApiSpecificStreamInfo?=[False]
]
Started! Please speak

0:  THIS IS A TEST
1:  THIS IS A SECOND TEST
```

8.11.4 colab

We provide a colab notebook for you to try the C# API examples of sherpa-onnx.

8.12 WebAssembly

In this section, we describe how to build sherpa-onnx for WebAssembly so that you can run real-time speech recognition with WebAssembly.

Please follow the steps below to build and run sherpa-onnx for WebAssembly.

Hint: We provide a colab notebook for you to try this section step by step.

If you are using Windows or you don't want to setup your local environment to build WebAssembly support, please

use the above colab notebook.

8.12.1 Install Emscripten

We need to compile the C/C++ files in sherpa-onnx with the help of emscripten.

Please refer to https://emscripten.org/docs/getting_started/downloads for detailed installation instructions.

The following is an example to show you how to install it on Linux/macOS.

```
git clone https://github.com/emscripten-core/emsdk.git
cd emsdk
git pull
./emsdk install latest
./emsdk activate latest
source ./emsdk_env.sh
```

To check that you have installed emscripten successfully, please run:

```
emcc -v
```

The above command should print something like below:

```
emcc (Emscripten gcc/clang-like replacement + linker emulating GNU ld) 3.1.48␣
↪(e967e20b4727956a30592165a3c1cde5c67fa0a8)
shared:INFO: (Emscripten: Running sanity checks)
(py38) fangjuns-MacBook-Pro:open-source fangjun$ emcc -v
emcc (Emscripten gcc/clang-like replacement + linker emulating GNU ld) 3.1.48␣
↪(e967e20b4727956a30592165a3c1cde5c67fa0a8)
clang version 18.0.0 (https://github.com/llvm/llvm-project␣
↪a54545ba6514802178cf7cf1c1dd9f7efbf3cde7)
Target: wasm32-unknown-emscripten
Thread model: posix
InstalledDir: /Users/fangjun/open-source/emsdk/upstream/bin
```

Congratulations! You have successfully installed emscripten.

8.12.2 Build

After installing emscripten, we can build sherpa-onnx for WebAssembly now.

Please use the following command to build it:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx

cd wasm/asr/assets

wget -q https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-bilingual-zh-en-2023-02-20.tar.bz2
tar xvf sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20.tar.bz2
rm sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20.tar.bz2
```

(continues on next page)

(continued from previous page)

```
# Note it is not an error that we rename encoder.int8.onnx to encoder.onnx

mv sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/encoder-epoch-99-avg-1.
↪int8.onnx encoder.onnx
mv sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/decoder-epoch-99-avg-1.
↪onnx decoder.onnx
mv sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/joiner-epoch-99-avg-1.int8.
↪onnx joiner.onnx
mv sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/tokens.txt ./
rm -rf sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/

cd ../../..

./build-wasm-simd-asr.sh
```

Hint: You can visit https://github.com/k2-fsa/sherpa-onnx/releases/tag/asr-models to download a different model.

If you want to use a streaming Paraformer model, please see https://github.com/k2-fsa/sherpa-onnx/blob/master/wasm/asr/assets/README.md#paraformer

After building, you should see the following output:

```
Install the project...
-- Install configuration: "Release"
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-asr/install/bin/
↪wasm/asr/sherpa-onnx-wasm-asr-main.js
-- Up-to-date: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-asr/install/bin/
↪wasm/asr/sherpa-onnx-wasm-asr-main.js
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-asr/install/bin/
↪wasm/asr/index.html
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-asr/install/bin/
↪wasm/asr/sherpa-onnx.js
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-asr/install/bin/
↪wasm/asr/app.js
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-asr/install/bin/
↪wasm/asr/sherpa-onnx-wasm-asr-main.wasm
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-asr/install/bin/
↪wasm/asr/sherpa-onnx-wasm-asr-main.data
+ ls -lh install/bin/wasm/asr
total 440080
-rw-r--r--  1 fangjun  staff   9.0K Feb 23 17:39 app.js
-rw-r--r--  1 fangjun  staff   978B Feb 23 17:39 index.html
-rw-r--r--  1 fangjun  staff   199M Feb 23 18:34 sherpa-onnx-wasm-asr-main.data
-rw-r--r--  1 fangjun  staff    90K Feb 23 18:38 sherpa-onnx-wasm-asr-main.js
-rw-r--r--  1 fangjun  staff    10M Feb 23 18:38 sherpa-onnx-wasm-asr-main.wasm
-rw-r--r--  1 fangjun  staff   9.1K Feb 23 17:39 sherpa-onnx.js
```

Now you can use the following command to run it:

```
cd build-wasm-simd-asr/install/bin/wasm/asr
python3 -m http.server 6006
```

Start your browser and visit http://localhost:6006/; you should see the following page:

Now click start and speak! You should see the recognition results in the text box.

Warning: We are using a bilingual model (Chinese + English) in the above example, which means you can only speak Chinese or English in this case.

A screenshot is given below:

Congratulations! You have successfully run real-time speech recognition with WebAssembly in your browser.

8.12.3 Huggingface Spaces (WebAssembly)

We provide four Huggingface spaces so that you can try real-time speech recognition with WebAssembly in your browser.

English only (Zipformer)

https://huggingface.co/spaces/k2-fsa/web-assembly-asr-sherpa-onnx-en

Hint: If you don't have access to Huggingface, please visit the following mirror:

https://modelscope.cn/studios/k2-fsa/web-assembly-asr-sherpa-onnx-en/summary

Note: The script for building this space can be found at https://github.com/k2-fsa/sherpa-onnx/blob/master/.github/workflows/wasm-simd-hf-space-en-asr-zipformer.yaml

Chinese + English (Zipformer)

https://huggingface.co/spaces/k2-fsa/web-assembly-asr-sherpa-onnx-zh-en

Hint: If you don't have access to Huggingface, please visit the following mirror:

https://modelscope.cn/studios/k2-fsa/web-assembly-asr-sherpa-onnx-zh-en/summary

Note: The script for building this space can be found at https://github.com/k2-fsa/sherpa-onnx/blob/master/.github/workflows/wasm-simd-hf-space-zh-en-asr-zipformer.yaml

Chinese + English (Paraformer)

https://huggingface.co/spaces/k2-fsa/web-assembly-asr-sherpa-onnx-zh-en-paraformer

Hint: If you don't have access to Huggingface, please visit the following mirror:

https://modelscope.cn/studios/k2-fsa/web-assembly-asr-sherpa-onnx-zh-en-paraformer/summary

Note: The script for building this space can be found at https://github.com/k2-fsa/sherpa-onnx/blob/master/.github/workflows/wasm-simd-hf-space-zh-en-asr-paraformer.yaml

Chinese + English + Cantonese (Paraformer)

https://huggingface.co/spaces/k2-fsa/web-assembly-asr-sherpa-onnx-zh-cantonese-en-paraformer

Hint: If you don't have access to Huggingface, please visit the following mirror:

https://modelscope.cn/studios/k2-fsa/web-assembly-asr-sherpa-onnx-zh-cantonese-en-paraformer/summary

Note: The script for building this space can be found at https://github.com/k2-fsa/sherpa-onnx/blob/master/.github/workflows/wasm-simd-hf-space-zh-cantonese-en-asr-paraformer.yaml

8.13 Android

In this section, we describe how to build an Android app for `real-time` speech recognition with sherpa-onnx.

Hint: During speech recognition, it does not need to access the Internet. Everyting is processed locally on your phone.

8.13.1 Pre-built APKs

Links for pre-built APKs can be found in the following table:

Hint: It runs locally, without internet connection.

		URL
Streaming speech recognition		https://k2-fsa.github.io/sherpa/onnx/android/apk.html
Text-to-speech engine		https://k2-fsa.github.io/sherpa/onnx/tts/apk-engine.html
Voice activity detection (VAD)		https://k2-fsa.github.io/sherpa/onnx/vad/apk.html
VAD + non-streaming speech recognition		https://k2-fsa.github.io/sherpa/onnx/vad/apk-asr.html
Two-pass speech recognition		https://k2-fsa.github.io/sherpa/onnx/android/apk-2pass.html
Audio tagging		https://k2-fsa.github.io/sherpa/onnx/audio-tagging/apk.html
Audio tagging (WearOS)		https://k2-fsa.github.io/sherpa/onnx/audio-tagging/apk-wearos.html
Speaker identification		https://k2-fsa.github.io/sherpa/onnx/speaker-identification/apk.html
Spoken language identification		https://k2-fsa.github.io/sherpa/onnx/spoken-language-identification/apk.html
Keyword spotting		https://k2-fsa.github.io/sherpa/onnx/kws/apk.html

8.13.2 Build sherpa-onnx for Android

You can use this section for both `speech-to-text` (STT, ASR) and `text-to-speech` (TTS).

Hint: The build scripts mentioned in this section run on both Linux and macOS.

If you are using Windows or if you don't want to build the shared libraries, you can download pre-built shared libraries by visiting the release page https://github.com/k2-fsa/sherpa-onnx/releases/

For instance, for the relase `v1.9.14`, you can visit https://github.com/k2-fsa/sherpa-onnx/releases/tag/v1.9.14 and download the file `sherpa-onnx-v1.9.14-android.tar.bz2` using the following command:

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/v1.9.14/sherpa-
↪onnx-v1.9.14-android.tar.bz2
```

Please always use the latest release.

Hint: This section is originally written for speech-to-text. However, it is also applicable to other folders in https://github.com/k2-fsa/sherpa-onnx/tree/master/android.

For instance, you can replace `SherpaOnnx` in this section with

- `SherpaOnnx2Pass`
- `SherpaOnnxTts` (this is for text-to-speech)
- `SherpaOnnxTtsEngine` (this is for text-to-speech)
- `SherpaOnnxVad`
- `SherpaOnnxVadAsr`
- `SherpaOnnxSpeakerIdentification`
- `SherpaOnnxAudioTagging`
- `SherpaOnnxAudioTaggingWearOs`

Install Android Studio

The first step is to download and install Android Studio.

Please refer to https://developer.android.com/studio for how to install Android Studio.

Hint: Any recent version of Android Studio should work fine. Also, you can use the default settings of Android Studio during installation.

For reference, we post the version we are using below:

Download sherpa-onnx

Next, download the source code of sherpa-onnx:

```
git clone https://github.com/k2-fsa/sherpa-onnx
```

Install NDK

Step 1, start Android Studio.

Step 2, Open `sherpa-onnx/android/SherpaOnnx`.

Step 3, Select `Tools -> SDK Manager`.

Step 4, `Install NDK`.

Fig. 8.1: Step 1: Click `Open` to select `sherpa-onnx/android/SherpaOnnx`

Fig. 8.2: Step 2: Open `SherpaOnnx`.

Fig. 8.3: Step 3: Select `Tools -> SDK Manager`.

Fig. 8.4: Step 4: Install NDK.

In the following, we assume `Android SDK location` was set to `/Users/fangjun/software/my-android`. You can change it accordingly below.

After installing NDK, you can find it in

```
/Users/fangjun/software/my-android/ndk/22.1.7171670
```

Warning: If you selected a different version of NDK, please replace `22.1.7171670` accordingly.

Next, let us set the environment variable `ANDROID_NDK` for later use.

```
export ANDROID_NDK=/Users/fangjun/software/my-android/ndk/22.1.7171670
```

Note: Note from https://github.com/Tencent/ncnn/wiki/how-to-build#build-for-android

(Important) remove the hardcoded debug flag in Android NDK to fix the android-ndk issue: https://github.com/android/ndk/issues/243

1. open `$ANDROID_NDK/build/cmake/android.toolchain.cmake` for ndk < r23 or `$ANDROID_NDK/build/cmake/android-legacy.toolchain.cmake` for ndk >= r23
2. delete the line containing "-g"

```
list(APPEND ANDROID_COMPILER_FLAGS
-g
-DANDROID
```

Build sherpa-onnx (C++ code)

After installing `NDK`, it is time to build the C++ code of sherpa-onnx.

In the following, we show how to build sherpa-onnx for the following Android ABIs:

- `arm64-v8a`
- `armv7-eabi`
- `x86_64`
- `x86`

Caution: You only need to select one and only one ABI. `arm64-v8a` is probably the most common one.

If you want to test the app on an emulator, you probably need `x86_64`.

Hint: Building scripts for this section are for macOS and Linux. If you are using Windows or if you don't want to build the shared libraries by yourself, you can download pre-compiled shared libraries for this section by visiting

https://github.com/k2-fsa/sherpa-onnx/releases

Hint: We provide a colab notebook for you to try this section step by step.

If you are using Windows or you don't want to setup your local environment to build the C++ libraries, please use the above colab notebook.

Build for arm64-v8a

```
cd sherpa-onnx # Go to the root repo
./build-android-arm64-v8a.sh
```

After building, you will find the following shared libraries:

```
ls -lh build-android-arm64-v8a/install/lib/lib*.so
-rwxr-xr-x  1 fangjun  staff   848K Feb 26 15:54 build-android-arm64-v8a/install/lib/
↪libkaldi-native-fbank-core.so
-rw-r--r--@ 1 fangjun  staff    13M Feb 26 15:54 build-android-arm64-v8a/install/lib/
↪libonnxruntime.so
-rwxr-xr-x  1 fangjun  staff    29K Feb 26 15:54 build-android-arm64-v8a/install/lib/
↪libsherpa-onnx-c-api.so
-rwxr-xr-x  1 fangjun  staff   313K Feb 26 15:54 build-android-arm64-v8a/install/lib/
↪libsherpa-onnx-core.so
-rwxr-xr-x  1 fangjun  staff    34K Feb 26 15:54 build-android-arm64-v8a/install/lib/
↪libsherpa-onnx-jni.so
```

Please copy them to `android/SherpaOnnx/app/src/main/jniLibs/arm64-v8a/`:

```
cp build-android-arm64-v8a/install/lib/lib*.so  android/SherpaOnnx/app/src/main/jniLibs/
↪arm64-v8a/
```

You should see the following screen shot after running the above copy `cp` command.

Hint: You may see more files than it is shown in the screenshot. That is totally fine since we are extending sherpa-onnx.

The first thing to remember is to always use the wildcard `lib*.so` in the `cp` command.

Build for armv7-eabi

```
cd sherpa-onnx # Go to the root repo
./build-android-armv7-eabi.sh
```

After building, you will find the following shared libraries:

```
ls -lh build-android-armv7-eabi/install/lib/lib*.so

-rwxr-xr-x  1 fangjun  staff   513K Mar  4 21:48 build-android-armv7-eabi/install/lib/
↪libkaldi-native-fbank-core.so
-rw-r--r--  1 fangjun  staff   9.1M Mar  4 21:48 build-android-armv7-eabi/install/lib/
↪libonnxruntime.so
-rwxr-xr-x  1 fangjun  staff    19K Mar  4 21:48 build-android-armv7-eabi/install/lib/
↪libsherpa-onnx-c-api.so
-rwxr-xr-x  1 fangjun  staff   298K Mar  4 21:48 build-android-armv7-eabi/install/lib/
↪libsherpa-onnx-core.so
-rwxr-xr-x  1 fangjun  staff    22K Mar  4 21:48 build-android-armv7-eabi/install/lib/
↪libsherpa-onnx-jni.so
```

Please copy them to `android/SherpaOnnx/app/src/main/jniLibs/armeabi-v7a`:

```
cp build-android-armv7-eabi/install/lib/lib*.so android/SherpaOnnx/app/src/main/jniLibs/
↪armeabi-v7a/
```

You should see the following screen shot after running the above copy `cp` command.

Build for x86_64

```
cd sherpa-onnx # Go to the root repo
./build-android-x86-64.sh
```

After building, you will find the following shared libraries:

```
ls -lh build-android-x86-64/install/lib/lib*.so

-rwxr-xr-x  1 fangjun  staff   901K Feb 26 16:00 build-android-x86-64/install/lib/
↪libkaldi-native-fbank-core.so
-rw-r--r--@ 1 fangjun  staff    15M Feb 26 16:00 build-android-x86-64/install/lib/
↪libonnxruntime.so
-rwxr-xr-x  1 fangjun  staff   347K Feb 26 16:00 build-android-x86-64/install/lib/
↪libsherpa-onnx-core.so
-rwxr-xr-x  1 fangjun  staff    32K Feb 26 16:00 build-android-x86-64/install/lib/
↪libsherpa-onnx-jni.so
```

Please copy them to `android/SherpaOnnx/app/src/main/jniLibs/x86_64/`:

```
build-android-x86-64/install/lib/lib*.so android/SherpaOnnx/app/src/main/jniLibs/x86_64/
```

You should see the following screen shot after running the above copy `cp` command.

Build for x86

```
cd sherpa-onnx # Go to the root repo
./build-android-x86.sh
```

Download pre-trained models

Please read *Pre-trained models* for all available pre-trained models.

In the following, we use a pre-trained model *csukuangfj/sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20 (Bilingual, Chinese + English)*, which supports both Chinese and English.

Hint: The model is trained using icefall and the original torchscript model is from https://huggingface.co/pfluo/k2fsa-zipformer-chinese-english-mixed.

Use the following command to download the pre-trained model and place it into `android/SherpaOnnx/app/src/main/assets/`:

```
cd android/SherpaOnnx/app/src/main/assets/

sudo apt-get install git-lfs

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-bilingual-zh-en-2023-02-20.tar.bz2

tar xvf sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20.tar.bz2
rm sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20.tar.bz2

cd sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20

# Now, remove extra files to reduce the file size of the generated apk
rm -rf .git test_wavs
rm -f *.sh README.md
```

In the end, you should have the following files:

```
ls -lh

total 696984
-rw-r--r--  1 fangjun  staff    13M Feb 21 21:45 decoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   315M Feb 23 21:18 encoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff    12M Feb 21 21:45 joiner-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff    55K Feb 21 21:45 tokens.txt

du -h .

340M    .
```

You should see the following screen shot after downloading the pre-trained model:

Hint: If you select a different pre-trained model, make sure that you also change the corresponding code listed in the following screen shot:

SherpaOnnx
SherpaOnnx 〉 app 〉 src 〉 main 〉 jniLibs
app
Pixel 4 API 31
Android
app
manifests
java
java (generated)
assets
sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20
decoder-epoch-99-avg-1.onnx
encoder-epoch-99-avg-1.onnx
joiner-epoch-99-avg-1.onnx
tokens.txt
jniLibs
res
res (generated)
Gradle Scripts
build.gradle (Project: SherpaOnnx)
build.gradle (Module: SherpaOnnx.app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for SherpaOnnx.app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
You should see the following files after downloading the model
Search Everywhere Double ⇧
Go to File ⇧⌘O
Recent Files ⌘E
Navigation Bar ⌘↑
Drop files here to open them

SherpaOnnx – MainActivity.kt [SherpaOnnx.app.main]
java 〉 com 〉 k2fsa 〉 sherpa 〉 onnx 〉 MainActivity.kt 〉 MainActivity 〉 initModel
app
Pixel 4 API 31
Android
MainActivity.kt
com.k2fsa.sherpa.onnx
MainActivity.kt
SherpaOnnx.kt
WaveReader
com.k2fsa.sherpa.onnx (androidTest)
com.k2fsa.sherpa.onnx (test)
val numBytes = AudioRecord.getMinBufferSize(sampleRateInHz, channel
Log.i(
TAG, msg: "buffer size in milliseconds: ${numBytes * 1000.0f / sampleRateInHz}"
)
audioRecord = AudioRecord(
audioSource,
sampleRateInHz,
channelConfig,
audioFormat,
bufferSizeInBytes: numBytes * 2 // a sample has two bytes as we are using 16-bit PCM
)
return true
}
private fun initModel() {
// Please change getModelConfig() to add new models
// See https://k2-fsa.github.io/sherpa/onnx/pretrained_models/index.html
// for a list of available models
val type = 0
println("Select model type ${type}")
val config = OnlineRecognizerConfig(
featConfig = getFeatureConfig(sampleRate = 16000.0f, featureDim = 80),
modelConfig = getModelConfig(type = type)!!,
endpointConfig = getEndpointConfig(),
enableEndpoint = true
)
If you select a different model, please remember to change the variable "type"

Generate APK

Finally, it is time to build sherpa-onnx to generate an APK package.

Select `Build -> Make Project`, as shown in the following screen shot.

You can find the generated APK in `android/SherpaOnnx/app/build/outputs/apk/debug/app-debug.apk`:

```
ls -lh android/SherpaOnnx/app/build/outputs/apk/debug/app-debug.apk

-rw-r--r--  1 fangjun  staff   331M Feb 26 16:17 android/SherpaOnnx/app/build/outputs/
↪apk/debug/app-debug.apk
```

Congratulations! You have successfully built an APK for Android.

Read below to learn more.

Analyze the APK

Select `Build -> Analyze APK ...` in the above screen shot, in the popped-up dialog select the generated APK `app-debug.apk`, and you will see the following screen shot:

You can see from the above screen shot that most part of the APK is occupied by the pre-trained model, while the runtime, including the shared libraries, is only `5.4 MB`.

Caution: You can see that `libonnxruntime.so` alone occupies `5MB` out of `5.4MB`.

We use a so-called `Full build` instead of `Mobile build`, so the file size of the library is somewhat a bit larger.

`libonnxruntime.so` is donwloaded from

https://mvnrepository.com/artifact/com.microsoft.onnxruntime/onnxruntime-android/1.14.0

Please refer to https://onnxruntime.ai/docs/build/custom.html for a custom build to reduce the file size of `libonnxruntime.so`.

Note that we are constantly updating the version of `onnxruntime`. By the time you are reading this section, we may be using the latest version of `onnxruntime`.

Hint: We recommend you to use sherpa-ncnn. Please see *Analyze the APK* for sherpa-ncnn. The total runtime of sherpa-ncnn is only `1.6 MB`, which is much smaller than sherpa-onnx.

8.14 iOS

In this section, we describe how to build an iOS app for `real-time` speech recognition with sherpa-onnx and run it within a simulator on your Mac, run it on you iPhone or iPad.

Hint: During speech recognition, it does not need to access the Internet. Everyting is processed locally on your device.

8.14.1 Build sherpa-onnx for iOS

This section describes how to build sherpa-onnx for `iPhone` and `iPad`.

Requirement

Warning: The minimum deployment requires the iOS version >= `13.0`.

Before we continue, please make sure the following requirements are satisfied:

- macOS. It won't work on Windows or Linux.
- Xcode. The version `14.2 (14C18)` is known to work. Other versions may also work.
- CMake. CMake 3.25.1 is known to work. Other versions may also work.
- (Optional) iPhone or iPad. This is for testing the app on your device. If you don't have a device, you can still run the app within a simulator on your Mac.

Caution:

If you get the following error:

```
CMake Error at toolchains/ios.toolchain.cmake:544 (get_filename_component):
  get_filename_component called with incorrect number of arguments
Call Stack (most recent call first):
  /usr/local/Cellar/cmake/3.29.0/share/cmake/Modules/CMakeDetermineSystem.
↪cmake:146 (include)
  CMakeLists.txt:2 (project)
```

please run:

```
sudo xcode-select --install
sudo xcodebuild -license
```

And then delete the build directory `./build-ios` and re-build.

Please see also https://github.com/k2-fsa/sherpa-onnx/issues/702.

Download sherpa-onnx

First, let us download the source code of sherpa-onnx.

Note: In the following, I will download sherpa-onnx to `$HOME/open-source`, i.e., `/Users/fangjun/open-source`, on my Mac.

You can put it anywhere as you like.

```
mkdir -p $HOME/open-source
cd $HOME/open-source
git clone https://github.com/k2-fsa/sherpa-onnx
```

Build sherpa-onnx (in commandline, C++ Part)

After downloading sherpa-onnx, let us build the C++ part of sherpa-onnx.

```
cd $HOME/open-source/sherpa-onnx/
./build-ios.sh
```

It will generate a directory `$HOME/open-source/sherpa-onnx/build-ios`, which we have already pre-configured for you in Xcode.

Build sherpa-onnx (in Xcode)

Use the following command to open sherpa-onnx in Xcode:

```
cd $HOME/open-source/sherpa-onnx/ios-swift/SherpaOnnx
open SherpaOnnx.xcodeproj
```

It will start Xcode and you will see the following screenshot:

Please select `Product -> Build` to build the project. See the screenshot below:

After finishing the build, you should see the following screenshot:

Congratulations! You have successfully built the project. Let us run the project by selecting `Product -> Run`, which is shown in the following screenshot:

Please wait for a few seconds before Xcode starts the simulator.

Unfortunately, it will throw the following error:

The reason for the above error is that we have not provided the pre-trained model yet.

The file ViewController.swift pre-selects the pre-trained model to be *csukuangfj/sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20 (Bilingual, Chinese + English)*, shown in the screenshot below:

Fig. 8.5: Screenshot after running the command `open SherpaOnnx.xcodeproj`

Fig. 8.6: Screenshot for selecting `Product -> Build`

Fig. 8.7: Screenshot after finishing the build.

Fig. 8.8: Screenshot for `Product -> Run`.

Fig. 8.9: Screenshot for the error

Fig. 8.10: Screenshot for the pre-selected pre-trained model

Let us add the pre-trained model *csukuangfj/sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20 (Bilingual, Chinese + English)* to Xcode. Please follow *csukuangfj/sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20 (Bilingual, Chinese + English)* to download it from huggingface. You can download it to any directory as you like.

Please right click the project `SherpaOnnx` and select `Add Files to "SherpaOnnx"...` in the popup menu, as is shown in the screenshot below:

Fig. 8.11: Screenshot for adding files to SherpaOnnx

In the popup dialog, switch to the folder where you just downloaded the pre-trained model.

In the screenshot below, it is the folder `/Users/fangjun/open-source/icefall-models/sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20`:

Select required files and click the button `Add`:

After adding pre-trained model files to Xcode, you should see the following screenshot:

Fig. 8.12: Screenshot for navigating to the folder containing the downloaded pre-trained

Fig. 8.13: Screenshot for selecting required files

Fig. 8.14: Screenshot after add pre-trained model files

At this point, you should be able to select the menu `Product -> Run` to run the project and you should finally see the following screenshot:

Click the button to start recording! A screenshot is given below:

Congratulations! You have finally succeeded in running sherpa-onnx with iOS, though it is in a simulator.

Please read below if you want to run sherpa-onnx on your iPhone or iPad.

Run sherpa-onnx on your iPhone/iPad

First, please make sure the iOS version of your iPhone/iPad is >= `13.0`.

Click the menu `Xcode -> Settings...`, as is shown in the following screenshot:

In the popup dialog, please select `Account` and click + to add your Apple ID, as is shown in the following `screenshots`.

After adding your Apple ID, please connect your iPhone or iPad to your Mac and select your device in Xcode. The following screenshot is an example to select my iPhone.

Now your Xcode should look like below after selecting a device:

Fig. 8.15: Screenshot for a successful run.

Please select `Product -> Run` again to run sherpa-onnx on your selected device, as is shown in the following screenshot:

After a successful build, check your iPhone/iPad and you should see the following screenshot:

At this point, you should be able to run the app on your device. The following is a screenshot about running it on my iPhone:

Congratulations! You have successfully run sherpa-onnx on your device!

8.15 Flutter

8.15.1 Pre-built Flutter Apps

Links for pre-built Apps can be found in the following table:

Hint: It runs locally, without internet connection.

Fig. 8.16: Screenshot for recording and recognition.

Fig. 8.17: Screenshot for `Xcode -> Settings...`

Fig. 8.18: Screenshot for selecting `Account` and click +.

Text to speech (TTS, Speech synthesis)

		URL
Android (arm64-v8a, armeabi-v7a, x86_64)		https://k2-fsa.github.io/sherpa/onnx/flutter/tts-android.html
Linux (x64)		https://k2-fsa.github.io/sherpa/onnx/flutter/tts-linux.html
macOS (x64)		https://k2-fsa.github.io/sherpa/onnx/flutter/tts-macos-x64.html
macOS (arm64)		https://k2-fsa.github.io/sherpa/onnx/flutter/tts-macos-arm64.html
Windows (x64)		https://k2-fsa.github.io/sherpa/onnx/flutter/tts-win.html

Fig. 8.19: Screenshot for selecting `Apple ID` and click `Continue`

Streaming Speech recognition (STT, ASR)

		URL
Streaming speech recognition		https://k2-fsa.github.io/sherpa/onnx/flutter/asr/app.html

8.16 WebSocket

In this section, we describe how to use the WebSocket server and client for real-time speech recognition with sherpa-onnx.

The WebSocket server is implemented in C++ with the help of websocketpp and asio.

Hint: It does not depend on boost.

It does not depend on boost.

It does not depend on boost.

Fig. 8.20: Screenshot for adding your Apple ID and click `Next`

8.16.1 Streaming WebSocket server and client

Hint: Please refer to *Installation* to install sherpa-onnx before you read this section.

Build *sherpa-onnx* with WebSocket support

By default, it will generate the following binaries after *Installation*:

```
sherpa-onnx fangjun$ ls -lh build/bin/*websocket*
-rwxr-xr-x  1 fangjun  staff   1.1M Mar 31 22:09 build/bin/sherpa-onnx-offline-websocket-
↪server
-rwxr-xr-x  1 fangjun  staff   1.0M Mar 31 22:09 build/bin/sherpa-onnx-online-websocket-
↪client
-rwxr-xr-x  1 fangjun  staff   1.2M Mar 31 22:09 build/bin/sherpa-onnx-online-websocket-
↪server
```

Please refer to *Non-streaming WebSocket server and client* for the usage of `sherpa-onnx-offline-websocket-server`.

Fig. 8.21: Screenshot for entering your password and click `Next`

View the server usage

Before starting the server, let us view the help message of `sherpa-onnx-online-websocket-server`:

```
build/bin/sherpa-onnx-online-websocket-server
```

The above command will print the following help information:

```


Automatic speech recognition with sherpa-onnx using websocket.

Usage:

./bin/sherpa-onnx-online-websocket-server --help

./bin/sherpa-onnx-online-websocket-server \
  --port=6006 \
  --num-work-threads=5 \
  --tokens=/path/to/tokens.txt \
  --encoder=/path/to/encoder.onnx \
  --decoder=/path/to/decoder.onnx \
  --joiner=/path/to/joiner.onnx \
  --log-file=./log.txt \
  --max-batch-size=5 \
```

(continues on next page)

Fig. 8.22: Screenshot after adding your Apple ID

Fig. 8.23: Screenshot for selecting your device

Fig. 8.24: Screenshot after selecting your device

Fig. 8.25: Screenshot for selecting `Product -> Run`

Fig. 8.26: Screenshot for running sherpa-onnx on your device

Fig. 8.27: Screenshot for running sherpa-onnx on iPhone

(continued from previous page)

```
  --loop-interval-ms=10

Please refer to
https://k2-fsa.github.io/sherpa/onnx/pretrained_models/index.html
for a list of pre-trained models to download.

Options:
  --max-batch-size            : Max batch size for recognition. (int, default = 5)
  --loop-interval-ms          : It determines how often the decoder loop runs.  (int,␣
↪default = 10)
  --max-active-paths          : beam size used in modified beam search. (int, default =␣
↪4)
  --decoding-method           : decoding method,now support greedy_search and modified_
↪beam_search. (string, default = "greedy_search")
  --rule3-min-utterance-length : This endpointing rule3 requires utterance-length (in␣
↪seconds) to be >= this value. (float, default = 20)
  --rule3-min-trailing-silence : This endpointing rule3 requires duration of trailing␣
↪silence in seconds) to be >= this value. (float, default = 0)
  --rule3-must-contain-nonsilence : If True, for this endpointing rule3 to apply there␣
↪must be nonsilence in the best-path traceback. For decoding, a non-blank token is␣
↪considered as non-silence (bool, default = false)
  --rule2-min-utterance-length : This endpointing rule2 requires utterance-length (in␣
↪seconds) to be >= this value. (float, default = 0)
  --rule1-min-trailing-silence : This endpointing rule1 requires duration of trailing␣
↪silence in seconds) to be >= this value. (float, default = 2.4)
  --feat-dim                  : Feature dimension. Must match the one expected by the␣
↪model. (int, default = 80)
  --rule1-must-contain-nonsilence : If True, for this endpointing rule1 to apply there␣
↪must be nonsilence in the best-path traceback. For decoding, a non-blank token is␣
↪considered as non-silence (bool, default = false)
  --enable-endpoint           : True to enable endpoint detection. False to disable it.␣
↪(bool, default = true)
  --num_threads               : Number of threads to run the neural network (int,␣
↪default = 2)
  --debug                     : true to print model information while loading it. (bool,␣
↪default = false)
  --port                      : The port on which the server will listen. (int, default␣
↪= 6006)
  --num-io-threads            : Thread pool size for network connections. (int, default␣
↪= 1)
  --rule2-must-contain-nonsilence : If True, for this endpointing rule2 to apply there␣
↪must be nonsilence in the best-path traceback. For decoding, a non-blank token is␣
↪considered as non-silence (bool, default = true)
  --joiner                    : Path to joiner.onnx (string, default = "")
  --tokens                    : Path to tokens.txt (string, default = "")
  --num-work-threads          : Thread pool size for for neural network computation and␣
↪decoding. (int, default = 3)
  --encoder                   : Path to encoder.onnx (string, default = "")
  --sample-rate               : Sampling rate of the input waveform. Note: You can have␣
↪a different sample rate for the input waveform. We will do resampling inside the␣
↪feature extractor (int, default = 16000)
  --rule2-min-trailing-silence : This endpointing rule2 requires duration of trailing␣
↪silence in seconds) to be >= this value. (float, default = 1.2)
```

(continues on next page)

(continued from previous page)

```
  --log-file                  : Path to the log file. Logs are appended to this file␣
↪(string, default = "./log.txt")
  --rule1-min-utterance-length : This endpointing rule1 requires utterance-length (in␣
↪seconds) to be >= this value. (float, default = 0)
  --decoder                   : Path to decoder.onnx (string, default = "")

Standard options:
  --config                    : Configuration file to read (this option may be repeated)␣
↪(string, default = "")
  --help                      : Print out usage message (bool, default = false)
  --print-args                : Print the command line arguments (to stderr) (bool,␣
↪default = true)
```

Start the server

Hint: Please refer to *Pre-trained models* for a list of pre-trained models.

```
./build/bin/sherpa-onnx-online-websocket-server \
  --port=6006 \
  --num-work-threads=3 \
  --num-io-threads=2 \
  --tokens=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/encoder-epoch-
↪99-avg-1.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/decoder-epoch-
↪99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/joiner-epoch-99-
↪avg-1.onnx \
  --log-file=./log.txt \
  --max-batch-size=5 \
  --loop-interval-ms=20
```

Caution: The arguments are in the form `--key=value`.

It does not support `--key value`.

It does not support `--key value`.

It does not support `--key value`.

Hint: In the above demo, the model files are from *csukuangfj/sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20 (Bilingual, Chinese + English)*.

Note: Note that the server supports processing multiple clients in a batch in parallel. You can use `--max-batch-size`

to limit the batch size.

View the usage of the client (C++)

Let us view the usage of the C++ WebSocket client:

```
./build/bin/sherpa-onnx-online-websocket-client
```

The above command will print the following help information:

```
[I] /Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:484:int␣
↪sherpa_onnx::ParseOptions::Read(int, const char *const *) ./build/bin/sherpa-onnx-
↪online-websocket-client
[I] /Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:525:void␣
↪sherpa_onnx::ParseOptions::PrintUsage(bool) const

Automatic speech recognition with sherpa-onnx using websocket.

Usage:

./bin/sherpa-onnx-online-websocket-client --help

./bin/sherpa-onnx-online-websocket-client \
  --server-ip=127.0.0.1 \
  --server-port=6006 \
  --samples-per-message=8000 \
  --seconds-per-message=0.2 \
  /path/to/foo.wav

It support only wave of with a single channel, 16kHz, 16-bit samples.

Options:
  --seconds-per-message       : We will simulate that each message takes this number of␣
↪seconds to send. If you select a very large value, it will take a long time to send␣
↪all the samples (float, default = 0.2)
  --samples-per-message       : Send this number of samples per message. (int, default =␣
↪8000)
  --sample-rate               : Sample rate of the input wave. Should be the one␣
↪expected by the server (int, default = 16000)
  --server-port               : Port of the websocket server (int, default = 6006)
  --server-ip                 : IP address of the websocket server (string, default =
↪"127.0.0.1")

Standard options:
  --help                      : Print out usage message (bool, default = false)
  --print-args                : Print the command line arguments (to stderr) (bool,␣
↪default = true)
  --config                    : Configuration file to read (this option may be repeated)␣
↪(string, default = "")
```

Caution: We only support using IP address for `--server-ip`.

For instance, please don't use `--server-ip=localhost`, use `--server-ip=127.0.0.1` instead.

For instance, please don't use `--server-ip=localhost`, use `--server-ip=127.0.0.1` instead.

For instance, please don't use `--server-ip=localhost`, use `--server-ip=127.0.0.1` instead.

Start the client (C++)

To start the C++ WebSocket client, use:

```
build/bin/sherpa-onnx-online-websocket-client \
  --seconds-per-message=0.1 \
  --server-port=6006 \
  ./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/0.wav
```

Since the server is able to process multiple clients at the same time, you can use the following command to start multiple clients:

```
for i in $(seq 0 10); do
  k=$(expr $i % 5)
  build/bin/sherpa-onnx-online-websocket-client \
    --seconds-per-message=0.1 \
    --server-port=6006 \
    ./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/${k}.wav &
done

wait

echo "done"
```

View the usage of the client (Python)

Use the following command to view the usage:

```
python3 ./python-api-examples/online-websocket-client-decode-file.py  --help
```

Hint: `online-websocket-client-decode-file.py` is from https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/online-websocket-client-decode-file.py

It will print:

```
usage: online-websocket-client-decode-file.py [-h] [--server-addr SERVER_ADDR]
                                              [--server-port SERVER_PORT]
                                              [--samples-per-message SAMPLES_PER_MESSAGE]
                                              [--seconds-per-message SECONDS_PER_MESSAGE]
                                              sound_file

positional arguments:
  sound_file            The input sound file. Must be wave with a single
```

(continues on next page)

(continued from previous page)

```
                        channel, 16kHz sampling rate, 16-bit of each sample.

optional arguments:
  -h, --help            show this help message and exit
  --server-addr SERVER_ADDR
                        Address of the server (default: localhost)
  --server-port SERVER_PORT
                        Port of the server (default: 6006)
  --samples-per-message SAMPLES_PER_MESSAGE
                        Number of samples per message (default: 8000)
  --seconds-per-message SECONDS_PER_MESSAGE
                        We will simulate that the duration of two messages is
                        of this value (default: 0.1)
```

Hint: For the Python client, you can use either a domain name or an IP address for `--server-addr`. For instance, you can use either `--server-addr localhost` or `--server-addr 127.0.0.1`.

For the input argument, you can either use `--key=value` or `--key value`.

Start the client (Python)

```
python3 ./python-api-examples/online-websocket-client-decode-file.py \
  --server-addr localhost \
  --server-port 6006 \
  --seconds-per-message 0.1 \
  ./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/4.wav
```

Start the client (Python, with microphone)

```
python3 ./python-api-examples/online-websocket-client-microphone.py \
  --server-addr localhost \
  --server-port 6006

 ``online-websocket-client-microphone.py `` is from
 `<https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/online-
↪websocket-client-microphone.py>`_
```

8.16.2 Non-streaming WebSocket server and client

Hint: Please refer to *Installation* to install sherpa-onnx before you read this section.

Build *sherpa-onnx* with WebSocket support

By default, it will generate the following binaries after *Installation*:

```
sherpa-onnx fangjun$ ls -lh build/bin/*websocket*
-rwxr-xr-x  1 fangjun  staff   1.1M Mar 31 22:09 build/bin/sherpa-onnx-offline-websocket-
↪server
-rwxr-xr-x  1 fangjun  staff   1.0M Mar 31 22:09 build/bin/sherpa-onnx-online-websocket-
↪client
-rwxr-xr-x  1 fangjun  staff   1.2M Mar 31 22:09 build/bin/sherpa-onnx-online-websocket-
↪server
```

Please refer to *Streaming WebSocket server and client* for the usage of `sherpa-onnx-online-websocket-server` and `sherpa-onnx-online-websocket-client`.

View the server usage

Before starting the server, let us view the help message of `sherpa-onnx-offline-websocket-server`:

```
build/bin/sherpa-onnx-offline-websocket-server
```

The above command will print the following help information:

```
Automatic speech recognition with sherpa-onnx using websocket.

Usage:

./bin/sherpa-onnx-offline-websocket-server --help

(1) For transducer models

./bin/sherpa-onnx-offline-websocket-server \
  --port=6006 \
  --num-work-threads=5 \
  --tokens=/path/to/tokens.txt \
  --encoder=/path/to/encoder.onnx \
  --decoder=/path/to/decoder.onnx \
  --joiner=/path/to/joiner.onnx \
  --log-file=./log.txt \
  --max-batch-size=5

(2) For Paraformer

./bin/sherpa-onnx-offline-websocket-server \
  --port=6006 \
  --num-work-threads=5 \
  --tokens=/path/to/tokens.txt \
  --paraformer=/path/to/model.onnx \
  --log-file=./log.txt \
  --max-batch-size=5

Please refer to
https://k2-fsa.github.io/sherpa/onnx/pretrained_models/index.html
```

(continues on next page)

(continued from previous page)

```
for a list of pre-trained models to download.

Options:
  --log-file                  : Path to the log file. Logs are appended to this file␣
↪(string, default = "./log.txt")
  --max-utterance-length      : Max utterance length in seconds. If we receive an␣
↪utterance longer than this value, we will reject the connection. If you have enough␣
↪memory, you can select a large value for it. (float, default = 300)
  --decoding-method           : decoding method,Valid values: greedy_search. (string,␣
↪default = "greedy_search")
  --num-threads               : Number of threads to run the neural network (int,␣
↪default = 2)
  --feat-dim                  : Feature dimension. Must match the one expected by the␣
↪model. (int, default = 80)
  --port                      : The port on which the server will listen. (int, default␣
↪= 6006)
  --debug                     : true to print model information while loading it. (bool,␣
↪default = false)
  --joiner                    : Path to joiner.onnx (string, default = "")
  --tokens                    : Path to tokens.txt (string, default = "")
  --encoder                   : Path to encoder.onnx (string, default = "")
  --num-work-threads          : Thread pool size for for neural network computation and␣
↪decoding. (int, default = 3)
  --paraformer                : Path to model.onnx of paraformer. (string, default = "")
  --num-io-threads            : Thread pool size for network connections. (int, default␣
↪= 1)
  --max-batch-size            : Max batch size for decoding. (int, default = 5)
  --decoder                   : Path to decoder.onnx (string, default = "")

Standard options:
  --help                      : Print out usage message (bool, default = false)
  --print-args                : Print the command line arguments (to stderr) (bool,␣
↪default = true)
  --config                    : Configuration file to read (this option may be repeated)␣
↪(string, default = "")
```

Start the server

Hint: Please refer to *Pre-trained models* for a list of pre-trained models.

Start the server with a transducer model

```
./build/bin/sherpa-onnx-offline-websocket-server \
  --port=6006 \
  --num-work-threads=5 \
  --tokens=./sherpa-onnx-zipformer-en-2023-03-30/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-en-2023-03-30/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-en-2023-03-30/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-en-2023-03-30/joiner-epoch-99-avg-1.onnx \
  --log-file=./log.txt \
  --max-batch-size=5
```

Caution: The arguments are in the form `--key=value`.

It does not support `--key value`.

It does not support `--key value`.

It does not support `--key value`.

Hint: In the above demo, the model files are from *csukuangfj/sherpa-onnx-zipformer-en-2023-03-30 (English)*.

Note: Note that the server supports processing multiple clients in a batch in parallel. You can use `--max-batch-size` to limit the batch size.

Start the server with a paraformer model

```
./build/bin/sherpa-onnx-offline-websocket-server \
  --port=6006 \
  --num-work-threads=5 \
  --tokens=./sherpa-onnx-paraformer-zh-2023-03-28/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-zh-2023-03-28/model.onnx \
  --log-file=./log.txt \
  --max-batch-size=5
```

Hint: In the above demo, the model files are from *csukuangfj/sherpa-onnx-paraformer-zh-2023-03-28 (Chinese + English)*.

Start the client (Python)

We provide two clients written in Python:

- offline-websocket-client-decode-files-paralell.py: It decodes multiple files in parallel by creating a separate connection for each file
- offline-websocket-client-decode-files-sequential.py: It decodes multiple files sequentially by creating only a single connection

offline-websocket-client-decode-files-paralell.py

```
python3 ./python-api-examples/offline-websocket-client-decode-files-paralell.py \
  --server-addr localhost \
  --server-port 6006 \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav
```

offline-websocket-client-decode-files-sequential.py

```
python3 ./python-api-examples/offline-websocket-client-decode-files-sequential.py \
  --server-addr localhost \
  --server-port 6006 \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav
```

8.17 Hotwords (Contextual biasing)

In this section, we describe how we implement the hotwords (aka contextual biasing) feature with an Aho-corasick automaton and how to use it in sherpa-onnx.

8.17.1 What are hotwords

Current ASR systems work very well for general cases, but they sometimes fail to recognize special words/phrases (aka hotwords) like rare words, personalized information etc. Usually, those words/phrases will be recognized as the words/phrases that pronounce similar to them (for example, recognize `LOUIS FOURTEEN` as `LEWIS FOURTEEN`). So we have to provide some kind of contexts information (for example, the `LOUIS FOURTEEN`) to the ASR systems to boost those words/phrases. Normally, we call this kind of boosting task contextual biasing (aka hotwords recognition).

8.17.2 How do we implement it with an Aho-corasick

We first construct an Aho-corasick automaton on those given hotwords (after tokenizing into tokens). Please refer to https://en.wikipedia.org/wiki/Aho%E2%80%93Corasick_algorithm for the construction details of Aho-corasick.

The figure below is the aho-corasick on "HE/SHE/SHELL/HIS/THIS" with `hotwords-score==1`.

Fig. 8.28: The Aho-corasick for "HE/SHE/SHELL/HIS/THIS"

The `black` arrows in the graph are the goto arcs, the `red` arrows are the failure arcs, the `green` arrows are the output arcs. On each goto arc, there are token and boosting score (**Note: we will boost the path when any partial sequence is matched, if the path finally fails to full match any hotwords, the boosted score will be canceled**). Currentlly, the boosting score distributes on the arcs evenly along the path. On each state, there are two scores, the first one is the node score (mainly used to cancel score) the second one is output score, the output score is the total scores of the full matched hotwords of this state.

The following are several matching examples of the graph above.

Note: For simplicity, we assume that the system emits a token each frame.

Hint: We have an extra `finalize` step to force the graph state to go back to the root state.

The path is "SHELF"

Frame	Boost score	Total boost score	Graph state	Matched hotwords
init	0	0	0	
1	1	1	3	
2	1	2	4	
3	1 + 5	8	5	HE, SHE
4	1	9	6	
5	-4	5	0	
finalize	0	5	0	

At `frame 3` we reach `state 5` and match `HE, SHE`, so we get a boosting score `1 + 5`, the score `1` here because the `SHEL` still might be the prefix of other hotwords. At `frame 5 F` can not match any tokens and fail back to root, so we cancel the score for `SHEL` which is `4` (the node score of `state 6`).

The path is "HI"

Frame	Boost score	Total boost score	Graph state	Matched hotwords
init	0	0	0	
1	1	1	1	
2	1	2	8	
finalize	-2	0	0	

`H` and `I` all match the tokens in the graph, unfortunately, we have to go back to root state when finishing matching a path, so we cancel the boosting score of `HI` which is `2` (the node score of `state 8`).

The path is "THE"

Frame	Boost score	Total boost score	Graph state	Matched hotwords
init	0	0	0	
1	1	1	10	
2	1	2	11	
3	0 + 2	4	2	HE
finalize	-2	3	0	

At `frame 3` we jump from `state 11` to `state 2` and get a boosting score of `0 + 2`, `0` because the node score of `state 2` is the same as `state 11` so we don't get score by partial match (the prefix of `state 11` is `TH` has the same length of the prefix of `state 2` which is `HE`), but we do get the output score (at `state 2` it outputs `HE`).

Note: We implement the hotwords feature during inference time, you don't have to re-train the models to use this feature.

8.17.3 How to use hotwords in sherpa-onnx

Caution: Currentlly, the hotwords feature is only supported in the `modified_beam_search` decoding method of the **transducer models** (both streaming and non-streaming).

The use of the hotwords is no different for streaming and non-streaming models, and in fact it is even no different for all the API supported by sherpa onnx. We add **FOUR** extra arguments for hotwords:

- `hotwords-file`

  The file path of the hotwords, one hotwords per line. They could be Chinese words, English words or both according to the modeling units used to train the model. Here are some examples:

  For Chinese models trained on `cjkchar` it looks like:

  ```
  ```

  For English like language models trained on `bpe` it looks like:

  ```
  SPEECH RECOGNITION
  DEEP LEARNING
  ```

  For multilingual models trained on `cjkchar+bpe` (Chinese + English) it looks like:

```
SPEECH
SPEECH RECOGNITION
```

You can also specify the boosting score for each hotword, the score should follow the predefined character *:*, for example:

```
:3.5
:2.0
```

It means, hotword will have a boosting score of 3.5, hotword will have a boosting score of 2.0. For those hotwords that don't have specific scores, they will use the global score provided by *hotword-score* below.

> **Caution:** The specific score MUST BE the last item of each hotword (i.e You shouldn't break the hotword into two parts by the score). SPEECH :2.0 # This is invalid

- `hotwords-score`

  The boosting score for each matched token.

  > **Note:** We match the hotwords at token level, so the `hotwords-score` is applied at token level.

- `modeling-unit`

  The modeling unit of the used model, currently support *cjkchar* (for Chinese), *bpe* (for English like languages) and *cjkchar+bpe* (for multilingual models). We need this modeling-unit to select tokenizer to encode words/phrases into tokens, so **do provide correct modeling-unit according to your model**.

- `bpe-vocab`

  The bpe vocabulary generated by sentencepiece toolkit, it also can be exported from *bpe.model* (see *script/export_bpe_vocab.py* for details). This vocabulary is used to tokenize words/phrases into bpe units. It is only used when *modeling-unit* is *bpe* or *cjkchar+bpe*.

  > **Hint:** We need *bpe.vocab* rather than *bpe.model*, because we don't introduce sentencepiece c++ codebase into *sherpa-onnx* (which has a depandancy issue of protobuf) , we implement a simple sentencepiece encoder and decoder which takes *bpe.vocab* as input.

The main difference of using hotwords feature is about the modeling units. The following shows how to use it for different modeling units.

> **Hint:** You can use any transducer models here https://k2-fsa.github.io/sherpa/onnx/pretrained_models/index.html, we just choose three of them randomly for the following examples.

> **Note:** In the following example, we use a non-streaming model, if you are using a streaming model, you should use `sherpa-onnx`. `sherpa-onnx-alsa`, `sherpa-onnx-microphone`, `sherpa-onnx-microphone-offline`, `sherpa-onnx-online-websocket-server` and `sherpa-onnx-offline-websocket-server` all support hotwords.

Modeling unit is bpe

Download the model

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-en-2023-04-01.tar.bz2
tar xvf sherpa-onnx-zipformer-en-2023-04-01.tar.bz2
rm sherpa-onnx-zipformer-en-2023-04-01.tar.bz2
ln -s sherpa-onnx-zipformer-en-2023-04-01 exp
```

Export bpe.vocab if you can't find bpe.vocab in the model directory.

```
python script/export_bpe_vocab.py --bpe-model exp/bpe.model
```

The `hotwords_en.txt` contains:

```
QUARTERS
FOREVER
```

C++ api

Decoding without hotwords

```
./build/bin/sherpa-onnx-offline \
  --encoder=exp/encoder-epoch-99-avg-1.onnx \
  --decoder=exp/decoder-epoch-99-avg-1.onnx \
  --joiner=exp/joiner-epoch-99-avg-1.onnx \
  --decoding-method=modified_beam_search \
  --tokens=exp/tokens.txt \
  exp/test_wavs/0.wav exp/test_wavs/1.wav
```

The output is:

```
/star-kw/kangwei/code/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./build/bin/
↪sherpa-onnx-offline --encoder=exp/encoder-epoch-99-avg-1.onnx --decoder=exp/decoder-
↪epoch-99-avg-1.onnx --joiner=exp/joiner-epoch-99-avg-1.onnx --decoding-method=modified_
↪beam_search --tokens=exp/tokens.txt exp/test_wavs/0.wav exp/test_wavs/1.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_config=OfflineModelConfig(transducer=OfflineTran
↪$ducerModelConfig(encoder_filename="exp/encoder-epoch-99-avg-1.onnx", decoder_filename=
↪"exp/decoder-epoch-99-avg-1.onnx", joiner_filename="exp/joiner-epoch-99-$vg-1.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder=$", decoder="", language="", task="transcribe
↪"), tdnn=OfflineTdnnModelConfig(model=""), tokens="exp/tokens.txt", num_threads$2,␣
↪debug=False, provider="cpu", model_type=""), lm_config=OfflineLMConfig(model="",␣
↪scale=0.5), decoding_method="modified_beam_search", max_active_paths=4, ho$words_file=,
↪ hotwords_score=1.5)
Creating recognizer ...
Started
```

(continues on next page)

(continued from previous page)

```
Done!

exp/test_wavs/0.wav
{"text":"ALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUARTER OF THE␣
↪BROTHELS","timestamps":"[1.44, 1.48, 1.56, 1.72, 1.88, 1.96, 2.16, 2.28$ 2.36, 2.48, 2.
↪60, 2.80, 3.08, 3.28, 3.40, 3.60, 3.80, 4.08, 4.24, 4.32, 4.48, 4.64, 4.84, 4.88, 5.00,
↪ 5.08, 5.32, 5.48, 5.60, 5.68, 5.84, 6.04, 6.24]","token$":["A","LL"," THE"," YE","LL",
↪"OW"," LA","M","P","S"," WOULD"," LIGHT"," UP"," HE","RE"," AND"," THERE"," THE"," S",
↪"QUA","LI","D"," ","QUA","R","TER"," OF","THE"," B","RO","TH","EL","S"]}
----
exp/test_wavs/1.wav
{"text":"IN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY CHILD WHOSE PLACE WAS ON THAT␣
↪SAME DISHONOURED BOSOM TO CONNECT HER PARENT FOR EVER WITH THE RACE AN
D DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN","timestamps":"[2.44, 2.
↪64, 2.88, 3.16, 3.28, 3.48, 3.60, 3.80, 3.96, 4.12, 4.36, 4.52, 4.72, 4
.92, 5.16, 5.44, 5.68, 6.04, 6.24, 6.48, 6.84, 7.08, 7.32, 7.56, 7.84, 8.12, 8.24, 8.32,␣
↪8.44, 8.60, 8.76, 8.88, 9.08, 9.28, 9.44, 9.56, 9.64, 9.76, 9.96, 10.0
4, 10.20, 10.40, 10.64, 10.76, 11.04, 11.20, 11.36, 11.60, 11.80, 12.00, 12.12, 12.28,␣
↪12.32, 12.52, 12.72, 12.84, 12.96, 13.04, 13.24, 13.40, 13.60, 13.76, 13
.96, 14.12, 14.24, 14.36, 14.52, 14.68, 14.76, 15.04, 15.28, 15.52, 15.76, 16.00, 16.16,␣
↪16.24, 16.32]","tokens":["IN"," WHICH"," MAN"," TH","US"," P","UN","IS
H","ED"," HAD"," GIVE","N"," HER"," A"," LOVE","LY"," CHILD"," WHO","SE"," PLACE"," WAS",
↪" ON"," THAT"," SAME"," DIS","HO","N","OUR","ED"," BO","S","OM"," TO",
" CON","NE","C","T"," HER"," P","AR","ENT"," FOR"," E","VER"," WITH"," THE"," RA","CE","␣
↪AND"," DE","S","C","ENT"," OF"," MO","R","T","AL","S"," AND"," TO"," B
E"," FI","N","AL","LY"," A"," B","LESS","ED"," SO","UL"," IN"," HE","A","VE","N"]}
----
num threads: 2
decoding method: modified_beam_search
max active paths: 4
Elapsed seconds: 1.775 s
Real time factor (RTF): 1.775 / 23.340 = 0.076
```

Decoding with hotwords

```
./build/bin/sherpa-onnx-offline \
    --encoder=exp/encoder-epoch-99-avg-1.onnx \
    --decoder=exp/decoder-epoch-99-avg-1.onnx \
    --joiner=exp/joiner-epoch-99-avg-1.onnx \
    --decoding-method=modified_beam_search \
    --tokens=exp/tokens.txt \
    --modeling-unit=bpe \
    --bpe-vocab=exp/bpe.vocab \
    --hotwords-file=hotwords_en.txt \
    --hotwords-score=2.0 \
    exp/test_wavs/0.wav exp/test_wavs/1.wav
```

The output is:

```
/star-kw/kangwei/code/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./build/bin/
↪sherpa-onnx-offline --encoder=exp/encoder-epoch-99-avg-1.onnx --decoder=exp/decoder-
↪epoch-99-avg-1.onnx --joiner=exp/joiner-epoch-99-avg-1.onnx --decoding-method=modified_
↪beam_search --tokens=exp/tokens.txt --hotwords-file=hotwords_en.txt --hotwords-score=2.
↪0 exp/test_wavs/0.wav exp/test_wavs/1.wav
```

(continues on next page)

(continued from previous page)

```
OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename=
↪"exp/encoder-epoch-99-avg-1.onnx", decoder_filename="exp/decoder-epoch-99-avg-1.onnx",␣
↪joiner_filename="exp/joiner-epoch-99-avg-1.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="
", decoder="", language="", task="transcribe"), tdnn=OfflineTdnnModelConfig(model=""),␣
↪tokens="exp/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type=""),␣
↪lm_config=OfflineLMConfig(model="", scale=0.5), decoding_method="modified_beam_search",
↪ max_active_paths=4, hotwords_file=hotwords_en.txt, hotwords_score=2)
Creating recognizer ...
Started
Done!

exp/test_wavs/0.wav
{"text":"ALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUARTERS OF THE␣
↪BROTHELS","timestamps":"[1.44, 1.48, 1.56, 1.72, 1.88, 1.96, 2.16, 2.28
, 2.36, 2.48, 2.60, 2.80, 3.08, 3.28, 3.40, 3.60, 3.80, 4.08, 4.24, 4.32, 4.48, 4.64, 4.
↪84, 4.88, 5.00, 5.08, 5.12, 5.36, 5.48, 5.60, 5.68, 5.84, 6.04, 6.24]",
"tokens":["A","LL"," THE"," YE","LL","OW"," LA","M","P","S"," WOULD"," LIGHT"," UP"," HE
↪","RE"," AND"," THERE"," THE"," S","QUA","LI","D"," ","QUA","R","TER",$
S"," OF"," THE"," B","RO","TH","EL","S"]}
----
exp/test_wavs/1.wav
{"text":"IN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY CHILD WHOSE PLACE WAS ON THAT␣
↪SAME DISHONOURED BOSOM TO CONNECT HER PARENT FOREVER WITH THE RACE AN$
 DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN","timestamps":"[2.44, 2.
↪64, 2.88, 3.16, 3.28, 3.48, 3.60, 3.80, 3.96, 4.12, 4.36, 4.52, 4.72, 4$
92, 5.16, 5.44, 5.68, 6.04, 6.24, 6.48, 6.84, 7.08, 7.32, 7.56, 7.84, 8.12, 8.24, 8.32,␣
↪8.44, 8.60, 8.76, 8.88, 9.08, 9.28, 9.44, 9.56, 9.64, 9.76, 9.96, 10.0$
, 10.20, 10.40, 10.68, 10.76, 11.04, 11.20, 11.36, 11.60, 11.80, 12.00, 12.12, 12.28, 12.
↪32, 12.52, 12.72, 12.84, 12.96, 13.04, 13.24, 13.40, 13.60, 13.76, 13$
96, 14.12, 14.24, 14.36, 14.52, 14.68, 14.76, 15.04, 15.28, 15.52, 15.76, 16.00, 16.16,␣
↪16.24, 16.32]","tokens":["IN"," WHICH"," MAN"," TH","US"," P","UN","IS$
","ED"," HAD"," GIVE","N"," HER"," A"," LOVE","LY"," CHILD"," WHO","SE"," PLACE"," WAS",
↪" ON"," THAT"," SAME"," DIS","HO","N","OUR","ED"," BO","S","OM"," TO",$
 CON","NE","C","T"," HER"," P","AR","ENT"," FOR","E","VER"," WITH"," THE"," RA","CE","␣
↪AND"," DE","S","C","ENT"," OF"," MO","R","T","AL","S"," AND"," TO"," BE$
," FI","N","AL","LY"," A"," B","LESS","ED"," SO","UL"," IN"," HE","A","VE","N"]}
----
num threads: 2
decoding method: modified_beam_search
max active paths: 4
Elapsed seconds: 1.522 s
Real time factor (RTF): 1.522 / 23.340 = 0.065
```

Hint: QUARTER -> QUARTERS

```
FOR EVER -> FOREVER
```

Python api

Decoding without hotwords

```
python python-api-examples/offline-decode-files.py \
    --encoder exp/encoder-epoch-99-avg-1.onnx \
    --decoder exp/decoder-epoch-99-avg-1.onnx \
    --joiner exp/joiner-epoch-99-avg-1.onnx \
    --decoding modified_beam_search \
    --tokens exp/tokens.txt \
    exp/test_wavs/0.wav exp/test_wavs/1.wav
```

The output is:

```
Started!
Done!
exp/test_wavs/0.wav
ALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUARTER OF THE BROTHELS
----------
exp/test_wavs/1.wav
IN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY CHILD WHOSE PLACE WAS ON THAT SAME␣
↪DISHONOURED BOSOM TO CONNECT HER PARENT FOR EVER WITH THE RACE AND DESCENT OF MORTALS␣
↪AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
----------
num_threads: 1
decoding_method: modified_beam_search
Wave duration: 23.340 s
Elapsed time: 2.546 s
Real time factor (RTF): 2.546/23.340 = 0.109
```

Decoding with hotwords

```
python python-api-examples/offline-decode-files.py \
    --encoder exp/encoder-epoch-99-avg-1.onnx \
    --decoder exp/decoder-epoch-99-avg-1.onnx \
    --joiner exp/joiner-epoch-99-avg-1.onnx \
    --decoding modified_beam_search \
    --tokens exp/tokens.txt \
    --modeling-unit bpe \
    --bpe-vocab exp/bpe.vocab \
    --hotwords-file hotwords_en.txt \
    --hotwords-score 2.0 \
    exp/test_wavs/0.wav exp/test_wavs/1.wav
```

The output is:

```
Started!
Done!
exp/test_wavs/0.wav
ALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUARTERS OF THE BROTHELS
```

(continues on next page)

(continued from previous page)

```
----------
exp/test_wavs/1.wav
IN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY CHILD WHOSE PLACE WAS ON THAT SAME␣
↪DISHONOURED BOSOM TO CONNECT HER PARENT FOREVER WITH THE RACE AND DESCENTOF MORTALS␣
↪AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
----------
num_threads: 1
decoding_method: modified_beam_search
Wave duration: 23.340 s
Elapsed time: 2.463 s
Real time factor (RTF): 2.463/23.340 = 0.106
```

Hint: QUARTER -> QUARTERS

FOR EVER -> FOREVER

Modeling unit is cjkchar

Download the model

```
cd /path/to/sherpa-onnx
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪conformer-zh-stateless2-2023-05-23.tar.bz2
tar xvf sherpa-onnx-conformer-zh-stateless2-2023-05-23.tar.bz2
rm sherpa-onnx-conformer-zh-stateless2-2023-05-23.tar.bz2

ln -s sherpa-onnx-conformer-zh-stateless2-2023-05-23 exp-zh
```

The `hotwords_cn.txt` contains:

```

```

C++ api

Decoding without hotwords

```
./build/bin/sherpa-onnx-offline \
    --encoder=exp-zh/encoder-epoch-99-avg-1.onnx \
    --decoder=exp-zh/decoder-epoch-99-avg-1.onnx \
    --joiner=exp-zh/joiner-epoch-99-avg-1.onnx \
    --tokens=exp-zh/tokens.txt \
    --decoding-method=modified_beam_search \
    exp-zh/test_wavs/3.wav exp-zh/test_wavs/4.wav exp-zh/test_wavs/5.wav exp-zh/test_
↪wavs/6.wav
```

The output is:

```
/star-kw/kangwei/code/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./build/bin/
↪sherpa-onnx-offline --encoder=exp-zh/encoder-epoch-99-avg-1.onnx --decoder=exp-zh/
↪decoder-epoch-99-avg-1.onnx --joiner=exp-zh/joiner-epoch-99-avg-1.onnx --tokens=exp-zh/
↪tokens.txt --decoding-method=modified_beam_search exp-zh/test_wavs/3.wav exp-zh/test_
↪wavs/4.wav exp-zh/test_wavs/5.wav exp-zh/test_wavs/6.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename=
↪"exp-zh/encoder-epoch-99-avg-1.onnx", decoder_filename="exp-zh/decoder-epoch-99-avg-1.
↪onnx", joiner_filename="exp-zh/joiner-$poch-99-avg-1.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), whisper=OfflineWhisperModelConfig
↪$encoder="", decoder="", language="", task="transcribe"),␣
↪tdnn=OfflineTdnnModelConfig(model=""), tokens="exp-zh/tokens.txt", num_threads=2,␣
↪debug=False, provider="cpu", model_type=""), lm_config=OfflineLMConfig(model="",␣
↪scale=0.5), decoding_method="modified_beam_search", max_active$paths=4, hotwords_file=,
↪ hotwords_score=1.5)
Creating recognizer ...
Started
Done!

exp-zh/test_wavs/3.wav
{"text":"","timestamps":"[0.00, 0.16, 0.68, 1.32, 1.72, 2.08, 2.60, 2.88, 3.20, 3.52, 3.
↪92, 4.40, 4.68, 5.12, 5.44, 6.36, $.96, 7.32]","tokens":["","","","","","","","","","",
↪"","","","","","","","","]}
----
exp-zh/test_wavs/4.wav
{"text":"","timestamps":"[0.00, 0.20, 0.88, 1.36, 1.76, 2.08, 2.28, 2.68, 2.92, 3.16, 3.
↪44, 3.80]","tokens":["","","","","",$
","","","","","","]}
----
exp-zh/test_wavs/5.wav
{"text":"","timestamps":"[0.00, 0.16, 0.88, 1.24, 1.64, 1.96, 2.76, 3.04, 3.32]","tokens
↪":["","","","","","","","","]}
----
exp-zh/test_wavs/6.wav
{"text":"","timestamps":"[0.00, 0.16, 0.80, 1.12, 1.44, 1.68, 1.92, 2.16, 2.36, 2.60, 2.
↪84, 3.12]","tokens":["","","","","",$
","","","","","","]}
----
num threads: 2
decoding method: modified_beam_search
max active paths: 4
Elapsed seconds: 1.883 s
Real time factor (RTF): 1.883 / 20.328 = 0.093
```

Decoding with hotwords

```
./build/bin/sherpa-onnx-offline \
  --encoder=exp-zh/encoder-epoch-99-avg-1.onnx \
  --decoder=exp-zh/decoder-epoch-99-avg-1.onnx \
  --joiner=exp-zh/joiner-epoch-99-avg-1.onnx \
```

(continues on next page)

(continued from previous page)

```
  --tokens=exp-zh/tokens.txt \
  --decoding-method=modified_beam_search \
  --modeling-unit=cjkchar \
  --hotwords-file=hotwords_cn.txt \
  --hotwords-score=2.0 \
  exp-zh/test_wavs/3.wav exp-zh/test_wavs/4.wav exp-zh/test_wavs/5.wav exp-zh/test_wavs/
↪6.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename=
↪"exp-zh/encoder-epoch-99-avg-1.onnx", decoder_filename="exp-zh/decoder-epoch-99-avg-1.
↪onnx", joiner_filename="exp-zh/joiner-$poch-99-avg-1.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), whisper=OfflineWhisperModelConfig
↪$encoder="", decoder="", language="", task="transcribe"),␣
↪tdnn=OfflineTdnnModelConfig(model=""), tokens="exp-zh/tokens.txt", num_threads=2,␣
↪debug=False, provider="cpu", model_type=""), lm_config=OfflineLMConfig(model="",␣
↪scale=0.5), decoding_method="modified_beam_search", max_active$paths=4, hotwords_
↪file=hotwords_cn.txt, hotwords_score=2)
Creating recognizer ...
Started
Done!

exp-zh/test_wavs/3.wav
{"text":"","timestamps":"[0.00, 0.16, 0.64, 1.28, 1.64, 2.04, 2.60, 2.88, 3.20, 3.52, 3.
↪92, 4.40, 4.68, 5.12, 5.44, 6.36, $.96, 7.32]","tokens":["","","","","","","","","","",
↪"","","","","","","","","]}
----
exp-zh/test_wavs/4.wav
{"text":"","timestamps":"[0.00, 0.12, 0.80, 1.36, 1.76, 2.08, 2.28, 2.68, 2.92, 3.16, 3.
↪44, 3.80]","tokens":["","","","","","$
","","","","","","","]}
----
exp-zh/test_wavs/5.wav
{"text":"","timestamps":"[0.00, 0.12, 0.80, 1.24, 1.56, 1.96, 2.68, 3.08, 3.32]","tokens
↪":["","","","","","","","","","]}
----
exp-zh/test_wavs/6.wav
{"text":"","timestamps":"[0.00, 0.12, 0.80, 1.12, 1.44, 1.68, 1.92, 2.16, 2.36, 2.60, 2.
↪84, 3.12]","tokens":["","","","","","$
","","","","","","","]}
----
num threads: 2
decoding method: modified_beam_search
max active paths: 4
Elapsed seconds: 1.810 s
Real time factor (RTF): 1.810 / 20.328 = 0.089
```

Hint: ->

->

```
->

->
```

Python api

Decoding without hotwords

```
python python-api-examples/offline-decode-files.py \
--encoder exp-zh/encoder-epoch-99-avg-1.onnx \
--decoder exp-zh/decoder-epoch-99-avg-1.onnx \
--joiner exp-zh/joiner-epoch-99-avg-1.onnx \
--tokens exp-zh/tokens.txt \
--decoding-method modified_beam_search \
exp-zh/test_wavs/3.wav exp-zh/test_wavs/4.wav exp-zh/test_wavs/5.wav exp-zh/test_wavs/6.
↪wav
```

The output is:

```
Started!
Done!
exp-zh/test_wavs/3.wav

----------
exp-zh/test_wavs/4.wav

----------
exp-zh/test_wavs/5.wav

----------
exp-zh/test_wavs/6.wav

----------
num_threads: 1
decoding_method: modified_beam_search
Wave duration: 20.328 s
Elapsed time: 2.653 s
Real time factor (RTF): 2.653/20.328 = 0.131
```

Decoding with hotwords

```
python python-api-examples/offline-decode-files.py \
    --encoder exp-zh/encoder-epoch-99-avg-1.onnx \
    --decoder exp-zh/decoder-epoch-99-avg-1.onnx \
    --joiner exp-zh/joiner-epoch-99-avg-1.onnx \
    --tokens exp-zh/tokens.txt \
    --decoding-method modified_beam_search \
    --modeling-unit=cjkchar \
    --hotwords-file hotwords_cn.txt \
    --hotwords-score 2.0 \
    exp-zh/test_wavs/3.wav exp-zh/test_wavs/4.wav exp-zh/test_wavs/5.wav exp-zh/test_
↪wavs/6.wav
```

The output is:

```
Started!
Done!
exp-zh/test_wavs/3.wav

----------
exp-zh/test_wavs/4.wav

----------
exp-zh/test_wavs/5.wav

----------
exp-zh/test_wavs/6.wav

----------
num_threads: 1
decoding_method: modified_beam_search
Wave duration: 20.328 s
Elapsed time: 2.636 s
Real time factor (RTF): 2.636/20.328 = 0.130
```

Hint: ->

->

->

->

Modeling unit is cjkchar+bpe

Download the model

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-bilingual-zh-en-2023-02-20.tar.bz2
tar xvf sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20.tar.bz2
rm sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20.tar.bz2

ln -s sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20 exp-mixed
```

Export bpe.vocab if you can't find bpe.vocab in the model directory.

```
python script/export_bpe_vocab.py --bpe-model exp/bpe.model
```

The `hotwords_mix.txt` contains:

```

```

C++ api

Decoding without hotwords

```
./build/bin/sherpa-onnx \
    --encoder=exp-mixed/encoder-epoch-99-avg-1.onnx \
    --decoder=exp-mixed/decoder-epoch-99-avg-1.onnx \
    --joiner=exp-mixed/joiner-epoch-99-avg-1.onnx \
    --decoding-method=modified_beam_search \
    --tokens=exp-mixed/tokens.txt \
    exp-mixed/test_wavs/0.wav exp-mixed/test_wavs/2.wav
```

The output is:

```
/star-kw/kangwei/code/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./build/bin/
↪sherpa-onnx --encoder=exp-mixed/encoder-epoch-99-avg-1.onnx --decoder=exp-mixed/
↪decoder-epoch-99-avg-1.onnx --joiner=exp-mixed/joiner-epoch-99-avg-1.onnx --decoding-
↪method=modified_beam_search --tokens=exp-mixed/tokens.txt exp-mixed/test_wavs/0.wav␣
↪exp-mixed/test_wavs/2.wav
OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"exp-mixed/encoder-epoch-99-avg-1.onnx", decoder="exp-mixed/decoder-epoch-99-avg-1.onnx
↪", joiner="exp-mixed/joiner-epoch-99-avg-1.onnx"),␣
↪paraformer=OnlineParaformerModelConfig(encoder="", decoder=""), tokens="exp-mixed/
↪tokens.txt", num_threads=1, debug=False, provider="cpu", model_type=""), lm_
↪config=OnlineLMConfig(model="", scale=0.5), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=True, max_active_paths=4, hotwords_score=1.5, hotwords_file="", decoding_
↪method="modified_beam_search")

exp-mixed/test_wavs/0.wav
Elapsed seconds: 3, Real time factor (RTF): 0.3
 MONDAY TODAY IS LIBR THE DAY AFTER TOMORROW
{"is_final":false,"segment":0,"start_time":0.0,"text":" MONDAY TODAY IS LIBR THE DAY␣
↪AFTER TOMORROW","timestamps":"[0.64, 1.04, 1.60, 2.08, 2.20, 2.40, 4.16, 4.40, 4.88, 5.
↪56, 5.80, 6.16, 6.84, 7.12, 7.44, 8.04, 8.16, 8.24, 8.28, 9.04, 9.40, 9.64, 9.88]",
↪"tokens":["","","","" MO","N","DAY"," TO","DAY"," IS"," LI","B","R"," THE"," DAY","␣
↪AFTER"," TO","M","OR","ROW","","","",""]}

exp-mixed/test_wavs/2.wav
Elapsed seconds: 1.7, Real time factor (RTF): 0.37
 FREQUENTLY
{"is_final":false,"segment":0,"start_time":0.0,"text":" FREQUENTLY","timestamps":"[0.00,␣
↪0.40, 0.52, 0.96, 1.08, 1.28, 1.48, 1.68, 1.84, 2.00, 2.24, 2.36, 2.52, 2.68, 2.92, 3.
↪00, 3.12, 3.32, 3.64, 3.96, 4.36]","tokens":["","","","","","","","","","","","","","" F
↪","RE","QU","ENT","LY","","",""]}
```

Decoding with hotwords

```
./build/bin/sherpa-onnx \
```

(continues on next page)

(continued from previous page)

```
    --encoder=exp-mixed/encoder-epoch-99-avg-1.onnx \
    --decoder=exp-mixed/decoder-epoch-99-avg-1.onnx \
    --joiner=exp-mixed/joiner-epoch-99-avg-1.onnx \
    --decoding-method=modified_beam_search \
    --tokens=exp-mixed/tokens.txt \
    --modeling-unit=cjkchar+bpe \
    --bpe-vocab=exp/bpe.vocab \
    --hotwords-file=hotwords_mix.txt \
    --hotwords-score=2.0 \
    exp-mixed/test_wavs/0.wav exp-mixed/test_wavs/2.wav
```

The output is:

```
/star-kw/kangwei/code/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./build/bin/
↪sherpa-onnx --encoder=exp-mixed/encoder-epoch-99-avg-1.onnx --decoder=exp-mixed/
↪decoder-epoch-99-avg-1.onnx --joiner=exp-mixed/joiner-epoch-99-avg-1.onnx --decoding-
↪method=modified_beam_search --tokens=exp-mixed/tokens.txt --tokens-type=cjkchar+bpe --
↪bpe-model=exp-mixed/bpe.model --hotwords-file=hotwords_mix.txt --hotwords-score=2.0␣
↪exp-mixed/test_wavs/0.wav exp-mixed/test_wavs/2.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"exp-mixed/encoder-epoch-99-avg-1.onnx", decoder="exp-mixed/decoder-epoch-99-avg-1.onnx
↪", joiner="exp-mixed/joiner-epoch-99-avg-1.onnx"),␣
↪paraformer=OnlineParaformerModelConfig(encoder="", decoder=""), tokens="exp-mixed/
↪tokens.txt", num_threads=1, debug=False, provider="cpu", model_type=""), lm_
↪config=OnlineLMConfig(model="", scale=0.5), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=True, max_active_paths=4, hotwords_score=2, hotwords_file="hotwords_mix.txt",␣
↪decoding_method="modified_beam_search")

exp-mixed/test_wavs/0.wav
Elapsed seconds: 3.2, Real time factor (RTF): 0.32
 MONDAY TODAY IS THE DAY AFTER TOMORROW
{"is_final":false,"segment":0,"start_time":0.0,"text":" MONDAY TODAY IS THE DAY AFTER␣
↪TOMORROW","timestamps":"[0.64, 1.04, 1.60, 2.08, 2.20, 2.40, 4.16, 4.40, 4.88, 5.56, 5.
↪68, 6.00, 6.84, 7.12, 7.44, 8.04, 8.16, 8.24, 8.28, 9.04, 9.40, 9.64, 9.88]","tokens":[
↪"","","","," MO","N","DAY"," TO","DAY"," IS","","","," THE"," DAY"," AFTER"," TO","M",
↪"OR","ROW","","","","]}

exp-mixed/test_wavs/2.wav
Elapsed seconds: 1.9, Real time factor (RTF): 0.4
 FREQUENTLY
{"is_final":false,"segment":0,"start_time":0.0,"text":" FREQUENTLY","timestamps":"[0.00,␣
↪0.40, 0.52, 0.96, 1.08, 1.28, 1.48, 1.68, 1.84, 2.00, 2.24, 2.36, 2.52, 2.68, 2.92, 3.
↪00, 3.12, 3.32, 3.64, 3.96, 4.36]","tokens":["","","","","","","","","","","","","","," F
↪","RE","QU","ENT","LY","","","]}
```

Hint: LIBR ->

```
->
```

Python api

Decoding without hotwords

```
python python-api-examples/online-decode-files.py \
  --encoderexp-mixed/encoder-epoch-99-avg-1.onnx \
  --decoder exp-mixed/decoder-epoch-99-avg-1.onnx \
  --joiner exp-mixed/joiner-epoch-99-avg-1.onnx \
  --decoding-method modified_beam_search \
  --tokens exp-mixed/tokens.txt
  exp-mixed/test_wavs/0.wav exp-mixed/test_wavs/2.wav
```

The output is:

```
Started!
Done!
exp-mixed/test_wavs/0.wav
 MONDAY TODAY IS LIBR THE DAY AFTER TOMORROW
----------
exp-mixed/test_wavs/2.wav
 FREQUENTLY
----------
num_threads: 1
decoding_method: modified_beam_search
Wave duration: 14.743 s
Elapsed time: 3.052 s
Real time factor (RTF): 3.052/14.743 = 0.207
```

Decoding with hotwords

```
python python-api-examples/online-decode-files.py \
    --encoder exp-mixed/encoder-epoch-99-avg-1.onnx \
    --decoder exp-mixed/decoder-epoch-99-avg-1.onnx \
    --joiner exp-mixed/joiner-epoch-99-avg-1.onnx \
    --decoding-method modified_beam_search \
    --tokens exp-mixed/tokens.txt \
    --modeling-unit cjkchar+bpe \
    --bpe-vocab exp-mixed/bpe.vocab \
    --hotwords-file hotwords_mix.txt \
    --hotwords-score 2.0 \
    exp-mixed/test_wavs/0.wav exp-mixed/test_wavs/2.wav
```

The output is:

```
Started!
Done!
exp-mixed/test_wavs/0.wav
 MONDAY TODAY IS THE DAY AFTER TOMORROW
----------
exp-mixed/test_wavs/2.wav
```

(continues on next page)

(continued from previous page)

```
 FREQUENTLY
----------
num_threads: 1
decoding_method: modified_beam_search
Wave duration: 14.743 s
Elapsed time: 3.060 s
Real time factor (RTF): 3.060/14.743 = 0.208
```

Hint: `LIBR ->`

`->`

8.18 Keyword spotting

In this section, we describe how we implement the open vocabulary keyword spotting (aka customized keyword spotting) feature and how to use it in sherpa-onnx.

8.18.1 What is open vocabulary keyword spotting

Basically, an open vocabulary keyword spotting system is just like a tiny ASR system, but it can only decode words/phrases in the given keywords. For example, if the given keyword is `HELLO WORLD`, then the decoded result should be either `HELLO WORLD` or empty. As for open vocabulary (or customized), it means you can specify any keywords without re-training the model. For building a conventional keyword spotting systems, people need to prepare a lot of audio-text pairs for the selected keywords and the trained model can only be used to detect those selected keywords. While an open vocabulary keyword spotting system allows people using one system to detect different keywords, even the keywords might not be in the training data.

8.18.2 Decoder for open vocabulary keyword spotting

For now, we only implement a beam search decoder to make the system only trigger the given keywords (i.e. the model itself is actually a tiny ASR). To make it is able to balance between the `trigged rate` and `false alarm`, we introduce two parameters for each keyword, `boosting score` and `trigger threshold`. The `boosting score` works like the hotwords recognition, it help the paths containing keywords to survive beam search, the larger this score is the easier the corresponding keyword will be triggered, read *Hotwords (Contextual biasing)* for more details. The `trigger threshold` defines the minimum acoustic probability of decoded sequences (token sequences) that can be triggered, it is a float value between 0 to 1, the lower this threshold is the easier the corresponding keyword will be triggered.

8.18.3 Keywords file

The input keywords looks like (the keywords are `HELLO WORLD`, `HI GOOGLE` and `HEY SIRI`):

```
HE LL O WORLD :1.5 #0.35
HI GO O G LE :1.0 #0.25
HE Y S I RI
```

Each line contains a keyword, the first several tokens (separated by spaces) are encoded tokens of the keyword, the item starts with `:` is the `boosting score` and the item starts with `#` is the `trigger threshold`. Note: No spaces between `:` (or `#`) and the float value.

To get the tokens you need to use the command line tool in sherpa-onnx to convert the original keywords, you can see the usage as follows:

```
sherpa-onnx-cli text2token --help
Usage: sherpa-onnx-cli text2token [OPTIONS] INPUT OUTPUT

Options:

  --text TEXT         Path to the input texts. Each line in the texts contains the␣
↪original phrase, it might also contain some extra items,
                      for example, the boosting score (startting with :), the triggering␣
↪threshold
                      (startting with #, only used in keyword spotting task) and the␣
↪original phrase (startting with @).
                      Note: extra items will be kept in the output.

                      example input 1 (tokens_type = ppinyin):
                           :2.0 #0.6 @
                           :3.5 @
                           #0.6 @
                      example output 1:
                          x iǎo ài tóng x ué :2.0 #0.6 @
                          n ǐ h ǎo w èn w èn :3.5 @
                          x iǎo y ì x iǎo y ì #0.6 @

                      example input 2 (tokens_type = bpe):
                          HELLO WORLD :1.5 #0.4
                          HI GOOGLE :2.0 #0.8
                          HEY SIRI #0.35
                      example output 2:
                          HE LL O WORLD :1.5 #0.4
                          HI GO O G LE :2.0 #0.8
                          HE Y S I RI #0.35

  --tokens TEXT       The path to tokens.txt.
  --tokens-type TEXT  The type of modeling units, should be cjkchar, bpe, cjkchar+bpe,␣
↪fpinyin or ppinyin.
                      fpinyin means full pinyin, each cjkchar has a pinyin(with tone).␣
↪ppinyin
                      means partial pinyin, it splits pinyin into initial and final,
  --bpe-model TEXT    The path to bpe.model. Only required when tokens-type is bpe or␣
↪cjkchar+bpe.
```

(continues on next page)

(continued from previous page)

```
  --help              Show this message and exit.
```

Note: If the tokens-type is `fpinyin` or `ppinyin`, you MUST provide the original keyword (starting with `@`).

Note: If you install sherpa-onnx from sources (i.e. not by pip), you can use the alternative script in *scripts*, the usage is almost the same as the command line tool, read the help information by:

```
python3 scripts/text2token.py --help
```

8.18.4 How to use keyword spotting in sherpa-onnx

Currently, we provide command-line tool and android app for keyword spotting.

command-line tool

After installing sherpa-onnx, type `sherpa-onnx-keyword-spotter --help` for the help message.

8.18.5 Android application

You can find pre-built Android APKs for keyword spotting at

https://k2-fsa.github.io/sherpa/onnx/kws/apk.html

Here is a demo video (Note: It is in Chinese).

8.18.6 Pretrained models

You can find the pre-trained models in sherpa-onnx-kws-pre-trained-models.

8.19 Punctuation

This section introduces the models that sherpa-onnx supports for adding punctuations to text.

Hint: After getting text from speech using speech-to-text, you can use models from this section to add punctuations to text.

8.19.1 Pre-trained models

This section lists pre-trained models for adding punctuations to text.

You can find all models at the following URL:

https://github.com/k2-fsa/sherpa-onnx/releases/tag/punctuation-models

sherpa-onnx-punct-ct-transformer-zh-en-vocab272727-2024-04-12

This model is converted from

https://modelscope.cn/models/iic/punc_ct-transformer_zh-cn-common-vocab272727-pytorch/summary

and it supports both Chinese and English.

Hint: If you want to know how the model is converted to sherpa-onnx, please download it and you can find related scripts in the downloaded model directory.

In the following, we describe how to download and use it with sherpa-onnx.

Download the model

Please use the following commands to download it:

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/punctuation-models/sherpa-
↪onnx-punct-ct-transformer-zh-en-vocab272727-2024-04-12.tar.bz2

# For Chinese users, you can aso use the following mirror:
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/punctuation-models/
↪sherpa-onnx-punct-ct-transformer-zh-en-vocab272727-2024-04-12.tar.bz2

tar xvf sherpa-onnx-punct-ct-transformer-zh-en-vocab272727-2024-04-12.tar.bz2
rm sherpa-onnx-punct-ct-transformer-zh-en-vocab272727-2024-04-12.tar.bz2
```

You will find the following files after unzipping:

```
-rw-r--r--  1 fangjun  staff   1.4K Apr 12 12:32 README.md
-rwxr-xr-x  1 fangjun  staff   1.6K Apr 12 14:40 add-model-metadata.py
-rw-r--r--  1 fangjun  staff   810B Apr 12 11:56 config.yaml
-rw-r--r--  1 fangjun  staff    42B Apr 12 11:45 configuration.json
-rw-r--r--  1 fangjun  staff   281M Apr 12 14:40 model.onnx
-rwxr-xr-x  1 fangjun  staff   745B Apr 12 11:53 show-model-input-output.py
-rwxr-xr-x  1 fangjun  staff   4.9K Apr 13 18:45 test.py
-rw-r--r--  1 fangjun  staff   4.0M Apr 12 11:56 tokens.json
```

Only `model.onnx` is needed in sherpa-onnx. All other files are for your information about how the model is converted to sherpa-onnx.

C++ binary examples

After installing sherpa-onnx, you can use the following command to add punctuations to text:

```
./bin/sherpa-onnx-offline-punctuation \
  --ct-transformer=./sherpa-onnx-punct-ct-transformer-zh-en-vocab272727-2024-04-12/model.
↪onnx \
  ""
```

The output is given below:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./bin/
↪sherpa-onnx-offline-punctuation --ct-transformer=./sherpa-onnx-punct-ct-transformer-zh-
↪en-vocab272727-2024-04-12/model.onnx ''

OfflinePunctuationConfig(model=OfflinePunctuationModelConfig(ct_transformer="./sherpa-
↪onnx-punct-ct-transformer-zh-en-vocab272727-2024-04-12/model.onnx", num_threads=1,␣
↪debug=False, provider="cpu"))
Creating OfflinePunctuation ...
Started
Done
Num threads: 1
Elapsed seconds: 0.007 s
Input text:
Output text:
```

The second example is for text containing both Chinese and English:

```
./bin/sherpa-onnx-offline-punctuation \
  --ct-transformer=./sherpa-onnx-punct-ct-transformer-zh-en-vocab272727-2024-04-12/model.
↪onnx \
  "How are youthank you are you ok"
```

Its output is given below:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./bin/
↪sherpa-onnx-offline-punctuation --ct-transformer=./sherpa-onnx-punct-ct-transformer-zh-
↪en-vocab272727-2024-04-12/model.onnx 'How are youthank you are you ok'

OfflinePunctuationConfig(model=OfflinePunctuationModelConfig(ct_transformer="./sherpa-
↪onnx-punct-ct-transformer-zh-en-vocab272727-2024-04-12/model.onnx", num_threads=1,␣
↪debug=False, provider="cpu"))
Creating OfflinePunctuation ...
Started
Done
Num threads: 1
Elapsed seconds: 0.005 s
Input text: How are youthank you are you ok
Output text: How are youthank youare you ok
```

The last example is for text containing only English:

```
./bin/sherpa-onnx-offline-punctuation \
  --ct-transformer=./sherpa-onnx-punct-ct-transformer-zh-en-vocab272727-2024-04-12/model.
↪onnx \
```

(continues on next page)

(continued from previous page)

```
  "The African blogosphere is rapidly expanding bringing more voices online in the form␣
↪of commentaries opinions analyses rants and poetry"
```

Its output is given below:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./bin/
↪sherpa-onnx-offline-punctuation --ct-transformer=./sherpa-onnx-punct-ct-transformer-zh-
↪en-vocab272727-2024-04-12/model.onnx 'The African blogosphere is rapidly expanding␣
↪bringing more voices online in the form of commentaries opinions analyses rants and␣
↪poetry'

OfflinePunctuationConfig(model=OfflinePunctuationModelConfig(ct_transformer="./sherpa-
↪onnx-punct-ct-transformer-zh-en-vocab272727-2024-04-12/model.onnx", num_threads=1,␣
↪debug=False, provider="cpu"))
Creating OfflinePunctuation ...
Started
Done
Num threads: 1
Elapsed seconds: 0.003 s
Input text: The African blogosphere is rapidly expanding bringing more voices online in␣
↪the form of commentaries opinions analyses rants and poetry
Output text: The African blogosphere is rapidly expandingbringing more voices online in␣
↪the form of commentariesopinionsanalysesrants and poetry
```

Python API examples

Please see

https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/add-punctuation.py

Huggingface space examples

Please see

- https://huggingface.co/spaces/k2-fsa/generate-subtitles-for-videos
- https://huggingface.co/spaces/k2-fsa/automatic-speech-recognition

Hint: For Chinese users, please visit the following mirrors:

- https://hf-mirror.com/spaces/k2-fsa/generate-subtitles-for-videos
- https://hf-mirror.com/spaces/k2-fsa/automatic-speech-recognition

Video demos

The following video is in Chinese.

8.20 Audio tagging

This section introduces the models that sherpa-onnx supports for audio tagging, which aims to recognize sound events within an audio clip without its temporal localization.

8.20.1 Pre-trained models

This section lists pre-trained models for audio tagging.

You can find all models at the following URL:

https://github.com/k2-fsa/sherpa-onnx/releases/tag/audio-tagging-models

sherpa-onnx-zipformer-small-audio-tagging-2024-04-15

This model is trained by https://github.com/k2-fsa/icefall/pull/1421 using the dataset audioset.

In the following, we describe how to download and use it with sherpa-onnx.

Download the model

Please use the following commands to download it:

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/audio-tagging-models/sherpa-
↪onnx-zipformer-small-audio-tagging-2024-04-15.tar.bz2

# For Chinese users, you can aso use the following mirror:
wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/audio-tagging-models/
↪sherpa-onnx-zipformer-small-audio-tagging-2024-04-15.tar.bz2

tar xvf sherpa-onnx-zipformer-small-audio-tagging-2024-04-15.tar.bz2
rm sherpa-onnx-zipformer-small-audio-tagging-2024-04-15.tar.bz2
```

You will find the following files after unzipping:

```
-rw-r--r--   1 fangjun  staff   243B Apr 15 16:14 README.md
-rw-r--r--   1 fangjun  staff    14K Apr 15 16:14 class_labels_indices.csv
-rw-r--r--   1 fangjun  staff    26M Apr 15 16:14 model.int8.onnx
-rw-r--r--   1 fangjun  staff    88M Apr 15 16:14 model.onnx
drwxr-xr-x  15 fangjun  staff   480B Apr 15 16:14 test_wavs
```

C++ binary examples

Hint: You can find the binary executable file `sherpa-onnx-offline-audio-tagging` after installing sherpa-onnx either from source or using ``pip install sherpa-onnx``_.

Cat

For the following test wave,

the command:

```
./bin/sherpa-onnx-offline-audio-tagging \
  --zipformer-model=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/model.int8.
↪onnx \
  --labels=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/class_labels_indices.
↪csv \
  ./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/test_wavs/1.wav
```

prints the following:

```
0: AudioEvent(name="Animal", index=72, prob=0.947886)
1: AudioEvent(name="Cat", index=81, prob=0.938876)
2: AudioEvent(name="Domestic animals, pets", index=73, prob=0.931975)
3: AudioEvent(name="Caterwaul", index=85, prob=0.178876)
4: AudioEvent(name="Meow", index=83, prob=0.176177)
Num threads: 1
Wave duration: 10.000
Elapsed seconds: 0.297 s
Real time factor (RTF): 0.297 / 10.000 = 0.030
```

Hint: By default, it outputs the top 5 events. The first event has the largest probability.

Whistle

For the following test wave,

the command:

```
./bin/sherpa-onnx-offline-audio-tagging \
  --zipformer-model=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/model.int8.
↪onnx \
  --labels=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/class_labels_indices.
↪csv \
  ./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/test_wavs/2.wav
```

prints the following:

```
0: AudioEvent(name="Whistling", index=40, prob=0.804928)
1: AudioEvent(name="Music", index=137, prob=0.27548)
2: AudioEvent(name="Piano", index=153, prob=0.135418)
3: AudioEvent(name="Keyboard (musical)", index=152, prob=0.0580414)
4: AudioEvent(name="Musical instrument", index=138, prob=0.0400399)
Num threads: 1
Wave duration: 10.000
Elapsed seconds: 0.289 s
Real time factor (RTF): 0.289 / 10.000 = 0.029
```

Music

For the following test wave,

the command:

```
./bin/sherpa-onnx-offline-audio-tagging \
  --zipformer-model=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/model.int8.
↪onnx \
  --labels=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/class_labels_indices.
↪csv \
  ./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/test_wavs/3.wav
```

prints the following:

```
0: AudioEvent(name="Music", index=137, prob=0.79673)
1: AudioEvent(name="A capella", index=255, prob=0.765521)
2: AudioEvent(name="Singing", index=27, prob=0.473899)
3: AudioEvent(name="Vocal music", index=254, prob=0.459337)
4: AudioEvent(name="Choir", index=28, prob=0.458174)
Num threads: 1
Wave duration: 10.000
Elapsed seconds: 0.279 s
Real time factor (RTF): 0.279 / 10.000 = 0.028
```

Laughter

For the following test wave,

the command:

```
./bin/sherpa-onnx-offline-audio-tagging \
  --zipformer-model=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/model.int8.
↪onnx \
  --labels=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/class_labels_indices.
↪csv \
  ./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/test_wavs/4.wav
```

prints the following:

```
0: AudioEvent(name="Laughter", index=16, prob=0.929239)
1: AudioEvent(name="Snicker", index=19, prob=0.321969)
2: AudioEvent(name="Giggle", index=18, prob=0.149667)
3: AudioEvent(name="Inside, small room", index=506, prob=0.119332)
4: AudioEvent(name="Belly laugh", index=20, prob=0.100728)
Num threads: 1
Wave duration: 10.000
Elapsed seconds: 0.314 s
Real time factor (RTF): 0.314 / 10.000 = 0.031
```

Finger snapping

For the following test wave,

the command:

```
./bin/sherpa-onnx-offline-audio-tagging \
  --zipformer-model=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/model.int8.
↪onnx \
  --labels=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/class_labels_indices.
↪csv \
  ./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/test_wavs/5.wav
```

prints the following:

```
0: AudioEvent(name="Finger snapping", index=62, prob=0.690543)
1: AudioEvent(name="Slap, smack", index=467, prob=0.452133)
2: AudioEvent(name="Clapping", index=63, prob=0.179213)
3: AudioEvent(name="Sound effect", index=504, prob=0.101151)
4: AudioEvent(name="Whack, thwack", index=468, prob=0.0294559)
Num threads: 1
Wave duration: 8.284
Elapsed seconds: 0.225 s
Real time factor (RTF): 0.225 / 8.284 = 0.027
```

Baby cry

For the following test wave,

the command:

```
./bin/sherpa-onnx-offline-audio-tagging \
  --zipformer-model=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/model.int8.
↪onnx \
  --labels=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/class_labels_indices.
↪csv \
  ./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/test_wavs/6.wav
```

prints the following:

```
0: AudioEvent(name="Baby cry, infant cry", index=23, prob=0.912273)
1: AudioEvent(name="Crying, sobbing", index=22, prob=0.670927)
2: AudioEvent(name="Whimper", index=24, prob=0.187221)
3: AudioEvent(name="Inside, small room", index=506, prob=0.0314955)
4: AudioEvent(name="Sound effect", index=504, prob=0.0118726)
Num threads: 1
Wave duration: 8.719
Elapsed seconds: 0.232 s
Real time factor (RTF): 0.232 / 8.719 = 0.027
```

Smoke alarm

For the following test wave,

the command:

```
./bin/sherpa-onnx-offline-audio-tagging \
  --zipformer-model=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/model.int8.
↪onnx \
  --labels=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/class_labels_indices.
↪csv \
  ./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/test_wavs/7.wav
```

prints the following:

```
0: AudioEvent(name="Smoke detector, smoke alarm", index=399, prob=0.781478)
1: AudioEvent(name="Beep, bleep", index=481, prob=0.641056)
2: AudioEvent(name="Buzzer", index=398, prob=0.218576)
3: AudioEvent(name="Fire alarm", index=400, prob=0.140145)
4: AudioEvent(name="Alarm", index=388, prob=0.012525)
Num threads: 1
Wave duration: 2.819
Elapsed seconds: 0.080 s
Real time factor (RTF): 0.080 / 2.819 = 0.028
```

Siren

For the following test wave,

the command:

```
./bin/sherpa-onnx-offline-audio-tagging \
  --zipformer-model=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/model.int8.
↪onnx \
  --labels=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/class_labels_indices.
↪csv \
  ./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/test_wavs/8.wav
```

prints the following:

```
0: AudioEvent(name="Siren", index=396, prob=0.877108)
1: AudioEvent(name="Civil defense siren", index=397, prob=0.732789)
2: AudioEvent(name="Vehicle", index=300, prob=0.0113797)
3: AudioEvent(name="Inside, small room", index=506, prob=0.00537381)
4: AudioEvent(name="Outside, urban or manmade", index=509, prob=0.00261939)
Num threads: 1
Wave duration: 7.721
Elapsed seconds: 0.220 s
Real time factor (RTF): 0.220 / 7.721 = 0.028
```

Stream water

For the following test wave,

the command:

```
./bin/sherpa-onnx-offline-audio-tagging \
  --zipformer-model=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/model.int8.
↪onnx \
  --labels=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/class_labels_indices.
↪csv \
  ./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/test_wavs/10.wav
```

prints the following:

```
0: AudioEvent(name="Stream", index=292, prob=0.247785)
1: AudioEvent(name="Water", index=288, prob=0.231587)
2: AudioEvent(name="Gurgling", index=297, prob=0.170981)
3: AudioEvent(name="Trickle, dribble", index=450, prob=0.108859)
4: AudioEvent(name="Liquid", index=444, prob=0.0693812)
Num threads: 1
Wave duration: 7.837
Elapsed seconds: 0.212 s
Real time factor (RTF): 0.212 / 7.837 = 0.027
```

Meow

For the following test wave,

the command:

```
./bin/sherpa-onnx-offline-audio-tagging \
  --zipformer-model=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/model.int8.
↪onnx \
  --labels=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/class_labels_indices.
↪csv \
  ./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/test_wavs/11.wav
```

prints the following:

```
0: AudioEvent(name="Meow", index=83, prob=0.814944)
1: AudioEvent(name="Cat", index=81, prob=0.698858)
2: AudioEvent(name="Domestic animals, pets", index=73, prob=0.564516)
3: AudioEvent(name="Animal", index=72, prob=0.535303)
4: AudioEvent(name="Music", index=137, prob=0.105332)
Num threads: 1
Wave duration: 11.483
Elapsed seconds: 0.361 s
Real time factor (RTF): 0.361 / 11.483 = 0.031
```

Dog bark

For the following test wave,

the command:

```
./bin/sherpa-onnx-offline-audio-tagging \
  --zipformer-model=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/model.int8.
↪onnx \
  --labels=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/class_labels_indices.
↪csv \
  ./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/test_wavs/12.wav
```

prints the following:

```
0: AudioEvent(name="Animal", index=72, prob=0.688237)
1: AudioEvent(name="Dog", index=74, prob=0.637803)
2: AudioEvent(name="Bark", index=75, prob=0.608597)
3: AudioEvent(name="Bow-wow", index=78, prob=0.515501)
4: AudioEvent(name="Domestic animals, pets", index=73, prob=0.495074)
Num threads: 1
Wave duration: 8.974
Elapsed seconds: 0.261 s
Real time factor (RTF): 0.261 / 8.974 = 0.029
```

Oink (pig)

For the following test wave,

the command:

```
./bin/sherpa-onnx-offline-audio-tagging \
  --zipformer-model=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/model.int8.
↪onnx \
  --labels=./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/class_labels_indices.
↪csv \
  ./sherpa-onnx-zipformer-small-audio-tagging-2024-04-15/test_wavs/13.wav
```

prints the following:

```
0: AudioEvent(name="Oink", index=94, prob=0.888416)
1: AudioEvent(name="Pig", index=93, prob=0.164295)
2: AudioEvent(name="Animal", index=72, prob=0.160802)
3: AudioEvent(name="Speech", index=0, prob=0.0276513)
4: AudioEvent(name="Snort", index=46, prob=0.0201952)
Num threads: 1
Wave duration: 9.067
Elapsed seconds: 0.261 s
Real time factor (RTF): 0.261 / 9.067 = 0.029
```

Python API examples

Please see

https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/audio-tagging-from-a-file.py

Huggingface space

You can try audio tagging with sherpa-onnx from within you browser by visiting the following URL:

https://huggingface.co/spaces/k2-fsa/audio-tagging

Note: For Chinese users, please use

https://hf-mirror.com/spaces/k2-fsa/audio-tagging

8.20.2 Android

You can find Android APKs for each model at the following page

https://k2-fsa.github.io/sherpa/onnx/audio-tagging/apk.html

Please follow *Android* to build Android APKs from source.

If you want to run audio tagging on your WearOS watches, please see *WearOS*.

8.20.3 WearOS

You can find APKs for WearOS of each model at the following page

https://k2-fsa.github.io/sherpa/onnx/audio-tagging/apk-wearos.html

Please follow *Android* to build APKs for WearOS from source.

If you want to run audio tagging on your Android phones, please see *Android*.

8.21 Spoken language identification

This section describes how to use sherpa-onnx for spoken language identification.

8.21.1 Pre-trained models

whisper

Currently, we support whisper multilingual models for spoken language identification.

Model type	Huggingface repo
`tiny`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-tiny
`base`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-base
`small`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-small
`medium`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-medium

Hint: You can also download them from

https://github.com/k2-fsa/sherpa-onnx/releases/tag/asr-models

In the following, we use the `tiny` model as an example. You can replace `tiny` with `base`, `small`, or `medium` and everything still holds.

Download the model

Please use the following commands to download the `tiny` model:

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪whisper-tiny.tar.bz2

# For Chinese users, please use
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪whisper-tiny.tar.bz2

tar xvf sherpa-onnx-whisper-tiny.tar.bz2
rm sherpa-onnx-whisper-tiny.tar.bz2
```

You should find the following files after unzipping:

```
-rw-r--r--  1 fangjun  staff   427B Jan 31 16:21 README.md
-rwxr-xr-x  1 fangjun  staff    19K Jan 31 16:21 export-onnx.py
-rw-r--r--  1 fangjun  staff    15B Jan 31 16:21 requirements.txt
-rwxr-xr-x  1 fangjun  staff    12K Jan 31 16:21 test.py
drwxr-xr-x  6 fangjun  staff   192B Jan 31 16:22 test_wavs
-rw-r--r--  1 fangjun  staff    86M Jan 31 16:22 tiny-decoder.int8.onnx
-rw-r--r--  1 fangjun  staff   109M Jan 31 16:22 tiny-decoder.onnx
-rw-r--r--  1 fangjun  staff    12M Jan 31 16:22 tiny-encoder.int8.onnx
-rw-r--r--  1 fangjun  staff    36M Jan 31 16:22 tiny-encoder.onnx
-rw-r--r--  1 fangjun  staff   798K Jan 31 16:22 tiny-tokens.txt
```

Download test waves

Please use the following command to download test data:

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/spoken-language-
↪identification-test-wavs.tar.bz2

# For Chinese users, please use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/spoken-
↪language-identification-test-wavs.tar.bz2

tar xvf spoken-language-identification-test-wavs.tar.bz2
rm spoken-language-identification-test-wavs.tar.bz2
```

You can find the following test files after unzipping:

```
-rw-r--r--  1 fangjun  staff   222K Mar 24 12:51 ar-arabic.wav
-rw-r--r--@ 1 fangjun  staff   137K Mar 24 13:09 bg-bulgarian.wav
-rw-r--r--  1 fangjun  staff    83K Mar 24 13:07 cs-czech.wav
-rw-r--r--  1 fangjun  staff   112K Mar 24 13:07 da-danish.wav
-rw-r--r--  1 fangjun  staff   199K Mar 24 12:50 de-german.wav
-rw-r--r--  1 fangjun  staff   207K Mar 24 13:06 el-greek.wav
-rw-r--r--  1 fangjun  staff    31K Mar 24 12:45 en-english.wav
-rw-r--r--@ 1 fangjun  staff    77K Mar 24 12:23 es-spanish.wav
-rw-r--r--@ 1 fangjun  staff   371K Mar 24 12:21 fa-persian.wav
-rw-r--r--  1 fangjun  staff   136K Mar 24 13:08 fi-finnish.wav
-rw-r--r--  1 fangjun  staff   112K Mar 24 12:49 fr-french.wav
-rw-r--r--  1 fangjun  staff   179K Mar 24 12:47 hi-hindi.wav
-rw-r--r--@ 1 fangjun  staff   177K Mar 24 12:29 hr-croatian.wav
-rw-r--r--  1 fangjun  staff   167K Mar 24 12:53 id-indonesian.wav
-rw-r--r--  1 fangjun  staff   136K Mar 24 12:54 it-italian.wav
-rw-r--r--  1 fangjun  staff    46K Mar 24 12:44 ja-japanese.wav
-rw-r--r--@ 1 fangjun  staff   122K Mar 24 12:52 ko-korean.wav
-rw-r--r--  1 fangjun  staff    85K Mar 24 12:54 nl-dutch.wav
-rw-r--r--@ 1 fangjun  staff   241K Mar 24 12:38 no-norwegian.wav
-rw-r--r--@ 1 fangjun  staff   121K Mar 24 12:35 po-polish.wav
-rw-r--r--  1 fangjun  staff   166K Mar 24 12:48 pt-portuguese.wav
-rw-r--r--@ 1 fangjun  staff   144K Mar 24 12:33 ro-romanian.wav
-rw-r--r--  1 fangjun  staff   111K Mar 24 12:51 ru-russian.wav
-rw-r--r--@ 1 fangjun  staff   239K Mar 24 12:40 sk-slovak.wav
-rw-r--r--  1 fangjun  staff   196K Mar 24 13:01 sv-swedish.wav
-rw-r--r--  1 fangjun  staff   106K Mar 24 13:14 ta-tamil.wav
-rw-r--r--  1 fangjun  staff   104K Mar 24 13:02 tl-tagalog.wav
-rw-r--r--  1 fangjun  staff    76K Mar 24 13:00 tr-turkish.wav
-rw-r--r--  1 fangjun  staff   188K Mar 24 13:05 uk-ukrainian.wav
-rw-r--r--  1 fangjun  staff   181K Mar 24 13:20 zh-chinese.wav
```

Test with Python APIs

After installing sherpa-onnx either from source or from using `pip install sherpa-onnx`, you can run:

```
python3 ./python-api-examples/spoken-language-identification.py \
  --whisper-encoder ./sherpa-onnx-whisper-tiny/tiny-encoder.int8.onnx \
  --whisper-decoder ./sherpa-onnx-whisper-tiny/tiny-decoder.onnx \
  ./spoken-language-identification-test-wavs/de-german.wav
```

You should see the following output:

```
2024-04-17 15:53:23,104 INFO [spoken-language-identification.py:158] File: ./spoken-
↪language-identification-test-wavs/de-german.wav
2024-04-17 15:53:23,104 INFO [spoken-language-identification.py:159] Detected language:␣
↪de
2024-04-17 15:53:23,104 INFO [spoken-language-identification.py:160] Elapsed seconds: 0.
↪275
2024-04-17 15:53:23,105 INFO [spoken-language-identification.py:161] Audio duration in␣
↪seconds: 6.374
2024-04-17 15:53:23,105 INFO [spoken-language-identification.py:162] RTF: 0.275/6.374 =␣
↪0.043
```

Hint: You can find `spoken-language-identification.py` at

https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/spoken-language-identification.py

Android APKs

You can find pre-built Android APKs for spoken language identification at the following address:

https://k2-fsa.github.io/sherpa/onnx/spoken-language-identification/apk.html

Huggingface space

We provide a huggingface space for spoken language identification.

You can visit the following URL:

http://huggingface.co/spaces/k2-fsa/spoken-language-identification

Note: For Chinese users, you can use the following mirror:

http://hf-mirror.com/spaces/k2-fsa/spoken-language-identification

8.22 VAD

We support silero-vad for voice activity detection.

You can find pre-built Android APKs for VAD at:

https://k2-fsa.github.io/sherpa/onnx/vad/apk.html

APKs for VAD + speech recognition can be found at:

https://k2-fsa.github.io/sherpa/onnx/vad/apk-asr.html

8.23 Pre-trained models

The following table lists links for all pre-trained models.

Description	URL
Speech recognition (speech to text, ASR)	https://github.com/k2-fsa/sherpa-onnx/releases/tag/asr-models
Text to speech (TTS)	https://github.com/k2-fsa/sherpa-onnx/releases/tag/tts-models
VAD	https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/silero_vad.onnx
Keyword spotting	https://github.com/k2-fsa/sherpa-onnx/releases/tag/kws-models
Speech identification (Speaker ID)	https://github.com/k2-fsa/sherpa-onnx/releases/tag/speaker-recongition-models
Spoken language identification (Language ID)	https://github.com/k2-fsa/sherpa-onnx/releases/tag/asr-models (multilingual whisper)
Audio tagging	https://github.com/k2-fsa/sherpa-onnx/releases/tag/audio-tagging-models
Punctuation	https://github.com/k2-fsa/sherpa-onnx/releases/tag/punctuation-models

In this section, we describe how to download and use all available pre-trained models for speech recognition.

Hint: Please install git-lfs before you continue.

Otherwise, you will be `SAD` later.

8.23.1 Online transducer models

This section lists available online transducer models.

Zipformer-transducer-based Models

Hint: Please refer to *Installation* to install sherpa-onnx before you read this section.

sherpa-onnx-streaming-zipformer-korean-2024-06-16 (Korean)

Training code for this model can be found at https://github.com/k2-fsa/icefall/pull/1651. It supports only Korean.

PyTorch checkpoint with optimizer state can be found at https://huggingface.co/johnBamma/icefall-asr-ksponspeech-pruned-transducer-stateless7-streaming-2024-06-12

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-korean-2024-06-16.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-korean-2024-06-16.tar.bz2

tar xvf sherpa-onnx-streaming-zipformer-korean-2024-06-16.tar.bz2
rm sherpa-onnx-streaming-zipformer-korean-2024-06-16.tar.bz2
ls -lh sherpa-onnx-streaming-zipformer-korean-2024-06-16
```

The output is given below:

```
$ ls -lh sherpa-onnx-streaming-zipformer-korean-2024-06-16

total 907104
-rw-r--r--  1 fangjun  staff   307K Jun 16 17:36 bpe.model
-rw-r--r--  1 fangjun  staff   2.7M Jun 16 17:36 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff    11M Jun 16 17:36 decoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   121M Jun 16 17:36 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   279M Jun 16 17:36 encoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   2.5M Jun 16 17:36 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   9.8M Jun 16 17:36 joiner-epoch-99-avg-1.onnx
drwxr-xr-x  7 fangjun  staff   224B Jun 16 17:36 test_wavs
-rw-r--r--  1 fangjun  staff    59K Jun 16 17:36 tokens.txt
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/encoder-epoch-99-avg-1.
↪onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/decoder-epoch-99-avg-1.
↪onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/joiner-epoch-99-avg-1.
↪onnx \
  ./sherpa-onnx-streaming-zipformer-korean-2024-06-16/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/
↪tokens.txt --encoder=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/encoder-epoch-
↪99-avg-1.onnx --decoder=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/decoder-
↪epoch-99-avg-1.onnx --joiner=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/
↪joiner-epoch-99-avg-1.onnx ./sherpa-onnx-streaming-zipformer-korean-2024-06-16/test_
↪wavs/0.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80, low_freq=20, high_freq=-400, dither=0), model_
↪config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder="./sherpa-onnx-
↪streaming-zipformer-korean-2024-06-16/encoder-epoch-99-avg-1.onnx", decoder="./sherpa-
↪onnx-streaming-zipformer-korean-2024-06-16/decoder-epoch-99-avg-1.onnx", joiner="./
↪sherpa-onnx-streaming-zipformer-korean-2024-06-16/joiner-epoch-99-avg-1.onnx"),␣
↪paraformer=OnlineParaformerModelConfig(encoder="", decoder=""), wenet_
↪ctc=OnlineWenetCtcModelConfig(model="", chunk_size=16, num_left_chunks=4), zipformer2_
↪ctc=OnlineZipformer2CtcModelConfig(model=""), nemo_ctc=OnlineNeMoCtcModelConfig(model="
↪"), tokens="./sherpa-onnx-streaming-zipformer-korean-2024-06-16/tokens.txt", num_
↪threads=1, warm_up=0, debug=False, provider="cpu", model_type="", modeling_unit=
↪"cjkchar", bpe_vocab=""), lm_config=OnlineLMConfig(model="", scale=0.5), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), ctc_fst_decoder_
```

(continued from next page)

(continued from previous page)

```
./sherpa-onnx-streaming-zipformer-korean-2024-06-16/test_wavs/0.wav
Elapsed seconds: 0.56, Real time factor (RTF): 0.16

{ "text": "       ", "tokens": [" ", " ", " ", "", "", "", " ", " ", " ", " ", "", ""],
↪"timestamps": [0.52, 0.96, 1.28, 1.44, 1.52, 1.84, 2.28, 2.48, 2.88, 3.04, 3.20, 3.44],
↪ "ys_probs": [-1.701665, -0.208116, -0.527190, -1.777411, -1.853504, -0.768175, -1.
↪029222, -1.657714, -0.514807, -0.360788, -0.842238, -0.218511], "lm_probs": [],
↪"context_scores": [], "segment": 0, "words": [], "start_time": 0.00, "is_final": false}
```

int8

The following code shows how to use `int8` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/encoder-epoch-99-avg-1.
↪int8.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/decoder-epoch-99-avg-1.
↪onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/joiner-epoch-99-avg-1.
↪int8.onnx \
  ./sherpa-onnx-streaming-zipformer-korean-2024-06-16/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/
↪tokens.txt --encoder=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/encoder-epoch-
↪99-avg-1.int8.onnx --decoder=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/
↪decoder-epoch-99-avg-1.onnx --joiner=./sherpa-onnx-streaming-zipformer-korean-2024-06-
↪16/joiner-epoch-99-avg-1.int8.onnx ./sherpa-onnx-streaming-zipformer-korean-2024-06-16/
↪test_wavs/0.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80, low_freq=20, high_freq=-400, dither=0), model_
↪config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder="./sherpa-onnx-
↪streaming-zipformer-korean-2024-06-16/encoder-epoch-99-avg-1.int8.onnx", decoder="./
↪sherpa-onnx-streaming-zipformer-korean-2024-06-16/decoder-epoch-99-avg-1.onnx", joiner=
↪"./sherpa-onnx-streaming-zipformer-korean-2024-06-16/joiner-epoch-99-avg-1.int8.onnx"),
↪ paraformer=OnlineParaformerModelConfig(encoder="", decoder=""), wenet_
↪ctc=OnlineWenetCtcModelConfig(model="", chunk_size=16, num_left_chunks=4), zipformer2_
↪ctc=OnlineZipformer2CtcModelConfig(model=""), nemo_ctc=OnlineNeMoCtcModelConfig(model="
↪"), tokens="./sherpa-onnx-streaming-zipformer-korean-2024-06-16/tokens.txt", num_
↪threads=1, warm_up=0, debug=False, provider="cpu", model_type="", modeling_unit=
↪"cjkchar", bpe_vocab=""), lm_config=OnlineLMConfig(model="", scale=0.5), endpoint_
```

(continues on next page)

(continued from previous page)

```
./sherpa-onnx-streaming-zipformer-korean-2024-06-16/test_wavs/0.wav
Elapsed seconds: 0.37, Real time factor (RTF): 0.1

{ "text": "       ", "tokens": [" ", " ", " ", "", "", "", " ", " ", " ", " ", "", ""],
↪"timestamps": [0.52, 0.96, 1.28, 1.44, 1.52, 1.84, 2.28, 2.48, 2.88, 3.04, 3.20, 3.44],
↪ "ys_probs": [-1.750286, -0.241571, -0.621155, -1.862032, -1.977561, -0.789718, -1.
↪002497, -1.627276, -0.554654, -0.443969, -0.852731, -0.218611], "lm_probs": [],
↪"context_scores": [], "segment": 0, "words": [], "start_time": 0.00, "is_final": false}
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --tokens=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/encoder-epoch-99-avg-1.
↪int8.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/decoder-epoch-99-avg-1.
↪onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-korean-2024-06-16/joiner-epoch-99-avg-1.
↪int8.onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12 (Chinese)

Training code for this model can be found at https://github.com/k2-fsa/icefall/pull/1369. It supports only Chinese.

Please refer to https://github.com/k2-fsa/icefall/tree/master/egs/multi_zh-hans/ASR#included-training-sets for the detailed information about the training data. In total, there are 14k hours of training data.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-multi-zh-hans-2023-12-12.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-multi-zh-hans-2023-12-12.tar.bz2
```

(continues on next page)

(continued from previous page)

```
tar xf sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12.tar.bz2
rm sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12.tar.bz2
ls -lh sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12
```

The output is given below:

```
$ ls -lh sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12
total 668864
-rw-r--r--  1 fangjun  staff    28B Dec 12 18:59 README.md
-rw-r--r--  1 fangjun  staff   131B Dec 12 18:59 bpe.model
-rw-r--r--  1 fangjun  staff   1.2M Dec 12 18:59 decoder-epoch-20-avg-1-chunk-16-left-
↪128.int8.onnx
-rw-r--r--  1 fangjun  staff   4.9M Dec 12 18:59 decoder-epoch-20-avg-1-chunk-16-left-
↪128.onnx
-rw-r--r--  1 fangjun  staff    67M Dec 12 18:59 encoder-epoch-20-avg-1-chunk-16-left-
↪128.int8.onnx
-rw-r--r--  1 fangjun  staff   249M Dec 12 18:59 encoder-epoch-20-avg-1-chunk-16-left-
↪128.onnx
-rw-r--r--  1 fangjun  staff   1.0M Dec 12 18:59 joiner-epoch-20-avg-1-chunk-16-left-128.
↪int8.onnx
-rw-r--r--  1 fangjun  staff   3.9M Dec 12 18:59 joiner-epoch-20-avg-1-chunk-16-left-128.
↪onnx
drwxr-xr-x  8 fangjun  staff   256B Dec 12 18:59 test_wavs
-rw-r--r--  1 fangjun  staff    18K Dec 12 18:59 tokens.txt
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/encoder-epoch-20-
↪avg-1-chunk-16-left-128.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/decoder-epoch-20-
↪avg-1-chunk-16-left-128.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/joiner-epoch-20-
↪avg-1-chunk-16-left-128.onnx \
  ./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/test_wavs/DEV_T0000000000.
↪wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-
↪12/tokens.txt --encoder=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/
↪encoder-epoch-20-avg-1-chunk-16-left-128.onnx --decoder=./sherpa-onnx-streaming-
↪zipformer-multi-zh-hans-2023-12-12/decoder-epoch-20-avg-1-chunk-16-left-128.onnx --
↪joiner=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/joiner-epoch-20-avg-
↪1-chunk-16-left-128.onnx ./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/
↪test_wavs/DEV_T0000000000.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/encoder-epoch-20-avg-1-
↪chunk-16-left-128.onnx", decoder="./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-
↪12-12/decoder-epoch-20-avg-1-chunk-16-left-128.onnx", joiner="./sherpa-onnx-streaming-
↪zipformer-multi-zh-hans-2023-12-12/joiner-epoch-20-avg-1-chunk-16-left-128.onnx"),␣
↪paraformer=OnlineParaformerModelConfig(encoder="", decoder=""), wenet_
↪ctc=OnlineWenetCtcModelConfig(model="", chunk_size=16, num_left_chunks=4), tokens="./
↪sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/tokens.txt", num_threads=1,␣
↪debug=False, provider="cpu", model_type=""), lm_config=OnlineLMConfig(model="",␣
↪scale=0.5), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0),␣
↪rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_
↪utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=0, min_utterance_length=20)), enable_endpoint=True, max_active_paths=4,␣
↪hotwords_score=1.5, hotwords_file="", decoding_method="greedy_search")
./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/test_wavs/DEV_T0000000000.wav
Elapsed seconds: 0.65, Real time factor (RTF): 0.12

{"is_final":false, "segment":0, "start_time":0.00, "text": " ", "timestamps": [0.32, 0.
↪64, 0.76, 0.84, 1.08, 1.24, 1.96, 2.04, 2.24, 2.36, 2.56, 2.68, 2.80, 3.28, 3.40, 3.60,
↪ 3.72, 3.84, 3.96, 4.04, 4.16, 4.28, 4.36, 4.60, 4.72], "tokens":[" ", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
```

int8

The following code shows how to use `int8` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/encoder-epoch-20-
↪avg-1-chunk-16-left-128.int8.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/decoder-epoch-20-
↪avg-1-chunk-16-left-128.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/joiner-epoch-20-
↪avg-1-chunk-16-left-128.int8.onnx \
  ./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/test_wavs/DEV_T0000000000.
↪wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-
↪12/tokens.txt --encoder=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/
↪encoder-epoch-20-avg-1-chunk-16-left-128.int8.onnx --decoder=./sherpa-onnx-streaming-
↪zipformer-multi-zh-hans-2023-12-12/decoder-epoch-20-avg-1-chunk-16-left-128.onnx --
↪joiner=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/joiner-epoch-20-avg-
↪1-chunk-16-left-128.int8.onnx ./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-
↪12/test_wavs/DEV_T0000000000.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/encoder-epoch-20-avg-1-
↪chunk-16-left-128.int8.onnx", decoder="./sherpa-onnx-streaming-zipformer-multi-zh-hans-
↪2023-12-12/decoder-epoch-20-avg-1-chunk-16-left-128.onnx", joiner="./sherpa-onnx-
↪streaming-zipformer-multi-zh-hans-2023-12-12/joiner-epoch-20-avg-1-chunk-16-left-128.
↪int8.onnx"), paraformer=OnlineParaformerModelConfig(encoder="", decoder=""), wenet_
↪ctc=OnlineWenetCtcModelConfig(model="", chunk_size=16, num_left_chunks=4), tokens="./
↪sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/tokens.txt", num_threads=1,␣
↪debug=False, provider="cpu", model_type=""), lm_config=OnlineLMConfig(model="",␣
↪scale=0.5), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0),␣
↪rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_
↪utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=0, min_utterance_length=20)), enable_endpoint=True, max_active_paths=4,␣
↪hotwords_score=1.5, hotwords_file="", decoding_method="greedy_search")
```

(continues on next page)

(continued from previous page)

```
./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/test_wavs/DEV_T0000000000.wav
Elapsed seconds: 0.5, Real time factor (RTF): 0.088

{"is_final":false, "segment":0, "start_time":0.00, "text": " ", "timestamps": [0.32, 0.
↪64, 0.76, 0.84, 1.04, 1.24, 1.96, 2.04, 2.24, 2.36, 2.56, 2.68, 2.88, 3.28, 3.40, 3.60,
↪ 3.72, 3.84, 3.96, 4.04, 4.16, 4.28, 4.36, 4.60, 4.72], "tokens":[" ", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --tokens=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/encoder-epoch-20-
↪avg-1-chunk-16-left-128.int8.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/decoder-epoch-20-
↪avg-1-chunk-16-left-128.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-multi-zh-hans-2023-12-12/joiner-epoch-20-
↪avg-1-chunk-16-left-128.int8.onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

pkufool/icefall-asr-zipformer-streaming-wenetspeech-20230615 (Chinese)

This model is from

https://huggingface.co/pkufool/icefall-asr-zipformer-streaming-wenetspeech-20230615

which supports only Chinese as it is trained on the WenetSpeech corpus.

If you are interested in how the model is trained, please refer to https://github.com/k2-fsa/icefall/pull/1130.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/icefall-asr-
↪zipformer-streaming-wenetspeech-20230615.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/icefall-asr-
↪zipformer-streaming-wenetspeech-20230615.tar.bz2
```

(continues on next page)

(continued from previous page)

```
tar xvf icefall-asr-zipformer-streaming-wenetspeech-20230615.tar.bz2
rm icefall-asr-zipformer-streaming-wenetspeech-20230615.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
icefall-asr-zipformer-streaming-wenetspeech-20230615 fangjun$ ls -lh exp/*chunk-16-left-
↪128.*onnx
-rw-r--r--  1 fangjun  staff    11M Jun 26 15:42 exp/decoder-epoch-12-avg-4-chunk-16-
↪left-128.int8.onnx
-rw-r--r--  1 fangjun  staff    12M Jun 26 15:42 exp/decoder-epoch-12-avg-4-chunk-16-
↪left-128.onnx
-rw-r--r--  1 fangjun  staff    68M Jun 26 15:42 exp/encoder-epoch-12-avg-4-chunk-16-
↪left-128.int8.onnx
-rw-r--r--  1 fangjun  staff   250M Jun 26 15:43 exp/encoder-epoch-12-avg-4-chunk-16-
↪left-128.onnx
-rw-r--r--  1 fangjun  staff   2.7M Jun 26 15:42 exp/joiner-epoch-12-avg-4-chunk-16-left-
↪128.int8.onnx
-rw-r--r--  1 fangjun  staff    11M Jun 26 15:42 exp/joiner-epoch-12-avg-4-chunk-16-left-
↪128.onnx
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./icefall-asr-zipformer-streaming-wenetspeech-20230615/data/lang_char/tokens.
↪txt \
  --encoder=./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/encoder-epoch-12-
↪avg-4-chunk-16-left-128.onnx \
  --decoder=./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/decoder-epoch-12-
↪avg-4-chunk-16-left-128.onnx \
  --joiner=./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/joiner-epoch-12-
↪avg-4-chunk-16-left-128.onnx \
  ./icefall-asr-zipformer-streaming-wenetspeech-20230615/test_wavs/DEV_T0000000000.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./icefall-asr-zipformer-streaming-wenetspeech-20230615/
↪data/lang_char/tokens.txt --encoder=./icefall-asr-zipformer-streaming-wenetspeech-
↪20230615/exp/encoder-epoch-12-avg-4-chunk-16-left-128.onnx --decoder=./icefall-asr-
↪zipformer-streaming-wenetspeech-20230615/exp/decoder-epoch-12-avg-4-chunk-16-left-128.
↪onnx --joiner=./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/joiner-epoch-
↪12-avg-4-chunk-16-left-128.onnx ./icefall-asr-zipformer-streaming-wenetspeech-20230615/
↪test_wavs/DEV_T0000000000.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./icefall-asr-
↪zipformer-streaming-wenetspeech-20230615/exp/encoder-epoch-12-avg-4-chunk-16-left-128.
↪onnx", decoder_filename="./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/
↪decoder-epoch-12-avg-4-chunk-16-left-128.onnx", joiner_filename="./icefall-asr-
↪zipformer-streaming-wenetspeech-20230615/exp/joiner-epoch-12-avg-4-chunk-16-left-128.
↪onnx", tokens="./icefall-asr-zipformer-streaming-wenetspeech-20230615/data/lang_char/
↪tokens.txt", num_threads=2, provider="cpu", debug=False), lm_
↪config=OnlineLMConfig(model="", scale=0.5), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=True, max_active_paths=4, decoding_method="greedy_search")
./icefall-asr-zipformer-streaming-wenetspeech-20230615/test_wavs/DEV_T0000000000.wav
Elapsed seconds: 0.56, Real time factor (RTF): 0.1

{"is_final":false,"segment":0,"start_time":0.0,"text":"","timestamps":"[0.36, 0.64, 0.76,
↪ 0.88, 1.08, 1.24, 1.96, 2.04, 2.24, 2.36, 2.56, 2.68, 2.92, 3.32, 3.40, 3.64, 3.72, 3.
↪84, 3.96, 4.04, 4.16, 4.32, 4.48, 4.64, 4.72]","tokens":["","","","","","","","","","",
↪"","","","","","","","","","","","","","","","","","]}
```

int8

The following code shows how to use `int8` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./icefall-asr-zipformer-streaming-wenetspeech-20230615/data/lang_char/tokens.
↪txt \
  --encoder=./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/encoder-epoch-12-
↪avg-4-chunk-16-left-128.int8.onnx \
  --decoder=./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/decoder-epoch-12-
↪avg-4-chunk-16-left-128.onnx \
```

(continues on next page)

(continued from previous page)

```
  --joiner=./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/joiner-epoch-12-
↪avg-4-chunk-16-left-128.int8.onnx \
  ./icefall-asr-zipformer-streaming-wenetspeech-20230615/test_wavs/DEV_T0000000000.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./icefall-asr-zipformer-streaming-wenetspeech-20230615/
↪data/lang_char/tokens.txt --encoder=./icefall-asr-zipformer-streaming-wenetspeech-
↪20230615/exp/encoder-epoch-12-avg-4-chunk-16-left-128.int8.onnx --decoder=./icefall-
↪asr-zipformer-streaming-wenetspeech-20230615/exp/decoder-epoch-12-avg-4-chunk-16-left-
↪128.onnx --joiner=./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/joiner-
↪epoch-12-avg-4-chunk-16-left-128.int8.onnx ./icefall-asr-zipformer-streaming-
↪wenetspeech-20230615/test_wavs/DEV_T0000000000.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./icefall-asr-
↪zipformer-streaming-wenetspeech-20230615/exp/encoder-epoch-12-avg-4-chunk-16-left-128.
↪int8.onnx", decoder_filename="./icefall-asr-zipformer-streaming-wenetspeech-20230615/
↪exp/decoder-epoch-12-avg-4-chunk-16-left-128.onnx", joiner_filename="./icefall-asr-
↪zipformer-streaming-wenetspeech-20230615/exp/joiner-epoch-12-avg-4-chunk-16-left-128.
↪int8.onnx", tokens="./icefall-asr-zipformer-streaming-wenetspeech-20230615/data/lang_
↪char/tokens.txt", num_threads=2, provider="cpu", debug=False), lm_
↪config=OnlineLMConfig(model="", scale=0.5), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=True, max_active_paths=4, decoding_method="greedy_search")
./icefall-asr-zipformer-streaming-wenetspeech-20230615/test_wavs/DEV_T0000000000.wav
Elapsed seconds: 0.38, Real time factor (RTF): 0.068

{"is_final":false,"segment":0,"start_time":0.0,"text":"","timestamps":"[0.40, 0.52, 0.72,
↪ 0.84, 1.08, 1.24, 1.48, 1.92, 2.00, 2.24, 2.32, 2.48, 2.68, 2.80, 3.00, 3.28, 3.36, 3.
↪60, 3.72, 3.84, 3.92, 4.00, 4.16, 4.28, 4.36, 4.64, 4.68, 5.00]","tokens":["","","","",
↪ "","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","]}
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --tokens=./icefall-asr-zipformer-streaming-wenetspeech-20230615/data/lang_char/tokens.
↪txt \
  --encoder=./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/encoder-epoch-12-
↪avg-4-chunk-16-left-128.int8.onnx \
  --decoder=./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/decoder-epoch-12-
↪avg-4-chunk-16-left-128.onnx \
  --joiner=./icefall-asr-zipformer-streaming-wenetspeech-20230615/exp/joiner-epoch-12-
↪avg-4-chunk-16-left-128.int8.onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

csukuangfj/sherpa-onnx-streaming-zipformer-en-2023-06-26 (English)

This model is converted from

https://huggingface.co/Zengwei/icefall-asr-librispeech-streaming-zipformer-2023-05-17

which supports only English as it is trained on the LibriSpeech corpus.

If you are interested in how the model is trained, please refer to https://github.com/k2-fsa/icefall/pull/1058.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-en-2023-06-26.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-en-2023-06-26.tar.bz2

tar xvf sherpa-onnx-streaming-zipformer-en-2023-06-26.tar.bz2
rm sherpa-onnx-streaming-zipformer-en-2023-06-26.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes below.

```
-rw-r--r-- 1 1001 127  240K Apr 23 06:45 bpe.model
-rw-r--r-- 1 1001 127  1.3M Apr 23 06:45 decoder-epoch-99-avg-1-chunk-16-left-128.int8.
↪onnx
-rw-r--r-- 1 1001 127  2.0M Apr 23 06:45 decoder-epoch-99-avg-1-chunk-16-left-128.onnx
```

(continues on next page)

(continued from previous page)

```
-rw-r--r-- 1 1001 127   68M Apr 23 06:45 encoder-epoch-99-avg-1-chunk-16-left-128.int8.
↪onnx
-rw-r--r-- 1 1001 127  250M Apr 23 06:45 encoder-epoch-99-avg-1-chunk-16-left-128.onnx
-rwxr-xr-x 1 1001 127   814 Apr 23 06:45 export-onnx-zipformer-online.sh
-rw-r--r-- 1 1001 127  254K Apr 23 06:45 joiner-epoch-99-avg-1-chunk-16-left-128.int8.
↪onnx
-rw-r--r-- 1 1001 127 1003K Apr 23 06:45 joiner-epoch-99-avg-1-chunk-16-left-128.onnx
-rw-r--r-- 1 1001 127   216 Apr 23 06:45 README.md
drwxr-xr-x 2 1001 127  4.0K Apr 23 06:45 test_wavs
-rw-r--r-- 1 1001 127  5.0K Apr 23 06:45 tokens.txt
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-en-2023-06-26/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-2023-06-26/decoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-2023-06-26/joiner-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  ./sherpa-onnx-streaming-zipformer-en-2023-06-26/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-zipformer-en-2023-06-26/tokens.
↪txt --encoder=./sherpa-onnx-streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-
↪chunk-16-left-128.onnx --decoder=./sherpa-onnx-streaming-zipformer-en-2023-06-26/
↪decoder-epoch-99-avg-1-chunk-16-left-128.onnx --joiner=./sherpa-onnx-streaming-
↪zipformer-en-2023-06-26/joiner-epoch-99-avg-1-chunk-16-left-128.onnx ./sherpa-onnx-
↪streaming-zipformer-en-2023-06-26/test_wavs/0.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-
↪streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-chunk-16-left-128.onnx",␣
↪decoder_filename="./sherpa-onnx-streaming-zipformer-en-2023-06-26/decoder-epoch-99-avg-
↪1-chunk-16-left-128.onnx", joiner_filename="./sherpa-onnx-streaming-zipformer-en-2023-
↪06-26/joiner-epoch-99-avg-1-chunk-16-left-128.onnx", tokens="./sherpa-onnx-streaming-
↪zipformer-en-2023-06-26/tokens.txt", num_threads=2, provider="cpu", debug=False), lm_
↪config=OnlineLMConfig(model="", scale=0.5), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,
```

(continues on next page)

(continued from previous page)

```
./sherpa-onnx-streaming-zipformer-en-2023-06-26/test_wavs/0.wav
Elapsed seconds: 0.51, Real time factor (RTF): 0.077
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
{"is_final":false,"segment":0,"start_time":0.0,"text":" AFTER EARLY NIGHTFALL THE YELLOW␣
↪LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUARTER OF THE BROTHELS","timestamps":
↪"[0.68, 1.04, 1.16, 1.24, 1.60, 1.76, 1.80, 1.92, 2.04, 2.24, 2.32, 2.36, 2.52, 2.68,␣
↪2.72, 2.80, 2.92, 3.12, 3.40, 3.64, 3.76, 3.92, 4.12, 4.48, 4.68, 4.72, 4.84, 5.00, 5.
↪20, 5.24, 5.36, 5.40, 5.64, 5.76, 5.92, 5.96, 6.08, 6.24, 6.52]","tokens":[" AFTER"," E
↪","AR","LY"," NIGHT","F","A","LL"," THE"," YE","LL","OW"," LA","M","P","S"," WOULD","␣
↪LIGHT"," UP"," HE","RE"," AND"," THERE"," THE"," S","QUA","LI","D"," ","QUA","R","TER",
↪" OF"," THE"," B","RO","TH","EL","S"]}
```

int8

The following code shows how to use `int8` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-en-2023-06-26/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-chunk-
↪16-left-128.int8.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-2023-06-26/decoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-2023-06-26/joiner-epoch-99-avg-1-chunk-
↪16-left-128.int8.onnx \
  ./sherpa-onnx-streaming-zipformer-en-2023-06-26/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-zipformer-en-2023-06-26/tokens.
↪txt --encoder=./sherpa-onnx-streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-
↪chunk-16-left-128.int8.onnx --decoder=./sherpa-onnx-streaming-zipformer-en-2023-06-26/
↪decoder-epoch-99-avg-1-chunk-16-left-128.onnx --joiner=./sherpa-onnx-streaming-
↪zipformer-en-2023-06-26/joiner-epoch-99-avg-1-chunk-16-left-128.int8.onnx ./sherpa-
↪onnx-streaming-zipformer-en-2023-06-26/test_wavs/0.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-
↪streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-chunk-16-left-128.int8.onnx",␣
↪decoder_filename="./sherpa-onnx-streaming-zipformer-en-2023-06-26/decoder-epoch-99-avg-
↪1-chunk-16-left-128.onnx", joiner_filename="./sherpa-onnx-streaming-zipformer-en-2023-
↪06-26/joiner-epoch-99-avg-1-chunk-16-left-128.int8.onnx", tokens="./sherpa-onnx-
↪streaming-zipformer-en-2023-06-26/tokens.txt", num_threads=2, provider="cpu",␣
↪debug=False), lm_config=OnlineLMConfig(model="", scale=0.5), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=True, max_active_paths=4, decoding_method="greedy_search")
```

(continues on next page)

(continued from previous page)

```
./sherpa-onnx-streaming-zipformer-en-2023-06-26/test_wavs/0.wav
Elapsed seconds: 0.41, Real time factor (RTF): 0.062
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
{"is_final":false,"segment":0,"start_time":0.0,"text":" AFTER EARLY NIGHTFALL THE YELLOW␣
↪LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUARTER OF THE BROTHELS","timestamps":
↪"[0.68, 1.04, 1.16, 1.24, 1.60, 1.76, 1.80, 1.92, 2.04, 2.24, 2.32, 2.36, 2.52, 2.68,␣
↪2.72, 2.80, 2.92, 3.12, 3.40, 3.64, 3.76, 3.92, 4.12, 4.48, 4.68, 4.72, 4.84, 5.00, 5.
↪20, 5.24, 5.36, 5.44, 5.64, 5.76, 5.92, 5.96, 6.08, 6.24, 6.52]","tokens":[" AFTER"," E
↪","AR","LY"," NIGHT","F","A","LL"," THE"," YE","LL","OW"," LA","M","P","S"," WOULD","␣
↪LIGHT"," UP"," HE","RE"," AND"," THERE"," THE"," S","QUA","LI","D"," ","QUA","R","TER",
↪" OF"," THE"," B","RO","TH","EL","S"]}
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --tokens=./sherpa-onnx-streaming-zipformer-en-2023-06-26/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-2023-06-26/encoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-2023-06-26/decoder-epoch-99-avg-1-chunk-
↪16-left-128.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-2023-06-26/joiner-epoch-99-avg-1-chunk-
↪16-left-128.onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

csukuangfj/sherpa-onnx-streaming-zipformer-en-2023-06-21 (English)

This model is converted from

https://huggingface.co/marcoyang/icefall-libri-giga-pruned-transducer-stateless7-streaming-2023-04-04

which supports only English as it is trained on the LibriSpeech and GigaSpeech corpus.

If you are interested in how the model is trained, please refer to https://github.com/k2-fsa/icefall/pull/984.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-en-2023-06-21.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-en-2023-06-21.tar.bz2

tar xvf sherpa-onnx-streaming-zipformer-en-2023-06-21.tar.bz2
rm sherpa-onnx-streaming-zipformer-en-2023-06-21.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-streaming-zipformer-en-2023-06-21 fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff   1.2M Jun 21 15:34 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   2.0M Jun 21 15:34 decoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   179M Jun 21 15:36 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   337M Jun 21 15:37 encoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   253K Jun 21 15:34 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   1.0M Jun 21 15:34 joiner-epoch-99-avg-1.onnx
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-en-2023-06-21/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-2023-06-21/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-2023-06-21/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-2023-06-21/joiner-epoch-99-avg-1.onnx \
  ./sherpa-onnx-streaming-zipformer-en-2023-06-21/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-zipformer-en-2023-06-21/tokens.
↪txt --encoder=./sherpa-onnx-streaming-zipformer-en-2023-06-21/encoder-epoch-99-avg-1.
↪onnx --decoder=./sherpa-onnx-streaming-zipformer-en-2023-06-21/decoder-epoch-99-avg-1.
↪onnx --joiner=./sherpa-onnx-streaming-zipformer-en-2023-06-21/joiner-epoch-99-avg-1.
↪onnx ./sherpa-onnx-streaming-zipformer-en-2023-06-21/test_wavs/0.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-
↪streaming-zipformer-en-2023-06-21/encoder-epoch-99-avg-1.onnx", decoder_filename="./
↪sherpa-onnx-streaming-zipformer-en-2023-06-21/decoder-epoch-99-avg-1.onnx", joiner_
↪filename="./sherpa-onnx-streaming-zipformer-en-2023-06-21/joiner-epoch-99-avg-1.onnx",␣
↪tokens="./sherpa-onnx-streaming-zipformer-en-2023-06-21/tokens.txt", num_threads=2,␣
↪provider="cpu", debug=False), lm_config=OnlineLMConfig(model="", scale=0.5), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=True, max_active_paths=4, decoding_method="greedy_search")
./sherpa-onnx-streaming-zipformer-en-2023-06-21/test_wavs/0.wav
Elapsed seconds: 0.5, Real time factor (RTF): 0.076
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
{"is_final":false,"segment":0,"start_time":0.0,"text":" AFTER EARLY NIGHTFALL THE YELLOW␣
↪LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUARTER OF THE BROTHELS","timestamps":
↪"[0.64, 1.00, 1.12, 1.20, 1.60, 1.76, 1.84, 1.96, 2.08, 2.24, 2.36, 2.40, 2.60, 2.72,␣
↪2.80, 2.88, 3.00, 3.20, 3.44, 3.68, 3.76, 3.96, 4.24, 4.52, 4.72, 4.76, 4.88, 5.04, 5.
↪24, 5.28, 5.36, 5.48, 5.64, 5.76, 5.92, 5.96, 6.04, 6.24, 6.36]","tokens":[" AFTER"," E
↪","AR","LY"," NIGHT","F","A","LL"," THE"," YE","LL","OW"," LA","M","P","S"," WOULD","␣
↪LIGHT"," UP"," HE","RE"," AND"," THERE"," THE"," S","QUA","LI","D"," ","QUA","R","TER",
↪" OF"," THE"," B","RO","TH","EL","S"]}
```

int8

The following code shows how to use `int8` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-en-2023-06-21/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-2023-06-21/encoder-epoch-99-avg-1.int8.
↪onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-2023-06-21/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-2023-06-21/joiner-epoch-99-avg-1.int8.
↪onnx \
  ./sherpa-onnx-streaming-zipformer-en-2023-06-21/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-zipformer-en-2023-06-21/tokens.
↪txt --encoder=./sherpa-onnx-streaming-zipformer-en-2023-06-21/encoder-epoch-99-avg-1.
↪int8.onnx --decoder=./sherpa-onnx-streaming-zipformer-en-2023-06-21/decoder-epoch-99-
↪avg-1.onnx --joiner=./sherpa-onnx-streaming-zipformer-en-2023-06-21/joiner-epoch-99-
↪avg-1.int8.onnx ./sherpa-onnx-streaming-zipformer-en-2023-06-21/test_wavs/0.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-
↪streaming-zipformer-en-2023-06-21/encoder-epoch-99-avg-1.int8.onnx", decoder_filename=
↪"./sherpa-onnx-streaming-zipformer-en-2023-06-21/decoder-epoch-99-avg-1.onnx", joiner_
↪filename="./sherpa-onnx-streaming-zipformer-en-2023-06-21/joiner-epoch-99-avg-1.int8.
↪onnx", tokens="./sherpa-onnx-streaming-zipformer-en-2023-06-21/tokens.txt", num_
↪threads=2, provider="cpu", debug=False), lm_config=OnlineLMConfig(model="", scale=0.5),
↪ endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.2, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)), enable_endpoint=True, max_active_paths=4, decoding_method=
↪"greedy_search")
./sherpa-onnx-streaming-zipformer-en-2023-06-21/test_wavs/0.wav
Elapsed seconds: 0.41, Real time factor (RTF): 0.062
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
{"is_final":false,"segment":0,"start_time":0.0,"text":" AFTER EARLY NIGHTFALL THE YELLOW␣
↪LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUARTER OF THE BROTHELS","timestamps":
↪"[0.64, 1.00, 1.12, 1.20, 1.60, 1.76, 1.80, 1.96, 2.08, 2.24, 2.36, 2.40, 2.60, 2.72,␣
↪2.80, 2.88, 3.00, 3.20, 3.44, 3.68, 3.76, 3.96, 4.24, 4.52, 4.72, 4.76, 4.88, 5.04, 5.
↪24, 5.28, 5.36, 5.48, 5.64, 5.76, 5.92, 5.96, 6.04, 6.24, 6.36]","tokens":[" AFTER"," E
↪","AR","LY"," NIGHT","F","A","LL"," THE"," YE","LL","OW"," LA","M","P","S"," WOULD","␣
↪LIGHT"," UP"," HE","RE"," AND"," THERE"," THE"," S","QUA","LI","D"," ","QUA","R","TER",
↪" OF"," THE"," B","RO","TH","EL","S"]}
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --tokens=./sherpa-onnx-streaming-zipformer-en-2023-06-21/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-2023-06-21/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-2023-06-21/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-2023-06-21/joiner-epoch-99-avg-1.onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

csukuangfj/sherpa-onnx-streaming-zipformer-en-2023-02-21 (English)

This model is converted from

https://huggingface.co/Zengwei/icefall-asr-librispeech-pruned-transducer-stateless7-streaming-2022-12-29

which supports only English as it is trained on the LibriSpeech corpus.

You can find the training code at

https://github.com/k2-fsa/icefall/tree/master/egs/librispeech/ASR/pruned_transducer_stateless7_streaming

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

GitHub

ModelScope

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-en-2023-02-21.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-en-2023-02-21.tar.bz2

tar xvf sherpa-onnx-streaming-zipformer-en-2023-02-21.tar.bz2
rm sherpa-onnx-streaming-zipformer-en-2023-02-21.tar.bz2
```

```
cd /path/to/sherpa-onnx

GIT_LFS_SKIP_SMUDGE=1 git clone https://www.modelscope.cn/pkufool/sherpa-onnx-streaming-
↪zipformer-en-2023-02-21.git
cd sherpa-onnx-streaming-zipformer-en-2023-02-21
git lfs pull --include "*.onnx"
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-streaming-zipformer-en-2023-02-21$ ls -lh *.onnx
-rw-r--r-- 1 kuangfangjun root  1.3M Mar 31 23:06 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root  2.0M Feb 21 20:51 decoder-epoch-99-avg-1.onnx
-rw-r--r-- 1 kuangfangjun root  180M Mar 31 23:07 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root  338M Feb 21 20:51 encoder-epoch-99-avg-1.onnx
-rw-r--r-- 1 kuangfangjun root  254K Mar 31 23:06 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root 1003K Feb 21 20:51 joiner-epoch-99-avg-1.onnx
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-en-2023-02-21/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-2023-02-21/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-2023-02-21/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-2023-02-21/joiner-epoch-99-avg-1.onnx \
  ./sherpa-onnx-streaming-zipformer-en-2023-02-21/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

You should see the following output:

```
OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-
↪streaming-zipformer-en-2023-02-21/encoder-epoch-99-avg-1.onnx", decoder_filename="./
↪sherpa-onnx-streaming-zipformer-en-2023-02-21/decoder-epoch-99-avg-1.onnx", joiner_
↪filename="./sherpa-onnx-streaming-zipformer-en-2023-02-21/joiner-epoch-99-avg-1.onnx",␣
↪tokens="./sherpa-onnx-streaming-zipformer-en-2023-02-21/tokens.txt", num_threads=2,␣
↪debug=False), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0),␣
↪rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_
↪utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=0, min_utterance_length=20)), enable_endpoint=True, max_active_paths=4,␣
↪decoding_method="greedy_search")
2023-04-01 06:16:29.128344485 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 604840, index: 15, mask: {16, 52, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
2023-04-01 06:16:29.128346568 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 604841, index: 16, mask: {17, 53, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
sampling rate of input file: 16000
wav filename: ./sherpa-onnx-streaming-zipformer-en-2023-02-21/test_wavs/0.wav
wav duration (s): 6.625
Started
Done!
Recognition result for ./sherpa-onnx-streaming-zipformer-en-2023-02-21/test_wavs/0.wav:
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
```

(continues on next page)

(continued from previous page)

```
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.825 s
Real time factor (RTF): 0.825 / 6.625 = 0.125
```

int8

The following code shows how to use `int8` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-en-2023-02-21/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-2023-02-21/encoder-epoch-99-avg-1.int8.
↪onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-2023-02-21/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-2023-02-21/joiner-epoch-99-avg-1.int8.
↪onnx \
  ./sherpa-onnx-streaming-zipformer-en-2023-02-21/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

You should see the following output:

```
OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-
↪streaming-zipformer-en-2023-02-21/encoder-epoch-99-avg-1.int8.onnx", decoder_filename=
↪"./sherpa-onnx-streaming-zipformer-en-2023-02-21/decoder-epoch-99-avg-1.onnx", joiner_
↪filename="./sherpa-onnx-streaming-zipformer-en-2023-02-21/joiner-epoch-99-avg-1.int8.
↪onnx", tokens="./sherpa-onnx-streaming-zipformer-en-2023-02-21/tokens.txt", num_
↪threads=2, debug=False), endpoint_config=EndpointConfig(rule1=EndpointRule(must_
↪contain_nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0),␣
↪rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_
↪utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=0, min_utterance_length=20)), enable_endpoint=True, max_active_paths=4,␣
↪decoding_method="greedy_search")
2023-04-01 06:18:47.466564998 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 604880, index: 15, mask: {16, 52, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
2023-04-01 06:18:47.466566863 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 604881, index: 16, mask: {17, 53, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
sampling rate of input file: 16000
wav filename: ./sherpa-onnx-streaming-zipformer-en-2023-02-21/test_wavs/0.wav
wav duration (s): 6.625
Started
Done!
```

(continues on next page)

(continued from previous page)

```
Recognition result for ./sherpa-onnx-streaming-zipformer-en-2023-02-21/test_wavs/0.wav:
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.633 s
Real time factor (RTF): 0.633 / 6.625 = 0.096
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --tokens=./sherpa-onnx-streaming-zipformer-en-2023-02-21/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-2023-02-21/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-2023-02-21/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-2023-02-21/joiner-epoch-99-avg-1.onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

csukuangfj/sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20 (Bilingual, Chinese + English)

This model is converted from

https://huggingface.co/csukuangfj/k2fsa-zipformer-chinese-english-mixed

which supports both Chinese and English. The model is contributed by the community and is trained on tens of thousands of some internal dataset.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-bilingual-zh-en-2023-02-20.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-bilingual-zh-en-2023-02-20.tar.bz2

tar xvf sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20.tar.bz2
rm sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20$ ls -lh *.onnx
-rw-r--r-- 1 kuangfangjun root  13M Mar 31 21:11 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root  14M Feb 20 20:13 decoder-epoch-99-avg-1.onnx
-rw-r--r-- 1 kuangfangjun root 174M Mar 31 21:11 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root 315M Feb 20 20:13 encoder-epoch-99-avg-1.onnx
-rw-r--r-- 1 kuangfangjun root 3.1M Mar 31 21:11 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root  13M Feb 20 20:13 joiner-epoch-99-avg-1.onnx
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/encoder-epoch-
↪99-avg-1.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/decoder-epoch-
↪99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/joiner-epoch-99-
↪avg-1.onnx \
  ./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/1.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-
↪streaming-zipformer-en-2023-02-21/encoder-epoch-99-avg-1.onnx", decoder_filename="./
↪sherpa-onnx-streaming-zipformer-en-2023-02-21/decoder-epoch-99-avg-1.onnx", joiner_
↪filename="./sherpa-onnx-streaming-zipformer-en-2023-02-21/joiner-epoch-99-avg-1.onnx",␣
↪tokens="./sherpa-onnx-streaming-zipformer-en-2023-02-21/tokens.txt", num_threads=2,␣
↪debug=False), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0),␣
↪rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_
↪utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=0, min_utterance_length=20)), enable_endpoint=True, max_active_paths=4,␣
↪decoding_method="greedy_search")
```

(continues on next page)

(continued from previous page)

```
2023-04-01 06:22:23.030317206 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 604942, index: 16, mask: {17, 53, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
2023-04-01 06:22:23.030315351 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 604941, index: 15, mask: {16, 52, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
sampling rate of input file: 16000
wav filename: ./sherpa-onnx-streaming-zipformer-en-2023-02-21/test_wavs/0.wav
wav duration (s): 6.625
Started
Done!
Recognition result for ./sherpa-onnx-streaming-zipformer-en-2023-02-21/test_wavs/0.wav:
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.815 s
Real time factor (RTF): 0.815 / 6.625 = 0.123
```

int8

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/encoder-epoch-
↪99-avg-1.int8.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/decoder-epoch-
↪99-avg-1.int8.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/joiner-epoch-99-
↪avg-1.int8.onnx \
  ./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/1.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-
↪streaming-zipformer-bilingual-zh-en-2023-02-20/encoder-epoch-99-avg-1.int8.onnx",␣
↪decoder_filename="./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/decoder-
↪epoch-99-avg-1.int8.onnx", joiner_filename="./sherpa-onnx-streaming-zipformer-
↪bilingual-zh-en-2023-02-20/joiner-epoch-99-avg-1.int8.onnx", tokens="./sherpa-onnx-
↪streaming-zipformer-bilingual-zh-en-2023-02-20/tokens.txt", num_threads=2,␣
↪debug=False), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0),␣
↪rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_
↪utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=0, min_utterance_length=20)), enable_endpoint=True, max_active_paths=4,␣
↪decoding_method="greedy_search")
2023-04-01 06:24:10.503505750 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 604982, index: 16, mask: {17, 53, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
2023-04-01 06:24:10.503503942 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 604981, index: 15, mask: {16, 52, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
sampling rate of input file: 16000
wav filename: ./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_wavs/1.
↪wav
wav duration (s): 5.100
Started
Done!
Recognition result for ./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/test_
↪wavs/1.wav:
 ALWAYS ALWAYS
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.551 s
Real time factor (RTF): 0.551 / 5.100 = 0.108
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --tokens=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/encoder-epoch-
↪99-avg-1.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/decoder-epoch-
↪99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-bilingual-zh-en-2023-02-20/joiner-epoch-99-
↪avg-1.onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

shaojieli/sherpa-onnx-streaming-zipformer-fr-2023-04-14 (French)

This model is converted from

https://huggingface.co/shaojieli/icefall-asr-commonvoice-fr-pruned-transducer-stateless7-streaming-2023-04-02

which supports only French as it is trained on the CommonVoice corpus. In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-fr-2023-04-14.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-fr-2023-04-14.tar.bz2

tar xvf sherpa-onnx-streaming-zipformer-fr-2023-04-14.tar.bz2
rm sherpa-onnx-streaming-zipformer-fr-2023-04-14.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-streaming-zipformer-fr-2023-04-14 shaojieli$ ls -lh *.bin

-rw-r--r-- 1 lishaojie Students  1.3M 4  14 14:09 decoder-epoch-29-avg-9-with-averaged-
↪model.int8.onnx
-rw-r--r-- 1 lishaojie Students  2.0M 4  14 14:09 decoder-epoch-29-avg-9-with-averaged-
↪model.onnx
-rw-r--r-- 1 lishaojie Students  121M 4  14 14:09 encoder-epoch-29-avg-9-with-averaged-
↪model.int8.onnx
-rw-r--r-- 1 lishaojie Students  279M 4  14 14:09 encoder-epoch-29-avg-9-with-averaged-
↪model.onnx
-rw-r--r-- 1 lishaojie Students  254K 4  14 14:09 joiner-epoch-29-avg-9-with-averaged-
↪model.int8.onnx
-rw-r--r-- 1 lishaojie Students 1003K 4  14 14:09 joiner-epoch-29-avg-9-with-averaged-
↪model.onnx
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-fr-2023-04-14/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-fr-2023-04-14/encoder-epoch-29-avg-9-with-
↪averaged-model.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-fr-2023-04-14/decoder-epoch-29-avg-9-with-
↪averaged-model.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-fr-2023-04-14/joiner-epoch-29-avg-9-with-
↪averaged-model.onnx \
  ./sherpa-onnx-streaming-zipformer-fr-2023-04-14/test_wavs/common_voice_fr_19364697.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-
↪streaming-zipformer-fr-2023-04-14/encoder-epoch-29-avg-9-with-averaged-model.onnx",␣
↪decoder_filename="./sherpa-onnx-streaming-zipformer-fr-2023-04-14/decoder-epoch-29-avg-
↪9-with-averaged-model.onnx", joiner_filename="./sherpa-onnx-streaming-zipformer-fr-
↪2023-04-14/joiner-epoch-29-avg-9-with-averaged-model.onnx", tokens="./sherpa-onnx-
↪streaming-zipformer-fr-2023-04-14/tokens.txt", num_threads=2, debug=False), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=True, max_active_paths=4, decoding_method="greedy_search")
sampling rate of input file: 16000
wav filename: ./sherpa-onnx-streaming-zipformer-fr-2023-04-14/test_wavs/common_voice_fr_
↪19364697.wav
wav duration (s): 7.128
Started
Done!
Recognition result for ./sherpa-onnx-streaming-zipformer-fr-2023-04-14/test_wavs/common_
↪voice_fr_19364697.wav:
 CE SITE CONTIENT QUATRE TOMBEAUX DE LA DYNASTIE ASHÉMÉNIDE ET SEPT DES SASSANDIDES
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.458 s
Real time factor (RTF): 0.458 / 7.128 = 0.064
```

int8

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-fr-2023-04-14/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-fr-2023-04-14/encoder-epoch-29-avg-9-with-
↪averaged-model.int8.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-fr-2023-04-14/decoder-epoch-29-avg-9-with-
↪averaged-model.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-fr-2023-04-14/joiner-epoch-29-avg-9-with-
↪averaged-model.int8.onnx \
  ./sherpa-onnx-streaming-zipformer-fr-2023-04-14/test_wavs/common_voice_fr_19364697.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-
↪streaming-zipformer-fr-2023-04-14/encoder-epoch-29-avg-9-with-averaged-model.int8.onnx
↪", decoder_filename="./sherpa-onnx-streaming-zipformer-fr-2023-04-14/decoder-epoch-29-
↪avg-9-with-averaged-model.onnx", joiner_filename="./sherpa-onnx-streaming-zipformer-fr-
↪2023-04-14/joiner-epoch-29-avg-9-with-averaged-model.int8.onnx", tokens="./sherpa-onnx-
↪streaming-zipformer-fr-2023-04-14/tokens.txt", num_threads=2, debug=False), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=True, max_active_paths=4, decoding_method="greedy_search")
sampling rate of input file: 16000
wav filename: ./sherpa-onnx-streaming-zipformer-fr-2023-04-14/test_wavs/common_voice_fr_
↪19364697.wav
wav duration (s): 7.128
Started
Done!
Recognition result for ./sherpa-onnx-streaming-zipformer-fr-2023-04-14/test_wavs/common_
↪voice_fr_19364697.wav:
 CE SITE CONTIENT QUATRE TOMBEAUX DE LA DYNASTIE ASHÉMÉNIDE ET SEPT DES SASSANDIDES
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.485 s
Real time factor (RTF): 0.485 / 7.128 = 0.068
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx
./build/bin/sherpa-onnx-microphone \
  --tokens=./sherpa-onnx-streaming-zipformer-fr-2023-04-14/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-fr-2023-04-14/encoder-epoch-29-avg-9-with-
↪averaged-model.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-fr-2023-04-14/decoder-epoch-29-avg-9-with-
↪averaged-model.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-fr-2023-04-14/joiner-epoch-29-avg-9-with-
↪averaged-model.onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16 (Bilingual, Chinese + English)

Hint: It is a small model.

This model is converted from

https://huggingface.co/csukuangfj/k2fsa-zipformer-bilingual-zh-en-t

which supports both Chinese and English. The model is contributed by the community and is trained on tens of thousands of some internal dataset.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-small-bilingual-zh-en-2023-02-16.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-small-bilingual-zh-en-2023-02-16.tar.bz2

tar xf sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16.tar.bz2
rm sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16 fangjun$ ls -lh *.onnx
total 158M
drwxr-xr-x 2 1001 127 4.0K Mar 20 13:11 64
```

(continues on next page)

(continued from previous page)

```
drwxr-xr-x 2 1001 127 4.0K Mar 20 13:11 96
-rw-r--r-- 1 1001 127 240K Mar 20 13:11 bpe.model
-rw-r--r-- 1 1001 127 3.4M Mar 20 13:11 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 1001 127  14M Mar 20 13:11 decoder-epoch-99-avg-1.onnx
-rw-r--r-- 1 1001 127  41M Mar 20 13:11 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 1001 127  85M Mar 20 13:11 encoder-epoch-99-avg-1.onnx
-rw-r--r-- 1 1001 127 3.1M Mar 20 13:11 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 1001 127  13M Mar 20 13:11 joiner-epoch-99-avg-1.onnx
drwxr-xr-x 2 1001 127 4.0K Mar 20 13:11 test_wavs
-rw-r--r-- 1 1001 127  55K Mar 20 13:11 tokens.txt
```

Hint: There are two sub-folders in the model directory: `64` and `96`. The number represents chunk size. The larger the number, the lower the RTF. The default chunk size is 32.

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/tokens.txt␣
↪\
  --encoder=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/encoder-
↪epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/decoder-
↪epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/joiner-
↪epoch-99-avg-1.onnx \
  ./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/project/sherpa-onnx/csrc/parse-options.cc:Read:361 sherpa-onnx --tokens=./sherpa-onnx-
↪streaming-zipformer-small-bilingual-zh-en-2023-02-16/tokens.txt --encoder=./sherpa-
↪onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/encoder-epoch-99-avg-1.onnx -
↪-decoder=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/decoder-
↪epoch-99-avg-1.onnx --joiner=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-
↪2023-02-16/joiner-epoch-99-avg-1.onnx ./sherpa-onnx-streaming-zipformer-small-
↪bilingual-zh-en-2023-02-16/test_wavs/0.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/encoder-epoch-99-
↪avg-1.onnx", decoder="./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-
↪16/decoder-epoch-99-avg-1.onnx", joiner="./sherpa-onnx-streaming-zipformer-small-
↪bilingual-zh-en-2023-02-16/joiner-epoch-99-avg-1.onnx"),␣
↪paraformer=OnlineParaformerModelConfig(encoder="", decoder=""), wenet_
↪ctc=OnlineWenetCtcModelConfig(model="", chunk_size=16, num_left_chunks=4), zipformer2_
↪ctc=OnlineZipformer2CtcModelConfig(model=""), tokens="./sherpa-onnx-streaming-
↪zipformer-small-bilingual-zh-en-2023-02-16/tokens.txt", num_threads=1, debug=False,␣
↪provider="cpu", model_type=""), lm_config=OnlineLMConfig(model="", scale=0.5),␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.2, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)), enable_endpoint=True, max_active_paths=4, hotwords_score=1.5,␣
↪hotwords_file="", decoding_method="greedy_search", blank_penalty=0)
./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/test_wavs/0.wav
Elapsed seconds: 1, Real time factor (RTF): 0.1
 MONDAY TODAY IS THEY AFTER TOMORROW
{ "text": " MONDAY TODAY IS THEY AFTER TOMORROW", "tokens": [ "", "", "", " MO", "N",
↪"DAY", " TO", "DAY", " IS", " THEY", " AFTER", " TO", "M", "OR", "ROW", "", "", "", ""␣
↪], "timestamps": [ 0.64, 1.08, 1.64, 2.08, 2.20, 2.36, 4.16, 4.36, 5.12, 7.16, 7.44, 8.
↪00, 8.12, 8.20, 8.44, 9.08, 9.44, 9.64, 9.88 ], "ys_probs": [ -0.000507, -0.056152, -0.
↪007374, -0.213242, -0.362640, -0.117561, -1.036179, -0.219900, -0.150360, -0.734749, -
↪0.113281, -0.060974, -0.117775, -0.361603, -0.039993, -0.217766, -0.042011, -0.108857,␣
↪-0.135108 ], "lm_probs": [  ], "context_scores": [  ], "segment": 0, "start_time": 0.
↪00, "is_final": false}
```

int8

The following code shows how to use `int8` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/tokens.txt␣
↪\
  --encoder=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/encoder-
↪epoch-99-avg-1.int8.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/decoder-
↪epoch-99-avg-1.onnx \
```

(continues on next page)

(continued from previous page)

```
  --joiner=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/joiner-
↪epoch-99-avg-1.int8.onnx \
  ./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/project/sherpa-onnx/csrc/parse-options.cc:Read:361 sherpa-onnx --tokens=./sherpa-onnx-
↪streaming-zipformer-small-bilingual-zh-en-2023-02-16/tokens.txt --encoder=./sherpa-
↪onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/encoder-epoch-99-avg-1.int8.
↪onnx --decoder=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/
↪decoder-epoch-99-avg-1.onnx --joiner=./sherpa-onnx-streaming-zipformer-small-bilingual-
↪zh-en-2023-02-16/joiner-epoch-99-avg-1.int8.onnx ./sherpa-onnx-streaming-zipformer-
↪small-bilingual-zh-en-2023-02-16/test_wavs/0.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/encoder-epoch-99-
↪avg-1.int8.onnx", decoder="./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-
↪2023-02-16/decoder-epoch-99-avg-1.onnx", joiner="./sherpa-onnx-streaming-zipformer-
↪small-bilingual-zh-en-2023-02-16/joiner-epoch-99-avg-1.int8.onnx"),␣
↪paraformer=OnlineParaformerModelConfig(encoder="", decoder=""), wenet_
↪ctc=OnlineWenetCtcModelConfig(model="", chunk_size=16, num_left_chunks=4), zipformer2_
↪ctc=OnlineZipformer2CtcModelConfig(model=""), tokens="./sherpa-onnx-streaming-
↪zipformer-small-bilingual-zh-en-2023-02-16/tokens.txt", num_threads=1, debug=False,␣
↪provider="cpu", model_type=""), lm_config=OnlineLMConfig(model="", scale=0.5),␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.2, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)), enable_endpoint=True, max_active_paths=4, hotwords_score=1.5,␣
↪hotwords_file="", decoding_method="greedy_search", blank_penalty=0)
./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/test_wavs/0.wav
Elapsed seconds: 0.69, Real time factor (RTF): 0.069
 MONDAY TODAY IS THEY AFTER TOMORROW
{ "text": " MONDAY TODAY IS THEY AFTER TOMORROW", "tokens": [ "", "", "", " MO", "N",
↪"DAY", " TO", "DAY", " IS", " THEY", " AFTER", " TO", "M", "OR", "ROW", "", "", "", ""␣
↪], "timestamps": [ 0.64, 1.08, 1.64, 2.08, 2.20, 2.36, 4.20, 4.36, 5.12, 7.16, 7.44, 8.
↪00, 8.12, 8.20, 8.40, 9.04, 9.44, 9.64, 9.88 ], "ys_probs": [ -0.000305, -0.152557, -0.
↪007835, -0.156221, -0.622139, -0.081843, -1.140152, -0.418322, -0.198410, -0.939461, -
↪0.224989, -0.052963, -0.098366, -0.081665, -0.453255, -0.335670, -0.039482, -0.381765,␣
↪-0.192475 ], "lm_probs": [  ], "context_scores": [  ], "segment": 0, "start_time": 0.
↪00, "is_final": false}
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --tokens=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/tokens.txt␣
↪\
  --encoder=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/encoder-
↪epoch-99-avg-1.int8.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/decoder-
↪epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16/joiner-
↪epoch-99-avg-1.int8.onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

csukuangfj/sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23 (Chinese)

Hint: It is a small model.

This model is from

https://huggingface.co/marcoyang/sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23/

which supports only Chinese as it is trained on the WenetSpeech corpus.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-zh-14M-2023-02-23.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-zh-14M-2023-02-23.tar.bz2

tar xvf sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23.tar.bz2
rm sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23 fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff   1.8M Sep 10 15:31 decoder-epoch-99-avg-1.int8.onnx
```

(continues on next page)

(continued from previous page)

```
-rw-r--r--  1 fangjun  staff   7.2M Sep 10 15:31 decoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff    21M Sep 10 15:31 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff    39M Sep 10 15:31 encoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   1.7M Sep 10 15:31 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   6.8M Sep 10 15:31 joiner-epoch-99-avg-1.onnx
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/encoder-epoch-99-avg-1.
↪onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/decoder-epoch-99-avg-1.
↪onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/joiner-epoch-99-avg-1.
↪onnx \
  ./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/
↪tokens.txt --encoder=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/encoder-epoch-
↪99-avg-1.onnx --decoder=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/decoder-
↪epoch-99-avg-1.onnx --joiner=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/
↪joiner-epoch-99-avg-1.onnx ./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/test_
↪wavs/0.wav
```

(continues on next page)

(continued from previous page)

```
OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/encoder-epoch-99-avg-1.onnx",␣
↪decoder="./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/decoder-epoch-99-avg-1.
↪onnx", joiner="./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/joiner-epoch-99-avg-
↪1.onnx"), paraformer=OnlineParaformerModelConfig(encoder="", decoder=""), tokens="./
↪sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/tokens.txt", num_threads=1,␣
↪debug=False, provider="cpu", model_type=""), lm_config=OnlineLMConfig(model="",␣
↪scale=0.5), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0),␣
↪rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_
↪utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=0, min_utterance_length=20)), enable_endpoint=True, max_active_paths=4,␣
↪context_score=1.5, decoding_method="greedy_search")
./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/test_wavs/0.wav
Elapsed seconds: 0.21, Real time factor (RTF): 0.038

{"is_final":false,"segment":0,"start_time":0.0,"text":"","timestamps":"[0.32, 0.64, 0.76,
↪ 0.96, 1.08, 1.16, 1.96, 2.04, 2.24, 2.36, 2.56, 2.68, 2.80, 3.36, 3.52, 3.64, 3.72, 3.
↪84, 3.92, 4.00, 4.08, 4.24, 4.48, 4.56, 4.72]","tokens":["","","","","","","","","","",
↪ "","","","","","","","","","","","","","","","","","]}
```

int8

The following code shows how to use `int8` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/encoder-epoch-99-avg-1.
↪int8.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/decoder-epoch-99-avg-1.
↪onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/joiner-epoch-99-avg-1.
↪int8.onnx \
  ./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/
↪tokens.txt --encoder=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/encoder-epoch-
↪99-avg-1.int8.onnx --decoder=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/
↪decoder-epoch-99-avg-1.onnx --joiner=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-
↪23/joiner-epoch-99-avg-1.int8.onnx ./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/
↪test_wavs/0.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/encoder-epoch-99-avg-1.int8.onnx",
↪ decoder="./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/decoder-epoch-99-avg-1.
↪onnx", joiner="./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/joiner-epoch-99-avg-
↪1.int8.onnx"), paraformer=OnlineParaformerModelConfig(encoder="", decoder=""), tokens=
↪"./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/tokens.txt", num_threads=1,␣
↪debug=False, provider="cpu", model_type=""), lm_config=OnlineLMConfig(model="",␣
↪scale=0.5), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0),␣
↪rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_
↪utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=0, min_utterance_length=20)), enable_endpoint=True, max_active_paths=4,␣
↪context_score=1.5, decoding_method="greedy_search")
./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/test_wavs/0.wav
Elapsed seconds: 0.16, Real time factor (RTF): 0.028

{"is_final":false,"segment":0,"start_time":0.0,"text":"","timestamps":"[0.32, 0.64, 0.76,
↪ 0.96, 1.08, 1.16, 1.92, 2.04, 2.24, 2.36, 2.56, 2.68, 2.76, 3.36, 3.52, 3.64, 3.72, 3.
↪84, 3.92, 4.00, 4.08, 4.24, 4.48, 4.56, 4.72]","tokens":["","","","","","","","","","",
↪"","","","","","","","","","","","","","","]}
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --tokens=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/encoder-epoch-99-avg-1.
↪onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/decoder-epoch-99-avg-1.
↪onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23/joiner-epoch-99-avg-1.onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

csukuangfj/sherpa-onnx-streaming-zipformer-en-20M-2023-02-17 (English)

Hint: It is a small model.

This model is from

https://huggingface.co/desh2608/icefall-asr-librispeech-pruned-transducer-stateless7-streaming-small

which supports only English as it is trained on the LibriSpeech corpus.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-en-20M-2023-02-17.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-en-20M-2023-02-17.tar.bz2

tar xvf sherpa-onnx-streaming-zipformer-en-20M-2023-02-17.tar.bz2
rm sherpa-onnx-streaming-zipformer-en-20M-2023-02-17.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-streaming-zipformer-en-20M-2023-02-17 fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff   527K Sep 10 17:06 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   2.0M Sep 10 17:06 decoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff    41M Sep 10 17:06 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff    85M Sep 10 17:06 encoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   253K Sep 10 17:06 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   1.0M Sep 10 17:06 joiner-epoch-99-avg-1.onnx
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/encoder-epoch-99-avg-1.
↪onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/decoder-epoch-99-avg-1.
↪onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/joiner-epoch-99-avg-1.
↪onnx \
  ./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/
↪tokens.txt --encoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/encoder-epoch-
↪99-avg-1.onnx --decoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/decoder-
↪epoch-99-avg-1.onnx --joiner=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/
↪joiner-epoch-99-avg-1.onnx ./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/test_
↪wavs/0.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/encoder-epoch-99-avg-1.onnx",␣
↪decoder="./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/decoder-epoch-99-avg-1.
↪onnx", joiner="./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/joiner-epoch-99-avg-
↪1.onnx"), paraformer=OnlineParaformerModelConfig(encoder="", decoder=""), tokens="./
↪sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/tokens.txt", num_threads=1,␣
↪debug=False, provider="cpu", model_type=""), lm_config=OnlineLMConfig(model="",␣
↪scale=0.5), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0),␣
↪rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_
↪utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=0, min_utterance_length=20)), enable_endpoint=True, max_active_paths=4,␣
↪context_score=1.5, decoding_method="greedy_search")
./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/test_wavs/0.wav
Elapsed seconds: 0.32, Real time factor (RTF): 0.049
 THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUARTER OF THE BRAFFLELS
{"is_final":false,"segment":0,"start_time":0.0,"text":" THE YELLOW LAMPS WOULD LIGHT UP␣
↪HERE AND THERE THE SQUALID QUARTER OF THE BRAFFLELS","timestamps":"[2.04, 2.16, 2.28,␣
↪2.36, 2.52, 2.64, 2.68, 2.76, 2.92, 3.08, 3.40, 3.60, 3.72, 3.88, 4.12, 4.48, 4.64, 4.
↪68, 4.84, 4.96, 5.16, 5.20, 5.32, 5.36, 5.60, 5.72, 5.92, 5.96, 6.08, 6.24, 6.36, 6.52]
↪","tokens":[" THE"," YE","LL","OW"," LA","M","P","S"," WOULD"," LIGHT"," UP"," HE","RE
↪"," AND"," THERE"," THE"," S","QUA","LI","D"," ","QUA","R","TER"," OF"," THE"," B","RA
↪","FF","L","EL","S"]}
```

(continues on next page)

(continued from previous page)

int8

The following code shows how to use `int8` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/encoder-epoch-99-avg-1.
↪int8.onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/decoder-epoch-99-avg-1.
↪onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/joiner-epoch-99-avg-1.
↪int8.onnx \
  ./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/
↪tokens.txt --encoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/encoder-epoch-
↪99-avg-1.int8.onnx --decoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/
↪decoder-epoch-99-avg-1.onnx --joiner=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-
↪17/joiner-epoch-99-avg-1.int8.onnx ./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/
↪test_wavs/0.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/encoder-epoch-99-avg-1.int8.onnx",
↪ decoder="./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/decoder-epoch-99-avg-1.
↪onnx", joiner="./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/joiner-epoch-99-avg-
↪1.int8.onnx"), paraformer=OnlineParaformerModelConfig(encoder="", decoder=""), tokens=
↪"./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/tokens.txt", num_threads=1,␣
↪debug=False, provider="cpu", model_type=""), lm_config=OnlineLMConfig(model="",␣
↪scale=0.5), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0),␣
↪rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_
↪utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=0, min_utterance_length=20)), enable_endpoint=True, max_active_paths=4,␣
↪context_score=1.5, decoding_method="greedy_search")
./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/test_wavs/0.wav
Elapsed seconds: 0.25, Real time factor (RTF): 0.038
 THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID QUARTER OF THE BRAFFLS
{"is_final":false,"segment":0,"start_time":0.0,"text":" THE YELLOW LAMPS WOULD LIGHT UP␣
↪HERE AND THERE THE SQUALID QUARTER OF THE BRAFFLS","timestamps":"[2.04, 2.20, 2.28, 2.
↪36, 2.52, 2.64, 2.68, 2.76, 2.92, 3.08, 3.40, 3.60, 3.72, 3.88, 4.12, 4.48, 4.64, 4.68,
↪ 4.84, 4.96, 5.16, 5.20, 5.32, 5.36, 5.60, 5.72, 5.92, 5.96, 6.08, 6.24, 6.36]","tokens
↪":[" THE"," YE","LL","OW"," LA","M","P","S"," WOULD"," LIGHT"," UP"," HE","RE"," AND",
↪" THERE"," THE"," S","QUA","LI","D"," ","QUA","R","TER"," OF"," THE"," B","RA","FF","L
↪","S"]}
```

(continues on next page)

(continued from previous page)

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --tokens=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/tokens.txt \
  --encoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/encoder-epoch-99-avg-1.
↪onnx \
  --decoder=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/decoder-epoch-99-avg-1.
↪onnx \
  --joiner=./sherpa-onnx-streaming-zipformer-en-20M-2023-02-17/joiner-epoch-99-avg-1.onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

Conformer-transducer-based Models

Hint: Please refer to *Installation* to install sherpa-onnx before you read this section.

csukuangfj/sherpa-onnx-streaming-conformer-zh-2023-05-23 (Chinese)

This model is converted from

https://huggingface.co/luomingshuang/icefall_asr_wenetspeech_pruned_transducer_stateless5_streaming

which supports only Chinese as it is trained on the WenetSpeech corpus.

You can find the training code at

https://github.com/k2-fsa/icefall/tree/master/egs/wenetspeech/ASR/pruned_transducer_stateless5

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-conformer-zh-2023-05-23.tar.bz2

# For Chinese users, please use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-conformer-zh-2023-05-23.tar.bz2
```

(continues on next page)

(continued from previous page)

```
tar xvf sherpa-onnx-streaming-conformer-zh-2023-05-23.tar.bz2
rm sherpa-onnx-streaming-conformer-zh-2023-05-23.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-streaming-conformer-zh-2023-05-23 fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff    11M May 23 14:44 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff    12M May 23 14:44 decoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   160M May 23 14:46 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   345M May 23 14:47 encoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   2.7M May 23 14:44 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff    11M May 23 14:44 joiner-epoch-99-avg-1.onnx
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-conformer-zh-2023-05-23/tokens.txt \
  --encoder=./sherpa-onnx-streaming-conformer-zh-2023-05-23/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-streaming-conformer-zh-2023-05-23/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-streaming-conformer-zh-2023-05-23/joiner-epoch-99-avg-1.onnx \
  ./sherpa-onnx-streaming-conformer-zh-2023-05-23/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

You should see the following output:

```
OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-
↪streaming-conformer-zh-2023-05-23/encoder-epoch-99-avg-1.onnx", decoder_filename="./
↪sherpa-onnx-streaming-conformer-zh-2023-05-23/decoder-epoch-99-avg-1.onnx", joiner_
↪filename="./sherpa-onnx-streaming-conformer-zh-2023-05-23/joiner-epoch-99-avg-1.onnx",␣
↪tokens="./sherpa-onnx-streaming-conformer-zh-2023-05-23/tokens.txt", num_threads=2,␣
↪debug=False), lm_config=OnlineLMConfig(model="", scale=0.5), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=True, max_active_paths=4, decoding_method="greedy_search")
```

(continues on next page)

(continued from previous page)

```
sampling rate of input file: 16000
wav filename: ./sherpa-onnx-streaming-conformer-zh-2023-05-23/test_wavs/0.wav
wav duration (s): 5.611
Started
Done!
Recognition result for ./sherpa-onnx-streaming-conformer-zh-2023-05-23/test_wavs/0.wav:
{"is_final":false,"segment":0,"start_time":0.0,"text":"","timestamps":"[0.00, 0.48, 0.76,
↪ 0.88, 1.08, 1.24, 2.00, 2.04, 2.16, 2.36, 2.56, 2.72, 2.92, 3.36, 3.44, 3.64, 3.72, 3.
↪84, 3.96, 4.04, 4.16, 4.28, 4.48, 4.64, 4.84, 5.16]","tokens":["","","","","","","","",
↪"","","","","","","","
```

(continued from previous page)

```
sampling rate of input file: 16000
wav filename: ./sherpa-onnx-streaming-conformer-zh-2023-05-23/test_wavs/0.wav
wav duration (s): 5.611
Started
Done!
Recognition result for ./sherpa-onnx-streaming-conformer-zh-2023-05-23/test_wavs/0.wav:
{"is_final":false,"segment":0,"start_time":0.0,"text":"","timestamps":"[0.00, 0.48, 0.72,
↪ 0.88, 1.08, 1.24, 2.00, 2.04, 2.16, 2.28, 2.56, 2.72, 2.92, 3.36, 3.44, 3.60, 3.72, 3.
↪84, 3.92, 4.04, 4.16, 4.28, 4.48, 4.60, 4.84, 5.16]","tokens":["","","","","","","","",
↪"","","","","","","","
```

LSTM-transducer-based Models

Hint: Please refer to *Installation* to install sherpa-onnx before you read this section.

csukuangfj/sherpa-onnx-lstm-en-2023-02-17 (English)

This model trained using the GigaSpeech and the LibriSpeech dataset.

Please see https://github.com/k2-fsa/icefall/pull/558 for how the model is trained.

You can find the training code at

https://github.com/k2-fsa/icefall/tree/master/egs/librispeech/ASR/lstm_transducer_stateless2

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-lstm-
↪en-2023-02-17.tar.bz2

# For Chinese users, please use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪lstm-en-2023-02-17.tar.bz2

tar xvf sherpa-onnx-lstm-en-2023-02-17.tar.bz2
rm sherpa-onnx-lstm-en-2023-02-17.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-lstm-en-2023-02-17$ ls -lh *.onnx
-rw-r--r-- 1 kuangfangjun root  1.3M Mar 31 22:41 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root  2.0M Mar 31 22:41 decoder-epoch-99-avg-1.onnx
-rw-r--r-- 1 kuangfangjun root   80M Mar 31 22:41 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root  319M Mar 31 22:41 encoder-epoch-99-avg-1.onnx
-rw-r--r-- 1 kuangfangjun root  254K Mar 31 22:41 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root 1003K Mar 31 22:41 joiner-epoch-99-avg-1.onnx
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-lstm-en-2023-02-17/tokens.txt \
  --encoder=./sherpa-onnx-lstm-en-2023-02-17/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-lstm-en-2023-02-17/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-lstm-en-2023-02-17/joiner-epoch-99-avg-1.onnx \
  ./sherpa-onnx-lstm-en-2023-02-17/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

You should see the following output:

```
OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-lstm-
↪en-2023-02-17/encoder-epoch-99-avg-1.onnx", decoder_filename="./sherpa-onnx-lstm-en-
↪2023-02-17/decoder-epoch-99-avg-1.onnx", joiner_filename="./sherpa-onnx-lstm-en-2023-
↪02-17/joiner-epoch-99-avg-1.onnx", tokens="./sherpa-onnx-lstm-en-2023-02-17/tokens.txt
↪", num_threads=2, debug=False), endpoint_config=EndpointConfig(rule1=EndpointRule(must_
↪contain_nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0),␣
↪rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_
↪utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=0, min_utterance_length=20)), enable_endpoint=True,
max_active_paths=4, decoding_method="greedy_search")
2023-03-31 22:53:22.120185169 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 576406, index: 16, mask: {17, 53, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
2023-03-31 22:53:22.120183162 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 576405, index: 15, mask: {16, 52, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
sampling rate of input file: 16000
wav filename: ./sherpa-onnx-lstm-en-2023-02-17/test_wavs/0.wav
wav duration (s): 6.625
Started
Done!
Recognition result for ./sherpa-onnx-lstm-en-2023-02-17/test_wavs/0.wav:
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
```

(continues on next page)

(continued from previous page)

```
num threads: 2
decoding method: greedy_search
Elapsed seconds: 2.927 s
Real time factor (RTF): 2.927 / 6.625 = 0.442
```

int8

The following code shows how to use `int8` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-lstm-en-2023-02-17/tokens.txt \
  --encoder=./sherpa-onnx-lstm-en-2023-02-17/encoder-epoch-99-avg-1.int8.onnx \
  --decoder=./sherpa-onnx-lstm-en-2023-02-17/decoder-epoch-99-avg-1.int8.onnx \
  --joiner=./sherpa-onnx-lstm-en-2023-02-17/joiner-epoch-99-avg-1.int8.onnx \
  ./sherpa-onnx-lstm-en-2023-02-17/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

You should see the following output:

```
OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-lstm-
↪en-2023-02-17/encoder-epoch-99-avg-1.int8.onnx", decoder_filename="./sherpa-onnx-lstm-
↪en-2023-02-17/decoder-epoch-99-avg-1.int8.onnx", joiner_filename="./sherpa-onnx-lstm-
↪en-2023-02-17/joiner-epoch-99-avg-1.int8.onnx", tokens="./sherpa-onnx-lstm-en-2023-02-
↪17/tokens.txt", num_threads=2, debug=False), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=True, max_active_paths=4, decoding_method="greedy_search")
2023-03-31 22:55:46.608941959 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 578689, index: 16, mask: {17, 53, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
2023-03-31 22:55:46.608939862 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 578688, index: 15, mask: {16, 52, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
sampling rate of input file: 16000
wav filename: ./sherpa-onnx-lstm-en-2023-02-17/test_wavs/0.wav
wav duration (s): 6.625
Started
Done!
Recognition result for ./sherpa-onnx-lstm-en-2023-02-17/test_wavs/0.wav:
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
```

(continues on next page)

(continued from previous page)

```
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.009 s
Real time factor (RTF): 1.009 / 6.625 = 0.152
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --tokens=./sherpa-onnx-lstm-en-2023-02-17/tokens.txt \
  --encoder=./sherpa-onnx-lstm-en-2023-02-17/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-lstm-en-2023-02-17/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-lstm-en-2023-02-17/joiner-epoch-99-avg-1.onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

csukuangfj/sherpa-onnx-lstm-zh-2023-02-20 (Chinese)

This is a model trained using the WenetSpeech dataset.

Please see https://github.com/k2-fsa/icefall/pull/595 for how the model is trained.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-lstm-
↪zh-2023-02-20.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪lstm-zh-2023-02-20.tar.bz2

tar xvf sherpa-onnx-lstm-zh-2023-02-20.tar.bz2
rm sherpa-onnx-lstm-zh-2023-02-20.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-lstm-zh-2023-02-20$ ls -lh *.onnx
-rw-r--r-- 1 kuangfangjun root  12M Mar 31 20:55 decoder-epoch-11-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root  12M Mar 31 20:55 decoder-epoch-11-avg-1.onnx
-rw-r--r-- 1 kuangfangjun root  80M Mar 31 20:55 encoder-epoch-11-avg-1.int8.onnx
```

(continues on next page)

(continued from previous page)

```
-rw-r--r-- 1 kuangfangjun root 319M Mar 31 20:55 encoder-epoch-11-avg-1.onnx
-rw-r--r-- 1 kuangfangjun root 2.8M Mar 31 20:55 joiner-epoch-11-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root  11M Mar 31 20:55 joiner-epoch-11-avg-1.onnx
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-lstm-zh-2023-02-20/tokens.txt \
  --encoder=./sherpa-onnx-lstm-zh-2023-02-20/encoder-epoch-11-avg-1.onnx \
  --decoder=./sherpa-onnx-lstm-zh-2023-02-20/decoder-epoch-11-avg-1.onnx \
  --joiner=./sherpa-onnx-lstm-zh-2023-02-20/joiner-epoch-11-avg-1.onnx \
  ./sherpa-onnx-lstm-zh-2023-02-20/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-lstm-
↪zh-2023-02-20/encoder-epoch-11-avg-1.onnx", decoder_filename="./sherpa-onnx-lstm-zh-
↪2023-02-20/decoder-epoch-11-avg-1.onnx", joiner_filename="./sherpa-onnx-lstm-zh-2023-
↪02-20/joiner-epoch-11-avg-1.onnx", tokens="./sherpa-onnx-lstm-zh-2023-02-20/tokens.txt
↪", num_threads=2, debug=False), endpoint_config=EndpointConfig(rule1=EndpointRule(must_
↪contain_nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0),␣
↪rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_
↪utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=0, min_utterance_length=20)), enable_endpoint=True, max_active_paths=4,␣
↪decoding_method="greedy_search")
2023-03-31 22:58:59.348229346 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 578800, index: 15, mask: {16, 52, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
```

(continues on next page)

(continued from previous page)

```
2023-03-31 22:58:59.348231417 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 578801, index: 16, mask: {17, 53, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
sampling rate of input file: 16000
wav filename: ./sherpa-onnx-lstm-zh-2023-02-20/test_wavs/0.wav
wav duration (s): 5.611
Started
Done!
Recognition result for ./sherpa-onnx-lstm-zh-2023-02-20/test_wavs/0.wav:

num threads: 2
decoding method: greedy_search
Elapsed seconds: 3.030 s
Real time factor (RTF): 3.030 / 5.611 = 0.540
```

int8

The following code shows how to use `int8` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-lstm-zh-2023-02-20/tokens.txt \
  --encoder=./sherpa-onnx-lstm-zh-2023-02-20/encoder-epoch-11-avg-1.int8.onnx \
  --decoder=./sherpa-onnx-lstm-zh-2023-02-20/decoder-epoch-11-avg-1.int8.onnx \
  --joiner=./sherpa-onnx-lstm-zh-2023-02-20/joiner-epoch-11-avg-1.int8.onnx \
  ./sherpa-onnx-lstm-zh-2023-02-20/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineTransducerModelConfig(encoder_filename="./sherpa-onnx-lstm-
↪zh-2023-02-20/encoder-epoch-11-avg-1.int8.onnx", decoder_filename="./sherpa-onnx-lstm-
↪zh-2023-02-20/decoder-epoch-11-avg-1.int8.onnx", joiner_filename="./sherpa-onnx-lstm-
↪zh-2023-02-20/joiner-epoch-11-avg-1.int8.onnx", tokens="./sherpa-onnx-lstm-zh-2023-02-
↪20/tokens.txt", num_threads=2, debug=False), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=True, max_active_paths=4, decoding_method="greedy_search")
```

(continues on next page)

(continued from previous page)

```
2023-03-31 23:01:05.737519659 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 578880, index: 15, mask: {16, 52, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
2023-03-31 23:01:05.737521655 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 578881, index: 16, mask: {17, 53, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
sampling rate of input file: 16000
wav filename: ./sherpa-onnx-lstm-zh-2023-02-20/test_wavs/0.wav
wav duration (s): 5.611
Started
Done!
Recognition result for ./sherpa-onnx-lstm-zh-2023-02-20/test_wavs/0.wav:

num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.091 s
Real time factor (RTF): 1.091 / 5.611 = 0.194
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --tokens=./sherpa-onnx-lstm-zh-2023-02-20/tokens.txt \
  --encoder=./sherpa-onnx-lstm-zh-2023-02-20/encoder-epoch-11-avg-1.onnx \
  --decoder=./sherpa-onnx-lstm-zh-2023-02-20/decoder-epoch-11-avg-1.onnx \
  --joiner=./sherpa-onnx-lstm-zh-2023-02-20/joiner-epoch-11-avg-1.onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

8.23.2 Online paraformer models

This section lists available online paraformer models.

Paraformer models

Hint: Please refer to *Installation* to install sherpa-onnx before you read this section.

csukuangfj/sherpa-onnx-streaming-paraformer-bilingual-zh-en (Chinese + English)

Note: This model does not support timestamps. It is a bilingual model, supporting both Chinese and English. ()

This model is converted from

https://www.modelscope.cn/models/damo/speech_paraformer_asr_nat-zh-cn-16k-common-vocab8404-online/summary

The code for converting can be found at

https://huggingface.co/csukuangfj/streaming-paraformer-zh

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-paraformer-bilingual-zh-en.tar.bz2

# For Chinese users
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-paraformer-bilingual-zh-en.tar.bz2

tar xvf sherpa-onnx-streaming-paraformer-bilingual-zh-en.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-streaming-paraformer-bilingual-zh-en fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff    68M Aug 14 09:53 decoder.int8.onnx
-rw-r--r--  1 fangjun  staff   218M Aug 14 09:55 decoder.onnx
-rw-r--r--  1 fangjun  staff   158M Aug 14 09:54 encoder.int8.onnx
-rw-r--r--  1 fangjun  staff   607M Aug 14 09:57 encoder.onnx
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/tokens.txt \
  --paraformer-encoder=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/encoder.onnx \
  --paraformer-decoder=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/decoder.onnx \
  ./sherpa-onnx-streaming-paraformer-bilingual-zh-en/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/
↪tokens.txt --paraformer-encoder=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/
↪encoder.onnx --paraformer-decoder=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/
↪decoder.onnx ./sherpa-onnx-streaming-paraformer-bilingual-zh-en/test_wavs/0.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"", decoder="", joiner=""), paraformer=OnlineParaformerModelConfig(encoder="./sherpa-
↪onnx-streaming-paraformer-bilingual-zh-en/encoder.onnx", decoder="./sherpa-onnx-
↪streaming-paraformer-bilingual-zh-en/decoder.onnx"), tokens="./sherpa-onnx-streaming-
↪paraformer-bilingual-zh-en/tokens.txt", num_threads=1, debug=False, provider="cpu",␣
↪model_type=""), lm_config=OnlineLMConfig(model="", scale=0.5), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=True, max_active_paths=4, context_score=1.5, decoding_method="greedy_search")
./sherpa-onnx-streaming-paraformer-bilingual-zh-en/test_wavs/0.wav
Elapsed seconds: 2.2, Real time factor (RTF): 0.21
 monday today day is  the day after tomorrow
{"is_final":false,"segment":0,"start_time":0.0,"text":" monday today day is  the day␣
↪after tomorrow ","timestamps":"[]","tokens":["","","","mon@@","day","today","day","is",
↪"","","","the","day","after","tom@@","or@@","row","","","",""]}
```

int8

The following code shows how to use `int8` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/tokens.txt \
  --paraformer-encoder=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/encoder.int8.
↪onnx \
  --paraformer-decoder=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/decoder.int8.
↪onnx \
  ./sherpa-onnx-streaming-paraformer-bilingual-zh-en/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/
↪tokens.txt --paraformer-encoder=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/
↪encoder.int8.onnx --paraformer-decoder=./sherpa-onnx-streaming-paraformer-bilingual-zh-
↪en/decoder.int8.onnx ./sherpa-onnx-streaming-paraformer-bilingual-zh-en/test_wavs/0.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"", decoder="", joiner=""), paraformer=OnlineParaformerModelConfig(encoder="./sherpa-
↪onnx-streaming-paraformer-bilingual-zh-en/encoder.int8.onnx", decoder="./sherpa-onnx-
↪streaming-paraformer-bilingual-zh-en/decoder.int8.onnx"), tokens="./sherpa-onnx-
↪streaming-paraformer-bilingual-zh-en/tokens.txt", num_threads=1, debug=False, provider=
↪"cpu", model_type=""), lm_config=OnlineLMConfig(model="", scale=0.5), endpoint_
↪config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_nonsilence=True,␣
↪min_trailing_silence=1.2, min_utterance_length=0), rule3=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=0, min_utterance_length=20)), enable_
↪endpoint=True, max_active_paths=4, context_score=1.5, decoding_method="greedy_search")
./sherpa-onnx-streaming-paraformer-bilingual-zh-en/test_wavs/0.wav
Elapsed seconds: 1.6, Real time factor (RTF): 0.15
 monday today day is  the day after tomorrow
{"is_final":false,"segment":0,"start_time":0.0,"text":" monday today day is  the day␣
↪after tomorrow ","timestamps":"[]","tokens":["","","","mon@@","day","today","day","is",
↪"","","","the","day","after","tom@@","or@@","row","","","",""]}
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --tokens=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/tokens.txt \
  --paraformer-encoder=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/encoder.int8.
↪onnx \
  --paraformer-decoder=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/decoder.int8.
↪onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

csukuangfj/sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en (Chinese + Cantonese + English)

Note: This model does not support timestamps. It is a trilingual model, supporting both Chinese and English. ()

This model is converted from

https://modelscope.cn/models/dengcunqin/speech_paraformer-large_asr_nat-zh-cantonese-en-16k-vocab8501-online/files

You can find the conversion code after downloading and unzipping the model.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-paraformer-trilingual-zh-cantonese-en.tar.bz2

# For Chinese users
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-paraformer-trilingual-zh-cantonese-en.tar.bz2

tar xvf sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff    69M Feb 29 19:44 decoder.int8.onnx
-rw-r--r--  1 fangjun  staff   218M Feb 29 19:44 decoder.onnx
```

(continues on next page)

(continued from previous page)

```
-rw-r--r--  1 fangjun  staff   159M Feb 29 19:44 encoder.int8.onnx
-rw-r--r--  1 fangjun  staff   607M Feb 29 19:44 encoder.onnx
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en/tokens.txt \
  --paraformer-encoder=./sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en/
↪encoder.onnx \
  --paraformer-decoder=./sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en/
↪decoder.onnx \
  ./sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en/test_wavs/1.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-paraformer-trilingual-zh-
↪cantonese-en/tokens.txt --paraformer-encoder=./sherpa-onnx-streaming-paraformer-
↪trilingual-zh-cantonese-en/encoder.int8.onnx --paraformer-decoder=./sherpa-onnx-
↪streaming-paraformer-trilingual-zh-cantonese-en/decoder.int8.onnx ./sherpa-onnx-
↪streaming-paraformer-trilingual-zh-cantonese-en/test_wavs/1.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"", decoder="", joiner=""), paraformer=OnlineParaformerModelConfig(encoder="./sherpa-
↪onnx-streaming-paraformer-trilingual-zh-cantonese-en/encoder.int8.onnx", decoder="./
↪sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en/decoder.int8.onnx"), wenet_
↪ctc=OnlineWenetCtcModelConfig(model="", chunk_size=16, num_left_chunks=4), zipformer2_
↪ctc=OnlineZipformer2CtcModelConfig(model=""), tokens="./sherpa-onnx-streaming-
↪paraformer-trilingual-zh-cantonese-en/tokens.txt", num_threads=1, debug=False,␣
↪provider="cpu", model_type=""), lm_config=OnlineLMConfig(model="", scale=0.5),␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.2, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)), enable_endpoint=True, max_active_paths=4, hotwords_score=1.5
```

(continues on next page)

(continued from previous page)

```
./sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en/test_wavs/1.wav
Elapsed seconds: 0.98, Real time factor (RTF): 0.16

{ "text": "", "tokens": [ "", "", "", "", "", "", "", "", "", "", "", "" ], "timestamps
↪": [  ], "ys_probs": [  ], "lm_probs": [  ], "context_scores": [  ], "segment": 0,
↪"start_time": 0.00, "is_final": false}
```

int8

The following code shows how to use `int8` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --tokens=./sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en/tokens.txt \
  --paraformer-encoder=./sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en/
↪encoder.int8.onnx \
  --paraformer-decoder=./sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en/
↪decoder.int8.onnx \
  ./sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en/test_wavs/1.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --tokens=./sherpa-onnx-streaming-paraformer-trilingual-zh-
↪cantonese-en/tokens.txt --paraformer-encoder=./sherpa-onnx-streaming-paraformer-
↪trilingual-zh-cantonese-en/encoder.int8.onnx --paraformer-decoder=./sherpa-onnx-
↪streaming-paraformer-trilingual-zh-cantonese-en/decoder.int8.onnx ./sherpa-onnx-
↪streaming-paraformer-trilingual-zh-cantonese-en/test_wavs/1.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"", decoder="", joiner=""), paraformer=OnlineParaformerModelConfig(encoder="./sherpa-
↪onnx-streaming-paraformer-trilingual-zh-cantonese-en/encoder.int8.onnx", decoder="./
↪sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en/decoder.int8.onnx"), wenet_
↪ctc=OnlineWenetCtcModelConfig(model="", chunk_size=16, num_left_chunks=4), zipformer2_
↪ctc=OnlineZipformer2CtcModelConfig(model=""), tokens="./sherpa-onnx-streaming-
↪paraformer-trilingual-zh-cantonese-en/tokens.txt", num_threads=1, debug=False,␣
↪provider="cpu", model_type=""), lm_config=OnlineLMConfig(model="", scale=0.5),␣
↪endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_nonsilence=False, min_
↪trailing_silence=2.4, min_utterance_length=0), rule2=EndpointRule(must_contain_
↪nonsilence=True, min_trailing_silence=1.2, min_utterance_length=0),␣
↪rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_silence=0, min_
↪utterance_length=20)), enable_endpoint=True, max_active_paths=4, hotwords_score=1.5,␣
↪hotwords_file="", decoding_method="greedy_search", blank_penalty=0)
```

(continues on next page)

(continued from previous page)

```
./sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en/test_wavs/1.wav
Elapsed seconds: 0.84, Real time factor (RTF): 0.14

{ "text": "", "tokens": [ "", "", "", "", "", "", "", "", "", "", "", "" ], "timestamps
↪": [  ], "ys_probs": [  ], "lm_probs": [  ], "context_scores": [  ], "segment": 0,
↪"start_time": 0.00, "is_final": false}
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --tokens=./sherpa-onnx-streaming-paraformer-bilingual-zh-en/tokens.txt \
  --paraformer-encoder=./sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en/
↪encoder.int8.onnx \
  --paraformer-decoder=./sherpa-onnx-streaming-paraformer-trilingual-zh-cantonese-en/
↪decoder.int8.onnx
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

8.23.3 Online CTC models

This section lists available online CTC models.

Zipformer-CTC-based Models

Hint: Please refer to *Installation* to install sherpa-onnx before you read this section.

sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13 (Chinese)

Training code for this model can be found at https://github.com/k2-fsa/icefall/pull/1369. It supports only Chinese.

Please refer to https://github.com/k2-fsa/icefall/tree/master/egs/multi_zh-hans/ASR#included-training-sets for the detailed information about the training data. In total, there are 14k hours of training data.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-ctc-multi-zh-hans-2023-12-13.tar.bz2

# For Chinese users, please use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪streaming-zipformer-ctc-multi-zh-hans-2023-12-13.tar.bz2

tar xvf sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13.tar.bz2
rm sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13.tar.bz2
ls -lh sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13
```

The output is given below:

```
$ ls -lh sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13
total 654136
-rw-r--r--@ 1 fangjun  staff    28B Dec 13 16:19 README.md
-rw-r--r--@ 1 fangjun  staff   258K Dec 13 16:19 bpe.model
-rw-r--r--@ 1 fangjun  staff    68M Dec 13 16:19 ctc-epoch-20-avg-1-chunk-16-left-128.
↪int8.onnx
-rw-r--r--@ 1 fangjun  staff   252M Dec 13 16:19 ctc-epoch-20-avg-1-chunk-16-left-128.
↪onnx
drwxr-xr-x@ 8 fangjun  staff   256B Dec 13 16:19 test_wavs
-rw-r--r--@ 1 fangjun  staff    18K Dec 13 16:19 tokens.txt
```

Decode a single wave file

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --zipformer2-ctc-model=./sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13/
↪ctc-epoch-20-avg-1-chunk-16-left-128.onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13/tokens.txt \
  ./sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13/test_wavs/DEV_
↪T0000000000.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --zipformer2-ctc-model=./sherpa-onnx-streaming-zipformer-ctc-
↪multi-zh-hans-2023-12-13/ctc-epoch-20-avg-1-chunk-16-left-128.onnx --tokens=./sherpa-
↪onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13/tokens.txt ./sherpa-onnx-
↪streaming-zipformer-ctc-multi-zh-hans-2023-12-13/test_wavs/DEV_T0000000000.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"", decoder="", joiner=""), paraformer=OnlineParaformerModelConfig(encoder="", decoder=
↪""), wenet_ctc=OnlineWenetCtcModelConfig(model="", chunk_size=16, num_left_chunks=4),␣
↪zipformer2_ctc=OnlineZipformer2CtcModelConfig(model="./sherpa-onnx-streaming-zipformer-
↪ctc-multi-zh-hans-2023-12-13/ctc-epoch-20-avg-1-chunk-16-left-128.onnx"), tokens="./
↪sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13/tokens.txt", num_
↪threads=1, debug=False, provider="cpu", model_type=""), lm_config=OnlineLMConfig(model=
↪"", scale=0.5), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0),␣
↪rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_
↪utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=0, min_utterance_length=20)), enable_endpoint=True, max_active_paths=4,␣
↪hotwords_score=1.5, hotwords_file="", decoding_method="greedy_search")
./sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13/test_wavs/DEV_T0000000000.
↪wav
Elapsed seconds: 0.66, Real time factor (RTF): 0.12

{"is_final":false, "segment":0, "start_time":0.00, "text": " ", "timestamps": [0.00, 0.
↪52, 0.76, 0.84, 1.08, 1.24, 1.96, 2.04, 2.24, 2.36, 2.56, 2.68, 2.80, 3.28, 3.40, 3.60,
↪ 3.72, 3.84, 3.96, 4.04, 4.16, 4.28, 4.36, 4.60, 4.80], "tokens":[" ", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
```

int8

The following code shows how to use `int8` models to decode a wave file:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx \
  --zipformer2-ctc-model=./sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13/
↪ctc-epoch-20-avg-1-chunk-16-left-128.int8.onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13/tokens.txt \
```

(continues on next page)

(continued from previous page)

```
./sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13/test_wavs/DEV_
↪T0000000000.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx --zipformer2-ctc-model=./sherpa-onnx-streaming-zipformer-ctc-
↪multi-zh-hans-2023-12-13/ctc-epoch-20-avg-1-chunk-16-left-128.int8.onnx --tokens=./
↪sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13/tokens.txt ./sherpa-onnx-
↪streaming-zipformer-ctc-multi-zh-hans-2023-12-13/test_wavs/DEV_T0000000000.wav

OnlineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80), model_config=OnlineModelConfig(transducer=OnlineTransducerModelConfig(encoder=
↪"", decoder="", joiner=""), paraformer=OnlineParaformerModelConfig(encoder="", decoder=
↪""), wenet_ctc=OnlineWenetCtcModelConfig(model="", chunk_size=16, num_left_chunks=4),␣
↪zipformer2_ctc=OnlineZipformer2CtcModelConfig(model="./sherpa-onnx-streaming-zipformer-
↪ctc-multi-zh-hans-2023-12-13/ctc-epoch-20-avg-1-chunk-16-left-128.int8.onnx"), tokens=
↪"./sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13/tokens.txt", num_
↪threads=1, debug=False, provider="cpu", model_type=""), lm_config=OnlineLMConfig(model=
↪"", scale=0.5), endpoint_config=EndpointConfig(rule1=EndpointRule(must_contain_
↪nonsilence=False, min_trailing_silence=2.4, min_utterance_length=0),␣
↪rule2=EndpointRule(must_contain_nonsilence=True, min_trailing_silence=1.2, min_
↪utterance_length=0), rule3=EndpointRule(must_contain_nonsilence=False, min_trailing_
↪silence=0, min_utterance_length=20)), enable_endpoint=True, max_active_paths=4,␣
↪hotwords_score=1.5, hotwords_file="", decoding_method="greedy_search")
./sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13/test_wavs/DEV_T0000000000.
↪wav
Elapsed seconds: 0.44, Real time factor (RTF): 0.078

{"is_final":false, "segment":0, "start_time":0.00, "text": " ", "timestamps": [0.00, 0.
↪52, 0.76, 0.84, 1.04, 1.24, 1.96, 2.04, 2.24, 2.36, 2.56, 2.68, 2.80, 3.28, 3.40, 3.60,
↪ 3.72, 3.84, 3.96, 4.04, 4.16, 4.28, 4.36, 4.60, 4.84], "tokens":[" ", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
```

Real-time speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone \
  --zipformer2-ctc-model=./sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13/
↪ctc-epoch-20-avg-1-chunk-16-left-128.onnx \
  --tokens=./sherpa-onnx-streaming-zipformer-ctc-multi-zh-hans-2023-12-13/tokens.txt
```

Hint: If your system is Linux (including embedded Linux), you can also use *sherpa-onnx-alsa* to do real-time speech recognition with your microphone if `sherpa-onnx-microphone` does not work for you.

8.23.4 Offline transducer models

This section lists available offline transducer models.

Zipformer-transducer-based Models

Hint: Please refer to *Installation* to install *sherpa-onnx* before you read this section.

sherpa-onnx-zipformer-korean-2024-06-24 (Korean,)

PyTorch checkpoints of this model can be found at *<https://huggingface.co/johnBamma/icefall-asr-ksponspeech-zipformer-2024-06-24>*.

The training dataset can be found at https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=123.

Paper about the dataset is https://www.mdpi.com/2076-3417/10/19/6936

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-korean-2024-06-24.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-korean-2024-06-24.tar.bz2

tar xf sherpa-onnx-zipformer-korean-2024-06-24.tar.bz2
rm sherpa-onnx-zipformer-korean-2024-06-24.tar.bz2

ls -lh sherpa-onnx-zipformer-korean-2024-06-24
```

You should see the following output:

```
-rw-r--r--  1 fangjun  staff   307K Jun 24 15:33 bpe.model
-rw-r--r--  1 fangjun  staff   2.7M Jun 24 15:33 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff    11M Jun 24 15:33 decoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff    68M Jun 24 15:33 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   249M Jun 24 15:33 encoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   2.5M Jun 24 15:33 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   9.8M Jun 24 15:33 joiner-epoch-99-avg-1.onnx
drwxr-xr-x  7 fangjun  staff   224B Jun 24 15:32 test_wavs
-rw-r--r--  1 fangjun  staff    59K Jun 24 15:33 tokens.txt
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-korean-2024-06-24/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-korean-2024-06-24/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-korean-2024-06-24/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-korean-2024-06-24/joiner-epoch-99-avg-1.onnx \
  ./sherpa-onnx-zipformer-korean-2024-06-24/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:360 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-zipformer-korean-2024-06-24/
↪tokens.txt --encoder=./sherpa-onnx-zipformer-korean-2024-06-24/encoder-epoch-99-avg-1.
↪onnx --decoder=./sherpa-onnx-zipformer-korean-2024-06-24/decoder-epoch-99-avg-1.onnx --
↪joiner=./sherpa-onnx-zipformer-korean-2024-06-24/joiner-epoch-99-avg-1.onnx ./sherpa-
↪onnx-zipformer-korean-2024-06-24/test_wavs/0.wav
```

(continues on next page)

(continued from previous page)

```
OfflineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80, low_freq=20, high_freq=-400, dither=0), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-korean-2024-06-24/encoder-epoch-99-avg-1.onnx", decoder_filename=
↪"./sherpa-onnx-zipformer-korean-2024-06-24/decoder-epoch-99-avg-1.onnx", joiner_
↪filename="./sherpa-onnx-zipformer-korean-2024-06-24/joiner-epoch-99-avg-1.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), telespeech_ctc="", tokens="./sherpa-onnx-
↪zipformer-korean-2024-06-24/tokens.txt", num_threads=2, debug=False, provider="cpu",␣
↪model_type="", modeling_unit="cjkchar", bpe_vocab=""), lm_config=OfflineLMConfig(model=
↪"", scale=0.5), ctc_fst_decoder_config=OfflineCtcFstDecoderConfig(graph="", max_
↪active=3000), decoding_method="greedy_search", max_active_paths=4, hotwords_file="",␣
↪hotwords_score=1.5, blank_penalty=0, rule_fsts="", rule_fars="")
Creating recognizer ...
Started
Done!

./sherpa-onnx-zipformer-korean-2024-06-24/test_wavs/0.wav
{"text": "      .", "timestamps": [0.12, 0.24, 0.56, 1.00, 1.20, 1.32, 2.00, 2.16, 2.32,␣
↪2.52, 2.68, 2.80, 3.08, 3.28], "tokens":[" ", "", " ", " ", "", "", " ", "", "", " ",
↪" ", "", "", "."], "words": []}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.119 s
Real time factor (RTF): 0.119 / 3.526 = 0.034
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-korean-2024-06-24/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-korean-2024-06-24/encoder-epoch-99-avg-1.int8.onnx \
  --decoder=./sherpa-onnx-zipformer-korean-2024-06-24/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-korean-2024-06-24/joiner-epoch-99-avg-1.int8.onnx \
  ./sherpa-onnx-zipformer-korean-2024-06-24/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:360 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-zipformer-korean-2024-06-24/
↪tokens.txt --encoder=./sherpa-onnx-zipformer-korean-2024-06-24/encoder-epoch-99-avg-1.
↪int8.onnx --decoder=./sherpa-onnx-zipformer-korean-2024-06-24/decoder-epoch-99-avg-1.
↪onnx --joiner=./sherpa-onnx-zipformer-korean-2024-06-24/joiner-epoch-99-avg-1.int8.
↪onnx ./sherpa-onnx-zipformer-korean-2024-06-24/test_wavs/0.wav

OfflineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80, low_freq=20, high_freq=-400, dither=0), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-korean-2024-06-24/encoder-epoch-99-avg-1.int8.onnx", decoder_
↪filename="./sherpa-onnx-zipformer-korean-2024-06-24/decoder-epoch-99-avg-1.onnx",␣
↪joiner_filename="./sherpa-onnx-zipformer-korean-2024-06-24/joiner-epoch-99-avg-1.int8.
↪onnx"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), telespeech_ctc="", tokens="./sherpa-onnx-
↪zipformer-korean-2024-06-24/tokens.txt", num_threads=2, debug=False, provider="cpu",␣
↪model_type="", modeling_unit="cjkchar", bpe_vocab=""), lm_config=OfflineLMConfig(model=
↪"", scale=0.5), ctc_fst_decoder_config=OfflineCtcFstDecoderConfig(graph="", max_
↪active=3000), decoding_method="greedy_search", max_active_paths=4, hotwords_file="",␣
↪hotwords_score=1.5, blank_penalty=0, rule_fsts="", rule_fars="")
Creating recognizer ...
Started
Done!

./sherpa-onnx-zipformer-korean-2024-06-24/test_wavs/0.wav
{"text": "      .", "timestamps": [0.12, 0.24, 0.56, 1.00, 1.20, 1.32, 2.00, 2.16, 2.32,␣
↪2.52, 2.68, 2.84, 3.08, 3.28], "tokens":[" ", "", " ", " ", "", "", " ", "", "", " ",
↪" ", "", "", "."], "words": []}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.092 s
Real time factor (RTF): 0.092 / 3.526 = 0.026
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-zipformer-korean-2024-06-24/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-korean-2024-06-24/encoder-epoch-99-avg-1.int8.onnx \
  --decoder=./sherpa-onnx-zipformer-korean-2024-06-24/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-korean-2024-06-24/joiner-epoch-99-avg-1.int8.onnx
```

sherpa-onnx-zipformer-thai-2024-06-20 (Thai,)

PyTorch checkpoints of this model can be found at https://huggingface.co/yfyeung/icefall-asr-gigaspeech2-th-zipformer-2024-06-20.

The training dataset can be found at https://github.com/SpeechColab/GigaSpeech2.

The paper about the dataset is https://arxiv.org/pdf/2406.11546.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-thai-2024-06-20.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-thai-2024-06-20.tar.bz2

tar xf sherpa-onnx-zipformer-thai-2024-06-20.tar.bz2
rm sherpa-onnx-zipformer-thai-2024-06-20.tar.bz2

ls -lh sherpa-onnx-zipformer-thai-2024-06-20
```

You should see the following output:

```
-rw-r--r--  1 fangjun  staff   277K Jun 20 16:47 bpe.model
-rw-r--r--  1 fangjun  staff   1.2M Jun 20 16:47 decoder-epoch-12-avg-5.int8.onnx
-rw-r--r--  1 fangjun  staff   4.9M Jun 20 16:47 decoder-epoch-12-avg-5.onnx
-rw-r--r--  1 fangjun  staff   148M Jun 20 16:47 encoder-epoch-12-avg-5.int8.onnx
-rw-r--r--  1 fangjun  staff   565M Jun 20 16:47 encoder-epoch-12-avg-5.onnx
-rw-r--r--  1 fangjun  staff   1.0M Jun 20 16:47 joiner-epoch-12-avg-5.int8.onnx
-rw-r--r--  1 fangjun  staff   3.9M Jun 20 16:47 joiner-epoch-12-avg-5.onnx
drwxr-xr-x  6 fangjun  staff   192B Jun 20 16:46 test_wavs
-rw-r--r--  1 fangjun  staff    38K Jun 20 16:47 tokens.txt
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-thai-2024-06-20/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-thai-2024-06-20/encoder-epoch-12-avg-5.onnx \
  --decoder=./sherpa-onnx-zipformer-thai-2024-06-20/decoder-epoch-12-avg-5.onnx \
  --joiner=./sherpa-onnx-zipformer-thai-2024-06-20/joiner-epoch-12-avg-5.onnx \
  ./sherpa-onnx-zipformer-thai-2024-06-20/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-thai-2024-06-20/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-thai-2024-06-20/encoder-epoch-12-avg-5.int8.onnx \
  --decoder=./sherpa-onnx-zipformer-thai-2024-06-20/decoder-epoch-12-avg-5.onnx \
  --joiner=./sherpa-onnx-zipformer-thai-2024-06-20/joiner-epoch-12-avg-5.int8.onnx \
  ./sherpa-onnx-zipformer-thai-2024-06-20/test_wavs/0.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-zipformer-thai-2024-06-20/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-thai-2024-06-20/encoder-epoch-12-avg-5.int8.onnx \
  --decoder=./sherpa-onnx-zipformer-thai-2024-06-20/decoder-epoch-12-avg-5.onnx \
  --joiner=./sherpa-onnx-zipformer-thai-2024-06-20/joiner-epoch-12-avg-5.int8.onnx
```

sherpa-onnx-zipformer-cantonese-2024-03-13 (Cantonese,)

Training code for this model can be found at https://github.com/k2-fsa/icefall/pull/1537. It supports only Cantonese since it is trained on a `Canatonese` dataset. The paper for the dataset can be found at https://arxiv.org/pdf/2201.02419.pdf.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-cantonese-2024-03-13.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-cantonese-2024-03-13.tar.bz2

tar xf sherpa-onnx-zipformer-cantonese-2024-03-13.tar.bz2
rm sherpa-onnx-zipformer-cantonese-2024-03-13.tar.bz2

ls -lh sherpa-onnx-zipformer-cantonese-2024-03-13
```

You should see the following output:

```
total 340M
-rw-r--r-- 1 1001 127 2.7M Mar 13 09:06 decoder-epoch-45-avg-35.int8.onnx
-rw-r--r-- 1 1001 127  11M Mar 13 09:06 decoder-epoch-45-avg-35.onnx
-rw-r--r-- 1 1001 127  67M Mar 13 09:06 encoder-epoch-45-avg-35.int8.onnx
-rw-r--r-- 1 1001 127 248M Mar 13 09:06 encoder-epoch-45-avg-35.onnx
-rw-r--r-- 1 1001 127 2.4M Mar 13 09:06 joiner-epoch-45-avg-35.int8.onnx
```

(continues on next page)

(continued from previous page)

```
-rw-r--r-- 1 1001 127 9.5M Mar 13 09:06 joiner-epoch-45-avg-35.onnx
drwxr-xr-x 2 1001 127 4.0K Mar 13 09:06 test_wavs
-rw-r--r-- 1 1001 127  42K Mar 13 09:06 tokens.txt
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --blank-penalty=1.2 \
  --tokens=./sherpa-onnx-zipformer-cantonese-2024-03-13/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-cantonese-2024-03-13/encoder-epoch-45-avg-35.onnx \
  --decoder=./sherpa-onnx-zipformer-cantonese-2024-03-13/decoder-epoch-45-avg-35.onnx \
  --joiner=./sherpa-onnx-zipformer-cantonese-2024-03-13/joiner-epoch-45-avg-35.onnx \
  ./sherpa-onnx-zipformer-cantonese-2024-03-13/test_wavs/test_wavs_1.wav \
  ./sherpa-onnx-zipformer-cantonese-2024-03-13/test_wavs/test_wavs_2.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/project/sherpa-onnx/csrc/parse-options.cc:Read:361 sherpa-onnx-offline --blank-
↪penalty=1.2 --tokens=./sherpa-onnx-zipformer-cantonese-2024-03-13/tokens.txt --
↪encoder=./sherpa-onnx-zipformer-cantonese-2024-03-13/encoder-epoch-45-avg-35.onnx --
↪decoder=./sherpa-onnx-zipformer-cantonese-2024-03-13/decoder-epoch-45-avg-35.onnx --
↪joiner=./sherpa-onnx-zipformer-cantonese-2024-03-13/joiner-epoch-45-avg-35.onnx ./
↪sherpa-onnx-zipformer-cantonese-2024-03-13/test_wavs/test_wavs_1.wav ./sherpa-onnx-
↪zipformer-cantonese-2024-03-13/test_wavs/test_wavs_2.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-cantonese-2024-03-13/encoder-epoch-45-avg-35.onnx", decoder_
↪filename="./sherpa-onnx-zipformer-cantonese-2024-03-13/decoder-epoch-45-avg-35.onnx",␣
↪joiner_filename="./sherpa-onnx-zipformer-cantonese-2024-03-13/joiner-epoch-45-avg-35.
↪onnx"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet
```

(continues on next page)

(continued from previous page)

```
Creating recognizer ...
Started
Done!

./sherpa-onnx-zipformer-cantonese-2024-03-13/test_wavs/test_wavs_1.wav
{"text": "", "timestamps": [0.00, 0.88, 1.28, 1.52, 1.84, 2.08, 2.32, 2.56, 2.80, 3.04,␣
↪3.20, 3.44, 3.68, 3.92], "tokens":["", "", "", "", "", "", "", "", "", "", "", "", "",
↪""]}
----
./sherpa-onnx-zipformer-cantonese-2024-03-13/test_wavs/test_wavs_2.wav
{"text": "", "timestamps": [0.00, 0.64, 0.88, 1.12, 1.28, 1.60, 1.80, 2.16, 2.36, 2.56,␣
↪2.88, 3.08, 3.32, 3.44, 3.60], "tokens":["", "", "", "", "", "", "", "", "", "", "", "
↪", "", "", ""]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.349 s
Real time factor (RTF): 1.349 / 10.320 = 0.131
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --blank-penalty=1.2 \
  --tokens=./sherpa-onnx-zipformer-cantonese-2024-03-13/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-cantonese-2024-03-13/encoder-epoch-45-avg-35.int8.
↪onnx \
  --decoder=./sherpa-onnx-zipformer-cantonese-2024-03-13/decoder-epoch-45-avg-35.onnx \
  --joiner=./sherpa-onnx-zipformer-cantonese-2024-03-13/joiner-epoch-45-avg-35.int8.onnx␣
↪\
  ./sherpa-onnx-zipformer-cantonese-2024-03-13/test_wavs/test_wavs_1.wav \
  ./sherpa-onnx-zipformer-cantonese-2024-03-13/test_wavs/test_wavs_2.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/project/sherpa-onnx/csrc/parse-options.cc:Read:361 sherpa-onnx-offline --blank-
↪penalty=1.2 --tokens=./sherpa-onnx-zipformer-cantonese-2024-03-13/tokens.txt --
↪encoder=./sherpa-onnx-zipformer-cantonese-2024-03-13/encoder-epoch-45-avg-35.int8.onnx␣
↪--decoder=./sherpa-onnx-zipformer-cantonese-2024-03-13/decoder-epoch-45-avg-35.onnx --
↪joiner=./sherpa-onnx-zipformer-cantonese-2024-03-13/joiner-epoch-45-avg-35.int8.onnx ./
↪sherpa-onnx-zipformer-cantonese-2024-03-13/test_wavs/test_wavs_1.wav ./sherpa-onnx-
↪zipformer-cantonese-2024-03-13/test_wavs/test_wavs_2.wav
```

(continues on next page)

(continued from previous page)

```
OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-cantonese-2024-03-13/encoder-epoch-45-avg-35.int8.onnx", decoder_
↪filename="./sherpa-onnx-zipformer-cantonese-2024-03-13/decoder-epoch-45-avg-35.onnx",␣
↪joiner_filename="./sherpa-onnx-zipformer-cantonese-2024-03-13/joiner-epoch-45-avg-35.
↪int8.onnx"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), tokens="./sherpa-onnx-zipformer-cantonese-
↪2024-03-13/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type=""), lm_
↪config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=1.2)
Creating recognizer ...
Started
Done!

./sherpa-onnx-zipformer-cantonese-2024-03-13/test_wavs/test_wavs_1.wav
{"text": "", "timestamps": [0.00, 0.88, 1.28, 1.52, 1.84, 2.08, 2.32, 2.56, 2.80, 3.04,␣
↪3.20, 3.44, 3.68, 3.92], "tokens":["", "", "", "", "", "", "", "", "", "", "", "", "",
↪""]}
----
./sherpa-onnx-zipformer-cantonese-2024-03-13/test_wavs/test_wavs_2.wav
{"text": "", "timestamps": [0.00, 0.64, 0.88, 1.12, 1.28, 1.60, 1.80, 2.16, 2.36, 2.56,␣
↪2.88, 3.08, 3.32, 3.44, 3.60], "tokens":["", "", "", "", "", "", "", "", "", "", "", "
↪", "", "", ""]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.907 s
Real time factor (RTF): 0.907 / 10.320 = 0.088
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-zipformer-cantonese-2024-03-13/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-cantonese-2024-03-13/encoder-epoch-45-avg-35.int8.
↪onnx \
  --decoder=./sherpa-onnx-zipformer-cantonese-2024-03-13/decoder-epoch-45-avg-35.onnx \
  --joiner=./sherpa-onnx-zipformer-cantonese-2024-03-13/joiner-epoch-45-avg-35.int8.onnx
```

sherpa-onnx-zipformer-gigaspeech-2023-12-12 (English)

Training code for this model is https://github.com/k2-fsa/icefall/pull/1254. It supports only English since it is trained on the GigaSpeech dataset.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-gigaspeech-2023-12-12.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-gigaspeech-2023-12-12.tar.bz2

tar xf sherpa-onnx-zipformer-gigaspeech-2023-12-12.tar.bz2
rm sherpa-onnx-zipformer-gigaspeech-2023-12-12.tar.bz2
ls -lh sherpa-onnx-zipformer-gigaspeech-2023-12-12
```

You should see the following output:

```
$ ls -lh sherpa-onnx-zipformer-gigaspeech-2023-12-12
total 656184
-rw-r--r--  1 fangjun  staff    28B Dec 12 19:00 README.md
-rw-r--r--  1 fangjun  staff   239K Dec 12 19:00 bpe.model
-rw-r--r--  1 fangjun  staff   528K Dec 12 19:00 decoder-epoch-30-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   2.0M Dec 12 19:00 decoder-epoch-30-avg-1.onnx
-rw-r--r--  1 fangjun  staff    68M Dec 12 19:00 encoder-epoch-30-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   249M Dec 12 19:00 encoder-epoch-30-avg-1.onnx
-rw-r--r--  1 fangjun  staff   253K Dec 12 19:00 joiner-epoch-30-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   1.0M Dec 12 19:00 joiner-epoch-30-avg-1.onnx
drwxr-xr-x  5 fangjun  staff   160B Dec 12 19:00 test_wavs
-rw-r--r--  1 fangjun  staff   4.9K Dec 12 19:00 tokens.txt
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/encoder-epoch-30-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/decoder-epoch-30-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/joiner-epoch-30-avg-1.onnx \
  ./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1089-134686-0001.wav \
  ./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1221-135766-0001.wav \
  ./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1221-135766-0002.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/
↪tokens.txt --encoder=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/encoder-epoch-30-
↪avg-1.onnx --decoder=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/decoder-epoch-30-
↪avg-1.onnx --joiner=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/joiner-epoch-30-avg-
↪1.onnx ./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1089-134686-0001.wav ./
↪sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1221-135766-0001.wav ./sherpa-
↪onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1221-135766-0002.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-gigaspeech-2023-12-12/encoder-epoch-30-avg-1.onnx", decoder_
↪filename="./sherpa-onnx-zipformer-gigaspeech-2023-12-12/decoder-epoch-30-avg-1.onnx",␣
↪joiner_filename="./sherpa-onnx-zipformer-gigaspeech-2023-12-12/joiner-epoch-30-avg-1.
↪onnx"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), tokens="./sherpa-onnx-zipformer-gigaspeech-
↪2023-12-12/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type=""), lm_
↪config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5)
```

(continues on next page)

(continued from previous page)

```
Creating recognizer ...
Started
Done!

./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1089-134686-0001.wav
{"text": " AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE␣
↪SQUALID QUARTER OF THE BROTHELS", "timestamps": [0.00, 0.36, 0.52, 0.68, 0.96, 1.00, 1.
↪08, 1.28, 1.40, 1.48, 1.60, 1.76, 1.80, 1.88, 1.92, 2.00, 2.20, 2.32, 2.36, 2.48, 2.60,
↪ 2.80, 2.84, 2.92, 3.12, 3.32, 3.56, 3.76, 4.04, 4.20, 4.32, 4.40, 4.56, 4.80, 4.92, 5.
↪08, 5.36, 5.48, 5.64, 5.72, 5.88, 6.04, 6.24], "tokens":[" AFTER", " E", "AR", "LY", "
↪", "N", "IGHT", "F", "AL", "L", " THE", " ", "Y", "E", "LL", "OW", " LA", "M", "P", "S
↪", " WOULD", " ", "L", "IGHT", " UP", " HERE", " AND", " THERE", " THE", " S", "QU",
↪"AL", "ID", " QU", "AR", "TER", " OF", " THE", " B", "RO", "TH", "EL", "S"]}
----
./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1221-135766-0001.wav
{"text": " GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER␣
↪A LOVELY CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT FOR␣
↪EVER WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN",
↪"timestamps": [0.00, 0.16, 0.40, 0.68, 0.84, 0.96, 1.04, 1.12, 1.32, 1.52, 1.68, 1.76,␣
↪2.00, 2.12, 2.28, 2.40, 2.64, 2.92, 3.20, 3.32, 3.52, 3.64, 3.76, 3.96, 4.12, 4.36, 4.
↪52, 4.72, 4.92, 5.16, 5.40, 5.64, 5.76, 5.88, 6.12, 6.28, 6.48, 6.84, 7.08, 7.32, 7.60,
↪ 7.92, 8.12, 8.24, 8.36, 8.48, 8.64, 8.76, 8.88, 9.12, 9.32, 9.48, 9.56, 9.60, 9.76,␣
↪10.00, 10.12, 10.20, 10.44, 10.68, 10.80, 11.00, 11.20, 11.36, 11.52, 11.76, 12.00, 12.
↪12, 12.24, 12.28, 12.52, 12.72, 12.84, 12.96, 13.04, 13.24, 13.40, 13.64, 13.76, 14.00,
↪ 14.08, 14.24, 14.52, 14.68, 14.80, 15.00, 15.04, 15.28, 15.52, 15.76, 16.00, 16.12,␣
↪16.20, 16.32], "tokens":[" GO", "D", " AS", " A", " DI", "RE", "C", "T", " CON", "SE",
↪"QU", "ENCE", " OF", " THE", " S", "IN", " WHICH", " MAN", " TH", "US", " P", "UN",
↪"ISH", "ED", " HAD", " GIVE", "N", " HER", " A", " LOVE", "LY", " CHI", "L", "D", " WHO
↪", "SE", " PLACE", " WAS", " ON", " THAT", " SAME", " DIS", "HO", "N", "OR", "ED", " BO
↪", "S", "OM", " TO", " CON", "NE", "C", "T", " HER", " PA", "R", "ENT", " FOR", " E",
↪"VER", " WITH", " THE", " RA", "CE", " AND", " DE", "S", "C", "ENT", " OF", " MO", "R",
↪ "T", "AL", "S", " AND", " TO", " BE", " F", "IN", "ALLY", " A", " B", "LES", "S", "ED
↪", " SO", "UL", " IN", " HE", "A", "VE", "N"]}
----
./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1221-135766-0002.wav
{"text": " YET THESE THOUGHTS AFFECTED HESTER PRYNE LESS WITH HOPE THAN APPREHENSION",
↪"timestamps": [0.00, 0.04, 0.12, 0.40, 0.68, 0.88, 0.96, 1.12, 1.20, 1.32, 1.44, 1.48,␣
↪1.64, 1.76, 1.88, 2.04, 2.16, 2.28, 2.52, 2.68, 2.72, 2.88, 3.12, 3.28, 3.52, 3.80, 4.
↪00, 4.16, 4.24, 4.40, 4.48], "tokens":[" ", "Y", "ET", " THESE", " THOUGH", "T", "S",
↪" A", "FF", "E", "C", "TED", " HE", "S", "TER", " P", "RY", "NE", " LE", "S", "S", "␣
↪WITH", " HO", "PE", " THAN", " APP", "RE", "HE", "N", "S", "ION"]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.407 s
Real time factor (RTF): 1.407 / 28.165 = 0.050
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/encoder-epoch-30-avg-1.int8.
↪onnx \
  --decoder=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/decoder-epoch-30-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/joiner-epoch-30-avg-1.int8.onnx␣
↪\
  ./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1089-134686-0001.wav \
  ./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1221-135766-0001.wav \
  ./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1221-135766-0002.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/
↪tokens.txt --encoder=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/encoder-epoch-30-
↪avg-1.int8.onnx --decoder=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/decoder-epoch-
↪30-avg-1.onnx --joiner=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/joiner-epoch-30-
↪avg-1.int8.onnx ./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1089-134686-
↪0001.wav ./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1221-135766-0001.wav .
↪/sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1221-135766-0002.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-gigaspeech-2023-12-12/encoder-epoch-30-avg-1.int8.onnx", decoder_
↪filename="./sherpa-onnx-zipformer-gigaspeech-2023-12-12/decoder-epoch-30-avg-1.onnx",␣
↪joiner_filename="./sherpa-onnx-zipformer-gigaspeech-2023-12-12/joiner-epoch-30-avg-1.
↪int8.onnx"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), tokens="./sherpa-onnx-zipformer-gigaspeech-
↪2023-12-12/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type=""), lm_
↪config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5)
Creating recognizer ...
Started
Done!

./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1089-134686-0001.wav
{"text": " AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE␣
↪SQUALID QUARTER OF THE BROTHELS", "timestamps": [0.00, 0.36, 0.52, 0.68, 0.96, 1.00, 1.
↪08, 1.28, 1.40, 1.48, 1.60, 1.76, 1.80, 1.88, 1.92, 2.00, 2.20, 2.32, 2.36, 2.48, 2.60,
↪2.80, 2.84, 2.92, 3.12, 3.32, 3.56, 3.76, 4.04, 4.24, 4.32, 4.40, 4.56, 4.80, 4.92, 5.
↪08, 5.36, 5.48, 5.64, 5.72, 5.88, 6.04, 6.24], "tokens":[" AFTER", " E", "AR", "LY",
↪", "N", "IGHT", "F", "AL", "L", " THE", " ", "Y", "E", "LL", "OW", " LA", "M", "P", "S
↪", " WOULD", " ", "L", "IGHT", " UP", " HERE", " AND", " THERE", " THE", " S", "QU",
↪"AL", "ID", " QU", "AR", "TER", " OF", " THE", " B", "RO", "TH", "EL", "S"]}
```

(continues on next page)

(continued from previous page)

```
----
./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1221-135766-0001.wav
{"text": " GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER␣
↪A LOVELY CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT FOR␣
↪EVER WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN",
↪"timestamps": [0.00, 0.16, 0.40, 0.68, 0.84, 0.96, 1.08, 1.12, 1.32, 1.52, 1.68, 1.76,␣
↪2.00, 2.12, 2.28, 2.40, 2.64, 2.92, 3.20, 3.32, 3.52, 3.64, 3.76, 3.96, 4.12, 4.36, 4.
↪52, 4.72, 4.92, 5.16, 5.40, 5.64, 5.76, 5.88, 6.12, 6.28, 6.52, 6.84, 7.08, 7.32, 7.60,
↪ 7.92, 8.12, 8.24, 8.36, 8.48, 8.64, 8.76, 8.88, 9.12, 9.32, 9.48, 9.56, 9.60, 9.76,␣
↪10.00, 10.12, 10.20, 10.44, 10.68, 10.80, 11.00, 11.20, 11.36, 11.52, 11.76, 12.00, 12.
↪12, 12.24, 12.28, 12.52, 12.72, 12.84, 12.96, 13.04, 13.24, 13.44, 13.64, 13.76, 14.00,
↪ 14.08, 14.24, 14.52, 14.68, 14.80, 15.00, 15.04, 15.28, 15.48, 15.76, 16.00, 16.12,␣
↪16.16, 16.32], "tokens":[" GO", "D", " AS", " A", " DI", "RE", "C", "T", " CON", "SE",
↪"QU", "ENCE", " OF", " THE", " S", "IN", " WHICH", " MAN", " TH", "US", " P", "UN",
↪"ISH", "ED", " HAD", " GIVE", "N", " HER", " A", " LOVE", "LY", " CHI", "L", "D", " WHO
↪", "SE", " PLACE", " WAS", " ON", " THAT", " SAME", " DIS", "HO", "N", "OR", "ED", " BO
↪", "S", "OM", " TO", " CON", "NE", "C", "T", " HER", " PA", "R", "ENT", " FOR", " E",
↪"VER", " WITH", " THE", " RA", "CE", " AND", " DE", "S", "C", "ENT", " OF", " MO", "R",
↪ "T", "AL", "S", " AND", " TO", " BE", " F", "IN", "ALLY", " A", " B", "LES", "S", "ED
↪", " SO", "UL", " IN", " HE", "A", "VE", "N"]}
----
./sherpa-onnx-zipformer-gigaspeech-2023-12-12/test_wavs/1221-135766-0002.wav
{"text": " YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION",
↪"timestamps": [0.00, 0.04, 0.12, 0.40, 0.68, 0.88, 0.96, 1.12, 1.24, 1.32, 1.44, 1.48,␣
↪1.64, 1.76, 1.88, 2.04, 2.16, 2.28, 2.32, 2.52, 2.68, 2.72, 2.88, 3.12, 3.32, 3.52, 3.
↪80, 4.00, 4.16, 4.24, 4.40, 4.48], "tokens":[" ", "Y", "ET", " THESE", " THOUGH", "T",
↪"S", " A", "FF", "E", "C", "TED", " HE", "S", "TER", " P", "RY", "N", "NE", " LE", "S",
↪ "S", " WITH", " HO", "PE", " THAN", " APP", "RE", "HE", "N", "S", "ION"]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.101 s
Real time factor (RTF): 1.101 / 28.165 = 0.039
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/encoder-epoch-30-avg-1.int8.
↪onnx \
  --decoder=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/decoder-epoch-30-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/joiner-epoch-30-avg-1.int8.onnx
```

Speech recognition from a microphone with VAD

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/silero_vad.onnx

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/silero_vad.
↪onnx

./build/bin/sherpa-onnx-vad-microphone-offline-asr \
  --silero-vad-model=./silero_vad.onnx \
  --tokens=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/encoder-epoch-30-avg-1.int8.
↪onnx \
  --decoder=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/decoder-epoch-30-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-gigaspeech-2023-12-12/joiner-epoch-30-avg-1.int8.onnx
```

zrjin/sherpa-onnx-zipformer-multi-zh-hans-2023-9-2 (Chinese)

This model is from

https://huggingface.co/zrjin/sherpa-onnx-zipformer-multi-zh-hans-2023-9-2

which supports Chinese as it is trained on whatever datasets involved in the multi-zh_hans recipe.

If you are interested in how the model is trained, please refer to https://github.com/k2-fsa/icefall/pull/1238.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-multi-zh-hans-2023-9-2.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-multi-zh-hans-2023-9-2.tar.bz2

tar xvf sherpa-onnx-zipformer-multi-zh-hans-2023-9-2.tar.bz2
rm sherpa-onnx-zipformer-multi-zh-hans-2023-9-2.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-zipformer-multi-zh-hans-2023-9-2 zengruijin$ ls -lh *.onnx
-rw-rw-r--@ 1 zengruijin  staff   1.2M Sep 18 07:04 decoder-epoch-20-avg-1.int8.onnx
-rw-rw-r--@ 1 zengruijin  staff   4.9M Sep 18 07:04 decoder-epoch-20-avg-1.onnx
-rw-rw-r--@ 1 zengruijin  staff    66M Sep 18 07:04 encoder-epoch-20-avg-1.int8.onnx
```

(continues on next page)

(continued from previous page)

```
-rw-rw-r--@ 1 zengruijin  staff   248M Sep 18 07:05 encoder-epoch-20-avg-1.onnx
-rw-rw-r--@ 1 zengruijin  staff   1.0M Sep 18 07:05 joiner-epoch-20-avg-1.int8.onnx
-rw-rw-r--@ 1 zengruijin  staff   3.9M Sep 18 07:05 joiner-epoch-20-avg-1.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/encoder-epoch-20-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/decoder-epoch-20-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/joiner-epoch-20-avg-1.onnx \
  ./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/runner/work/sherpa-onnx/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361␣
↪sherpa-onnx-offline --tokens=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/tokens.txt␣
↪--encoder=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/encoder-epoch-20-avg-1.onnx --
↪decoder=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/decoder-epoch-20-avg-1.onnx --
↪joiner=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/joiner-epoch-20-avg-1.onnx ./
↪sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/test_wavs/0.wav ./sherpa-onnx-zipformer-
↪multi-zh-hans-2023-9-2/test_wavs/1.wav ./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/
↪test_wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/encoder-epoch-20-avg-1.onnx", decoder_
↪filename="./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/decoder-epoch-20-avg-1.onnx",␣
↪joiner_filename="./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/joiner-epoch-20-avg-1.
↪onnx"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪"), tdnn=OfflineTdnnModelConfig(model=""), tokens="./sherpa-onnx-zipformer-multi-zh-
↪hans-2023-9-2/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type="
↪lm_config=OfflineLMConfig(model="", scale=0.5), decoding_method="greedy_search", max_
↪active_paths=4, hotwords_file="", hotwords_score=1.5)
```

(continues on next page)

(continued from previous page)

```
Creating recognizer ...
Started
/Users/runner/work/sherpa-onnx/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:117 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/test_wavs/0.wav
{"text":" ","timestamps":"[0.00, 0.16, 0.40, 0.60, 0.84, 1.08, 1.60, 1.72, 1.88, 2.04, 2.
↪24, 2.44, 2.60, 2.96, 3.12, 3.32, 3.40, 3.60, 3.72, 3.84, 4.00, 4.16, 4.32, 4.52, 4.68]
↪","tokens":[" ","","","","","","","","","","","","","","","","","","","","","","","","
↪"]}
----
./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/test_wavs/1.wav
{"text":" <0xE8><0x8D><0xA1>","timestamps":"[0.00, 0.12, 0.48, 0.68, 0.92, 1.12, 1.28, 1.
↪48, 1.80, 2.04, 2.40, 2.56, 2.76, 2.96, 3.08, 3.32, 3.48, 3.68, 3.84, 4.00, 4.20, 4.24,
↪ 4.28, 4.40, 4.60, 4.84]","tokens":[" ","","","","","","","","","","","","","","","","
↪","","","","<0xE8>","<0x8D>","<0xA1>","","",""]}
----
./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/test_wavs/8k.wav
{"text":" <0xE8><0x8D><0xA1>","timestamps":"[0.00, 0.04, 0.24, 0.52, 0.76, 1.00, 1.40, 1.
↪64, 1.80, 2.12, 2.32, 2.64, 2.80, 3.00, 3.20, 3.24, 3.28, 3.44, 3.64, 3.76, 3.96, 4.20]
↪","tokens":[" ","","","","","","","","","","","","","","<0xE8>","<0x8D>","<0xA1>","","
↪","","","",""]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.362 s
Real time factor (RTF): 0.362 / 15.289 = 0.024
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/encoder-epoch-20-avg-1.int8.
↪onnx \
  --decoder=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/decoder-epoch-20-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/joiner-epoch-20-avg-1.int8.
↪onnx \
  ./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/runner/work/sherpa-onnx/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361␣
↪sherpa-onnx-offline --tokens=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/tokens.txt␣
↪--encoder=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/encoder-epoch-20-avg-1.int8.
↪onnx --decoder=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/decoder-epoch-20-avg-1.
↪onnx --joiner=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/joiner-epoch-20-avg-1.
↪int8.onnx ./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/test_wavs/0.wav ./sherpa-onnx-
↪zipformer-multi-zh-hans-2023-9-2/test_wavs/1.wav ./sherpa-onnx-zipformer-multi-zh-hans-
↪2023-9-2/test_wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/encoder-epoch-20-avg-1.int8.onnx",␣
↪decoder_filename="./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/decoder-epoch-20-avg-
↪1.onnx", joiner_filename="./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/joiner-epoch-
↪20-avg-1.int8.onnx"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪"), tdnn=OfflineTdnnModelConfig(model=""), tokens="./sherpa-onnx-zipformer-multi-zh-
↪hans-2023-9-2/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type=""),␣
↪lm_config=OfflineLMConfig(model="", scale=0.5), decoding_method="greedy_search", max_
↪active_paths=4, hotwords_file="", hotwords_score=1.5)
Creating recognizer ...
Started
/Users/runner/work/sherpa-onnx/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:117 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/test_wavs/0.wav
{"text":" ","timestamps":"[0.00, 0.16, 0.40, 0.60, 0.84, 1.08, 1.60, 1.72, 1.88, 2.04, 2.
↪28, 2.44, 2.60, 2.96, 3.12, 3.32, 3.40, 3.60, 3.76, 3.84, 4.00, 4.16, 4.32, 4.52, 4.56]
↪","tokens":[" ","","","","","","","","","","","","","","","","","","","","","","","","
↪"]}
----
./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/test_wavs/1.wav
{"text":" <0xE8><0x8D><0xA1>","timestamps":"[0.00, 0.12, 0.48, 0.68, 0.92, 1.12, 1.28, 1.
↪48, 1.80, 2.04, 2.40, 2.56, 2.76, 2.96, 3.08, 3.32, 3.48, 3.68, 3.84, 4.00, 4.20, 4.24,
↪ 4.28, 4.40, 4.60, 4.84]","tokens":[" ","","","","","","","","","","","","","","","","
↪","","","","<0xE8>","<0x8D>","<0xA1>","","","]}
----
./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/test_wavs/8k.wav
{"text":" <0xE8><0x8D><0xA1>","timestamps":"[0.00, 0.04, 0.24, 0.52, 0.76, 1.00, 1.40, 1.
↪64, 1.80, 2.12, 2.36, 2.64, 2.80, 3.04, 3.16, 3.20, 3.24, 3.44, 3.64, 3.76, 3.96, 4.20]
↪","tokens":[" ","","","","","","","","","","","","","","","<0xE8>","<0x8D>","<0xA1>","","
↪","","","]}
```

(continues on next page)

(continued from previous page)

```
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.305 s
Real time factor (RTF): 0.305 / 15.289 = 0.020
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/encoder-epoch-20-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/decoder-epoch-0-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-multi-zh-hans-2023-9-2/joiner-epoch-20-avg-1.onnx
```

yfyeung/icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17 (English)

This model is from

https://huggingface.co/yfyeung/icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17

which supports only English as it is trained on the CommonVoice English dataset.

If you are interested in how the model is trained, please refer to https://github.com/k2-fsa/icefall/pull/997.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/icefall-asr-cv-
↪corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/icefall-asr-
↪cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17.tar.bz2

tar xvf icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17.
↪tar.bz2
rm icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17.tar.
↪bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17 fangjun
↪$ ls -lh exp/*epoch-60-avg-20*.onnx
-rw-r--r--  1 fangjun  staff   1.2M Jun 27 09:53 exp/decoder-epoch-60-avg-20.int8.onnx
-rw-r--r--  1 fangjun  staff   2.0M Jun 27 09:54 exp/decoder-epoch-60-avg-20.onnx
-rw-r--r--  1 fangjun  staff   121M Jun 27 09:54 exp/encoder-epoch-60-avg-20.int8.onnx
-rw-r--r--  1 fangjun  staff   279M Jun 27 09:55 exp/encoder-epoch-60-avg-20.onnx
-rw-r--r--  1 fangjun  staff   253K Jun 27 09:53 exp/joiner-epoch-60-avg-20.int8.onnx
-rw-r--r--  1 fangjun  staff   1.0M Jun 27 09:53 exp/joiner-epoch-60-avg-20.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-
↪04-17/data/lang_bpe_500/tokens.txt \
  --encoder=./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-
↪04-17/exp/encoder-epoch-60-avg-20.onnx \
  --decoder=./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-
↪04-17/exp/decoder-epoch-60-avg-20.onnx \
  --joiner=./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-
↪04-17/exp/joiner-epoch-60-avg-20.onnx \
  ./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/
↪test_wavs/1089-134686-0001.wav \
  ./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/
↪test_wavs/1221-135766-0001.wav \
  ./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/
↪test_wavs/1221-135766-0002.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./icefall-asr-cv-corpus-13.0-2023-03-09-en-
↪pruned-transducer-stateless7-2023-04-17/data/lang_bpe_500/tokens.txt --encoder=./
↪icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/exp/
↪encoder-epoch-60-avg-20.onnx --decoder=./icefall-asr-cv-corpus-13.0-2023-03-09-en-
↪pruned-transducer-stateless7-2023-04-17/exp/decoder-epoch-60-avg-20.onnx --joiner=./
↪icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/exp/
↪joiner-epoch-60-avg-20.onnx ./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-
↪transducer-stateless7-2023-04-17/test_wavs/1089-134686-0001.wav ./icefall-asr-cv-
↪corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/test_wavs/1221-
↪135766-0001.wav ./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-
↪stateless7-2023-04-17/test_wavs/1221-135766-0002.wav
```

(continues on next page)

(continued from previous page)

```
OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/exp/
↪encoder-epoch-60-avg-20.onnx", decoder_filename="./icefall-asr-cv-corpus-13.0-2023-03-
↪09-en-pruned-transducer-stateless7-2023-04-17/exp/decoder-epoch-60-avg-20.onnx",␣
↪joiner_filename="./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-
↪stateless7-2023-04-17/exp/joiner-epoch-60-avg-20.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./icefall-asr-cv-corpus-13.0-
↪2023-03-09-en-pruned-transducer-stateless7-2023-04-17/data/lang_bpe_500/tokens.txt",␣
↪num_threads=2, debug=False, provider="cpu"), lm_config=OfflineLMConfig(model="",␣
↪scale=0.5), decoding_method="greedy_search", max_active_paths=4, context_score=1.5)
Creating recognizer ...
Started
Done!

./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/test_
↪wavs/1089-134686-0001.wav
{"text":" AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE␣
↪SQUALID QUARTER OF THE BROTHELS","timestamps":"[0.00, 0.64, 0.76, 0.84, 1.04, 1.08, 1.
↪16, 1.32, 1.44, 1.56, 1.72, 1.84, 1.88, 1.92, 1.96, 2.04, 2.16, 2.32, 2.48, 2.56, 2.76,
↪ 2.80, 2.84, 3.08, 3.28, 3.40, 3.52, 3.68, 4.00, 4.24, 4.28, 4.52, 4.68, 4.84, 4.88, 4.
↪96, 5.04, 5.28, 5.40, 5.52, 5.72, 5.88, 6.08]","tokens":[" AFTER"," E","AR","LY"," ","N
↪","IGHT","F","AL","L"," THE"," ","Y","E","LL","OW"," LA","MP","S"," WOULD"," ","L",
↪"IGHT"," UP"," HE","RE"," AND"," THERE"," THE"," S","QUA","LI","D"," ","QUA","R","TER",
↪" OF"," THE"," BRO","TH","EL","S"]}
----
./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/test_
↪wavs/1221-135766-0001.wav
{"text":" GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A␣
↪LOVELY CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT␣
↪FOREVER WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
↪","timestamps":"[0.04, 0.44, 0.64, 0.84, 0.96, 1.32, 1.52, 1.68, 1.84, 1.88, 2.04, 2.
↪16, 2.32, 2.40, 2.64, 2.88, 3.12, 3.24, 3.44, 3.52, 3.72, 3.88, 4.20, 4.40, 4.48, 4.60,
↪ 4.76, 4.96, 5.08, 5.24, 5.36, 5.56, 5.80, 6.20, 6.32, 6.52, 6.92, 7.16, 7.36, 7.60, 7.
↪76, 7.92, 8.16, 8.28, 8.40, 8.48, 8.60, 8.76, 8.84, 9.08, 9.24, 9.44, 9.48, 9.72, 9.88,
↪ 10.04, 10.12, 10.52, 10.76, 10.84, 11.08, 11.24, 11.36, 11.60, 11.76, 11.96, 12.08,␣
↪12.24, 12.28, 12.48, 12.72, 12.84, 12.92, 13.00, 13.20, 13.52, 13.76, 13.88, 14.08, 14.
↪28, 14.52, 14.64, 14.76, 14.96, 15.04, 15.24, 15.48, 15.68, 15.84, 16.00, 16.04]",
↪"tokens":[" GO","D"," AS"," A"," DIRECT"," CON","SE","QUE","N","CE"," OF"," THE"," S",
↪"IN"," WHICH"," MAN"," TH","US"," P","UN","ISH","ED"," HAD"," G","IVE","N"," HER"," A",
↪" LO","VE","LY"," CHI","LD"," WHO","SE"," PLACE"," WAS"," ON"," THAT"," SA","ME"," DIS
↪","HO","N","OR","ED"," BO","S","OM"," TO"," CON","N","ECT"," HER"," PA","R","ENT"," FOR
↪","E","VER"," WITH"," THE"," RA","CE"," AND"," DE","S","C","ENT"," OF"," MO","R","T",
↪"AL","S"," AND"," TO"," BE"," FIN","ALLY"," A"," B","LES","S","ED"," SO","UL"," IN","␣
↪HE","A","VEN"]}
----
./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/test_
↪wavs/1221-135766-0002.wav
{"text":" YET THESE THOUGHTS AFFECTED HESTER PRIN LESS WITH HOPE THAN APPREHENSION",
↪"timestamps":"[0.00, 0.04, 0.12, 0.56, 0.80, 0.88, 1.00, 1.04, 1.12, 1.20, 1.28, 1.40,
↪1.52, 1.64, 1.76, 1.84, 2.04, 2.24, 2.40, 2.64, 2.68, 2.84, 3.04, 3.24, 3.44, 3.52, 3.
↪72, 3.92, 4.00, 4.16, 4.24, 4.36]","tokens":[" ","Y","ET"," THESE"," TH","O","UGH","T",
↪"S"," A","FF","ECT","ED"," HE","S","TER"," PRI","N"," LE","S","S"," WITH"," HO","PE",␣
↪TH","AN"," APP","RE","HE","N","S","ION"]}
```

(continued on next page)

(continued from previous page)

```
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.611 s
Real time factor (RTF): 1.611 / 28.165 = 0.057
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-
↪04-17/data/lang_bpe_500/tokens.txt \
  --encoder=./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-
↪04-17/exp/encoder-epoch-60-avg-20.int8.onnx \
  --decoder=./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-
↪04-17/exp/decoder-epoch-60-avg-20.onnx \
  --joiner=./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-
↪04-17/exp/joiner-epoch-60-avg-20.int8.onnx \
  ./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/
↪test_wavs/1089-134686-0001.wav \
  ./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/
↪test_wavs/1221-135766-0001.wav \
  ./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/
↪test_wavs/1221-135766-0002.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./icefall-asr-cv-corpus-13.0-2023-03-09-en-
↪pruned-transducer-stateless7-2023-04-17/data/lang_bpe_500/tokens.txt --encoder=./
↪icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/exp/
↪encoder-epoch-60-avg-20.int8.onnx --decoder=./icefall-asr-cv-corpus-13.0-2023-03-09-en-
↪pruned-transducer-stateless7-2023-04-17/exp/decoder-epoch-60-avg-20.onnx --joiner=./
↪icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/exp/
↪joiner-epoch-60-avg-20.int8.onnx ./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-
↪transducer-stateless7-2023-04-17/test_wavs/1089-134686-0001.wav ./icefall-asr-cv-
↪corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/test_wavs/1221-
↪135766-0001.wav ./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-
↪stateless7-2023-04-17/test_wavs/1221-135766-0002.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/exp/
↪encoder-epoch-60-avg-20.int8.onnx", decoder_filename="./icefall-asr-cv-corpus-13.0-
↪2023-03-09-en-pruned-transducer-stateless7-2023-04-17/exp/decoder-epoch-60-avg-20.onnx
↪", joiner_filename="./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-
↪stateless7-2023-04-17/exp/joiner-epoch-60-avg-20.int8.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./icefall-asr-cv-corpus-13.0-
↪2023-03-09-en-pruned-transducer-stateless7-2023-04-17/data/lang_bpe_500/tokens.txt",␣
↪num_threads=2, debug=False, provider="cpu"), lm_config=OfflineLMConfig(model="",
```

(continues on next page)

(continued from previous page)

```
Creating recognizer ...
Started
Done!

./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/test_
↪wavs/1089-134686-0001.wav
{"text":" AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE␣
↪SQUALID QUARTER OF THE BROTHELS","timestamps":"[0.00, 0.64, 0.76, 0.84, 1.04, 1.08, 1.
↪16, 1.36, 1.44, 1.56, 1.72, 1.84, 1.88, 1.92, 1.96, 2.04, 2.20, 2.32, 2.48, 2.56, 2.76,
↪ 2.80, 2.84, 3.08, 3.28, 3.40, 3.52, 3.68, 4.00, 4.24, 4.28, 4.52, 4.68, 4.84, 4.88, 4.
↪96, 5.04, 5.28, 5.36, 5.52, 5.72, 5.88, 6.08]","tokens":[" AFTER"," E","AR","LY"," ","N
↪","IGHT","F","AL","L"," THE"," ","Y","E","LL","OW"," LA","MP","S"," WOULD"," ","L",
↪"IGHT"," UP"," HE","RE"," AND"," THERE"," THE"," S","QUA","LI","D"," ","QUA","R","TER",
↪" OF"," THE"," BRO","TH","EL","S"]}
----
./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/test_
↪wavs/1221-135766-0001.wav
{"text":" GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A␣
↪LOVELY CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT␣
↪FOREVER WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
↪","timestamps":"[0.04, 0.44, 0.64, 0.84, 0.96, 1.32, 1.52, 1.68, 1.84, 1.88, 2.04, 2.
↪16, 2.32, 2.40, 2.64, 2.88, 3.12, 3.24, 3.44, 3.52, 3.72, 3.88, 4.20, 4.40, 4.48, 4.60,
↪ 4.76, 4.96, 5.08, 5.24, 5.36, 5.56, 5.80, 6.20, 6.32, 6.52, 6.92, 7.16, 7.32, 7.60, 7.
↪76, 7.92, 8.16, 8.28, 8.40, 8.48, 8.60, 8.76, 8.84, 9.08, 9.24, 9.44, 9.48, 9.72, 9.88,
↪ 10.04, 10.12, 10.52, 10.76, 10.84, 11.08, 11.24, 11.36, 11.60, 11.76, 11.96, 12.08,␣
↪12.24, 12.28, 12.48, 12.72, 12.84, 12.92, 13.00, 13.20, 13.52, 13.76, 13.88, 14.08, 14.
↪28, 14.52, 14.64, 14.76, 14.96, 15.04, 15.24, 15.48, 15.68, 15.84, 16.00, 16.04]",
↪"tokens":[" GO","D"," AS"," A"," DIRECT"," CON","SE","QUE","N","CE"," OF"," THE"," S",
↪"IN"," WHICH"," MAN"," TH","US"," P","UN","ISH","ED"," HAD"," G","IVE","N"," HER"," A",
↪" LO","VE","LY"," CHI","LD"," WHO","SE"," PLACE"," WAS"," ON"," THAT"," SA","ME"," DIS
↪","HO","N","OR","ED"," BO","S","OM"," TO"," CON","N","ECT"," HER"," PA","R","ENT"," FOR
↪","E","VER"," WITH"," THE"," RA","CE"," AND"," DE","S","C","ENT"," OF"," MO","R","T",
↪"AL","S"," AND"," TO"," BE"," FIN","ALLY"," A"," B","LES","S","ED"," SO","UL"," IN","␣
↪HE","A","VEN"]}
----
./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-04-17/test_
↪wavs/1221-135766-0002.wav
{"text":" YET THESE THOUGHTS AFFECTED HESTER PRIN LESS WITH HOPE THAN APPREHENSION",
↪"timestamps":"[0.00, 0.04, 0.12, 0.56, 0.80, 0.88, 1.00, 1.04, 1.12, 1.20, 1.28, 1.40,␣
↪1.52, 1.64, 1.76, 1.84, 2.04, 2.24, 2.40, 2.64, 2.68, 2.84, 3.04, 3.24, 3.44, 3.52, 3.
↪72, 3.92, 4.00, 4.16, 4.24, 4.36]","tokens":[" ","Y","ET"," THESE"," TH","O","UGH","T",
↪"S"," A","FF","ECT","ED"," HE","S","TER"," PRI","N"," LE","S","S"," WITH"," HO","PE","␣
↪TH","AN"," APP","RE","HE","N","S","ION"]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.368 s
Real time factor (RTF): 1.368 / 28.165 = 0.049
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-
↪04-17/data/lang_bpe_500/tokens.txt \
  --encoder=./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-
↪04-17/exp/encoder-epoch-60-avg-20.onnx \
  --decoder=./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-
↪04-17/exp/decoder-epoch-60-avg-20.onnx \
  --joiner=./icefall-asr-cv-corpus-13.0-2023-03-09-en-pruned-transducer-stateless7-2023-
↪04-17/exp/joiner-epoch-60-avg-20.onnx
```

pkufool/icefall-asr-zipformer-wenetspeech-20230615 (Chinese)

This model is from

https://huggingface.co/pkufool/icefall-asr-zipformer-wenetspeech-20230615

which supports only Chinese as it is trained on the WenetSpeech corpus.

If you are interested in how the model is trained, please refer to https://github.com/k2-fsa/icefall/pull/1130.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/icefall-asr-
↪zipformer-wenetspeech-20230615.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/icefall-asr-
↪zipformer-wenetspeech-20230615.tar.bz2

tar xvf icefall-asr-zipformer-wenetspeech-20230615.tar.bz2
rm icefall-asr-zipformer-wenetspeech-20230615.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
icefall-asr-zipformer-wenetspeech-20230615 fangjun$ ls -lh exp/*.onnx
-rw-r--r--  1 fangjun  staff    11M Jun 26 14:31 exp/decoder-epoch-12-avg-4.int8.onnx
-rw-r--r--  1 fangjun  staff    12M Jun 26 14:31 exp/decoder-epoch-12-avg-4.onnx
-rw-r--r--  1 fangjun  staff    66M Jun 26 14:32 exp/encoder-epoch-12-avg-4.int8.onnx
-rw-r--r--  1 fangjun  staff   248M Jun 26 14:34 exp/encoder-epoch-12-avg-4.onnx
-rw-r--r--  1 fangjun  staff   2.7M Jun 26 14:31 exp/joiner-epoch-12-avg-4.int8.onnx
-rw-r--r--  1 fangjun  staff    11M Jun 26 14:31 exp/joiner-epoch-12-avg-4.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./icefall-asr-zipformer-wenetspeech-20230615/data/lang_char/tokens.txt \
  --encoder=./icefall-asr-zipformer-wenetspeech-20230615/exp/encoder-epoch-12-avg-4.onnx␣
↪\
  --decoder=./icefall-asr-zipformer-wenetspeech-20230615/exp/decoder-epoch-12-avg-4.onnx␣
↪\
  --joiner=./icefall-asr-zipformer-wenetspeech-20230615/exp/joiner-epoch-12-avg-4.onnx \
  ./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_T0000000000.wav \
  ./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_T0000000001.wav \
  ./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_T0000000002.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./icefall-asr-zipformer-wenetspeech-20230615/
↪data/lang_char/tokens.txt --encoder=./icefall-asr-zipformer-wenetspeech-20230615/exp/
↪encoder-epoch-12-avg-4.onnx --decoder=./icefall-asr-zipformer-wenetspeech-20230615/exp/
↪decoder-epoch-12-avg-4.onnx --joiner=./icefall-asr-zipformer-wenetspeech-20230615/exp/
↪joiner-epoch-12-avg-4.onnx ./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_
↪T0000000000.wav ./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_T0000000001.
↪wav ./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_T0000000002.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪icefall-asr-zipformer-wenetspeech-20230615/exp/encoder-epoch-12-avg-4.onnx", decoder_
↪filename="./icefall-asr-zipformer-wenetspeech-20230615/exp/decoder-epoch-12-avg-4.onnx
↪", joiner_filename="./icefall-asr-zipformer-wenetspeech-20230615/exp/joiner-epoch-12-
↪avg-4.onnx"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./icefall-asr-zipformer-
↪wenetspeech-20230615/data/lang_char/tokens.txt", num_threads=2, debug=False, provider=
↪"cpu"), lm_config=OfflineLMConfig(model="", scale=0.5), decoding_method="greedy_search
↪", max_active_paths=4, context_score=1.5)
```

(continues on next page)

(continued from previous page)

```
Creating recognizer ...
Started
Done!

./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_T0000000000.wav
{"text":"","timestamps":"[0.00, 0.12, 0.48, 0.64, 0.88, 1.16, 1.64, 1.76, 1.92, 2.08, 2.
↪32, 2.48, 2.64, 3.08, 3.20, 3.40, 3.48, 3.64, 3.76, 3.88, 3.96, 4.12, 4.28, 4.52, 4.72,
↪ 4.84]","tokens":["","","","","","","","","","","","","","","","","","","","","","","",
↪"","",""]}
----
./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_T0000000001.wav
{"text":"","timestamps":"[0.00, 0.16, 0.48, 0.72, 0.92, 1.08, 1.28, 1.52, 1.92, 2.08, 2.
↪52, 2.64, 2.88, 3.04, 3.20, 3.40, 3.56, 3.76, 3.84, 4.00, 4.16, 4.32, 4.56, 4.84]",
↪"tokens":["","","","","","","","","","","","","","","","","","","","","","","","",""]}
----
./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_T0000000002.wav
{"text":"","timestamps":"[0.00, 0.32, 0.56, 0.84, 1.12, 1.44, 1.68, 1.84, 2.28, 2.48, 2.
↪76, 2.92, 3.12, 3.28, 3.44, 3.60, 3.72, 3.92, 4.20]","tokens":["","","","","","","","",
↪"","","","","","","","","","","","","","","","","","","","","","","","","","","",""]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.458 s
Real time factor (RTF): 0.458 / 15.289 = 0.030
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./icefall-asr-zipformer-wenetspeech-20230615/data/lang_char/tokens.txt \
  --encoder=./icefall-asr-zipformer-wenetspeech-20230615/exp/encoder-epoch-12-avg-4.int8.
↪onnx \
  --decoder=./icefall-asr-zipformer-wenetspeech-20230615/exp/decoder-epoch-12-avg-4.onnx␣
↪\
  --joiner=./icefall-asr-zipformer-wenetspeech-20230615/exp/joiner-epoch-12-avg-4.int8.
↪onnx \
  ./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_T0000000000.wav \
  ./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_T0000000001.wav \
  ./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_T0000000002.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./icefall-asr-zipformer-wenetspeech-20230615/
↪data/lang_char/tokens.txt --encoder=./icefall-asr-zipformer-wenetspeech-20230615/exp/
↪encoder-epoch-12-avg-4.int8.onnx --decoder=./icefall-asr-zipformer-wenetspeech-
↪20230615/exp/decoder-epoch-12-avg-4.onnx --joiner=./icefall-asr-zipformer-wenetspeech-
↪20230615/exp/joiner-epoch-12-avg-4.int8.onnx ./icefall-asr-zipformer-wenetspeech-
↪20230615/test_wavs/DEV_T0000000000.wav ./icefall-asr-zipformer-wenetspeech-20230615/
↪test_wavs/DEV_T0000000001.wav ./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/
↪DEV_T0000000002.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪icefall-asr-zipformer-wenetspeech-20230615/exp/encoder-epoch-12-avg-4.int8.onnx",␣
↪decoder_filename="./icefall-asr-zipformer-wenetspeech-20230615/exp/decoder-epoch-12-
↪avg-4.onnx", joiner_filename="./icefall-asr-zipformer-wenetspeech-20230615/exp/joiner-
↪epoch-12-avg-4.int8.onnx"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./icefall-asr-zipformer-
↪wenetspeech-20230615/data/lang_char/tokens.txt", num_threads=2, debug=False, provider=
↪"cpu"), lm_config=OfflineLMConfig(model="", scale=0.5), decoding_method="greedy_search
↪", max_active_paths=4, context_score=1.5)
Creating recognizer ...
Started
Done!

./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_T0000000000.wav
{"text":"","timestamps":"[0.00, 0.12, 0.48, 0.60, 0.80, 1.08, 1.64, 1.76, 1.92, 2.08, 2.
↪32, 2.48, 2.64, 3.08, 3.20, 3.28, 3.44, 3.60, 3.72, 3.84, 3.92, 4.12, 4.28, 4.48, 4.72,
↪ 4.84]","tokens":["","","","","","","","","","","","","","","","","","","","","","","
↪","","",""]}
----
./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_T0000000001.wav
{"text":"","timestamps":"[0.00, 0.16, 0.48, 0.68, 0.84, 1.08, 1.20, 1.48, 1.64, 2.08, 2.
↪36, 2.52, 2.64, 2.84, 3.00, 3.16, 3.40, 3.52, 3.72, 3.84, 4.00, 4.16, 4.32, 4.56, 4.84]
↪","tokens":["","","","","","","","","","","","","","","","","","","","","","","","
↪"]}
----
./icefall-asr-zipformer-wenetspeech-20230615/test_wavs/DEV_T0000000002.wav
{"text":"","timestamps":"[0.00, 0.12, 0.48, 0.84, 1.08, 1.44, 1.60, 1.84, 2.24, 2.48, 2.
↪76, 2.88, 3.12, 3.24, 3.28, 3.36, 3.60, 3.72, 3.84, 4.16]","tokens":["","","","","","
↪","","","","","","","","","","","","","","","]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.338 s
Real time factor (RTF): 0.338 / 15.289 = 0.022
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./icefall-asr-zipformer-wenetspeech-20230615/data/lang_char/tokens.txt \
  --encoder=./icefall-asr-zipformer-wenetspeech-20230615/exp/encoder-epoch-12-avg-4.onnx␣
↪\
  --decoder=./icefall-asr-zipformer-wenetspeech-20230615/exp/decoder-epoch-12-avg-4.onnx␣
↪\
  --joiner=./icefall-asr-zipformer-wenetspeech-20230615/exp/joiner-epoch-12-avg-4.onnx
```

csukuangfj/sherpa-onnx-zipformer-large-en-2023-06-26 (English)

This model is converted from

https://huggingface.co/Zengwei/icefall-asr-librispeech-zipformer-large-2023-05-16

which supports only English as it is trained on the LibriSpeech corpus.

You can find the training code at

https://github.com/k2-fsa/icefall/tree/master/egs/librispeech/ASR/zipformer

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-large-en-2023-06-26.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-large-en-2023-06-26.tar.bz2

tar xvf sherpa-onnx-zipformer-large-en-2023-06-26.tar.bz2
rm sherpa-onnx-zipformer-large-en-2023-06-26.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-zipformer-large-en-2023-06-26 fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff   1.2M Jun 26 13:19 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   2.0M Jun 26 13:19 decoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   145M Jun 26 13:20 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   564M Jun 26 13:22 encoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   253K Jun 26 13:19 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   1.0M Jun 26 13:19 joiner-epoch-99-avg-1.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-large-en-2023-06-26/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-large-en-2023-06-26/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-large-en-2023-06-26/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-large-en-2023-06-26/joiner-epoch-99-avg-1.onnx \
  ./sherpa-onnx-zipformer-large-en-2023-06-26/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-large-en-2023-06-26/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-large-en-2023-06-26/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-zipformer-large-en-2023-06-26/
↪tokens.txt --encoder=./sherpa-onnx-zipformer-large-en-2023-06-26/encoder-epoch-99-avg-
↪1.onnx --decoder=./sherpa-onnx-zipformer-large-en-2023-06-26/decoder-epoch-99-avg-1.
↪onnx --joiner=./sherpa-onnx-zipformer-large-en-2023-06-26/joiner-epoch-99-avg-1.onnx ./
↪sherpa-onnx-zipformer-large-en-2023-06-26/test_wavs/0.wav ./sherpa-onnx-zipformer-
↪large-en-2023-06-26/test_wavs/1.wav ./sherpa-onnx-zipformer-large-en-2023-06-26/test_
↪wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-large-en-2023-06-26/encoder-epoch-99-avg-1.onnx", decoder_
↪filename="./sherpa-onnx-zipformer-large-en-2023-06-26/decoder-epoch-99-avg-1.onnx",␣
↪joiner_filename="./sherpa-onnx-zipformer-large-en-2023-06-26/joiner-epoch-99-avg-1.onnx
↪"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./sherpa-onnx-zipformer-large-
↪en-2023-06-26/tokens.txt", num_threads=2, debug=False, provider="cpu"), lm_
↪config=OfflineLMConfig(model="", scale=0.5), decoding_method="greedy_search", max_
↪active_paths=4, context_score=1.5)
Creating recognizer ...
Started
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:108 Creating a resampler:
   in_sample_rate: 8000
```

(continues on next page)

(continued from previous page)

```
   output_sample_rate: 16000

Done!

./sherpa-onnx-zipformer-large-en-2023-06-26/test_wavs/0.wav
{"text":" AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE␣
↪SQUALID QUARTER OF THE BROTHELS","timestamps":"[0.00, 0.48, 0.60, 0.72, 1.04, 1.28, 1.
↪36, 1.48, 1.60, 1.84, 1.96, 2.00, 2.16, 2.32, 2.40, 2.48, 2.60, 2.80, 3.04, 3.28, 3.40,
↪ 3.56, 3.76, 4.04, 4.24, 4.28, 4.48, 4.64, 4.80, 4.84, 5.00, 5.04, 5.28, 5.40, 5.56, 5.
↪60, 5.76, 5.96, 6.16]","tokens":[" AFTER"," E","AR","LY"," NIGHT","F","A","LL"," THE",
↪" YE","LL","OW"," LA","M","P","S"," WOULD"," LIGHT"," UP"," HE","RE"," AND"," THERE","␣
↪THE"," S","QUA","LI","D"," ","QUA","R","TER"," OF"," THE"," B","RO","TH","EL","S"]}
----
./sherpa-onnx-zipformer-large-en-2023-06-26/test_wavs/1.wav
{"text":" GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A␣
↪LOVELY CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT FOR␣
↪EVER WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN",
↪"timestamps":"[0.00, 0.20, 0.48, 0.72, 0.88, 1.04, 1.12, 1.20, 1.36, 1.52, 1.68, 1.84,␣
↪1.88, 2.00, 2.12, 2.32, 2.36, 2.60, 2.84, 3.12, 3.24, 3.48, 3.56, 3.76, 3.92, 4.12, 4.
↪36, 4.56, 4.72, 4.96, 5.16, 5.44, 5.68, 6.12, 6.28, 6.48, 6.88, 7.12, 7.36, 7.56, 7.92,
↪ 8.16, 8.28, 8.40, 8.48, 8.60, 8.76, 8.88, 9.08, 9.28, 9.44, 9.52, 9.60, 9.72, 9.92,␣
↪10.00, 10.12, 10.48, 10.68, 10.76, 11.00, 11.20, 11.36, 11.56, 11.76, 12.00, 12.12, 12.
↪28, 12.32, 12.52, 12.72, 12.84, 12.92, 13.04, 13.20, 13.44, 13.64, 13.76, 14.00, 14.12,
↪ 14.24, 14.36, 14.52, 14.72, 14.80, 15.04, 15.28, 15.52, 15.76, 16.00, 16.20, 16.24,␣
↪16.32]","tokens":[" GO","D"," AS"," A"," DI","RE","C","T"," CON","SE","QUE","N","CE","␣
↪OF"," THE"," S","IN"," WHICH"," MAN"," TH","US"," P","UN","ISH","ED"," HAD"," GIVE","N
↪"," HER"," A"," LOVE","LY"," CHILD"," WHO","SE"," PLACE"," WAS"," ON"," THAT"," SAME",
↪" DIS","HO","N","OR","ED"," BO","S","OM"," TO"," CON","NE","C","T"," HER"," P","AR",
↪"ENT"," FOR"," E","VER"," WITH"," THE"," RA","CE"," AND"," DE","S","C","ENT"," OF"," MO
↪","R","T","AL","S"," AND"," TO"," BE"," FI","N","AL","LY"," A"," B","LESS","ED"," SO",
↪"UL"," IN"," HE","A","VE","N"]}
----
./sherpa-onnx-zipformer-large-en-2023-06-26/test_wavs/8k.wav
{"text":" YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION",
↪"timestamps":"[0.00, 0.12, 0.36, 0.48, 0.76, 0.96, 1.12, 1.24, 1.32, 1.44, 1.48, 1.68,␣
↪1.76, 1.88, 2.04, 2.12, 2.24, 2.28, 2.48, 2.56, 2.80, 3.08, 3.28, 3.52, 3.80, 3.92, 4.
↪00, 4.16, 4.24, 4.36, 4.44]","tokens":[" YE","T"," THE","SE"," THOUGHT","S"," A","FF",
↪"E","C","TED"," HE","S","TER"," P","RY","N","NE"," ","LESS"," WITH"," HO","PE"," THAN",
↪" A","PP","RE","HE","N","S","ION"]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.843 s
Real time factor (RTF): 1.843 / 28.165 = 0.065
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-large-en-2023-06-26/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-large-en-2023-06-26/encoder-epoch-99-avg-1.int8.onnx␣
↪\
  --decoder=./sherpa-onnx-zipformer-large-en-2023-06-26/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-large-en-2023-06-26/joiner-epoch-99-avg-1.int8.onnx \
  ./sherpa-onnx-zipformer-large-en-2023-06-26/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-large-en-2023-06-26/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-large-en-2023-06-26/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-zipformer-large-en-2023-06-26/
↪tokens.txt --encoder=./sherpa-onnx-zipformer-large-en-2023-06-26/encoder-epoch-99-avg-
↪1.int8.onnx --decoder=./sherpa-onnx-zipformer-large-en-2023-06-26/decoder-epoch-99-avg-
↪1.onnx --joiner=./sherpa-onnx-zipformer-large-en-2023-06-26/joiner-epoch-99-avg-1.int8.
↪onnx ./sherpa-onnx-zipformer-large-en-2023-06-26/test_wavs/0.wav ./sherpa-onnx-
↪zipformer-large-en-2023-06-26/test_wavs/1.wav ./sherpa-onnx-zipformer-large-en-2023-06-
↪26/test_wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-large-en-2023-06-26/encoder-epoch-99-avg-1.int8.onnx", decoder_
↪filename="./sherpa-onnx-zipformer-large-en-2023-06-26/decoder-epoch-99-avg-1.onnx",␣
↪joiner_filename="./sherpa-onnx-zipformer-large-en-2023-06-26/joiner-epoch-99-avg-1.
↪int8.onnx"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./sherpa-onnx-zipformer-large-
↪en-2023-06-26/tokens.txt", num_threads=2, debug=False, provider="cpu"), lm_
↪config=OfflineLMConfig(model="", scale=0.5), decoding_method="greedy_search", max_
↪active_paths=4, context_score=1.5)
Creating recognizer ...
Started
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:108 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-zipformer-large-en-2023-06-26/test_wavs/0.wav
{"text":" AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE␣
↪SQUALID QUARTER OF THE BROTHELS","timestamps":"[0.00, 0.48, 0.60, 0.72, 1.04, 1.28, 1.
↪36, 1.48, 1.60, 1.84, 1.96, 2.00, 2.16, 2.32, 2.40, 2.48, 2.60, 2.80, 3.04, 3.28, 3.40,
↪ 3.56, 3.76, 4.04, 4.24, 4.28, 4.48, 4.64, 4.80, 4.84, 5.00, 5.04, 5.28, 5.40, 5.56, 5.
↪60, 5.76, 5.96, 6.16]","tokens":[" AFTER"," E","AR","LY"," NIGHT","F","A","LL"," THE",
↪" YE","LL","OW"," LA","M","P","S"," WOULD"," LIGHT"," UP"," HE","RE"," AND"," THERE"," ␣
↪THE"," S","QUA","LI","D"," ","QUA","R","TER"," OF"," THE"," B","RO","TH","EL","S"]}
```

(continued on next page)

(continued from previous page)

```
----
./sherpa-onnx-zipformer-large-en-2023-06-26/test_wavs/1.wav
{"text":" GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A␣
↪LOVELY CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT FOR␣
↪EVER WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN",
↪"timestamps":"[0.00, 0.20, 0.48, 0.72, 0.88, 1.04, 1.12, 1.20, 1.36, 1.52, 1.64, 1.84,␣
↪1.88, 2.00, 2.12, 2.32, 2.36, 2.60, 2.84, 3.12, 3.24, 3.48, 3.56, 3.76, 3.92, 4.12, 4.
↪36, 4.52, 4.72, 4.96, 5.16, 5.44, 5.68, 6.12, 6.28, 6.48, 6.88, 7.12, 7.36, 7.56, 7.92,
↪ 8.16, 8.28, 8.40, 8.48, 8.60, 8.76, 8.88, 9.08, 9.28, 9.44, 9.52, 9.60, 9.72, 9.92,␣
↪10.00, 10.12, 10.48, 10.68, 10.76, 11.00, 11.20, 11.36, 11.56, 11.76, 12.00, 12.12, 12.
↪28, 12.32, 12.52, 12.72, 12.84, 12.92, 13.04, 13.20, 13.44, 13.64, 13.76, 14.00, 14.08,
↪ 14.24, 14.36, 14.52, 14.72, 14.76, 15.04, 15.28, 15.52, 15.76, 16.00, 16.20, 16.24,␣
↪16.32]","tokens":[" GO","D"," AS"," A"," DI","RE","C","T"," CON","SE","QUE","N","CE","␣
↪OF"," THE"," S","IN"," WHICH"," MAN"," TH","US"," P","UN","ISH","ED"," HAD"," GIVE","N
↪"," HER"," A"," LOVE","LY"," CHILD"," WHO","SE"," PLACE"," WAS"," ON"," THAT"," SAME",
↪" DIS","HO","N","OR","ED"," BO","S","OM"," TO"," CON","NE","C","T"," HER"," P","AR",
↪"ENT"," FOR"," E","VER"," WITH"," THE"," RA","CE"," AND"," DE","S","C","ENT"," OF"," MO
↪","R","T","AL","S"," AND"," TO"," BE"," FI","N","AL","LY"," A"," B","LESS","ED"," SO",
↪"UL"," IN"," HE","A","VE","N"]}
----
./sherpa-onnx-zipformer-large-en-2023-06-26/test_wavs/8k.wav
{"text":" YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION",
↪"timestamps":"[0.00, 0.12, 0.36, 0.48, 0.76, 0.96, 1.12, 1.24, 1.32, 1.44, 1.48, 1.68,␣
↪1.76, 1.88, 2.04, 2.12, 2.28, 2.32, 2.48, 2.52, 2.80, 3.08, 3.28, 3.52, 3.76, 3.92, 4.
↪00, 4.16, 4.24, 4.36, 4.44]","tokens":[" YE","T"," THE","SE"," THOUGHT","S"," A","FF",
↪"E","C","TED"," HE","S","TER"," P","RY","N","NE"," ","LESS"," WITH"," HO","PE"," THAN",
↪" A","PP","RE","HE","N","S","ION"]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.490 s
Real time factor (RTF): 1.490 / 28.165 = 0.053
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-zipformer-large-en-2023-06-26/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-large-en-2023-06-26/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-large-en-2023-06-26/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-large-en-2023-06-26/joiner-epoch-99-avg-1.onnx
```

csukuangfj/sherpa-onnx-zipformer-small-en-2023-06-26 (English)

This model is converted from

https://huggingface.co/Zengwei/icefall-asr-librispeech-zipformer-small-2023-05-16

which supports only English as it is trained on the LibriSpeech corpus.

You can find the training code at

https://github.com/k2-fsa/icefall/tree/master/egs/librispeech/ASR/zipformer

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-small-en-2023-06-26.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-small-en-2023-06-26.tar.bz2

tar xvf sherpa-onnx-zipformer-small-en-2023-06-26.tar.bz2
rm sherpa-onnx-zipformer-small-en-2023-06-26.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-zipformer-small-en-2023-06-26 fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff   1.2M Jun 26 13:04 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   2.0M Jun 26 13:04 decoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff    25M Jun 26 13:04 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff    87M Jun 26 13:04 encoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   253K Jun 26 13:04 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   1.0M Jun 26 13:04 joiner-epoch-99-avg-1.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-small-en-2023-06-26/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-small-en-2023-06-26/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-small-en-2023-06-26/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-small-en-2023-06-26/joiner-epoch-99-avg-1.onnx \
  ./sherpa-onnx-zipformer-small-en-2023-06-26/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-small-en-2023-06-26/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-small-en-2023-06-26/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-zipformer-small-en-2023-06-26/
↪tokens.txt --encoder=./sherpa-onnx-zipformer-small-en-2023-06-26/encoder-epoch-99-avg-
↪1.onnx --decoder=./sherpa-onnx-zipformer-small-en-2023-06-26/decoder-epoch-99-avg-1.
↪onnx --joiner=./sherpa-onnx-zipformer-small-en-2023-06-26/joiner-epoch-99-avg-1.onnx ./
↪sherpa-onnx-zipformer-small-en-2023-06-26/test_wavs/0.wav ./sherpa-onnx-zipformer-
↪small-en-2023-06-26/test_wavs/1.wav ./sherpa-onnx-zipformer-small-en-2023-06-26/test_
↪wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-small-en-2023-06-26/encoder-epoch-99-avg-1.onnx", decoder_
↪filename="./sherpa-onnx-zipformer-small-en-2023-06-26/decoder-epoch-99-avg-1.onnx",␣
↪joiner_filename="./sherpa-onnx-zipformer-small-en-2023-06-26/joiner-epoch-99-avg-1.onnx
↪"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./sherpa-onnx-zipformer-small-
↪en-2023-06-26/tokens.txt", num_threads=2, debug=False, provider="cpu"), lm_
↪config=OfflineLMConfig(model="", scale=0.5), decoding_method="greedy_search", max_
↪active_paths=4, context_score=1.5)
Creating recognizer ...
Started
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:108 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-zipformer-small-en-2023-06-26/test_wavs/0.wav
{"text":" AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE␣
↪SQUALID QUARTER OF THE BROTHELS","timestamps":"[0.00, 0.64, 0.76, 0.84, 1.12, 1.36, 1.
↪44, 1.56, 1.72, 1.84, 1.96, 2.04, 2.20, 2.32, 2.36, 2.44, 2.60, 2.76, 3.04, 3.24, 3.40,
↪ 3.52, 3.72, 4.04, 4.20, 4.28, 4.48, 4.64, 4.80, 4.84, 4.96, 5.00, 5.28, 5.40, 5.52, 5.
↪60, 5.76, 5.92, 6.08]","tokens":[" AFTER"," E","AR","LY"," NIGHT","F","A","LL"," THE",
↪" YE","LL","OW"," LA","M","P","S"," WOULD"," LIGHT"," UP"," HE","RE"," AND"," THERE"," ␣
↪THE"," S","QUA","LI","D"," ","QUA","R","TER"," OF"," THE"," B","RO","TH","EL","S"]}
```

(continued from previous page)

```
----
./sherpa-onnx-zipformer-small-en-2023-06-26/test_wavs/1.wav
{"text":" GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A␣
↪LOVELY CHILD WHOSE PLACE WAS ON THAT SAME DISHONOURED BOSOM TO CONNECT HER PARENT␣
↪FOREVER WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
↪","timestamps":"[0.00, 0.32, 0.64, 0.80, 0.96, 1.08, 1.16, 1.20, 1.32, 1.52, 1.68, 1.
↪80, 1.88, 2.04, 2.16, 2.32, 2.40, 2.64, 2.88, 3.16, 3.20, 3.44, 3.52, 3.72, 3.88, 4.16,
↪ 4.44, 4.60, 4.76, 4.96, 5.16, 5.36, 5.60, 6.16, 6.32, 6.52, 6.88, 7.16, 7.32, 7.60, 7.
↪96, 8.16, 8.28, 8.36, 8.48, 8.64, 8.76, 8.84, 9.04, 9.28, 9.44, 9.52, 9.60, 9.68, 9.88,
↪ 9.92, 10.12, 10.52, 10.76, 10.80, 11.08, 11.20, 11.36, 11.56, 11.76, 11.96, 12.08, 12.
↪24, 12.28, 12.48, 12.68, 12.80, 12.92, 13.00, 13.20, 13.48, 13.72, 13.84, 14.04, 14.20,
↪ 14.28, 14.40, 14.56, 14.68, 14.76, 15.00, 15.24, 15.48, 15.68, 15.92, 16.08, 16.12,␣
↪16.20]","tokens":[" GO","D"," AS"," A"," DI","RE","C","T"," CON","SE","QUE","N","CE","␣
↪OF"," THE"," S","IN"," WHICH"," MAN"," TH","US"," P","UN","ISH","ED"," HAD"," GIVE","N
↪"," HER"," A"," LOVE","LY"," CHILD"," WHO","SE"," PLACE"," WAS"," ON"," THAT"," SAME",
↪" DIS","HO","N","OUR","ED"," BO","S","OM"," TO"," CON","NE","C","T"," HER"," P","AR",
↪"ENT"," FOR","E","VER"," WITH"," THE"," RA","CE"," AND"," DE","S","C","ENT"," OF"," MO
↪","R","T","AL","S"," AND"," TO"," BE"," FI","N","AL","LY"," A"," B","LESS","ED"," SO",
↪"UL"," IN"," HE","A","VE","N"]}
----
./sherpa-onnx-zipformer-small-en-2023-06-26/test_wavs/8k.wav
{"text":" YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION",
↪"timestamps":"[0.00, 0.32, 0.48, 0.64, 0.84, 1.08, 1.20, 1.32, 1.36, 1.44, 1.48, 1.64,␣
↪1.76, 1.88, 2.08, 2.12, 2.24, 2.28, 2.44, 2.48, 2.80, 3.04, 3.24, 3.48, 3.72, 3.88, 3.
↪92, 4.08, 4.16, 4.24, 4.36]","tokens":[" YE","T"," THE","SE"," THOUGHT","S"," A","FF",
↪"E","C","TED"," HE","S","TER"," P","RY","N","NE"," ","LESS"," WITH"," HO","PE"," THAN",
↪" A","PP","RE","HE","N","S","ION"]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.953 s
Real time factor (RTF): 0.953 / 28.165 = 0.034
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-small-en-2023-06-26/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-small-en-2023-06-26/encoder-epoch-99-avg-1.int8.onnx␣
↪\
  --decoder=./sherpa-onnx-zipformer-small-en-2023-06-26/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-small-en-2023-06-26/joiner-epoch-99-avg-1.int8.onnx \
  ./sherpa-onnx-zipformer-small-en-2023-06-26/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-small-en-2023-06-26/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-small-en-2023-06-26/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-zipformer-small-en-2023-06-26/
↪tokens.txt --encoder=./sherpa-onnx-zipformer-small-en-2023-06-26/encoder-epoch-99-avg-
↪1.int8.onnx --decoder=./sherpa-onnx-zipformer-small-en-2023-06-26/decoder-epoch-99-avg-
↪1.onnx --joiner=./sherpa-onnx-zipformer-small-en-2023-06-26/joiner-epoch-99-avg-1.int8.
↪onnx ./sherpa-onnx-zipformer-small-en-2023-06-26/test_wavs/0.wav ./sherpa-onnx-
↪zipformer-small-en-2023-06-26/test_wavs/1.wav ./sherpa-onnx-zipformer-small-en-2023-06-
↪26/test_wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-small-en-2023-06-26/encoder-epoch-99-avg-1.int8.onnx", decoder_
↪filename="./sherpa-onnx-zipformer-small-en-2023-06-26/decoder-epoch-99-avg-1.onnx",␣
↪joiner_filename="./sherpa-onnx-zipformer-small-en-2023-06-26/joiner-epoch-99-avg-1.
↪int8.onnx"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./sherpa-onnx-zipformer-small-
↪en-2023-06-26/tokens.txt", num_threads=2, debug=False, provider="cpu"), lm_
↪config=OfflineLMConfig(model="", scale=0.5), decoding_method="greedy_search", max_
↪active_paths=4, context_score=1.5)
Creating recognizer ...
Started
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:108 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-zipformer-small-en-2023-06-26/test_wavs/0.wav
{"text":" AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE␣
↪SQUALID QUARTER OF THE BROTHELS","timestamps":"[0.00, 0.64, 0.76, 0.84, 1.08, 1.36, 1.
↪44, 1.56, 1.72, 1.84, 1.96, 2.04, 2.20, 2.32, 2.36, 2.44, 2.60, 2.76, 3.04, 3.24, 3.40,
↪ 3.52, 3.72, 4.00, 4.20, 4.28, 4.48, 4.64, 4.80, 4.84, 4.96, 5.00, 5.28, 5.40, 5.52, 5.
↪60, 5.76, 5.92, 6.08]","tokens":[" AFTER"," E","AR","LY"," NIGHT","F","A","LL"," THE",
↪" YE","LL","OW"," LA","M","P","S"," WOULD"," LIGHT"," UP"," HE","RE"," AND"," THERE","␣
↪THE"," S","QUA","LI","D"," ","QUA","R","TER"," OF"," THE"," B","RO","TH","EL","S"]}
----
./sherpa-onnx-zipformer-small-en-2023-06-26/test_wavs/1.wav
{"text":" GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A␣
↪LOVELY CHILD WHOSE PLACE WAS ON THAT SAME DISHONOURED BOSOM TO CONNECT HER PARENT␣
↪FOREVER WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
↪","timestamps":"[0.00, 0.32, 0.64, 0.80, 0.96, 1.08, 1.16, 1.20, 1.32, 1.52, 1.68, 1.
↪80, 1.88, 2.04, 2.16, 2.32, 2.40, 2.64, 2.88, 3.16, 3.20, 3.44, 3.52, 3.72, 3.88, 4.16,
↪ 4.44, 4.60, 4.76, 4.96, 5.16, 5.36, 5.60, 6.16, 6.32, 6.52, 6.88, 7.16, 7.32, 7.60, 7.
↪96, 8.16, 8.28, 8.36, 8.48, 8.64, 8.76, 8.84, 9.04, 9.28, 9.44, 9.52, 9.60, 9.68, 9.88,
↪ 9.92, 10.12, 10.52, 10.76, 10.80, 11.08, 11.20, 11.36, 11.56, 11.76, 11.96, 12.08, 12.
↪24, 12.28, 12.48, 12.68, 12.80, 12.92, 13.04, 13.16, 13.48, 13.72, 13.84, 14.04, 14.20,
↪ 14.28, 14.40, 14.56, 14.68, 14.76, 15.00, 15.28, 15.48, 15.68, 15.92, 16.08, 16.12,
↪16.20]","tokens":[" GO","D"," AS"," A"," DI","RE","C","T"," CON","SE","QUE","N","CE","␣
↪OF"," THE"," S","IN"," WHICH"," MAN"," TH","US"," P","UN","ISH","ED"," HAD"," GIVE","N
↪"," HER"," A"," LOVE","LY"," CHILD"," WHO","SE"," PLACE"," WAS"," ON"," THAT"," SAME",
↪" DIS","HO","N","OUR","ED"," BO","S","OM"," TO"," CON","NE","C","T"," HER"," P","AR",
↪"ENT"," FOR","E","VER"," WITH"," THE"," RA","CE"," AND"," DE","S","C","ENT"," OF"," MO
↪","R","T","AL","S"," AND"," TO"," BE"," FI","N","AL","LY"," A"," B","LESS","ED"," SO",
```

(continued on next page)

(continued from previous page)

```
----
./sherpa-onnx-zipformer-small-en-2023-06-26/test_wavs/8k.wav
{"text":" YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION",
↪"timestamps":"[0.00, 0.32, 0.48, 0.64, 0.84, 1.08, 1.20, 1.32, 1.36, 1.44, 1.48, 1.64,␣
↪1.76, 1.88, 2.08, 2.12, 2.24, 2.28, 2.44, 2.48, 2.80, 3.04, 3.24, 3.48, 3.72, 3.88, 3.
↪92, 4.08, 4.16, 4.24, 4.36]","tokens":[" YE","T"," THE","SE"," THOUGHT","S"," A","FF",
↪"E","C","TED"," HE","S","TER"," P","RY","N","NE"," ","LESS"," WITH"," HO","PE"," THAN",
↪" A","PP","RE","HE","N","S","ION"]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.891 s
Real time factor (RTF): 0.891 / 28.165 = 0.032
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-zipformer-small-en-2023-06-26/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-small-en-2023-06-26/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-small-en-2023-06-26/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-small-en-2023-06-26/joiner-epoch-99-avg-1.onnx
```

csukuangfj/sherpa-onnx-zipformer-en-2023-06-26 (English)

This model is converted from

https://huggingface.co/Zengwei/icefall-asr-librispeech-zipformer-2023-05-15

which supports only English as it is trained on the LibriSpeech corpus.

You can find the training code at

https://github.com/k2-fsa/icefall/tree/master/egs/librispeech/ASR/zipformer

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-en-2023-06-26.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-en-2023-06-26.tar.bz2
```

(continues on next page)

(continued from previous page)

```
tar xvf sherpa-onnx-zipformer-en-2023-06-26.tar.bz2
rm sherpa-onnx-zipformer-en-2023-06-26.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-zipformer-en-2023-06-26 fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff   1.2M Jun 26 12:45 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   2.0M Jun 26 12:45 decoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff    66M Jun 26 12:45 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   248M Jun 26 12:46 encoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   253K Jun 26 12:45 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   1.0M Jun 26 12:45 joiner-epoch-99-avg-1.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-en-2023-06-26/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-en-2023-06-26/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-en-2023-06-26/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-en-2023-06-26/joiner-epoch-99-avg-1.onnx \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-zipformer-en-2023-06-26/tokens.
↪txt --encoder=./sherpa-onnx-zipformer-en-2023-06-26/encoder-epoch-99-avg-1.onnx --
↪decoder=./sherpa-onnx-zipformer-en-2023-06-26/decoder-epoch-99-avg-1.onnx --joiner=./
↪sherpa-onnx-zipformer-en-2023-06-26/joiner-epoch-99-avg-1.onnx ./sherpa-onnx-zipformer-
↪en-2023-06-26/test_wavs/0.wav ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/1.wav ./
↪sherpa-onnx-zipformer-en-2023-06-26/test_wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-en-2023-06-26/encoder-epoch-99-avg-1.onnx", decoder_filename="./
↪sherpa-onnx-zipformer-en-2023-06-26/decoder-epoch-99-avg-1.onnx", joiner_filename="./
↪sherpa-onnx-zipformer-en-2023-06-26/joiner-epoch-99-avg-1.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./sherpa-onnx-zipformer-en-2023-
```

(continues on next page)

(continued from previous page)

```
Creating recognizer ...
Started
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:108 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/0.wav
{"text":" AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE␣
↪SQUALID QUARTER OF THE BROTHELS","timestamps":"[0.00, 0.56, 0.64, 0.80, 1.08, 1.36, 1.
↪40, 1.52, 1.68, 1.84, 1.96, 2.04, 2.20, 2.32, 2.40, 2.48, 2.60, 2.80, 3.04, 3.28, 3.40,
↪ 3.56, 3.76, 4.08, 4.24, 4.32, 4.48, 4.64, 4.80, 4.84, 5.00, 5.04, 5.28, 5.40, 5.56, 5.
↪60, 5.76, 5.96, 6.12]","tokens":[" AFTER"," E","AR","LY"," NIGHT","F","A","LL"," THE",
↪" YE","LL","OW"," LA","M","P","S"," WOULD"," LIGHT"," UP"," HE","RE"," AND"," THERE","␣
↪THE"," S","QUA","LI","D"," ","QUA","R","TER"," OF"," THE"," B","RO","TH","EL","S"]}
----
./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/1.wav
{"text":" GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A␣
↪LOVELY CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT␣
↪FOREVER WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
↪","timestamps":"[0.00, 0.24, 0.56, 0.76, 0.92, 1.04, 1.16, 1.20, 1.36, 1.52, 1.64, 1.
↪80, 1.88, 2.00, 2.16, 2.32, 2.40, 2.64, 2.88, 3.12, 3.24, 3.48, 3.56, 3.72, 3.92, 4.12,
↪ 4.40, 4.52, 4.72, 4.96, 5.16, 5.36, 5.64, 6.12, 6.28, 6.52, 6.88, 7.12, 7.32, 7.56, 7.
↪92, 8.16, 8.28, 8.40, 8.48, 8.64, 8.76, 8.88, 9.04, 9.28, 9.44, 9.52, 9.60, 9.72, 9.92,
↪ 9.96, 10.16, 10.48, 10.72, 10.80, 11.04, 11.20, 11.36, 11.56, 11.76, 12.00, 12.12, 12.
↪28, 12.32, 12.52, 12.72, 12.84, 12.92, 13.04, 13.20, 13.44, 13.68, 13.84, 14.00, 14.16,
↪ 14.28, 14.40, 14.56, 14.72, 14.76, 15.00, 15.28, 15.48, 15.68, 15.96, 16.16, 16.20,␣
↪16.28]","tokens":[" GO","D"," AS"," A"," DI","RE","C","T"," CON","SE","QUE","N","CE","␣
↪OF"," THE"," S","IN"," WHICH"," MAN"," TH","US"," P","UN","ISH","ED"," HAD"," GIVE","N
↪"," HER"," A"," LOVE","LY"," CHILD"," WHO","SE"," PLACE"," WAS"," ON"," THAT"," SAME",
↪" DIS","HO","N","OR","ED"," BO","S","OM"," TO"," CON","NE","C","T"," HER"," P","AR",
↪"ENT"," FOR","E","VER"," WITH"," THE"," RA","CE"," AND"," DE","S","C","ENT"," OF"," MO
↪","R","T","AL","S"," AND"," TO"," BE"," FI","N","AL","LY"," A"," B","LESS","ED"," SO",
↪"UL"," IN"," HE","A","VE","N"]}
----
./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/8k.wav
{"text":" YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION",
↪"timestamps":"[0.00, 0.24, 0.40, 0.60, 0.80, 1.04, 1.16, 1.28, 1.36, 1.44, 1.48, 1.68,␣
↪1.76, 1.88, 2.00, 2.12, 2.24, 2.28, 2.48, 2.52, 2.80, 3.08, 3.28, 3.52, 3.68, 3.84, 3.
↪96, 4.12, 4.20, 4.32, 4.44]","tokens":[" YE","T"," THE","SE"," THOUGHT","S"," A","FF",
↪"E","C","TED"," HE","S","TER"," P","RY","N","NE"," ","LESS"," WITH"," HO","PE"," THAN",
↪" A","PP","RE","HE","N","S","ION"]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.301 s
Real time factor (RTF): 1.301 / 28.165 = 0.046
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-en-2023-06-26/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-en-2023-06-26/encoder-epoch-99-avg-1.int8.onnx \
  --decoder=./sherpa-onnx-zipformer-en-2023-06-26/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-en-2023-06-26/joiner-epoch-99-avg-1.int8.onnx \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-zipformer-en-2023-06-26/tokens.
↪txt --encoder=./sherpa-onnx-zipformer-en-2023-06-26/encoder-epoch-99-avg-1.int8.onnx --
↪decoder=./sherpa-onnx-zipformer-en-2023-06-26/decoder-epoch-99-avg-1.onnx --joiner=./
↪sherpa-onnx-zipformer-en-2023-06-26/joiner-epoch-99-avg-1.int8.onnx ./sherpa-onnx-
↪zipformer-en-2023-06-26/test_wavs/0.wav ./sherpa-onnx-zipformer-en-2023-06-26/test_
↪wavs/1.wav ./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-en-2023-06-26/encoder-epoch-99-avg-1.int8.onnx", decoder_
↪filename="./sherpa-onnx-zipformer-en-2023-06-26/decoder-epoch-99-avg-1.onnx", joiner_
↪filename="./sherpa-onnx-zipformer-en-2023-06-26/joiner-epoch-99-avg-1.int8.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./sherpa-onnx-zipformer-en-2023-
↪06-26/tokens.txt", num_threads=2, debug=False, provider="cpu"), lm_
↪config=OfflineLMConfig(model="", scale=0.5), decoding_method="greedy_search", max_
↪active_paths=4, context_score=1.5)
Creating recognizer ...
Started
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:108 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/0.wav
{"text":" AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE␣
↪SQUALID QUARTER OF THE BROTHELS","timestamps":"[0.00, 0.56, 0.64, 0.80, 1.08, 1.36, 1.
↪40, 1.52, 1.68, 1.84, 1.96, 2.04, 2.20, 2.32, 2.40, 2.48, 2.60, 2.76, 3.04, 3.28, 3.40,
↪ 3.56, 3.76, 4.08, 4.24, 4.32, 4.48, 4.64, 4.80, 4.84, 5.00, 5.04, 5.28, 5.40, 5.56, 5.
↪60, 5.76, 5.96, 6.12]","tokens":[" AFTER"," E","AR","LY"," NIGHT","F","A","LL","L"," THE
↪" YE","LL","OW"," LA","M","P","S"," WOULD"," LIGHT"," UP"," HE","RE"," AND"," THERE","␣
↪THE"," S","QUA","LI","D"," ","QUA","R","TER"," OF"," THE"," B","RO","TH","EL","S"]}
```

(continues on next page)

(continued from previous page)

```
----
./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/1.wav
{"text":" GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A␣
↪LOVELY CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT␣
↪FOREVER WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
↪","timestamps":"[0.00, 0.24, 0.56, 0.76, 0.92, 1.04, 1.16, 1.20, 1.36, 1.52, 1.64, 1.
↪80, 1.88, 2.00, 2.16, 2.32, 2.40, 2.64, 2.88, 3.12, 3.24, 3.48, 3.56, 3.72, 3.92, 4.12,
↪ 4.40, 4.52, 4.72, 4.96, 5.12, 5.40, 5.64, 6.12, 6.28, 6.52, 6.88, 7.12, 7.32, 7.60, 7.
↪92, 8.16, 8.28, 8.40, 8.48, 8.64, 8.76, 8.88, 9.04, 9.28, 9.44, 9.52, 9.60, 9.72, 9.92,
↪ 9.96, 10.16, 10.48, 10.72, 10.80, 11.04, 11.20, 11.36, 11.56, 11.76, 12.00, 12.12, 12.
↪28, 12.32, 12.52, 12.72, 12.84, 12.92, 13.04, 13.20, 13.44, 13.68, 13.84, 14.00, 14.16,
↪ 14.28, 14.40, 14.56, 14.72, 14.76, 15.00, 15.28, 15.48, 15.68, 15.96, 16.16, 16.20,␣
↪16.28]","tokens":[" GO","D"," AS"," A"," DI","RE","C","T"," CON","SE","QUE","N","CE","␣
↪OF"," THE"," S","IN"," WHICH"," MAN"," TH","US"," P","UN","ISH","ED"," HAD"," GIVE","N
↪"," HER"," A"," LOVE","LY"," CHILD"," WHO","SE"," PLACE"," WAS"," ON"," THAT"," SAME",
↪" DIS","HO","N","OR","ED"," BO","S","OM"," TO"," CON","NE","C","T"," HER"," P","AR",
↪"ENT"," FOR","E","VER"," WITH"," THE"," RA","CE"," AND"," DE","S","C","ENT"," OF"," MO
↪","R","T","AL","S"," AND"," TO"," BE"," FI","N","AL","LY"," A"," B","LESS","ED"," SO",
↪"UL"," IN"," HE","A","VE","N"]}
----
./sherpa-onnx-zipformer-en-2023-06-26/test_wavs/8k.wav
{"text":" YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION",
↪"timestamps":"[0.00, 0.24, 0.40, 0.60, 0.80, 1.04, 1.16, 1.28, 1.36, 1.44, 1.48, 1.68,␣
↪1.76, 1.88, 2.00, 2.08, 2.24, 2.28, 2.48, 2.52, 2.80, 3.08, 3.28, 3.52, 3.68, 3.84, 3.
↪96, 4.12, 4.20, 4.32, 4.44]","tokens":[" YE","T"," THE","SE"," THOUGHT","S"," A","FF",
↪"E","C","TED"," HE","S","TER"," P","RY","N","NE"," ","LESS"," WITH"," HO","PE"," THAN",
↪" A","PP","RE","HE","N","S","ION"]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.106 s
Real time factor (RTF): 1.106 / 28.165 = 0.039
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-zipformer-en-2023-06-26/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-en-2023-06-26/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-en-2023-06-26/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-en-2023-06-26/joiner-epoch-99-avg-1.onnx
```

icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04 (English)

This model is trained using GigaSpeech + LibriSpeech + Common Voice 13.0 with zipformer

See https://github.com/k2-fsa/icefall/pull/1010 if you are interested in how it is trained.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/icefall-asr-
↪multidataset-pruned_transducer_stateless7-2023-05-04.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/icefall-asr-
↪multidataset-pruned_transducer_stateless7-2023-05-04.tar.bz2

tar xvf icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04.tar.bz2
rm icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff   1.2M May 15 11:11 decoder-epoch-30-avg-4.int8.onnx
-rw-r--r--  1 fangjun  staff   2.0M May 15 11:11 decoder-epoch-30-avg-4.onnx
-rw-r--r--  1 fangjun  staff   121M May 15 11:12 encoder-epoch-30-avg-4.int8.onnx
-rw-r--r--  1 fangjun  staff   279M May 15 11:13 encoder-epoch-30-avg-4.onnx
-rw-r--r--  1 fangjun  staff   253K May 15 11:11 joiner-epoch-30-avg-4.int8.onnx
-rw-r--r--  1 fangjun  staff   1.0M May 15 11:11 joiner-epoch-30-avg-4.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/data/lang_
↪bpe_500/tokens.txt \
  --encoder=./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/exp/
↪encoder-epoch-30-avg-4.onnx \
  --decoder=./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/exp/
↪decoder-epoch-30-avg-4.onnx \
```

(continues on next page)

(continued from previous page)

```
  --joiner=./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/exp/joiner-
↪epoch-30-avg-4.onnx \
  ./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1089-
↪134686-0001.wav \
  ./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1221-
↪135766-0001.wav \
  ./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1221-
↪135766-0002.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./icefall-asr-multidataset-pruned_transducer_
↪stateless7-2023-05-04/data/lang_bpe_500/tokens.txt --encoder=./icefall-asr-
↪multidataset-pruned_transducer_stateless7-2023-05-04/exp/encoder-epoch-30-avg-4.onnx --
↪decoder=./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/exp/decoder-
↪epoch-30-avg-4.onnx --joiner=./icefall-asr-multidataset-pruned_transducer_stateless7-
↪2023-05-04/exp/joiner-epoch-30-avg-4.onnx ./icefall-asr-multidataset-pruned_transducer_
↪stateless7-2023-05-04/test_wavs/1089-134686-0001.wav ./icefall-asr-multidataset-pruned_
↪transducer_stateless7-2023-05-04/test_wavs/1221-135766-0001.wav ./icefall-asr-
↪multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1221-135766-0002.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/exp/encoder-epoch-30-
↪avg-4.onnx", decoder_filename="./icefall-asr-multidataset-pruned_transducer_stateless7-
↪2023-05-04/exp/decoder-epoch-30-avg-4.onnx", joiner_filename="./icefall-asr-
↪multidataset-pruned_transducer_stateless7-2023-05-04/exp/joiner-epoch-30-avg-4.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./icefall-asr-multidataset-
↪pruned_transducer_stateless7-2023-05-04/data/lang_bpe_500/tokens.txt", num_threads=2,␣
↪debug=False, provider="cpu"), lm_config=OfflineLMConfig(model="", scale=0.5), decoding_
↪method="greedy_search", max_active_paths=4)
Creating recognizer ...
Started
Done!

./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1089-134686-
↪0001.wav
{"text":" AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE␣
↪SQUALID QUARTER OF THE BROTHELS","timestamps":"[0.00,0.40,0.56,0.64,0.96,1.24,1.32,1.
↪44,1.56,1.76,1.88,1.96,2.16,2.32,2.36,2.48,2.60,2.80,3.08,3.28,3.36,3.56,3.80,4.04,4.
↪24,4.32,4.48,4.64,4.84,4.88,5.00,5.08,5.32,5.44,5.56,5.64,5.80,5.96,6.20]","tokens":["␣
↪AFTER"," E","AR","LY"," NIGHT","F","A","LL"," THE"," YE","LL","OW"," LA","M","P","S","␣
↪WOULD"," LIGHT"," UP"," HE","RE"," AND"," THERE"," THE"," S","QUA","LI","D"," ","QUA",
↪"R","TER"," OF"," THE"," B","RO","TH","EL","S"]}
----
./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1221-135766-
↪0001.wav
```

(continues on next page)

(continued from previous page)

```
{"text":" GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A␣
↪LOVELY CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT FOR␣
↪EVER WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN",
↪"timestamps":"[0.00,0.16,0.44,0.68,0.84,1.00,1.12,1.16,1.32,1.48,1.64,1.80,1.84,2.00,2.
↪12,2.28,2.40,2.64,2.88,3.16,3.28,3.56,3.60,3.76,3.92,4.12,4.36,4.52,4.72,4.92,5.16,5.
↪44,5.72,6.12,6.24,6.48,6.84,7.08,7.28,7.56,7.88,8.12,8.28,8.36,8.48,8.60,8.76,8.88,9.
↪12,9.28,9.48,9.56,9.64,9.80,10.00,10.04,10.20,10.44,10.68,10.80,11.04,11.20,11.40,11.
↪56,11.80,12.00,12.12,12.28,12.32,12.52,12.72,12.84,12.96,13.04,13.24,13.40,13.64,13.80,
↪14.00,14.16,14.24,14.36,14.56,14.72,14.80,15.08,15.32,15.52,15.76,16.04,16.16,16.24,16.
↪36]","tokens":[" GO","D"," AS"," A"," DI","RE","C","T"," CON","SE","QUE","N","CE"," OF
↪"," THE"," S","IN"," WHICH"," MAN"," TH","US"," P","UN","ISH","ED"," HAD"," GIVE","N",
↪" HER"," A"," LOVE","LY"," CHILD"," WHO","SE"," PLACE"," WAS"," ON"," THAT"," SAME","␣
↪DIS","HO","N","OR","ED"," BO","S","OM"," TO"," CON","NE","C","T"," HER"," P","AR","ENT
↪"," FOR"," E","VER"," WITH"," THE"," RA","CE"," AND"," DE","S","C","ENT"," OF"," MO","R
↪","T","AL","S"," AND"," TO"," BE"," FI","N","AL","LY"," A"," B","LESS","ED"," SO","UL",
↪" IN"," HE","A","VE","N"]}
----
./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1221-135766-
↪0002.wav
{"text":" YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION",
↪"timestamps":"[0.00,0.08,0.32,0.48,0.68,0.92,1.08,1.20,1.28,1.40,1.44,1.64,1.76,1.88,2.
↪04,2.12,2.24,2.32,2.48,2.56,2.88,3.12,3.32,3.52,3.76,3.92,4.00,4.20,4.28,4.40,4.52]",
↪"tokens":[" YE","T"," THE","SE"," THOUGHT","S"," A","FF","E","C","TED"," HE","S","TER",
↪" P","RY","N","NE"," ","LESS"," WITH"," HO","PE"," THAN"," A","PP","RE","HE","N","S",
↪"ION"]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.662 s
Real time factor (RTF): 1.662 / 28.165 = 0.059
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/data/lang_
↪bpe_500/tokens.txt \
  --encoder=./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/exp/
↪encoder-epoch-30-avg-4.int8.onnx \
  --decoder=./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/exp/
↪decoder-epoch-30-avg-4.onnx \
  --joiner=./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/exp/joiner-
↪epoch-30-avg-4.int8.onnx \
  ./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1089-
↪134686-0001.wav \
  ./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1221-
↪135766-0001.wav \
  ./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1221-
↪135766-0002.wav
```

(continues on next page)

(continued from previous page)

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./icefall-asr-multidataset-pruned_transducer_
↪stateless7-2023-05-04/data/lang_bpe_500/tokens.txt --encoder=./icefall-asr-
↪multidataset-pruned_transducer_stateless7-2023-05-04/exp/encoder-epoch-30-avg-4.int8.
↪onnx --decoder=./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/exp/
↪decoder-epoch-30-avg-4.onnx --joiner=./icefall-asr-multidataset-pruned_transducer_
↪stateless7-2023-05-04/exp/joiner-epoch-30-avg-4.int8.onnx ./icefall-asr-multidataset-
↪pruned_transducer_stateless7-2023-05-04/test_wavs/1089-134686-0001.wav ./icefall-asr-
↪multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1221-135766-0001.wav ./
↪icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1221-135766-
↪0002.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/exp/encoder-epoch-30-
↪avg-4.int8.onnx", decoder_filename="./icefall-asr-multidataset-pruned_transducer_
↪stateless7-2023-05-04/exp/decoder-epoch-30-avg-4.onnx", joiner_filename="./icefall-asr-
↪multidataset-pruned_transducer_stateless7-2023-05-04/exp/joiner-epoch-30-avg-4.int8.
↪onnx"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./icefall-asr-multidataset-
↪pruned_transducer_stateless7-2023-05-04/data/lang_bpe_500/tokens.txt", num_threads=2,␣
↪debug=False, provider="cpu"), lm_config=OfflineLMConfig(model="", scale=0.5), decoding_
↪method="greedy_search", max_active_paths=4)
Creating recognizer ...
Started
Done!

./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1089-134686-
↪0001.wav
{"text":" AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE␣
↪SQUALID QUARTER OF THE BROTHELS","timestamps":"[0.00,0.40,0.56,0.64,0.96,1.24,1.32,1.
↪44,1.56,1.76,1.88,1.96,2.16,2.32,2.36,2.48,2.60,2.80,3.08,3.28,3.36,3.56,3.80,4.04,4.
↪24,4.32,4.48,4.64,4.84,4.88,5.00,5.08,5.32,5.44,5.56,5.64,5.80,5.96,6.20]","tokens":["␣
↪AFTER"," E","AR","LY"," NIGHT","F","A","LL"," THE"," YE","LL","OW"," LA","M","P","S","␣
↪WOULD"," LIGHT"," UP"," HE","RE"," AND"," THERE"," THE"," S","QUA","LI","D"," ","QUA",
↪"R","TER"," OF"," THE"," B","RO","TH","EL","S"]}
----
./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1221-135766-
↪0001.wav
{"text":" GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A␣
↪LOVELY CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT FOR␣
↪EVER WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN",
↪"timestamps":"[0.00,0.12,0.44,0.68,0.80,1.00,1.12,1.16,1.32,1.48,1.64,1.80,1.84,2.00,2.
↪12,2.28,2.40,2.64,2.88,3.16,3.28,3.56,3.60,3.76,3.92,4.12,4.36,4.52,4.72,4.92,5.16,5.
↪44,5.72,6.12,6.24,6.48,6.84,7.08,7.28,7.56,7.88,8.12,8.28,8.36,8.48,8.60,8.76,8.88,9.
↪12,9.28,9.48,9.56,9.64,9.80,10.00,10.04,10.16,10.44,10.68,10.80,11.04,11.20,11.40,11.
↪56,11.80,12.00,12.16,12.28,12.32,12.52,12.72,12.84,12.96,13.04,13.24,13.40,13.64,13.80,
↪14.00,14.16,14.24,14.36,14.56,14.72,14.80,15.08,15.32,15.52,15.76,[illegible]
↪36]","tokens":[" GO","D"," AS"," A"," DI","RE","C","T"," CON","SE","QUE","N","CE"," OF
↪"," THE"," S","IN"," WHICH"," MAN"," TH","US"," P","UN","ISH","ED"," HAD"," GIVE","N",
↪" HER"," A"," LOVE","LY"," CHILD"," WHO","SE"," PLACE"," WAS"," ON"," THAT"," SAME","␣
```

(continues on next page)

(continued from previous page)

```
----
./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/test_wavs/1221-135766-
↪0002.wav
{"text":" YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION",
↪"timestamps":"[0.00,0.08,0.32,0.48,0.68,0.92,1.08,1.20,1.28,1.40,1.44,1.64,1.76,1.88,2.
↪04,2.12,2.28,2.32,2.52,2.56,2.88,3.12,3.32,3.52,3.76,3.92,4.00,4.20,4.28,4.40,4.52]",
↪"tokens":[" YE","T"," THE","SE"," THOUGHT","S"," A","FF","E","C","TED"," HE","S","TER",
↪" P","RY","N","NE"," ","LESS"," WITH"," HO","PE"," THAN"," A","PP","RE","HE","N","S",
↪"ION"]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.424 s
Real time factor (RTF): 1.424 / 28.165 = 0.051
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/data/lang_
↪bpe_500/tokens.txt \
  --encoder=./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/exp/
↪encoder-epoch-30-avg-4.onnx \
  --decoder=./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/exp/
↪decoder-epoch-30-avg-4.onnx \
  --joiner=./icefall-asr-multidataset-pruned_transducer_stateless7-2023-05-04/exp/joiner-
↪epoch-30-avg-4.onnx
```

csukuangfj/sherpa-onnx-zipformer-en-2023-04-01 (English)

This model is converted from

https://huggingface.co/WeijiZhuang/icefall-asr-librispeech-pruned-transducer-stateless8-2022-12-02

which supports only English as it is trained on the LibriSpeech and GigaSpeech corpus.

You can find the training code at

https://github.com/k2-fsa/icefall/tree/master/egs/librispeech/ASR/pruned_transducer_stateless8

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-en-2023-04-01.tar.bz2

# For Chinese users, please use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-en-2023-04-01.tar.bz2

tar xvf sherpa-onnx-zipformer-en-2023-04-01.tar.bz2
rm sherpa-onnx-zipformer-en-2023-04-01.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-zipformer-en-2023-04-01$ ls -lh *.onnx
-rw-r--r-- 1 kuangfangjun root  1.3M Apr  1 14:34 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root  2.0M Apr  1 14:34 decoder-epoch-99-avg-1.onnx
-rw-r--r-- 1 kuangfangjun root  180M Apr  1 14:34 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root  338M Apr  1 14:34 encoder-epoch-99-avg-1.onnx
-rw-r--r-- 1 kuangfangjun root  254K Apr  1 14:34 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root 1003K Apr  1 14:34 joiner-epoch-99-avg-1.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-en-2023-04-01/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-en-2023-04-01/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-en-2023-04-01/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-en-2023-04-01/joiner-epoch-99-avg-1.onnx \
  ./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-en-2023-04-01/encoder-epoch-99-avg-1.onnx", decoder_filename="./
↪sherpa-onnx-zipformer-en-2023-04-01/decoder-epoch-99-avg-1.onnx", joiner_filename="./
↪sherpa-onnx-zipformer-en-2023-04-01/joiner-epoch-99-avg-1.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), tokens="./sherpa-onnx-zipformer-en-
↪2023-04-01/tokens.txt", num_threads=2, debug=False), decoding_method="greedy_search")
Creating recognizer ...
2023-04-01 14:40:56.353883875 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 638155, index: 16, mask: {17, 53, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
2023-04-01 14:40:56.353881478 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 638154, index: 15, mask: {16, 52, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
Started
Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/0.wav
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
----
./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/1.wav
 GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY␣
↪CHILD WHOSE PLACE WAS ON THAT SAME DISHONOURED BOSOM TO CONNECT HER PARENT FOR EVER␣
↪WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
----
./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/8k.wav
 YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 2.151 s
Real time factor (RTF): 2.151 / 28.165 = 0.076
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-en-2023-04-01/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-en-2023-04-01/encoder-epoch-99-avg-1.int8.onnx \
```

(continues on next page)

(continued from previous page)

```
  --decoder=./sherpa-onnx-zipformer-en-2023-04-01/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-en-2023-04-01/joiner-epoch-99-avg-1.int8.onnx \
  ./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-en-2023-04-01/encoder-epoch-99-avg-1.int8.onnx", decoder_
↪filename="./sherpa-onnx-zipformer-en-2023-04-01/decoder-epoch-99-avg-1.onnx", joiner_
↪filename="./sherpa-onnx-zipformer-en-2023-04-01/joiner-epoch-99-avg-1.int8.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), tokens="./sherpa-onnx-zipformer-en-
↪2023-04-01/tokens.txt", num_threads=2, debug=False), decoding_method="greedy_search")
Creating recognizer ...
2023-04-01 14:42:00.407939001 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 638195, index: 15, mask: {16, 52, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
2023-04-01 14:42:00.407940827 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 638196, index: 16, mask: {17, 53, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
Started
Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/0.wav
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
----
./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/1.wav
 GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY␣
↪CHILD WHOSE PLACE WAS ON THAT SAME DISHONOURED BOSOM TO CONNECT HER PARENT FOR EVER␣
↪WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
----
./sherpa-onnx-zipformer-en-2023-04-01/test_wavs/8k.wav
 YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.478 s
Real time factor (RTF): 1.478 / 28.165 = 0.052
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-zipformer-en-2023-04-01/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-en-2023-04-01/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-en-2023-04-01/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-en-2023-04-01/joiner-epoch-99-avg-1.onnx
```

csukuangfj/sherpa-onnx-zipformer-en-2023-03-30 (English)

This model is converted from

https://huggingface.co/csukuangfj/icefall-asr-librispeech-pruned-transducer-stateless7-2022-11-11

which supports only English as it is trained on the LibriSpeech corpus.

You can find the training code at

https://github.com/k2-fsa/icefall/tree/master/egs/librispeech/ASR/pruned_transducer_stateless7

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-en-2023-03-30.tar.bz2

# For Chinese users, please use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪zipformer-en-2023-03-30.tar.bz2

tar xvf sherpa-onnx-zipformer-en-2023-03-30.tar.bz2
rm sherpa-onnx-zipformer-en-2023-03-30.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-zipformer-en-2023-03-30$ ls -lh *.onnx
-rw-r--r-- 1 kuangfangjun root  1.3M Mar 31 00:37 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root  2.0M Mar 30 20:10 decoder-epoch-99-avg-1.onnx
-rw-r--r-- 1 kuangfangjun root  180M Mar 31 00:37 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root  338M Mar 30 20:10 encoder-epoch-99-avg-1.onnx
-rw-r--r-- 1 kuangfangjun root  254K Mar 31 00:37 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root 1003K Mar 30 20:10 joiner-epoch-99-avg-1.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-en-2023-03-30/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-en-2023-03-30/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-en-2023-03-30/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-en-2023-03-30/joiner-epoch-99-avg-1.onnx \
  ./sherpa-onnx-zipformer-en-2023-03-30/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-en-2023-03-30/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-en-2023-03-30/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-en-2023-03-30/encoder-epoch-99-avg-1.onnx", decoder_filename="./
↪sherpa-onnx-zipformer-en-2023-03-30/decoder-epoch-99-avg-1.onnx", joiner_filename="./
↪sherpa-onnx-zipformer-en-2023-03-30/joiner-epoch-99-avg-1.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), tokens="./sherpa-onnx-zipformer-en-
↪2023-03-30/tokens.txt", num_threads=2, debug=False), decoding_method="greedy_search")
Creating recognizer ...
2023-04-01 06:47:56.620698024 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 607690, index: 15, mask: {16, 52, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
2023-04-01 06:47:56.620700026 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 607691, index: 16, mask: {17, 53, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
Started
Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-zipformer-en-2023-03-30/test_wavs/0.wav
```

(continues on next page)

(continued from previous page)

```
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
----
./sherpa-onnx-zipformer-en-2023-03-30/test_wavs/1.wav
 GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY␣
↪CHILD WHOSE PLACE WAS ON THAT SAME DISHONOURED BOSOM TO CONNECT HER PARENT FOR EVER␣
↪WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
----
./sherpa-onnx-zipformer-en-2023-03-30/test_wavs/8k.wav
 YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.950 s
Real time factor (RTF): 1.950 / 28.165 = 0.069
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-zipformer-en-2023-03-30/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-en-2023-03-30/encoder-epoch-99-avg-1.int8.onnx \
  --decoder=./sherpa-onnx-zipformer-en-2023-03-30/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-en-2023-03-30/joiner-epoch-99-avg-1.int8.onnx \
  ./sherpa-onnx-zipformer-en-2023-03-30/test_wavs/0.wav \
  ./sherpa-onnx-zipformer-en-2023-03-30/test_wavs/1.wav \
  ./sherpa-onnx-zipformer-en-2023-03-30/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-zipformer-en-2023-03-30/encoder-epoch-99-avg-1.int8.onnx", decoder_
↪filename="./sherpa-onnx-zipformer-en-2023-03-30/decoder-epoch-99-avg-1.onnx", joiner_
↪filename="./sherpa-onnx-zipformer-en-2023-03-30/joiner-epoch-99-avg-1.int8.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), tokens="./sherpa-onnx-zipformer-en-
↪2023-03-30/tokens.txt", num_threads=2, debug=False), decoding_method="greedy_search")
Creating recognizer ...
2023-04-01 06:49:34.370117205 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 607732, index: 16, mask: {17, 53, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
2023-04-01 06:49:34.370115197 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 607731, index: 15, mask: {16, 52, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
```

(continues on next page)

(continued from previous page)

```
Started
Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-zipformer-en-2023-03-30/test_wavs/0.wav
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
----
./sherpa-onnx-zipformer-en-2023-03-30/test_wavs/1.wav
 GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY␣
↪CHILD WHOSE PLACE WAS ON THAT SAME DISHONOURED BOSOM TO CONNECT HER PARENT FOR EVER␣
↪WITH THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
----
./sherpa-onnx-zipformer-en-2023-03-30/test_wavs/8k.wav
 YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.710 s
Real time factor (RTF): 1.710 / 28.165 = 0.061
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-zipformer-en-2023-03-30/tokens.txt \
  --encoder=./sherpa-onnx-zipformer-en-2023-03-30/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-zipformer-en-2023-03-30/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-zipformer-en-2023-03-30/joiner-epoch-99-avg-1.onnx
```

Conformer-transducer-based Models

Hint: Please refer to *Installation* to install sherpa-onnx before you read this section.

csukuangfj/sherpa-onnx-conformer-zh-stateless2-2023-05-23 (Chinese)

This model is converted from

https://huggingface.co/luomingshuang/icefall_asr_wenetspeech_pruned_transducer_stateless2

which supports only Chinese as it is trained on the WenetSpeech corpus.

You can find the training code at

https://github.com/k2-fsa/icefall/tree/master/egs/wenetspeech/ASR/pruned_transducer_stateless2

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪conformer-zh-stateless2-2023-05-23.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪conformer-zh-stateless2-2023-05-23.tar.bz2

tar xvf sherpa-onnx-conformer-zh-stateless2-2023-05-23.tar.bz2
rm sherpa-onnx-conformer-zh-stateless2-2023-05-23.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-conformer-zh-stateless2-2023-05-23 fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff    11M May 23 15:29 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff    12M May 23 15:29 decoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   122M May 23 15:30 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   315M May 23 15:31 encoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   2.7M May 23 15:29 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff    11M May 23 15:29 joiner-epoch-99-avg-1.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/tokens.txt \
  --encoder=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/encoder-epoch-99-avg-1.onnx␣
↪\
  --decoder=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/decoder-epoch-99-avg-1.onnx␣
↪\
  --joiner=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/joiner-epoch-99-avg-1.onnx \
  ./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/0.wav \
  ./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/1.wav \
  ./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/2.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-conformer-zh-stateless2-2023-05-
↪23/tokens.txt --encoder=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/encoder-epoch-
↪99-avg-1.onnx --decoder=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/decoder-epoch-
↪99-avg-1.onnx --joiner=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/joiner-epoch-
↪99-avg-1.onnx ./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/0.wav ./
↪sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/1.wav ./sherpa-onnx-conformer-
↪zh-stateless2-2023-05-23/test_wavs/2.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-conformer-zh-stateless2-2023-05-23/encoder-epoch-99-avg-1.onnx", decoder_
↪filename="./sherpa-onnx-conformer-zh-stateless2-2023-05-23/decoder-epoch-99-avg-1.onnx
↪", joiner_filename="./sherpa-onnx-conformer-zh-stateless2-2023-05-23/joiner-epoch-99-
↪avg-1.onnx"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./sherpa-onnx-conformer-zh-
↪stateless2-2023-05-23/tokens.txt", num_threads=2, debug=False, provider="cpu"), lm_
↪config=OfflineLMConfig(model="", scale=0.5), decoding_method="greedy_search", max_
↪active_paths=4)
Creating recognizer ...
Started
Done!

./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/0.wav
{"text":"","timestamps":"[0.00, 0.12, 0.44, 0.64, 0.84, 1.04, 1.64, 1.72, 1.88, 2.08, 2.
↪28, 2.44, 2.56, 2.76, 3.08, 3.20, 3.32, 3.48, 3.64, 3.76, 3.88, 4.00, 4.16, 4.24, 4.44,
↪ 4.60, 4.84]","tokens":["","","","","","","","","","","","","","","","","","","","","",
↪"","","","","","","]}
----
./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/1.wav
```

(continues on next page)

(continued from previous page)

```
{"text":"","timestamps":"[0.00, 0.12, 0.48, 0.64, 0.88, 1.08, 1.28, 1.48, 1.80, 2.12, 2.
↪40, 2.56, 2.68, 2.88, 3.04, 3.16, 3.36, 3.56, 3.68, 3.84, 4.00, 4.16, 4.32, 4.56, 4.76]
↪","tokens":["","","","","","","","","","","","","","","","","","","","","","","","","
↪"]}
----
./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/2.wav
{"text":"","timestamps":"[0.00, 0.16, 0.60, 0.88, 1.08, 1.36, 1.64, 1.84, 2.24, 2.52, 2.
↪72, 2.92, 3.08, 3.24, 3.40, 3.56, 3.72, 3.88, 4.12]","tokens":["","","","","","","","",
↪"","","","","","","","","","","","]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.596 s
Real time factor (RTF): 0.596 / 15.289 = 0.039
```

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/tokens.txt \
  --encoder=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/encoder-epoch-99-avg-1.int8.
↪onnx \
  --decoder=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/decoder-epoch-99-avg-1.onnx␣
↪\
  --joiner=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/joiner-epoch-99-avg-1.int8.
↪onnx \
  ./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/0.wav \
  ./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/1.wav \
  ./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/2.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: We did not use `int8` for the decoder model above.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-conformer-zh-stateless2-2023-05-
↪23/tokens.txt --encoder=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/encoder-epoch-
↪99-avg-1.int8.onnx --decoder=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/decoder-
↪epoch-99-avg-1.onnx --joiner=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/joiner-
↪epoch-99-avg-1.int8.onnx ./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/0.
↪wav ./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/1.wav ./sherpa-onnx-
↪conformer-zh-stateless2-2023-05-23/test_wavs/2.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-conformer-zh-stateless2-2023-05-23/encoder-epoch-99-avg-1.int8.onnx",␣
↪decoder_filename="./sherpa-onnx-conformer-zh-stateless2-2023-05-23/decoder-epoch-99-
↪avg-1.onnx", joiner_filename="./sherpa-onnx-conformer-zh-stateless2-2023-05-23/joiner-
↪epoch-99-avg-1.int8.onnx"), paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./sherpa-onnx-conformer-zh-
↪stateless2-2023-05-23/tokens.txt", num_threads=2, debug=False, provider="cpu"), lm_
↪config=OfflineLMConfig(model="", scale=0.5), decoding_method="greedy_search", max_
↪active_paths=4)
Creating recognizer ...
Started
Done!

./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/0.wav
{"text":"","timestamps":"[0.00, 0.12, 0.44, 0.64, 0.84, 1.08, 1.64, 1.72, 1.88, 2.08, 2.
↪28, 2.44, 2.56, 2.76, 3.08, 3.20, 3.32, 3.48, 3.64, 3.76, 3.88, 4.00, 4.16, 4.24, 4.48,
↪ 4.60, 4.84]","tokens":["","","","","","","","","","","","","","","","","","","","","
↪","","","","","","]}
----
./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/1.wav
{"text":"","timestamps":"[0.00, 0.08, 0.48, 0.64, 0.88, 1.08, 1.28, 1.48, 1.80, 2.08, 2.
↪40, 2.56, 2.68, 2.88, 3.04, 3.16, 3.36, 3.56, 3.68, 3.84, 4.00, 4.16, 4.32, 4.56, 4.76]
↪","tokens":["","","","","","","","","","","","","","","","","","","","","","","","
↪"]}
----
./sherpa-onnx-conformer-zh-stateless2-2023-05-23/test_wavs/2.wav
{"text":"","timestamps":"[0.00, 0.12, 0.56, 0.84, 1.08, 1.40, 1.64, 1.84, 2.24, 2.52, 2.
↪72, 2.92, 3.08, 3.24, 3.40, 3.56, 3.72, 3.88, 4.12]","tokens":["","","","","","","","
↪","","","","","","","","","","","]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.439 s
Real time factor (RTF): 0.439 / 15.289 = 0.029
```

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/tokens.txt \
  --encoder=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/encoder-epoch-99-avg-1.onnx␣
↪\
  --decoder=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/decoder-epoch-99-avg-1.onnx␣
↪\
  --joiner=./sherpa-onnx-conformer-zh-stateless2-2023-05-23/joiner-epoch-99-avg-1.onnx
```

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

csukuangfj/sherpa-onnx-conformer-zh-2023-05-23 (Chinese)

This model is converted from

https://huggingface.co/luomingshuang/icefall_asr_wenetspeech_pruned_transducer_stateless5_offline

which supports only Chinese as it is trained on the WenetSpeech corpus.

You can find the training code at

https://github.com/k2-fsa/icefall/tree/master/egs/wenetspeech/ASR/pruned_transducer_stateless5

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪conformer-zh-2023-05-23.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪conformer-zh-2023-05-23.tar.bz2

tar xvf sherpa-onnx-conformer-zh-2023-05-23.tar.bz2
rm sherpa-onnx-conformer-zh-2023-05-23.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-conformer-zh-2023-05-23 fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff    11M May 23 13:45 decoder-epoch-99-avg-1.int8.onnx
```

(continues on next page)

(continued from previous page)

```
-rw-r--r--  1 fangjun  staff    12M May 23 13:45 decoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   129M May 23 13:47 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff   345M May 23 13:48 encoder-epoch-99-avg-1.onnx
-rw-r--r--  1 fangjun  staff   2.7M May 23 13:45 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r--  1 fangjun  staff    11M May 23 13:45 joiner-epoch-99-avg-1.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-conformer-zh-2023-05-23/tokens.txt \
  --encoder=./sherpa-onnx-conformer-zh-2023-05-23/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-conformer-zh-2023-05-23/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-conformer-zh-2023-05-23/joiner-epoch-99-avg-1.onnx \
  ./sherpa-onnx-conformer-zh-2023-05-23/test_wavs/0.wav \
  ./sherpa-onnx-conformer-zh-2023-05-23/test_wavs/1.wav \
  ./sherpa-onnx-conformer-zh-2023-05-23/test_wavs/2.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-conformer-zh-2023-05-23/tokens.
↪txt --encoder=./sherpa-onnx-conformer-zh-2023-05-23/encoder-epoch-99-avg-1.onnx --
↪decoder=./sherpa-onnx-conformer-zh-2023-05-23/decoder-epoch-99-avg-1.onnx --joiner=./
↪sherpa-onnx-conformer-zh-2023-05-23/joiner-epoch-99-avg-1.onnx ./sherpa-onnx-conformer-
↪zh-2023-05-23/test_wavs/0.wav ./sherpa-onnx-conformer-zh-2023-05-23/test_wavs/1.wav ./
↪sherpa-onnx-conformer-zh-2023-05-23/test_wavs/2.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-conformer-zh-2023-05-23/encoder-epoch-99-avg-1.onnx", decoder_filename="./
↪sherpa-onnx-conformer-zh-2023-05-23/decoder-epoch-99-avg-1.onnx", joiner_filename="./
↪sherpa-onnx-conformer-zh-2023-05-23/joiner-epoch-99-avg-1.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./sherpa-onnx-conformer-zh-2023-
↪05-23/tokens.txt", num_threads=2, debug=False, provider="cpu"), lm_
↪config=OfflineLMConfig(model="", scale=0.5), decoding_method="greedy_search", max_
↪active_paths=4)
```

(continues on next page)

(continued from previous page)

```
Creating recognizer ...
Started
Done!

./sherpa-onnx-conformer-zh-2023-05-23/test_wavs/0.wav
{"text":"","timestamps":"[0.00, 0.12, 0.52, 0.64, 0.84, 1.04, 1.68, 1.80, 1.92, 2.12, 2.
↪32, 2.48, 2.64, 2.76, 3.08, 3.20, 3.44, 3.52, 3.64, 3.76, 3.88, 4.00, 4.16, 4.32, 4.48,
↪ 4.64, 4.84]","tokens":["","","","","","","","","","","","","","","","","","","","","
↪","","","","","","]}
----
./sherpa-onnx-conformer-zh-2023-05-23/test_wavs/1.wav
{"text":"","timestamps":"[0.04, 0.16, 0.36, 0.48, 0.68, 0.92, 1.08, 1.24, 1.44, 1.84, 2.
↪08, 2.36, 2.52, 2.68, 2.88, 3.04, 3.16, 3.40, 3.56, 3.72, 3.84, 4.04, 4.16, 4.32, 4.56,
↪ 4.76]","tokens":["","","","","","","","","","","","","","","","","","","","","","
↪","","","]}
----
./sherpa-onnx-conformer-zh-2023-05-23/test_wavs/2.wav
{"text":"","timestamps":"[0.00, 0.12, 0.60, 0.84, 1.04, 1.44, 1.68, 1.84, 2.28, 2.52, 2.
↪80, 2.92, 3.08, 3.24, 3.40, 3.60, 3.72, 3.84, 4.12]","tokens":["","","","","","","","
↪","","","","","","","","","","","","]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.706 s
Real time factor (RTF): 0.706 / 15.289 = 0.046
```

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-conformer-zh-2023-05-23/tokens.txt \
  --encoder=./sherpa-onnx-conformer-zh-2023-05-23/encoder-epoch-99-avg-1.int8.onnx \
  --decoder=./sherpa-onnx-conformer-zh-2023-05-23/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-conformer-zh-2023-05-23/joiner-epoch-99-avg-1.int8.onnx \
  ./sherpa-onnx-conformer-zh-2023-05-23/test_wavs/0.wav \
  ./sherpa-onnx-conformer-zh-2023-05-23/test_wavs/1.wav \
  ./sherpa-onnx-conformer-zh-2023-05-23/test_wavs/2.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: We did not use `int8` for the decoder model above.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --decoding-method=greedy_search --tokens=./sherpa-onnx-
↪conformer-zh-2023-05-23/tokens.txt --encoder=./sherpa-onnx-conformer-zh-2023-05-23/
↪encoder-epoch-99-avg-1.int8.onnx --decoder=./sherpa-onnx-conformer-zh-2023-05-23/
↪decoder-epoch-99-avg-1.onnx --joiner=./sherpa-onnx-conformer-zh-2023-05-23/joiner-
↪epoch-99-avg-1.int8.onnx ./sherpa-onnx-conformer-zh-2023-05-23/test_wavs/0.wav ./
↪sherpa-onnx-conformer-zh-2023-05-23/test_wavs/1.wav ./sherpa-onnx-conformer-zh-2023-05-
↪23/test_wavs/2.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-conformer-zh-2023-05-23/encoder-epoch-99-avg-1.int8.onnx", decoder_
↪filename="./sherpa-onnx-conformer-zh-2023-05-23/decoder-epoch-99-avg-1.onnx", joiner_
↪filename="./sherpa-onnx-conformer-zh-2023-05-23/joiner-epoch-99-avg-1.int8.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""), tokens="./sherpa-onnx-conformer-zh-2023-
↪05-23/tokens.txt", num_threads=2, debug=False, provider="cpu"), lm_
↪config=OfflineLMConfig(model="", scale=0.5), decoding_method="greedy_search", max_
↪active_paths=4)
Creating recognizer ...
Started
Done!

./sherpa-onnx-conformer-zh-2023-05-23/test_wavs/0.wav
{"text":"","timestamps":"[0.00, 0.12, 0.52, 0.64, 0.84, 1.04, 1.68, 1.80, 1.92, 2.08, 2.
↪32, 2.48, 2.64, 2.76, 3.08, 3.20, 3.44, 3.52, 3.64, 3.76, 3.88, 4.00, 4.16, 4.32, 4.48,
↪ 4.60, 4.84]","tokens":["","","","","","","","","","","","","","","","","","","","","",
↪"","","","","","","]}
----
./sherpa-onnx-conformer-zh-2023-05-23/test_wavs/1.wav
{"text":"","timestamps":"[0.04, 0.16, 0.36, 0.48, 0.68, 0.92, 1.08, 1.24, 1.44, 1.88, 2.
↪08, 2.36, 2.52, 2.64, 2.88, 3.00, 3.16, 3.40, 3.56, 3.72, 3.84, 4.04, 4.20, 4.32, 4.56,
↪ 4.76]","tokens":["","","","","","","","","","","","","","","","","","","","","","",
↪"","","","]}
----
./sherpa-onnx-conformer-zh-2023-05-23/test_wavs/2.wav
{"text":"","timestamps":"[0.00, 0.12, 0.60, 0.84, 1.04, 1.44, 1.64, 1.84, 2.28, 2.52, 2.
↪80, 2.92, 3.08, 3.28, 3.36, 3.60, 3.72, 3.84, 4.12]","tokens":["","","","","","","","",
↪"","","","","","","","","","","","","]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.502 s
Real time factor (RTF): 0.502 / 15.289 = 0.033
```

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-conformer-zh-2023-05-23/tokens.txt \
  --encoder=./sherpa-onnx-conformer-zh-2023-05-23/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-conformer-zh-2023-05-23/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-conformer-zh-2023-05-23/joiner-epoch-99-avg-1.onnx
```

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

csukuangfj/sherpa-onnx-conformer-en-2023-03-18 (English)

This model is converted from

https://huggingface.co/csukuangfj/icefall-asr-librispeech-pruned-transducer-stateless3-2022-05-13

which supports only English as it is trained on the LibriSpeech corpus.

You can find the training code at

https://github.com/k2-fsa/icefall/tree/master/egs/librispeech/ASR/pruned_transducer_stateless3

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪conformer-en-2023-03-18.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪conformer-en-2023-03-18.tar.bz2

tar xvf sherpa-onnx-conformer-en-2023-03-18.tar.bz2
rm sherpa-onnx-conformer-en-2023-03-18.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-en-2023-03-18$ ls -lh *.onnx
-rw-r--r-- 1 kuangfangjun root  1.3M Apr  1 07:02 decoder-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root  2.0M Apr  1 07:02 decoder-epoch-99-avg-1.onnx
-rw-r--r-- 1 kuangfangjun root  122M Apr  1 07:02 encoder-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root  315M Apr  1 07:02 encoder-epoch-99-avg-1.onnx
-rw-r--r-- 1 kuangfangjun root  254K Apr  1 07:02 joiner-epoch-99-avg-1.int8.onnx
-rw-r--r-- 1 kuangfangjun root 1003K Apr  1 07:02 joiner-epoch-99-avg-1.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-conformer-en-2023-03-18/tokens.txt \
  --encoder=./sherpa-onnx-conformer-en-2023-03-18/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-conformer-en-2023-03-18/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-conformer-en-2023-03-18/joiner-epoch-99-avg-1.onnx \
  ./sherpa-onnx-conformer-en-2023-03-18/test_wavs/0.wav \
  ./sherpa-onnx-conformer-en-2023-03-18/test_wavs/1.wav \
  ./sherpa-onnx-conformer-en-2023-03-18/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-conformer-en-2023-03-18/encoder-epoch-99-avg-1.onnx", decoder_filename="./
↪sherpa-onnx-conformer-en-2023-03-18/decoder-epoch-99-avg-1.onnx", joiner_filename="./
↪sherpa-onnx-conformer-en-2023-03-18/joiner-epoch-99-avg-1.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), tokens="./sherpa-onnx-conformer-en-
↪2023-03-18/tokens.txt", num_threads=2, debug=False), decoding_method="greedy_search")
Creating recognizer ...
2023-04-01 07:11:51.666456713 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 608379, index: 15, mask: {16, 52, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
2023-04-01 07:11:51.666458525 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 608380, index: 16, mask: {17, 53, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
```

(continues on next page)

(continued from previous page)

```
Started
Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-conformer-en-2023-03-18/test_wavs/0.wav
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
----
./sherpa-onnx-conformer-en-2023-03-18/test_wavs/1.wav
 GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY␣
↪CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT FOREVER WITH␣
↪THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
----
./sherpa-onnx-conformer-en-2023-03-18/test_wavs/8k.wav
 YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 2.264 s
Real time factor (RTF): 2.264 / 28.165 = 0.080
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-conformer-en-2023-03-18/tokens.txt \
  --encoder=./sherpa-onnx-conformer-en-2023-03-18/encoder-epoch-99-avg-1.int8.onnx \
  --decoder=./sherpa-onnx-conformer-en-2023-03-18/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-conformer-en-2023-03-18/joiner-epoch-99-avg-1.int8.onnx \
  ./sherpa-onnx-conformer-en-2023-03-18/test_wavs/0.wav \
  ./sherpa-onnx-conformer-en-2023-03-18/test_wavs/1.wav \
  ./sherpa-onnx-conformer-en-2023-03-18/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="./
↪sherpa-onnx-conformer-en-2023-03-18/encoder-epoch-99-avg-1.int8.onnx", decoder_
↪filename="./sherpa-onnx-conformer-en-2023-03-18/decoder-epoch-99-avg-1.onnx", joiner_
↪filename="./sherpa-onnx-conformer-en-2023-03-18/joiner-epoch-99-avg-1.int8.onnx"),␣
↪paraformer=OfflineParaformerModelConfig(model=""), tokens="./sherpa-onnx-conformer-en-
↪2023-03-18/tokens.txt", num_threads=2, debug=False), decoding_method="greedy_search")
```

(continues on next page)

(continued from previous page)

```
Creating recognizer ...
2023-04-01 07:13:26.514109433 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 608419, index: 15, mask: {16, 52, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
2023-04-01 07:13:26.514112711 [E:onnxruntime:, env.cc:251 ThreadMain] pthread_
↪setaffinity_np failed for thread: 608420, index: 16, mask: {17, 53, }, error code: 22␣
↪error msg: Invalid argument. Specify the number of threads explicitly so the affinity␣
↪is not set.
Started
Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-conformer-en-2023-03-18/test_wavs/0.wav
 AFTER EARLY NIGHTFALL THE YELLOW LAMPS WOULD LIGHT UP HERE AND THERE THE SQUALID␣
↪QUARTER OF THE BROTHELS
----
./sherpa-onnx-conformer-en-2023-03-18/test_wavs/1.wav
 GOD AS A DIRECT CONSEQUENCE OF THE SIN WHICH MAN THUS PUNISHED HAD GIVEN HER A LOVELY␣
↪CHILD WHOSE PLACE WAS ON THAT SAME DISHONORED BOSOM TO CONNECT HER PARENT FOREVER WITH␣
↪THE RACE AND DESCENT OF MORTALS AND TO BE FINALLY A BLESSED SOUL IN HEAVEN
----
./sherpa-onnx-conformer-en-2023-03-18/test_wavs/8k.wav
 YET THESE THOUGHTS AFFECTED HESTER PRYNNE LESS WITH HOPE THAN APPREHENSION
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.370 s
Real time factor (RTF): 1.370 / 28.165 = 0.049
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-conformer-en-2023-03-18/tokens.txt \
  --encoder=./sherpa-onnx-conformer-en-2023-03-18/encoder-epoch-99-avg-1.onnx \
  --decoder=./sherpa-onnx-conformer-en-2023-03-18/decoder-epoch-99-avg-1.onnx \
  --joiner=./sherpa-onnx-conformer-en-2023-03-18/joiner-epoch-99-avg-1.onnx
```

8.23.5 Offline paraformer models

This section lists available offline paraformer models.

Paraformer models

Hint: Please refer to *Installation* to install sherpa-onnx before you read this section.

csukuangfj/sherpa-onnx-paraformer-trilingual-zh-cantonese-en (Chinese + English + Cantonese)

Note: This model does not support timestamps. It is a trilingual model, supporting both Chinese and English. ()

This model is converted from

https://www.modelscope.cn/models/dengcunqin/speech_seaco_paraformer_large_asr_nat-zh-cantonese-en-16k-common-vocab11666-pytorch/summary

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-trilingual-zh-cantonese-en.tar.bz2

# For Chinese users
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-trilingual-zh-cantonese-en.tar.bz2

tar xvf sherpa-onnx-paraformer-trilingual-zh-cantonese-en.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-paraformer-trilingual-zh-cantonese-en$ ls -lh *.onnx

-rw-r--r-- 1 1001 127 234M Mar 10 02:12 model.int8.onnx
-rw-r--r-- 1 1001 127 831M Mar 10 02:12 model.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/model.onnx \
  ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/2.wav \
  ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/3-sichuan.wav \
  ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/4-tianjin.wav \
  ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/5-henan.wav \
  ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/6-zh-en.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/project/sherpa-onnx/csrc/parse-options.cc:Read:361 sherpa-onnx-offline --tokens=./
↪sherpa-onnx-paraformer-trilingual-zh-cantonese-en/tokens.txt --paraformer=./sherpa-
↪onnx-paraformer-trilingual-zh-cantonese-en/model.onnx ./sherpa-onnx-paraformer-
↪trilingual-zh-cantonese-en/test_wavs/1.wav ./sherpa-onnx-paraformer-trilingual-zh-
↪cantonese-en/test_wavs/2.wav ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_
↪wavs/3-sichuan.wav ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/4-
↪tianjin.wav ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/5-henan.wav .
↪/sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/6-zh-en.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model="./sherpa-onnx-paraformer-trilingual-zh-
↪cantonese-en/model.onnx"), nemo_ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), tokens="./sherpa-onnx-paraformer-trilingual-
↪zh-cantonese-en/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type="
↪"), lm_config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=0)
```

(continues on next page)

(continued from previous page)

```
Creating recognizer ...
Started
/project/sherpa-onnx/csrc/offline-paraformer-greedy-search-decoder.cc:Decode:65 time␣
↪stamp for batch: 0, 13 vs -1
/project/sherpa-onnx/csrc/offline-paraformer-greedy-search-decoder.cc:Decode:65 time␣
↪stamp for batch: 1, 15 vs -1
/project/sherpa-onnx/csrc/offline-paraformer-greedy-search-decoder.cc:Decode:65 time␣
↪stamp for batch: 2, 40 vs -1
/project/sherpa-onnx/csrc/offline-paraformer-greedy-search-decoder.cc:Decode:65 time␣
↪stamp for batch: 3, 41 vs -1
/project/sherpa-onnx/csrc/offline-paraformer-greedy-search-decoder.cc:Decode:65 time␣
↪stamp for batch: 4, 37 vs -1
/project/sherpa-onnx/csrc/offline-paraformer-greedy-search-decoder.cc:Decode:65 time␣
↪stamp for batch: 5, 16 vs -1
Done!

./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/1.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪""]}
----
./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/2.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", ""]}
----
./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/3-sichuan.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/4-tianjin.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/5-henan.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", ""]}
----
./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/6-zh-en.wav
{"text": " yesterday was  today is tuesday ", "timestamps": [], "tokens":["yesterday",
↪"was", "", "", "", "today", "is", "tu@@", "es@@", "day", "", "", "", "", "", ""]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 6.871 s
Real time factor (RTF): 6.871 / 42.054 = 0.163
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/model.int8.onnx \
  ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/2.wav \
  ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/3-sichuan.wav \
  ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/4-tianjin.wav \
  ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/5-henan.wav \
  ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/6-zh-en.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/project/sherpa-onnx/csrc/parse-options.cc:Read:361 sherpa-onnx-offline --tokens=./
↪sherpa-onnx-paraformer-trilingual-zh-cantonese-en/tokens.txt --paraformer=./sherpa-
↪onnx-paraformer-trilingual-zh-cantonese-en/model.int8.onnx ./sherpa-onnx-paraformer-
↪trilingual-zh-cantonese-en/test_wavs/1.wav ./sherpa-onnx-paraformer-trilingual-zh-
↪cantonese-en/test_wavs/2.wav ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_
↪wavs/3-sichuan.wav ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/4-
↪tianjin.wav ./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/5-henan.wav .
↪/sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/6-zh-en.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model="./sherpa-onnx-paraformer-trilingual-zh-
↪cantonese-en/model.int8.onnx"), nemo_ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), tokens="./sherpa-onnx-paraformer-trilingual-
↪zh-cantonese-en/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type="
↪"), lm_config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=0)
Creating recognizer ...
Started
```

(continues on next page)

(continued from previous page)

```
/project/sherpa-onnx/csrc/offline-paraformer-greedy-search-decoder.cc:Decode:65 time␣
↪stamp for batch: 0, 13 vs -1
/project/sherpa-onnx/csrc/offline-paraformer-greedy-search-decoder.cc:Decode:65 time␣
↪stamp for batch: 1, 15 vs -1
/project/sherpa-onnx/csrc/offline-paraformer-greedy-search-decoder.cc:Decode:65 time␣
↪stamp for batch: 2, 40 vs -1
/project/sherpa-onnx/csrc/offline-paraformer-greedy-search-decoder.cc:Decode:65 time␣
↪stamp for batch: 3, 41 vs -1
/project/sherpa-onnx/csrc/offline-paraformer-greedy-search-decoder.cc:Decode:65 time␣
↪stamp for batch: 4, 37 vs -1
/project/sherpa-onnx/csrc/offline-paraformer-greedy-search-decoder.cc:Decode:65 time␣
↪stamp for batch: 5, 16 vs -1
Done!

./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/1.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪""]}
----
./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/2.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", ""]}
----
./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/3-sichuan.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/4-tianjin.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/5-henan.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", ""]}
----
./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/test_wavs/6-zh-en.wav
{"text": " yesterday was  today is tuesday ", "timestamps": [], "tokens":["yesterday",
↪"was", "", "", "", "today", "is", "tu@@", "es@@", "day", "", "", "", "", "", ""]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 6.290 s
Real time factor (RTF): 6.290 / 42.054 = 0.150
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-trilingual-zh-cantonese-en/model.int8.onnx
```

csukuangfj/sherpa-onnx-paraformer-en-2024-03-09 (English)

Note: This model does not support timestamps. It supports only English.

This model is converted from

https://www.modelscope.cn/models/iic/speech_paraformer_asr-en-16k-vocab4199-pytorch/summary

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-en-2024-03-09.tar.bz2

# For Chinese users
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-en-2024-03-09.tar.bz2

tar xvf sherpa-onnx-paraformer-en-2024-03-09.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-paraformer-en-2024-03-09$ ls -lh *.onnx

-rw-r--r-- 1 1001 127 220M Mar 10 02:12 model.int8.onnx
-rw-r--r-- 1 1001 127 817M Mar 10 02:12 model.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-paraformer-en-2024-03-09/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-en-2024-03-09/model.onnx \
  ./sherpa-onnx-paraformer-en-2024-03-09/test_wavs/0.wav \
  ./sherpa-onnx-paraformer-en-2024-03-09/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-en-2024-03-09/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/project/sherpa-onnx/csrc/parse-options.cc:Read:361 sherpa-onnx-offline --tokens=./
↪sherpa-onnx-paraformer-en-2024-03-09/tokens.txt --paraformer=./sherpa-onnx-paraformer-
↪en-2024-03-09/model.onnx ./sherpa-onnx-paraformer-en-2024-03-09/test_wavs/0.wav ./
↪sherpa-onnx-paraformer-en-2024-03-09/test_wavs/1.wav ./sherpa-onnx-paraformer-en-2024-
↪03-09/test_wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model="./sherpa-onnx-paraformer-en-2024-03-09/
↪model.onnx"), nemo_ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), tokens="./sherpa-onnx-paraformer-en-2024-03-
↪09/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type=""), lm_
↪config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=0)
Creating recognizer ...
Started
/project/sherpa-onnx/csrc/offline-stream.cc:AcceptWaveformImpl:119 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-paraformer-en-2024-03-09/test_wavs/0.wav
{"text": " after early nightfall the yellow lamps would light up here and there the␣
↪squalid quarter of the brothels", "timestamps": [], "tokens":["after", "early", "ni@@",
↪ "ght@@", "fall", "the", "yel@@", "low", "la@@", "mp@@", "s", "would", "light", "up",
↪"here", "and", "there", "the", "squ@@", "al@@", "id", "quarter", "of", "the", "bro@@",
↪"the@@", "ls"]}
----
```

(continues on next page)

(continued from previous page)

```
./sherpa-onnx-paraformer-en-2024-03-09/test_wavs/1.wav
{"text": " god as a direct consequence of the sin which man thus punished had given her␣
↪a lovely child whose place was 'on' that same dishonoured bosom to connect her parent␣
↪for ever with the race and descent of mortals and to be finally a blessed soul in␣
↪heaven", "timestamps": [], "tokens":["god", "as", "a", "direct", "con@@", "sequence",
↪"of", "the", "sin", "which", "man", "thus", "p@@", "uni@@", "shed", "had", "given",
↪"her", "a", "lo@@", "vely", "child", "whose", "place", "was", "'on'", "that", "same",
↪"di@@", "sh@@", "on@@", "ou@@", "red", "bo@@", "so@@", "m", "to", "connect", "her",
↪"paren@@", "t", "for", "ever", "with", "the", "race", "and", "des@@", "cent", "of",
↪"mor@@", "tal@@", "s", "and", "to", "be", "finally", "a", "bl@@", "essed", "soul", "in
↪", "hea@@", "ven"]}
----
./sherpa-onnx-paraformer-en-2024-03-09/test_wavs/8k.wav
{"text": " yet these thoughts affected hester prynne less with hope than apprehension",
↪"timestamps": [], "tokens":["yet", "these", "thoughts", "aff@@", "ected", "he@@", "ster
↪", "pr@@", "y@@", "n@@", "ne", "less", "with", "hope", "than", "ap@@", "pre@@", "hen@@
↪", "sion"]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 7.173 s
Real time factor (RTF): 7.173 / 28.165 = 0.255
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-paraformer-en-2024-03-09/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-en-2024-03-09/model.int8.onnx \
  ./sherpa-onnx-paraformer-en-2024-03-09/test_wavs/0.wav \
  ./sherpa-onnx-paraformer-en-2024-03-09/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-en-2024-03-09/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/project/sherpa-onnx/csrc/parse-options.cc:Read:361 sherpa-onnx-offline --tokens=./
↪sherpa-onnx-paraformer-en-2024-03-09/tokens.txt --paraformer=./sherpa-onnx-paraformer-
↪en-2024-03-09/model.int8.onnx ./sherpa-onnx-paraformer-en-2024-03-09/test_wavs/0.wav ./
↪sherpa-onnx-paraformer-en-2024-03-09/test_wavs/1.wav ./sherpa-onnx-paraformer-en-2024-
↪03-09/test_wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model="./sherpa-onnx-paraformer-en-2024-03-09/
↪model.int8.onnx"), nemo_ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), tokens="./sherpa-onnx-paraformer-en-2024-03-
↪09/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type="", lm
```

(continues on next page)

(continued from previous page)

```
Creating recognizer ...
Started
/project/sherpa-onnx/csrc/offline-stream.cc:AcceptWaveformImpl:119 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-paraformer-en-2024-03-09/test_wavs/0.wav
{"text": " after early nightfall the yellow lamps would light up here and there the␣
↪squalid quarter of the brothels", "timestamps": [], "tokens":["after", "early", "ni@@",
↪ "ght@@", "fall", "the", "yel@@", "low", "la@@", "mp@@", "s", "would", "light", "up",
↪"here", "and", "there", "the", "squ@@", "al@@", "id", "quarter", "of", "the", "bro@@",
↪"the@@", "ls"]}
----
./sherpa-onnx-paraformer-en-2024-03-09/test_wavs/1.wav
{"text": " god as a direct consequence of the sin which man thus punished had given her␣
↪a lovely child whose place was 'on' that same dishonoured bosom to connect her parent␣
↪for ever with the race and descent of mortals and to be finally a blessed soul in␣
↪heaven", "timestamps": [], "tokens":["god", "as", "a", "direct", "con@@", "sequence",
↪"of", "the", "sin", "which", "man", "thus", "p@@", "uni@@", "shed", "had", "given",
↪"her", "a", "lo@@", "vely", "child", "whose", "place", "was", "'on'", "that", "same",
↪"di@@", "sh@@", "on@@", "ou@@", "red", "bo@@", "so@@", "m", "to", "connect", "her",
↪"paren@@", "t", "for", "ever", "with", "the", "race", "and", "des@@", "cent", "of",
↪"mor@@", "tal@@", "s", "and", "to", "be", "finally", "a", "bl@@", "essed", "soul", "in
↪", "hea@@", "ven"]}
----
./sherpa-onnx-paraformer-en-2024-03-09/test_wavs/8k.wav
{"text": " yet these thoughts affected hester prynne less with hope than apprehension",
↪"timestamps": [], "tokens":["yet", "these", "thoughts", "aff@@", "ected", "he@@", "ster
↪", "pr@@", "y@@", "n@@", "ne", "less", "with", "hope", "than", "ap@@", "pre@@", "hen@@
↪", "sion"]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 5.492 s
Real time factor (RTF): 5.492 / 28.165 = 0.195
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-paraformer-en-2024-03-09/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-en-2024-03-09/model.int8.onnx
```

csukuangfj/sherpa-onnx-paraformer-zh-small-2024-03-09 (Chinese + English)

Note: This model does not support timestamps. It is a bilingual model, supporting both Chinese and English. ()

This model is converted from

https://www.modelscope.cn/models/crazyant/speech_paraformer_asr_nat-zh-cn-16k-common-vocab8358-onnx/summary

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-zh-small-2024-03-09.tar.bz2

# For Chinese users
wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-zh-small-2024-03-09.tar.bz2

tar xvf sherpa-onnx-paraformer-zh-small-2024-03-09.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-paraformer-zh-small-2024-03-09$ ls -lh *.onnx

-rw-r--r-- 1 1001 127 79M Mar 10 00:48 model.int8.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-paraformer-zh-small-2024-03-09/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-zh-small-2024-03-09/model.int8.onnx \
  ./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/0.wav \
  ./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/1.wav \
```

(continues on next page)

(continued from previous page)

```
  ./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/8k.wav \
  ./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/2-zh-en.wav \
  ./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/3-sichuan.wav \
  ./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/4-tianjin.wav \
  ./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/5-henan.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/project/sherpa-onnx/csrc/parse-options.cc:Read:361 sherpa-onnx-offline --tokens=./
↪sherpa-onnx-paraformer-zh-small-2024-03-09/tokens.txt --paraformer=./sherpa-onnx-
↪paraformer-zh-small-2024-03-09/model.int8.onnx ./sherpa-onnx-paraformer-zh-small-2024-
↪03-09/test_wavs/0.wav ./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/1.wav ./
↪sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/8k.wav ./sherpa-onnx-paraformer-
↪zh-small-2024-03-09/test_wavs/2-zh-en.wav ./sherpa-onnx-paraformer-zh-small-2024-03-09/
↪test_wavs/3-sichuan.wav ./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/4-
↪tianjin.wav ./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/5-henan.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model="./sherpa-onnx-paraformer-zh-small-2024-
↪03-09/model.int8.onnx"), nemo_ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), tokens="./sherpa-onnx-paraformer-zh-small-
↪2024-03-09/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type=""), lm_
↪config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=0)
Creating recognizer ...
Started
/project/sherpa-onnx/csrc/offline-stream.cc:AcceptWaveformImpl:119 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/0.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
```

(continues on next page)

(continued from previous page)

```
----
./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/1.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/8k.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/2-zh-en.wav
{"text": " yesterday was  today is tuesday ", "timestamps": [], "tokens":["ye@@", "ster@@
↪", "day", "was", "", "", "", "today", "is", "tu@@", "es@@", "day", "", "", "", "", "",
↪""]}
----
./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/3-sichuan.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/4-tianjin.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-small-2024-03-09/test_wavs/5-henan.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", ""]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 3.562 s
Real time factor (RTF): 3.562 / 47.023 = 0.076
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-paraformer-zh-small-2024-03-09/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-zh-small-2024-03-09/model.int8.onnx
```

csukuangfj/sherpa-onnx-paraformer-zh-2024-03-09 (Chinese + English)

Note: This model does not support timestamps. It is a bilingual model, supporting both Chinese and English. ()

This model is converted from

https://www.modelscope.cn/models/crazyant/speech_paraformer_asr_nat-zh-cn-16k-common-vocab8358-onnx/summary

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-zh-2024-03-09.tar.bz2

# For Chinese users
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-zh-2024-03-09.tar.bz2

tar xvf sherpa-onnx-paraformer-zh-2024-03-09.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-paraformer-zh-2024-03-09$ ls -lh *.onnx

-rw-r--r-- 1 1001 127 217M Mar 10 02:22 model.int8.onnx
-rw-r--r-- 1 1001 127 785M Mar 10 02:22 model.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-paraformer-zh-2024-03-09/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-zh-2024-03-09/model.onnx \
  ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/0.wav \
```

(continues on next page)

(continued from previous page)

```
  ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/8k.wav \
  ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/2-zh-en.wav \
  ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/3-sichuan.wav \
  ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/4-tianjin.wav \
  ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/5-henan.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/project/sherpa-onnx/csrc/parse-options.cc:Read:361 sherpa-onnx-offline --tokens=./
↪sherpa-onnx-paraformer-zh-2024-03-09/tokens.txt --paraformer=./sherpa-onnx-paraformer-
↪zh-2024-03-09/model.onnx ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/0.wav ./
↪sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/1.wav ./sherpa-onnx-paraformer-zh-2024-
↪03-09/test_wavs/8k.wav ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/2-zh-en.wav ./
↪sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/3-sichuan.wav ./sherpa-onnx-paraformer-
↪zh-2024-03-09/test_wavs/4-tianjin.wav ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/
↪5-henan.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model="./sherpa-onnx-paraformer-zh-2024-03-09/
↪model.onnx"), nemo_ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), tokens="./sherpa-onnx-paraformer-zh-2024-03-
↪09/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type=""), lm_
↪config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=0)
Creating recognizer ...
Started
/project/sherpa-onnx/csrc/offline-stream.cc:AcceptWaveformImpl:119 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/0.wav
```

(continues on next page)

(continued from previous page)

```
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/1.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/8k.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/2-zh-en.wav
{"text": " yesterday was  today is tuesday ", "timestamps": [], "tokens":["ye@@", "ster@@
↪", "day", "was", "", "", "", "today", "is", "tu@@", "es@@", "day", "", "", "", "", "",
↪""]}
----
./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/3-sichuan.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/4-tianjin.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/5-henan.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", ""]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 6.829 s
Real time factor (RTF): 6.829 / 47.023 = 0.145
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-paraformer-zh-2024-03-09/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-zh-2024-03-09/model.int8.onnx \
  ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/0.wav \
  ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/8k.wav \
  ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/2-zh-en.wav \
  ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/3-sichuan.wav \
  ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/4-tianjin.wav \
```

(continues on next page)

(continued from previous page)

```
./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/5-henan.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/project/sherpa-onnx/csrc/parse-options.cc:Read:361 sherpa-onnx-offline --tokens=./
↪sherpa-onnx-paraformer-zh-2024-03-09/tokens.txt --paraformer=./sherpa-onnx-paraformer-
↪zh-2024-03-09/model.onnx ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/0.wav ./
↪sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/1.wav ./sherpa-onnx-paraformer-zh-2024-
↪03-09/test_wavs/8k.wav ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/2-zh-en.wav ./
↪sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/3-sichuan.wav ./sherpa-onnx-paraformer-
↪zh-2024-03-09/test_wavs/4-tianjin.wav ./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/
↪5-henan.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model="./sherpa-onnx-paraformer-zh-2024-03-09/
↪model.onnx"), nemo_ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), tokens="./sherpa-onnx-paraformer-zh-2024-03-
↪09/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type=""), lm_
↪config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=0)
Creating recognizer ...
Started
/project/sherpa-onnx/csrc/offline-stream.cc:AcceptWaveformImpl:119 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/0.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/1.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
```

(continues on next page)

(continued from previous page)

```
----
./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/8k.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/2-zh-en.wav
{"text": " yesterday was  today is tuesday ", "timestamps": [], "tokens":["ye@@", "ster@@
↪", "day", "was", "", "", "", "today", "is", "tu@@", "es@@", "day", "", "", "", "", "",
↪""]}
----
./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/3-sichuan.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/4-tianjin.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2024-03-09/test_wavs/5-henan.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", ""]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 6.829 s
Real time factor (RTF): 6.829 / 47.023 = 0.145
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-paraformer-zh-2024-03-09/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-zh-2024-03-09/model.int8.onnx
```

csukuangfj/sherpa-onnx-paraformer-zh-2023-03-28 (Chinese + English)

Note: This model does not support timestamps. It is a bilingual model, supporting both Chinese and English. ()

This model is converted from

https://www.modelscope.cn/models/damo/speech_paraformer-large_asr_nat-zh-cn-16k-common-vocab8404-pytorch

The code for converting can be found at

https://huggingface.co/csukuangfj/paraformer-onnxruntime-python-example/tree/main

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-zh-2023-03-28.tar.bz2

# For Chinese users
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-zh-2023-03-28.tar.bz2

tar xvf sherpa-onnx-paraformer-zh-2023-03-28.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-paraformer-zh-2023-03-28$ ls -lh *.onnx
-rw-r--r-- 1 kuangfangjun root 214M Apr  1 07:28 model.int8.onnx
-rw-r--r-- 1 kuangfangjun root 824M Apr  1 07:28 model.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

fp32

The following code shows how to use `fp32` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-paraformer-zh-2023-03-28/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-zh-2023-03-28/model.onnx \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/3-sichuan.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/4-tianjin.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/5-henan.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/6-zh-en.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/project/sherpa-onnx/csrc/parse-options.cc:Read:361 sherpa-onnx-offline --tokens=./
↪sherpa-onnx-paraformer-zh-2023-03-28/tokens.txt --paraformer=./sherpa-onnx-paraformer-
↪zh-2023-03-28/model.onnx ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav ./
↪sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav ./sherpa-onnx-paraformer-zh-2023-
↪03-28/test_wavs/2.wav ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/3-sichuan.wav ./
↪sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/4-tianjin.wav ./sherpa-onnx-paraformer-
↪zh-2023-03-28/test_wavs/5-henan.wav ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/6-
↪zh-en.wav ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model="./sherpa-onnx-paraformer-zh-2023-03-28/
↪model.onnx"), nemo_ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), tokens="./sherpa-onnx-paraformer-zh-2023-03-
↪28/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type=""), lm_
↪config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=0)
Creating recognizer ...
Started
/project/sherpa-onnx/csrc/offline-stream.cc:AcceptWaveformImpl:119 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/3-sichuan.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", ""]}
```

(continues on next page)

(continued from previous page)

```
----
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/4-tianjin.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/5-henan.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/6-zh-en.wav
{"text": " yesterday was  today is tuesday ", "timestamps": [], "tokens":["ye@@", "ster@@
↪", "day", "was", "", "", "", "today", "is", "tu@@", "es@@", "day", "", "", "", "", "",
↪""]}
----
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪""]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 8.547 s
Real time factor (RTF): 8.547 / 51.236 = 0.167
```

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-paraformer-zh-2023-03-28/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-zh-2023-03-28/model.int8.onnx \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/3-sichuan.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/4-tianjin.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/5-henan.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/6-zh-en.wav \
  ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/project/sherpa-onnx/csrc/parse-options.cc:Read:361 sherpa-onnx-offline --tokens=./
↪sherpa-onnx-paraformer-zh-2023-03-28/tokens.txt --paraformer=./sherpa-onnx-paraformer-
↪zh-2023-03-28/model.int8.onnx ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav ./
↪sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav ./sherpa-onnx-paraformer-zh-2023-
↪03-28/test_wavs/2.wav ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/3-sichuan.wav ./
↪sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/4-tianjin.wav ./sherpa-onnx-paraformer-
↪zh-2023-03-28/test_wavs/5-henan.wav ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/6-
↪zh-en.wav ./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model="./sherpa-onnx-paraformer-zh-2023-03-28/
↪model.int8.onnx"), nemo_ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), tokens="./sherpa-onnx-paraformer-zh-2023-03-
↪28/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type=""), lm_
↪config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=0)
Creating recognizer ...
Started
/project/sherpa-onnx/csrc/offline-stream.cc:AcceptWaveformImpl:119 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/0.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/1.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/2.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/3-sichuan.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", ""]}
----
```

(continues on next page)

(continued from previous page)

```
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/4-tianjin.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/5-henan.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/6-zh-en.wav
{"text": " yesterday was  today is tuesday ", "timestamps": [], "tokens":["ye@@", "ster@@
↪", "day", "was", "", "", "", "today", "is", "tu@@", "es@@", "day", "", "", "", "", "",
↪""]}
----
./sherpa-onnx-paraformer-zh-2023-03-28/test_wavs/8k.wav
{"text": "", "timestamps": [], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪""]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 6.439 s
Real time factor (RTF): 6.439 / 51.236 = 0.126
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-paraformer-zh-2023-03-28/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-zh-2023-03-28/model.int8.onnx
```

csukuangfj/sherpa-onnx-paraformer-zh-2023-09-14 (Chinese + English))

Note: This model supports timestamps. It is a bilingual model, supporting both Chinese and English. ()

This model is converted from

https://www.modelscope.cn/models/iic/speech_paraformer-large-vad-punc_asr_nat-zh-cn-16k-common-vocab8404-onnx/summary

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-zh-2023-09-14.tar.bz2

# For Chinese users
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪paraformer-zh-2023-09-14.tar.bz2

tar xvf sherpa-onnx-paraformer-zh-2023-09-14.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-paraformer-zh-2023-09-14$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff   232M Sep 14 13:46 model.int8.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

int8

The following code shows how to use `int8` models to decode wave files:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-paraformer-zh-2023-09-14/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-zh-2023-09-14/model.int8.onnx \
  --model-type=paraformer \
  ./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/0.wav \
  ./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/1.wav \
  ./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/2.wav \
  ./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/3-sichuan.wav \
  ./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/4-tianjin.wav \
  ./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/5-henan.wav \
  ./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/6-zh-en.wav \
  ./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/project/sherpa-onnx/csrc/parse-options.cc:Read:361 sherpa-onnx-offline --tokens=./
↪sherpa-onnx-paraformer-zh-2023-09-14/tokens.txt --paraformer=./sherpa-onnx-paraformer-
↪zh-2023-09-14/model.int8.onnx --model-type=paraformer ./sherpa-onnx-paraformer-zh-2023-
↪09-14/test_wavs/0.wav ./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/1.wav ./sherpa-
↪onnx-paraformer-zh-2023-09-14/test_wavs/2.wav ./sherpa-onnx-paraformer-zh-2023-09-14/
↪test_wavs/3-sichuan.wav ./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/4-tianjin.wav␣
↪./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/5-henan.wav ./sherpa-onnx-paraformer-
↪zh-2023-09-14/test_wavs/6-zh-en.wav ./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/
↪8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model="./sherpa-onnx-paraformer-zh-2023-09-14/
↪model.int8.onnx"), nemo_ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), tokens="./sherpa-onnx-paraformer-zh-2023-09-
↪14/tokens.txt", num_threads=2, debug=False, provider="cpu", model_type="paraformer"),␣
↪lm_config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=0)
Creating recognizer ...
Started
/project/sherpa-onnx/csrc/offline-stream.cc:AcceptWaveformImpl:119 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/0.wav
{"text": "", "timestamps": [0.36, 0.48, 0.62, 0.72, 0.86, 1.02, 1.32, 1.74, 1.90, 2.12,␣
↪2.20, 2.38, 2.50, 2.62, 2.74, 3.18, 3.32, 3.52, 3.62, 3.74, 3.82, 3.90, 3.98, 4.08, 4.
↪20, 4.34, 4.56, 4.74, 5.10], "tokens":["", "", "", "", "", "", "", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/1.wav
{"text": "", "timestamps": [0.16, 0.30, 0.42, 0.56, 0.72, 0.96, 1.08, 1.20, 1.30, 2.08,␣
↪2.26, 2.44, 2.58, 2.72, 2.98, 3.14, 3.26, 3.46, 3.62, 3.80, 3.88, 4.02, 4.12, 4.20, 4.
↪36, 4.56], "tokens":["", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "
↪", "", "", "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/2.wav
```

(continues on next page)

(continued from previous page)

```
{"text": "", "timestamps": [0.34, 0.54, 0.66, 0.80, 1.08, 1.52, 1.72, 1.90, 2.40, 2.68,␣
↪2.86, 2.96, 3.16, 3.26, 3.46, 3.54, 3.66, 3.80, 3.90], "tokens":["", "", "", "", "", "
↪", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/3-sichuan.wav
{"text": "", "timestamps": [0.16, 0.30, 0.56, 0.72, 0.92, 1.18, 1.32, 1.88, 2.24, 2.40,␣
↪3.16, 3.28, 3.40, 3.54, 3.76, 3.88, 4.06, 4.24, 4.36, 4.56, 4.66, 4.88, 5.14, 5.30, 5.
↪44, 5.60, 5.72, 5.84, 5.96, 6.14, 6.24, 6.38, 6.56, 6.78, 6.98, 7.08, 7.22, 7.38, 7.50,
↪ 7.62], "tokens":["", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "
↪", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪""]}
----
./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/4-tianjin.wav
{"text": "", "timestamps": [0.08, 0.24, 0.36, 0.56, 0.66, 0.78, 1.04, 1.14, 1.26, 1.38,␣
↪1.50, 1.58, 1.70, 1.84, 2.28, 2.38, 2.64, 2.74, 3.08, 3.28, 3.66, 3.80, 3.94, 4.14, 4.
↪34, 4.64, 4.84, 4.94, 5.12, 5.24, 5.84, 6.10, 6.24, 6.44, 6.54, 6.66, 6.86, 7.02, 7.14,
↪ 7.24, 7.44], "tokens":["", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "
↪", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/5-henan.wav
{"text": "", "timestamps": [0.08, 0.20, 0.30, 0.42, 0.94, 1.14, 1.26, 1.46, 1.66, 2.28,␣
↪2.50, 2.62, 2.70, 2.82, 2.98, 3.14, 3.28, 3.52, 3.70, 3.86, 4.94, 5.06, 5.18, 5.30, 5.
↪42, 5.66, 5.76, 5.94, 6.08, 6.24, 6.38, 6.60, 6.78, 6.96, 7.10, 7.30, 7.50, 7.62],
↪"tokens":["", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "
↪", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/6-zh-en.wav
{"text": " yesterday was  today is tuesday ", "timestamps": [0.36, 0.60, 0.84, 1.22, 2.
↪24, 2.44, 2.74, 3.52, 4.06, 4.68, 5.00, 5.12, 5.76, 5.96, 6.24, 6.82, 7.02, 7.26],
↪"tokens":["ye@@", "ster@@", "day", "was", "", "", "", "today", "is", "tu@@", "es@@",
↪"day", "", "", "", "", "", ""]}
----
./sherpa-onnx-paraformer-zh-2023-09-14/test_wavs/8k.wav
{"text": "", "timestamps": [0.48, 0.78, 1.04, 1.18, 1.52, 1.78, 2.06, 2.18, 2.50, 2.66,␣
↪2.88, 3.10, 3.30], "tokens":["", "", "", "", "", "", "", "", "", "", "", "", ""]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 9.206 s
Real time factor (RTF): 9.206 / 51.236 = 0.180
```

Speech recognition from a microphone

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-microphone-offline \
  --tokens=./sherpa-onnx-paraformer-zh-2023-09-14/tokens.txt \
  --paraformer=./sherpa-onnx-paraformer-zh-2023-09-14/model.int8.onnx \
  --model-type=paraformer
```

8.23.6 Offline CTC models

This section lists available offline CTC models.

NeMo

This page lists all offline CTC models from NeMo.

Hint: Please refer to https://catalog.ngc.nvidia.com/orgs/nvidia/collections/nemo_asr for a list of pre-trained NeMo models.

How to export models from NeMo to sherpa-onnx

This section describes how to export CTC models from NeMo to sherpa-onnx.

Hint: Please refer to https://catalog.ngc.nvidia.com/orgs/nvidia/collections/nemo_asr for a list of pre-trained NeMo models.

You can use method described in this section to convert more models to sherpa-onnx.

Let us take the following model as an example:

https://ngc.nvidia.com/models/nvidia:nemo:stt_en_conformer_ctc_small.

Hint: You can find the exported files in this example by visiting

https://huggingface.co/csukuangfj/sherpa-onnx-nemo-ctc-en-conformer-small

The steps to export it to sherpa-onnx are given below.

Step 1: Export model.onnx

The first step is to obtain `model.onnx`.

```
import nemo.collections.asr as nemo_asr
m = nemo_asr.models.EncDecCTCModelBPE.from_pretrained('stt_en_conformer_ctc_small')
m.export('model.onnx')
```

Step 2: Add metadata

To be usable in sherpa-onnx, we have to use add-model-metadata.py to add metadata to `model.onnx`.

```
wget https://huggingface.co/csukuangfj/sherpa-onnx-nemo-ctc-en-conformer-small/resolve/
↪main/add-model-metadata.py

# The following command changes model.onnx in-place
python3 add-model-metadata.py
```

Step 3: Obtain model.int8.onnx

We can use quantize-model.py to obtain a quantized version of `model.onnx`:

```
wget https://huggingface.co/csukuangfj/sherpa-onnx-nemo-ctc-en-conformer-small/resolve/
↪main/quantize-model.py

# The following command will generate model.int8.onnx
python3 ./quantize-model.py
```

Step 4: Obtain tokens.txt

Use the following command to obtain `tokens.txt`:

```
import nemo.collections.asr as nemo_asr
m = nemo_asr.models.EncDecCTCModelBPE.from_pretrained('stt_en_conformer_ctc_small')

with open('tokens.txt', 'w') as f:
  for i, s in enumerate(m.decoder.vocabulary):
    f.write(f"{s} {i}\n")
  f.write(f"<blk> {i+1}\n")
```

English

Hint: Please refer to *Installation* to install sherpa-onnx before you read this section.

Note: We use ./build/bin/sherpa-offline as an example in this section. You can use other scripts such as

- ./build/bin/sherpa-onnx-microphone-offline
- ./build/bin/sherpa-onnx-offline-websocket-server
- python-api-examples/offline-decode-files.py

This page lists offline CTC models from NeMo for English.

stt_en_citrinet_512

This model is converted from

https://catalog.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_en_citrinet_512

Citrinet-512 model which has been trained on the ASR Set dataset with over 7000 hours of english speech.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-nemo-
↪ctc-en-citrinet-512.tar.bz2

# For Chinese users, please use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪nemo-ctc-en-citrinet-512.tar.bz2

tar xvf sherpa-onnx-nemo-ctc-en-citrinet-512.tar.bz2
rm sherpa-onnx-nemo-ctc-en-citrinet-512.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-nemo-ctc-en-citrinet-512 fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff    36M Apr  7 16:10 model.int8.onnx
-rw-r--r--  1 fangjun  staff   142M Apr  7 14:24 model.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

The following code shows how to use `fp32` models to decode wave files. Please replace `model.onnx` with `model.int8.onnx` to use `int8` quantized model.

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-nemo-ctc-en-citrinet-512/tokens.txt \
  --nemo-ctc-model=./sherpa-onnx-nemo-ctc-en-citrinet-512/model.onnx \
  --num-threads=2 \
  --decoding-method=greedy_search \
  --debug=false \
  ./sherpa-onnx-nemo-ctc-en-citrinet-512/test_wavs/0.wav \
  ./sherpa-onnx-nemo-ctc-en-citrinet-512/test_wavs/1.wav \
  ./sherpa-onnx-nemo-ctc-en-citrinet-512/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-nemo-ctc-en-citrinet-512/tokens.
↪txt --nemo-ctc-model=./sherpa-onnx-nemo-ctc-en-citrinet-512/model.onnx --num-threads=2␣
↪--decoding-method=greedy_search --debug=false ./sherpa-onnx-nemo-ctc-en-citrinet-512/
↪test_wavs/0.wav ./sherpa-onnx-nemo-ctc-en-citrinet-512/test_wavs/1.wav ./sherpa-onnx-
↪nemo-ctc-en-citrinet-512/test_wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model="./sherpa-onnx-nemo-ctc-en-citrinet-512/
↪model.onnx"), tokens="./sherpa-onnx-nemo-ctc-en-citrinet-512/tokens.txt", num_
↪threads=2, debug=False), decoding_method="greedy_search")
Creating recognizer ...
Started
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:105 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-nemo-ctc-en-citrinet-512/test_wavs/0.wav
 after early nightfall the yellow lamps would light up here and there the squalid␣
↪quarter of the brothels
```

(continues on next page)

(continued from previous page)

```
----
./sherpa-onnx-nemo-ctc-en-citrinet-512/test_wavs/1.wav
 god as a direct consequence of the sin which man thus punished had given her a lovely␣
↪child whose place was on that same dishonoured bosom to connect her parent for ever␣
↪with the race and descent of mortals and to be finally a blessed soul in heaven
----
./sherpa-onnx-nemo-ctc-en-citrinet-512/test_wavs/8k.wav
 yet these thoughts affected hester prynne less with hope than apprehension
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 4.963 s
Real time factor (RTF): 4.963 / 28.165 = 0.176
```

stt_en_conformer_ctc_small

This model is converted from

> https://registry.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_en_conformer_ctc_small

It contains small size versions of Conformer-CTC (13M parameters) trained on NeMo ASRSet with around 16000 hours of english speech. The model transcribes speech in lower case english alphabet along with spaces and apostrophes.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-nemo-
↪ctc-en-conformer-small.tar.bz2

# For Chinese users, please use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪nemo-ctc-en-conformer-small.tar.bz2

tar xvf sherpa-onnx-nemo-ctc-en-conformer-small.tar.bz2
rm sherpa-onnx-nemo-ctc-en-conformer-small.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-nemo-ctc-en-conformer-small fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff    44M Apr  7 20:24 model.int8.onnx
-rw-r--r--  1 fangjun  staff    81M Apr  7 18:56 model.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

The following code shows how to use `fp32` models to decode wave files. Please replace `model.onnx` with `model.int8.onnx` to use `int8` quantized model.

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-nemo-ctc-en-conformer-small/tokens.txt \
  --nemo-ctc-model=./sherpa-onnx-nemo-ctc-en-conformer-small/model.onnx \
  --num-threads=2 \
  --decoding-method=greedy_search \
  --debug=false \
  ./sherpa-onnx-nemo-ctc-en-conformer-small/test_wavs/0.wav \
  ./sherpa-onnx-nemo-ctc-en-conformer-small/test_wavs/1.wav \
  ./sherpa-onnx-nemo-ctc-en-conformer-small/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-nemo-ctc-en-conformer-small/
↪tokens.txt --nemo-ctc-model=./sherpa-onnx-nemo-ctc-en-conformer-small/model.onnx --num-
↪threads=2 --decoding-method=greedy_search --debug=false ./sherpa-onnx-nemo-ctc-en-
↪conformer-small/test_wavs/0.wav ./sherpa-onnx-nemo-ctc-en-conformer-small/test_wavs/1.
↪wav ./sherpa-onnx-nemo-ctc-en-conformer-small/test_wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model="./sherpa-onnx-nemo-ctc-en-conformer-small/
↪model.onnx"), tokens="./sherpa-onnx-nemo-ctc-en-conformer-small/tokens.txt", num_
↪threads=2, debug=False), decoding_method="greedy_search")
Creating recognizer ...
Started
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:105 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-nemo-ctc-en-conformer-small/test_wavs/0.wav
 after early nightfall the yellow lamps would light up here and there the squalid␣
↪quarter of the brothels
```

(continues on next page)

(continued from previous page)

```
----
./sherpa-onnx-nemo-ctc-en-conformer-small/test_wavs/1.wav
 god as a direct consequence of the sin which man thus punished had given her a lovely␣
↪child whose place was on that same dishonoured bosom to connect her parent for ever␣
↪with the race and descent of mortals and to be finally a blessed soul in heaven
----
./sherpa-onnx-nemo-ctc-en-conformer-small/test_wavs/8k.wav
 yet these thoughts affected hester prin less with hope than apprehension
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.665 s
Real time factor (RTF): 0.665 / 28.165 = 0.024
```

stt_en_conformer_ctc_medium

This model is converted from

> https://registry.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_en_conformer_ctc_medium

It contains medium size versions of Conformer-CTC (around 30M parameters) trained on NeMo ASRSet with around 16000 hours of english speech. The model transcribes speech in lower case english alphabet along with spaces and apostrophes.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-nemo-
↪ctc-en-conformer-medium.tar.bz2

# For Chinese users, please use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪nemo-ctc-en-conformer-medium.tar.bz2

tar xvf sherpa-onnx-nemo-ctc-en-conformer-medium.tar.bz2
rm sherpa-onnx-nemo-ctc-en-conformer-medium.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-nemo-ctc-en-conformer-medium fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff    64M Apr  7 20:44 model.int8.onnx
-rw-r--r--  1 fangjun  staff   152M Apr  7 20:43 model.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

The following code shows how to use `fp32` models to decode wave files. Please replace `model.onnx` with `model.int8.onnx` to use `int8` quantized model.

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-nemo-ctc-en-conformer-medium/tokens.txt \
  --nemo-ctc-model=./sherpa-onnx-nemo-ctc-en-conformer-medium/model.onnx \
  --num-threads=2 \
  --decoding-method=greedy_search \
  --debug=false \
  ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/0.wav \
  ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/1.wav \
  ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-nemo-ctc-en-conformer-medium/
↪tokens.txt --nemo-ctc-model=./sherpa-onnx-nemo-ctc-en-conformer-medium/model.onnx --
↪num-threads=2 --decoding-method=greedy_search --debug=false ./sherpa-onnx-nemo-ctc-en-
↪conformer-medium/test_wavs/0.wav ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/
↪1.wav ./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model="./sherpa-onnx-nemo-ctc-en-conformer-medium/
↪model.onnx"), tokens="./sherpa-onnx-nemo-ctc-en-conformer-medium/tokens.txt", num_
↪threads=2, debug=False), decoding_method="greedy_search")
Creating recognizer ...
Started
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:105 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/0.wav
 after early nightfall the yellow lamps would light up here and there the squalid␣
↪quarter of the brothels
```

(continues on next page)

(continued from previous page)

```
----
./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/1.wav
 god as a direct consequence of the sin which man thus punished had given her a lovely␣
↪child whose place was on that same dishonored bosom to connect her parent for ever␣
↪with the race and descent of mortals and to be finally a blessed soul in heaven
----
./sherpa-onnx-nemo-ctc-en-conformer-medium/test_wavs/8k.wav
 yet these thoughts affected hester pryne less with hope than apprehension
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 1.184 s
Real time factor (RTF): 1.184 / 28.165 = 0.042
```

stt_en_conformer_ctc_large

This model is converted from

> https://registry.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_en_conformer_ctc_large

It contains large size versions of Conformer-CTC (around 120M parameters) trained on NeMo ASRSet with around 24500 hours of english speech. The model transcribes speech in lower case english alphabet along with spaces and apostrophes

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-nemo-
↪ctc-en-conformer-large.tar.bz2

# For Chinese users, please use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪nemo-ctc-en-conformer-large.tar.bz2

tar xvf sherpa-onnx-nemo-ctc-en-conformer-large.tar.bz2
rm sherpa-onnx-nemo-ctc-en-conformer-large.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-nemo-ctc-en-conformer-large fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff   162M Apr  7 22:01 model.int8.onnx
-rw-r--r--  1 fangjun  staff   508M Apr  7 22:01 model.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

The following code shows how to use `fp32` models to decode wave files. Please replace `model.onnx` with `model.int8.onnx` to use `int8` quantized model.

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-nemo-ctc-en-conformer-large/tokens.txt \
  --nemo-ctc-model=./sherpa-onnx-nemo-ctc-en-conformer-large/model.onnx \
  --num-threads=2 \
  --decoding-method=greedy_search \
  --debug=false \
  ./sherpa-onnx-nemo-ctc-en-conformer-large/test_wavs/0.wav \
  ./sherpa-onnx-nemo-ctc-en-conformer-large/test_wavs/1.wav \
  ./sherpa-onnx-nemo-ctc-en-conformer-large/test_wavs/8k.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-nemo-ctc-en-conformer-large/
↪tokens.txt --nemo-ctc-model=./sherpa-onnx-nemo-ctc-en-conformer-large/model.onnx --num-
↪threads=2 --decoding-method=greedy_search --debug=false ./sherpa-onnx-nemo-ctc-en-
↪conformer-large/test_wavs/0.wav ./sherpa-onnx-nemo-ctc-en-conformer-large/test_wavs/1.
↪wav ./sherpa-onnx-nemo-ctc-en-conformer-large/test_wavs/8k.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model="./sherpa-onnx-nemo-ctc-en-conformer-large/
↪model.onnx"), tokens="./sherpa-onnx-nemo-ctc-en-conformer-large/tokens.txt", num_
↪threads=2, debug=False), decoding_method="greedy_search")
Creating recognizer ...
Started
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/offline-stream.
↪cc:AcceptWaveformImpl:105 Creating a resampler:
   in_sample_rate: 8000
   output_sample_rate: 16000

Done!

./sherpa-onnx-nemo-ctc-en-conformer-large/test_wavs/0.wav
 after early nightfall the yellow lamps would light up here and there the squalid␣
↪quarter of the brothels
```

(continues on next page)

(continued from previous page)

```
----
./sherpa-onnx-nemo-ctc-en-conformer-large/test_wavs/1.wav
 god as a direct consequence of the sin which man thus punished had given her a lovely␣
↪child whose place was on that same dishonored bosom to connect her parent for ever␣
↪with the race and descent of mortals and to be finally a blesed soul in heaven
----
./sherpa-onnx-nemo-ctc-en-conformer-large/test_wavs/8k.wav
 yet these thoughts afected hester pryne les with hope than aprehension
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 3.553 s
Real time factor (RTF): 3.553 / 28.165 = 0.126
```

yesno

This section describes how to use the tdnn model of the yesno dataset from icefall in sherpa-onnx.

Note: It is a **non-streaming** model and it can only recognize two words in Hebrew: `yes` and `no`.

To download the model, please use:

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-tdnn-
↪yesno.tar.bz2

# For Chinese users, please use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪tdnn-yesno.tar.bz2

tar xvf sherpa-onnx-tdnn-yesno.tar.bz2
rm sherpa-onnx-tdnn-yesno.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
sherpa-onnx-tdnn-yesno fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff    55K Aug 12 17:02 model-epoch-14-avg-2.int8.onnx
-rw-r--r--  1 fangjun  staff    54K Aug 12 17:02 model-epoch-14-avg-2.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

The following code shows how to use `fp32` models to decode wave files. Please replace `model-epoch-14-avg-2.int8.onnx` with `model-epoch-14-avg-2.int8.onnx` to use the `int8` quantized model.

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --sample-rate=8000 \
  --feat-dim=23 \
  --tokens=./sherpa-onnx-tdnn-yesno/tokens.txt \
  --tdnn-model=./sherpa-onnx-tdnn-yesno/model-epoch-14-avg-2.onnx \
  ./sherpa-onnx-tdnn-yesno/test_wavs/0_0_0_1_0_0_0_1.wav \
  ./sherpa-onnx-tdnn-yesno/test_wavs/0_0_1_0_0_0_1_0.wav \
  ./sherpa-onnx-tdnn-yesno/test_wavs/0_0_1_0_0_1_1_1.wav \
  ./sherpa-onnx-tdnn-yesno/test_wavs/0_0_1_0_1_0_0_1.wav \
  ./sherpa-onnx-tdnn-yesno/test_wavs/0_0_1_1_0_0_0_1.wav \
  ./sherpa-onnx-tdnn-yesno/test_wavs/0_0_1_1_0_1_1_0.wav
```

The output is given below:

```
OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=8000,␣
↪feature_dim=23), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder=""),␣
↪tdnn=OfflineTdnnModelConfig(model="./sherpa-onnx-tdnn-yesno/model-epoch-14-avg-2.onnx
↪"), tokens="./sherpa-onnx-tdnn-yesno/tokens.txt", num_threads=2, debug=False, provider=
↪"cpu", model_type=""), lm_config=OfflineLMConfig(model="", scale=0.5), decoding_method=
↪"greedy_search", max_active_paths=4, context_score=1.5)
Creating recognizer ...
Started
Done!

./sherpa-onnx-tdnn-yesno/test_wavs/0_0_0_1_0_0_0_1.wav
{"text":"NNNYNNNY","timestamps":"[]","tokens":["N","N","N","Y","N","N","N","Y"]}
----
./sherpa-onnx-tdnn-yesno/test_wavs/0_0_1_0_0_0_1_0.wav
{"text":"NNYNNNYN","timestamps":"[]","tokens":["N","N","Y","N","N","N","Y","N"]}
----
./sherpa-onnx-tdnn-yesno/test_wavs/0_0_1_0_0_1_1_1.wav
{"text":"NNYNNYYY","timestamps":"[]","tokens":["N","N","Y","N","N","Y","Y","Y"]}
----
./sherpa-onnx-tdnn-yesno/test_wavs/0_0_1_0_1_0_0_1.wav
{"text":"NNYNYNNY","timestamps":"[]","tokens":["N","N","Y","N","Y","N","N","Y"]}
----
./sherpa-onnx-tdnn-yesno/test_wavs/0_0_1_1_0_0_0_1.wav
{"text":"NNYYNNNY","timestamps":"[]","tokens":["N","N","Y","Y","N","N","N","Y"]}
----
./sherpa-onnx-tdnn-yesno/test_wavs/0_0_1_1_0_1_1_0.wav
{"text":"NNYYNYYN","timestamps":"[]","tokens":["N","N","Y","Y","N","Y","Y","N"]}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 0.071 s
Real time factor (RTF): 0.071 / 38.530 = 0.002
```

Note: In the above output, `N` represents `NO`, while `Y` is `YES`. So for the last wave, `NNYYNYYN` means `NO NO YES YES NO YES YES NO`.

In the filename of the last wave `0_0_1_1_0_1_1_0.wav`, 0 means `NO` and 1 means `YES`. So the ground truth of the last wave is `NO NO YES YES NO YES YES NO`.

8.23.7 TeleSpeech

This page lists all offline CTC models from https://github.com/Tele-AI/TeleSpeech-ASR.

Hint: Please see the license at TeleSpeech.pdf

Models from https://github.com/Tele-AI/TeleSpeech-ASR are subject to the above license if you want to use them for commercial purpose.

How to export models from Tele-AI/TeleSpeech-ASR to sherpa-onnx

This section describes how to export CTC models from *Tele-AI/TeleSpeech-ASR* to sherpa-onnx.

Step 1: Export model.onnx

The first step is to obtain `model.onnx`.

Please see https://github.com/lovemefan/telespeech-asr-python/blob/main/telespeechasr/onnx/onnx_export.py for details.

Step 2: Add metadata

To be usable in sherpa-onnx, we have to use add-metadata.py to add metadata to `model.onnx`.

Please see https://github.com/k2-fsa/sherpa-onnx/blob/master/scripts/tele-speech/run.sh for details.

Step 3: Obtain tokens.txt

Please also see https://github.com/k2-fsa/sherpa-onnx/blob/master/scripts/tele-speech/add-metadata.py

Models

sherpa-onnx-telespeech-ctc-int8-zh-2024-06-04 ()

Hint:

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪telespeech-ctc-int8-zh-2024-06-04.tar.bz2

# For Chinese users, please use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪telespeech-ctc-int8-zh-2024-06-04.tar.bz2

tar xvf sherpa-onnx-telespeech-ctc-int8-zh-2024-06-04.tar.bz2
rm sherpa-onnx-telespeech-ctc-int8-zh-2024-06-04.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff   325M Jun  4 11:56 model.int8.onnx
```

Decode wave files

Hint: It supports decoding only wave files of a single channel with 16-bit encoded samples, while the sampling rate does not need to be 16 kHz.

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --telespeech-ctc=./sherpa-onnx-telespeech-ctc-int8-zh-2024-06-04/model.int8.onnx \
  --tokens=./sherpa-onnx-telespeech-ctc-int8-zh-2024-06-04/tokens.txt \
  --model-type=telespeech_ctc \
  --num-threads=1 \
  ./sherpa-onnx-telespeech-ctc-int8-zh-2024-06-04/test_wavs/3-sichuan.wav \
  ./sherpa-onnx-telespeech-ctc-int8-zh-2024-06-04/test_wavs/4-tianjin.wav \
  ./sherpa-onnx-telespeech-ctc-int8-zh-2024-06-04/test_wavs/5-henan.wav
```

Note: Please use `./build/bin/Release/sherpa-onnx-offline.exe` for Windows.

Caution: If you use Windows and get encoding issues, please run:

```
CHCP 65001
```

in your commandline.

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪build/bin/sherpa-onnx-offline --telespeech-ctc=./sherpa-onnx-telespeech-ctc-int8-zh-
↪2024-06-04/model.int8.onnx --tokens=./sherpa-onnx-telespeech-ctc-int8-zh-2024-06-04/
↪tokens.txt --model-type=telespeech_ctc --num-threads=1 ./sherpa-onnx-telespeech-ctc-
↪int8-zh-2024-06-04/test_wavs/3-sichuan.wav ./sherpa-onnx-telespeech-ctc-int8-zh-2024-
↪06-04/test_wavs/4-tianjin.wav ./sherpa-onnx-telespeech-ctc-int8-zh-2024-06-04/test_
↪wavs/5-henan.wav
```

(continues on next page)

(continued from previous page)

```
OfflineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80, low_freq=20, high_freq=-400, dither=0), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), telespeech_ctc="./sherpa-onnx-telespeech-ctc-
↪int8-zh-2024-06-04/model.int8.onnx", tokens="./sherpa-onnx-telespeech-ctc-int8-zh-2024-
↪06-04/tokens.txt", num_threads=1, debug=False, provider="cpu", model_type="telespeech_
↪ctc", modeling_unit="cjkchar", bpe_vocab=""), lm_config=OfflineLMConfig(model="",␣
↪scale=0.5), ctc_fst_decoder_config=OfflineCtcFstDecoderConfig(graph="", max_
↪active=3000), decoding_method="greedy_search", max_active_paths=4, hotwords_file="",␣
↪hotwords_score=1.5, blank_penalty=0)
Creating recognizer ...
Started
Done!

./sherpa-onnx-telespeech-ctc-int8-zh-2024-06-04/test_wavs/3-sichuan.wav
{"text": "", "timestamps": [0.08, 0.36, 0.52, 0.72, 0.92, 1.16, 1.36, 1.88, 2.20, 2.36,␣
↪3.16, 3.28, 3.40, 3.60, 3.80, 3.92, 4.08, 4.24, 4.40, 4.56, 4.76, 5.16, 5.32, 5.44, 5.
↪64, 5.76, 5.88, 6.04, 6.16, 6.28, 6.40, 6.60, 6.88, 7.12, 7.40, 7.52, 7.64], "tokens":[
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""]}
----
./sherpa-onnx-telespeech-ctc-int8-zh-2024-06-04/test_wavs/4-tianjin.wav
{"text": "", "timestamps": [0.36, 0.56, 1.04, 1.16, 1.24, 1.64, 1.88, 2.24, 2.40, 2.60,␣
↪2.80, 3.12, 3.32, 3.64, 3.80, 3.96, 4.16, 4.44, 4.68, 4.80, 5.00, 5.16, 5.28, 6.12, 6.
↪28, 6.44, 6.60, 6.72, 6.88, 7.04, 7.12, 7.32, 7.52], "tokens":["", "", "", "", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", ""]}
----
./sherpa-onnx-telespeech-ctc-int8-zh-2024-06-04/test_wavs/5-henan.wav
{"text": "", "timestamps": [0.04, 0.12, 0.24, 0.40, 1.00, 1.24, 1.44, 1.68, 2.32, 2.48,␣
↪2.60, 2.64, 2.80, 3.00, 3.16, 3.32, 3.52, 3.68, 3.92, 5.00, 5.16, 5.28, 5.32, 5.44, 5.
↪84, 6.00, 6.12, 6.48, 6.68, 6.84, 7.00, 7.16, 7.32, 7.56, 7.68], "tokens":["", "", "",
↪"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
↪ "", "", "", "", "", "", "", "", "", ""]}
----
num threads: 1
decoding method: greedy_search
Elapsed seconds: 3.406 s
Real time factor (RTF): 3.406 / 23.634 = 0.144
```

Note: The `feature_dim=80` is incorrect in the above logs. The actual value is 40.

Hint: There is also a float32 model. Please see https://github.com/k2-fsa/sherpa-onnx/releases/download/

asr-models/sherpa-onnx-telespeech-ctc-zh-2024-06-04.tar.bz2

8.23.8 Whisper

This section describes how to use models from Whisper with sherpa-onnx for non-streaming speech recognition.

Export Whisper to ONNX

This section describes how to export Whisper models to onnx.

Available models

Note that we have already exported Whisper models to onnx and they are available from the following huggingface repositories:

Model type	Huggingface repo	
`tiny.en`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-tiny.en	
`base.en`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-base.en	
`small.en`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-small.en	
`distil-small.en`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-distil-small.en	
`medium.en`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-medium.en	
`distil-medium.en`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-distil-medium.en	
`tiny`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-tiny	
`base`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-base	
`small`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-small	
`medium`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-medium	
`large`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-large	
`large-v1`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-large-v1	
`large-v2`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-large-v2	
`large-v3`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-large-v3	
`distil-large-v2`	https://huggingface.co/csukuangfj/sherpa-onnx-whisper-distil-large-v2	

Note: You can also download them from

https://github.com/k2-fsa/sherpa-onnx/releases/tag/asr-models

Models end with `.en` support only English and all other models are multilingual.

If you want to export the models by yourself or/and want to learn how the models are exported, please read below.

Export to onnx

We use

https://github.com/k2-fsa/sherpa-onnx/blob/master/scripts/whisper/export-onnx.py

to export Whisper models to onnx.

First, let us install dependencies and download the export script

```
pip install torch openai-whisper onnxruntime onnx librosa soundfile

git clone https://github.com/k2-fsa/sherpa-onnx/
cd sherpa-onnx/scripts/whisper
python3 ./export-onnx.py --help
```

It will print the following message:

```
usage: export-onnx.py [-h] --model
                      {tiny,tiny.en,base,base.en,small,small.en,medium,medium.en,large,
↪large-v1,large-v2,large-v3,distil-medium.en,distil-small.en,distil-large-v2,medium-
↪aishell}

optional arguments:
  -h, --help            show this help message and exit
  --model {tiny,tiny.en,base,base.en,small,small.en,medium,medium.en,large,large-v1,
↪large-v2,large-v3,distil-medium.en,distil-small.en,distil-large-v2,medium-aishell}
```

Example 1: Export tiny.en

To export `tiny.en`, we can use:

```
python3 ./export-onnx.py --model tiny.en
```

It will generate the following files:

```
(py38) fangjuns-MacBook-Pro:whisper fangjun$ ls -lh tiny.en-*
-rw-r--r--  1 fangjun  staff   105M Aug  7 15:43 tiny.en-decoder.int8.onnx
-rw-r--r--  1 fangjun  staff   185M Aug  7 15:43 tiny.en-decoder.onnx
-rw-r--r--  1 fangjun  staff    12M Aug  7 15:43 tiny.en-encoder.int8.onnx
-rw-r--r--  1 fangjun  staff    36M Aug  7 15:43 tiny.en-encoder.onnx
-rw-r--r--  1 fangjun  staff   816K Aug  7 15:43 tiny.en-tokens.txt
```

`tiny.en-encoder.onnx` is the encoder model and `tiny.en-decoder.onnx` is the decoder model.

`tiny.en-encoder.int8.onnx` is the quantized encoder model and `tiny.en-decoder.onnx` is the quantized decoder model.

`tiny.en-tokens.txt` contains the token table, which maps an integer to a token and vice versa.

To check whether the exported model works correctly, we can use

https://github.com/k2-fsa/sherpa-onnx/blob/master/scripts/whisper/test.py

We use https://huggingface.co/csukuangfj/sherpa-onnx-whisper-tiny.en/resolve/main/test_wavs/0.wav as the test wave.

```
pip install kaldi-native-fbank
wget https://huggingface.co/csukuangfj/sherpa-onnx-whisper-tiny.en/resolve/main/test_
↪wavs/0.wav

python3 ./test.py \
  --encoder ./tiny.en-encoder.onnx \
  --decoder ./tiny.en-decoder.onnx \
  --tokens ./tiny.en-tokens.txt \
  ./0.wav
```

To test `int8` quantized models, we can use:

```
python3 ./test.py \
  --encoder ./tiny.en-encoder.int8.onnx \
  --decoder ./tiny.en-decoder.int8.onnx \
  --tokens ./tiny.en-tokens.txt \
  ./0.wav
```

Example 2: Export large-v3

To export `large-v3`, we can use:

```
python3 ./export-onnx.py --model large-v3
```

It will generate the following files:

```
(py38) fangjuns-MacBook-Pro:whisper fangjun$ ls -lh large-v3-*
-rw-r--r--  1 fangjun  staff   2.7M Jul 12 20:38 large-v3-decoder.onnx
-rw-r--r--  1 fangjun  staff   3.0G Jul 12 20:38 large-v3-decoder.weights
-rw-r--r--  1 fangjun  staff   744K Jul 12 20:35 large-v3-encoder.onnx
-rw-r--r--  1 fangjun  staff   2.8G Jul 12 20:35 large-v3-encoder.weights
-rw-r--r--  1 fangjun  staff   798K Jul 12 20:32 large-v3-tokens.txt
```

`large-v3-encoder.onnx` is the encoder model and `large-v3-decoder.onnx` is the decoder model.

Note that for `large` models, there will also be two additional `weights` files.

`large-v3-tokens.txt` contains the token table, which maps an integer to a token and vice versa.

To check whether the exported model works correctly, we can use
https://github.com/k2-fsa/sherpa-onnx/blob/master/scripts/whisper/test.py

We use https://huggingface.co/csukuangfj/sherpa-onnx-whisper-tiny.en/resolve/main/test_wavs/0.wav as the test wave.

```
pip install kaldi-native-fbank
wget https://huggingface.co/csukuangfj/sherpa-onnx-whisper-tiny.en/resolve/main/test_
↪wavs/0.wav

python3 ./test.py \
  --encoder ./large-v3-encoder.onnx \
  --decoder ./large-v3-decoder.onnx \
  --tokens ./large-v3-tokens.txt \
  ./0.wav
```

Hint: We provide a colab notebook for you to try the exported large-v3 onnx model with sherpa-onnx on CPU as well as on GPU.

You will find the RTF on GPU (Tesla T4) is less than 1.

tiny.en

You can use the following commands to download the exported onnx models of `tiny.en`:

Hint: Please replace `tiny.en` with `base.en`, `small.en`, `medium.en`, `distil-small.en`, `tiny`, `base`, `small`, and `medium` if you want to try a different type of model.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪whisper-tiny.en.tar.bz2
tar xvf sherpa-onnx-whisper-tiny.en.tar.bz2
```

Please check that the file sizes of the downloaded models are correct. See the file size of `*.onnx` files below.

```
(py38) fangjuns-MacBook-Pro:sherpa-onnx-whisper-tiny.en fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff   105M Aug  7 16:22 tiny.en-decoder.int8.onnx
-rw-r--r--  1 fangjun  staff   185M Aug  7 16:23 tiny.en-decoder.onnx
-rw-r--r--  1 fangjun  staff    12M Aug  7 16:22 tiny.en-encoder.int8.onnx
-rw-r--r--  1 fangjun  staff    36M Aug  7 16:22 tiny.en-encoder.onnx
```

To use the downloaded files to decode waves, please run:

Hint: Please first follow *Installation* to build sherpa-onnx before you continue.

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --whisper-encoder=./sherpa-onnx-whisper-tiny.en/tiny.en-encoder.onnx \
  --whisper-decoder=./sherpa-onnx-whisper-tiny.en/tiny.en-decoder.onnx \
  --tokens=./sherpa-onnx-whisper-tiny.en/tiny.en-tokens.txt \
  ./sherpa-onnx-whisper-tiny.en/test_wavs/0.wav \
  ./sherpa-onnx-whisper-tiny.en/test_wavs/1.wav \
  ./sherpa-onnx-whisper-tiny.en/test_wavs/8k.wav
```

To use `int8` quantized models, please use:

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline \
  --whisper-encoder=./sherpa-onnx-whisper-tiny.en/tiny.en-encoder.int8.onnx \
  --whisper-decoder=./sherpa-onnx-whisper-tiny.en/tiny.en-decoder.int8.onnx \
  --tokens=./sherpa-onnx-whisper-tiny.en/tiny.en-tokens.txt \
  ./sherpa-onnx-whisper-tiny.en/test_wavs/0.wav \
```

(continues on next page)

(continued from previous page)

```
  ./sherpa-onnx-whisper-tiny.en/test_wavs/1.wav \
  ./sherpa-onnx-whisper-tiny.en/test_wavs/8k.wav
```

Real-time factor (RTF) on Raspberry Pi 4 Model B

One of the test command is given below:

```
./sherpa-onnx-offline \
  --num-threads=1 \
  --whisper-encoder=./sherpa-onnx-whisper-tiny.en/tiny.en-encoder.onnx \
  --whisper-decoder=./sherpa-onnx-whisper-tiny.en/tiny.en-decoder.onnx \
  --tokens=./sherpa-onnx-whisper-tiny.en/tiny.en-tokens.txt \
  ./sherpa-onnx-whisper-tiny.en/test_wavs/1.wav
```

And its output is:

```
/root/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:361 ./
↪sherpa-onnx-offline --num-threads=1 --whisper-encoder=./sherpa-onnx-whisper-tiny.en/
↪tiny.en-encoder.onnx --whisper-decoder=./sherpa-onnx-whisper-tiny.en/tiny.en-decoder.
↪onnx --tokens=./sherpa-onnx-whisper-tiny.en/tiny.en-tokens.txt ./sherpa-onnx-whisper-
↪tiny.en/test_wavs/1.wav

OfflineRecognizerConfig(feat_config=OfflineFeatureExtractorConfig(sampling_rate=16000,␣
↪feature_dim=80), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="./sherpa-onnx-whisper-tiny.en/tiny.en-
↪encoder.onnx", decoder="./sherpa-onnx-whisper-tiny.en/tiny.en-decoder.onnx"), tokens=".
↪/sherpa-onnx-whisper-tiny.en/tiny.en-tokens.txt", num_threads=1, debug=False, provider=
↪"cpu", model_type=""), lm_config=OfflineLMConfig(model="", scale=0.5), decoding_method=
↪"greedy_search", max_active_paths=4, context_score=1.5)
Creating recognizer ...
Started
Done!

./sherpa-onnx-whisper-tiny.en/test_wavs/1.wav
{"text":" God, as a direct consequence of the sin which man thus punished, had given her␣
↪a lovely child, whose place was on that same dishonored bosom to connect her parent␣
↪forever with the race and descent of mortals, and to be finally a blessed soul in␣
↪heaven.","timestamps":"[]","tokens":[" God","," as"," a"," direct"," consequence","␣
↪of"," the"," sin"," which"," man"," thus"," punished","," had"," given"," her"," a",
↪" lovely"," child","," whose"," place"," was"," on"," that"," same"," dishon","ored",
↪" bos","om"," to"," connect"," her"," parent"," forever"," with"," the"," race"," and",
↪" descent"," of"," mortals","," and"," to"," be"," finally"," a"," blessed"," soul",
↪" in"," heaven","."]}
----
num threads: 1
decoding method: greedy_search
Elapsed seconds: 11.454 s
```

(continues on next page)

(continued from previous page)

```
Real time factor (RTF): 11.454 / 16.715 = 0.685
```

The following table compares the RTF between different number of threads and types of onnx models:

Model type	Number of threads	RTF
float32	1	0.685
float32	2	0.559
float32	3	0.526
float32	4	0.520
int8	1	0.547
int8	2	0.431
int8	3	0.398
int8	4	0.386

large-v3

Before we start, let us follow *Linux* to build a CUDA-enabled version of sherpa-onnx.

In the following, we assume you have run

```
cd /content

git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx

mkdir -p build
cd build
cmake \
  -DBUILD_SHARED_LIBS=ON \
  -DSHERPA_ONNX_ENABLE_GPU=ON ..

make -j2 sherpa-onnx-offline
```

You can use the following commands to download the exported onnx models of `large-v3`:

Hint: Please replace `large-v3` with `large`, `large-v1`, `large-v2`, and `distil-large-v2` if you want to try a different type of model.

```
cd /content

git lfs install
git clone https://huggingface.co/csukuangfj/sherpa-onnx-whisper-large-v3

ls -lh sherpa-onnx-whisper-large-v3
```

The logs of the above commands are given below:

```
Git LFS initialized.
Cloning into 'sherpa-onnx-whisper-large-v3'...
remote: Enumerating objects: 26, done.
```

(continues on next page)

(continued from previous page)

```
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 26 (delta 2), reused 0 (delta 0), pack-reused 4 (from 1)
Unpacking objects: 100% (26/26), 1.00 MiB | 9.10 MiB/s, done.
Filtering content: 100% (6/6), 7.40 GiB | 34.50 MiB/s, done.
total 7.5G
-rw-r--r-- 1 root root 962M Jul 13 14:19 large-v3-decoder.int8.onnx
-rw-r--r-- 1 root root 2.8M Jul 13 14:18 large-v3-decoder.onnx
-rw-r--r-- 1 root root 3.0G Jul 13 14:22 large-v3-decoder.weights
-rw-r--r-- 1 root root 732M Jul 13 14:19 large-v3-encoder.int8.onnx
-rw-r--r-- 1 root root 745K Jul 13 14:18 large-v3-encoder.onnx
-rw-r--r-- 1 root root 2.8G Jul 13 14:21 large-v3-encoder.weights
-rw-r--r-- 1 root root 798K Jul 13 14:18 large-v3-tokens.txt
drwxr-xr-x 2 root root 4.0K Jul 13 14:18 test_wavs
```

Caution: Please remember to run `git lfs install` before you run `git clone`. If you have any issues about `git lfs install`, please follow https://git-lfs.com/ to install `git-lfs`.

Caution: Please check the file sizes are correct before proceeding. Otherwise, you would be `SAD` later.

Caution: Please check the file sizes are correct before proceeding. Otherwise, you would be `SAD` later.

Caution: Please check the file sizes are correct before proceeding. Otherwise, you would be `SAD` later.

Run with CPU (float32)

```
cd /content

exe=$PWD/sherpa-onnx/build/bin/sherpa-onnx-offline

cd sherpa-onnx-whisper-large-v3

time $exe \
  --whisper-encoder=./large-v3-encoder.onnx \
  --whisper-decoder=./large-v3-decoder.onnx \
  --tokens=./large-v3-tokens.txt \
  --num-threads=2 \
  ./test_wavs/0.wav
```

The logs are given below:

```
/content/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:375 /content/sherpa-onnx/
↪build/bin/sherpa-onnx-offline --whisper-encoder=./large-v3-encoder.onnx --whisper-
↪decoder=./large-v3-decoder.onnx --tokens=./large-v3-tokens.txt --num-threads=2 ./test_
↪wavs/0.wav
```

(continues on next page)

(continued from previous page)

```
OfflineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80, low_freq=20, high_freq=-400, dither=0), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="./large-v3-encoder.onnx", decoder="./large-
↪v3-decoder.onnx", language="", task="transcribe", tail_paddings=-1),␣
↪tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), telespeech_ctc="", tokens="./large-v3-tokens.
↪txt", num_threads=2, debug=False, provider="cpu", model_type="", modeling_unit="cjkchar
↪", bpe_vocab=""), lm_config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=0,␣
↪rule_fsts="", rule_fars="")
Creating recognizer ...
Started
Done!

./test_wavs/0.wav
{"text": " after early nightfall the yellow lamps would light up here and there the␣
↪squalid quarter of the brothels", "timestamps": [], "tokens":[" after", " early", "␣
↪night", "fall", " the", " yellow", " lamps", " would", " light", " up", " here", " and
↪", " there", " the", " squ", "alid", " quarter", " of", " the", " broth", "els"],
↪"words": []}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 54.070 s
Real time factor (RTF): 54.070 / 6.625 = 8.162

real        1m32.107s
user        1m39.877s
sys 0m10.405s
```

Run with CPU (int8)

```
cd /content

exe=$PWD/sherpa-onnx/build/bin/sherpa-onnx-offline

cd sherpa-onnx-whisper-large-v3

time $exe \
  --whisper-encoder=./large-v3-encoder.int8.onnx \
  --whisper-decoder=./large-v3-decoder.int8.onnx \
  --tokens=./large-v3-tokens.txt \
  --num-threads=2 \
```

(continues on next page)

(continued from previous page)

```
./test_wavs/0.wav
```

The logs are given below:

```
/content/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:375 /content/sherpa-onnx/
↪build/bin/sherpa-onnx-offline --whisper-encoder=./large-v3-encoder.int8.onnx --whisper-
↪decoder=./large-v3-decoder.int8.onnx --tokens=./large-v3-tokens.txt --num-threads=2 ./
↪test_wavs/0.wav

OfflineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80, low_freq=20, high_freq=-400, dither=0), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="./large-v3-encoder.int8.onnx", decoder="./
↪large-v3-decoder.int8.onnx", language="", task="transcribe", tail_paddings=-1),␣
↪tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), telespeech_ctc="", tokens="./large-v3-tokens.
↪txt", num_threads=2, debug=False, provider="cpu", model_type="", modeling_unit="cjkchar
↪", bpe_vocab=""), lm_config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=0,␣
↪rule_fsts="", rule_fars="")
Creating recognizer ...
Started
Done!

./test_wavs/0.wav
{"text": " after early nightfall the yellow lamps would light up here and there the␣
↪squalid quarter of the brothels", "timestamps": [], "tokens":[" after", " early", "␣
↪night", "fall", " the", " yellow", " lamps", " would", " light", " up", " here", " and
↪", " there", " the", " squ", "alid", " quarter", " of", " the", " broth", "els"],
↪"words": []}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 49.991 s
Real time factor (RTF): 49.991 / 6.625 = 7.546

real  1m15.555s
user  1m41.488s
sys   0m9.156s
```

Run with GPU (float32)

```
cd /content
exe=$PWD/sherpa-onnx/build/bin/sherpa-onnx-offline

cd sherpa-onnx-whisper-large-v3

time $exe \
  --whisper-encoder=./large-v3-encoder.onnx \
  --whisper-decoder=./large-v3-decoder.onnx \
  --tokens=./large-v3-tokens.txt \
  --provider=cuda \
  --num-threads=2 \
  ./test_wavs/0.wav
```

The logs are given below:

```
/content/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:375 /content/sherpa-onnx/
↪build/bin/sherpa-onnx-offline --whisper-encoder=./large-v3-encoder.onnx --whisper-
↪decoder=./large-v3-decoder.onnx --tokens=./large-v3-tokens.txt --provider=cuda --num-
↪threads=2 ./test_wavs/0.wav

OfflineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80, low_freq=20, high_freq=-400, dither=0), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="./large-v3-encoder.onnx", decoder="./large-
↪v3-decoder.onnx", language="", task="transcribe", tail_paddings=-1),␣
↪tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), telespeech_ctc="", tokens="./large-v3-tokens.
↪txt", num_threads=2, debug=False, provider="cuda", model_type="", modeling_unit=
↪"cjkchar", bpe_vocab=""), lm_config=OfflineLMConfig(model="", scale=0.5), ctc_fst_
↪decoder_config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method=
↪"greedy_search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_
↪penalty=0, rule_fsts="", rule_fars="")
Creating recognizer ...
Started
Done!

./test_wavs/0.wav
{"text": " after early nightfall the yellow lamps would light up here and there the␣
↪squalid quarter of the brothels", "timestamps": [], "tokens":[" after", " early", "␣
↪night", "fall", " the", " yellow", " lamps", " would", " light", " up", " here", " and
↪", " there", " the", " squ", "alid", " quarter", " of", " the", " broth", "els"],
↪"words": []}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 5.910 s
Real time factor (RTF): 5.910 / 6.625 = 0.892
```

(continues on next page)

(continued from previous page)

```
real  0m26.996s
user  0m12.854s
sys   0m4.486s
```

Note: The above command is run within a colab notebook using Tesla T4 GPU. You can see the RTF is less than 1. If you has some more performant GPU, you would get an even lower RTF.

Run with GPU (int8)

```
cd /content
exe=$PWD/sherpa-onnx/build/bin/sherpa-onnx-offline

cd sherpa-onnx-whisper-large-v3

time $exe \
  --whisper-encoder=./large-v3-encoder.int8.onnx \
  --whisper-decoder=./large-v3-decoder.int8.onnx \
  --tokens=./large-v3-tokens.txt \
  --provider=cuda \
  --num-threads=2 \
  ./test_wavs/0.wav
```

The logs are given below:

```
/content/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:375 /content/sherpa-onnx/
↪build/bin/sherpa-onnx-offline --whisper-encoder=./large-v3-encoder.int8.onnx --whisper-
↪decoder=./large-v3-decoder.int8.onnx --tokens=./large-v3-tokens.txt --provider=cuda --
↪num-threads=2 ./test_wavs/0.wav

OfflineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80, low_freq=20, high_freq=-400, dither=0), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="./large-v3-encoder.int8.onnx", decoder="./
↪large-v3-decoder.int8.onnx", language="", task="transcribe", tail_paddings=-1),␣
↪tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), telespeech_ctc="", tokens="./large-v3-tokens.
↪txt", num_threads=2, debug=False, provider="cuda", model_type="", modeling_unit=
↪"cjkchar", bpe_vocab=""), lm_config=OfflineLMConfig(model="", scale=0.5), ctc_fst_
↪decoder_config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method=
↪"greedy_search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_
↪penalty=0, rule_fsts="", rule_fars="")
Creating recognizer ...
Started
```

(continues on next page)

(continued from previous page)

```
Done!

./test_wavs/0.wav
{"text": " after early nightfall the yellow lamps would light up here and there the␣
↪squalid quarter of the brothels", "timestamps": [], "tokens":[" after", " early", "␣
↪night", "fall", " the", " yellow", " lamps", " would", " light", " up", " here", " and
↪", " there", " the", " squ", "alid", " quarter", " of", " the", " broth", "els"],
↪"words": []}
----
num threads: 2
decoding method: greedy_search
Elapsed seconds: 19.190 s
Real time factor (RTF): 19.190 / 6.625 = 2.897

real  0m46.850s
user  0m50.007s
sys   0m8.013s
```

Fix issues about running on GPU

If you get errors like below:

```
what():  /onnxruntime_src/onnxruntime/core/session/provider_bridge_ort.cc:1426
onnxruntime::Provider& onnxruntime::ProviderLibrary::Get()
[ONNXRuntimeError] : 1 : FAIL :
Failed to load library libonnxruntime_providers_cuda.so with error:
libcublasLt.so.11: cannot open shared object file: No such file or directory
```

please follow https://www.google.com/url?q=https%3A%2F%2Fk2-fsa.github.io%2Fk2%2Finstallation%2Fcuda-cudnn.html to install CUDA toolkit.

To determine which version of CUDA toolkit to install, please read https://onnxruntime.ai/docs/execution-providers/CUDA-ExecutionProvider.html to figure it out.

For instance, if onnxruntime v1.18.1 is used in sherpa-onnx, we have to install CUDA 11.8 according to https://onnxruntime.ai/docs/execution-providers/CUDA-ExecutionProvider.html

colab

Please see the following colab notebook .

It walks you step by step to try the exported large-v3 onnx model with sherpa-onnx on CPU as well as on GPU.

colab

Non-large models

We provide a colab notebook for you to try Whisper models with sherpa-onnx step by step.

model	CPU or CUDA	RTF
float32 tiny.en	CPU	0.208
float32 tiny.en	CUDA	0.186
int8 tiny.en	CPU	0.142
int8 tiny.en	CUDA	0.146
float32 base.en	CPU	0.430
float32 base.en	CUDA	0.129
int8 base.en	CPU	0.332
int8 base.en	CUDA	0.259
float32 small.en	CPU	1.633
float32 small.en	CUDA	0.268

Large models

For `large` models of whisper, please see the following colab notebook . It walks you step by step to try the exported large-v3 onnx model with sherpa-onnx on CPU as well as on GPU.

You will find the RTF on GPU (Tesla T4) is less than 1.

Huggingface space

You can try Whisper models from within your browser without installing anything.

Please visit

https://huggingface.co/spaces/k2-fsa/automatic-speech-recognition

8.23.9 WeNet

This page lists all CTC models from WeNet.

How to export models from WeNet to sherpa-onnx

Suppose you have the following files from WeNet:

- `final.pt`
- `train.yaml`
- `global_cmvn`
- `units.txt`

We describe below how to use scripts from sherpa-onnx to export your files.

Hint: Both streaming and non-streaming models are supported.

Export for non-streaming inference

You can use the following script

https://github.com/k2-fsa/sherpa-onnx/blob/master/scripts/wenet/export-onnx.py

to export your model to sherpa-onnx. After running it, you should get two files:

- `model.onnx`
- `model.int8.onnx`.

Next, we rename `units.txt` to `tokens.txt` to follow the convention used in sherpa-onnx:

```
mv units.txt tokens.txt
```

Now you can use the following command for speech recognition with the exported models:

```
# with float32 models
./build/bin/sherpa-onnx-offline \
  --wenet-ctc-model=./model.onnx
  --tokens=./tokens.txt \
  /path/to/some.wav

# with int8 models
./build/bin/sherpa-onnx-offline \
  --wenet-ctc-model=./model.int8.onnx
  --tokens=./tokens.txt \
  /path/to/some.wav
```

Export for streaming inference

You can use the following script

https://github.com/k2-fsa/sherpa-onnx/blob/master/scripts/wenet/export-onnx-streaming.py

to export your model to sherpa-onnx. After running it, you should get two files:

- `model-streaming.onnx`
- `model-streaming.int8.onnx`.

Next, we rename `units.txt` to `tokens.txt` to follow the convention used in sherpa-onnx:

```
mv units.txt tokens.txt
```

Now you can use the following command for speech recognition with the exported models:

```
# with float32 models
./build/bin/sherpa-onnx \
  --wenet-ctc-model=./model-streaming.onnx
  --tokens=./tokens.txt \
  /path/to/some.wav

# with int8 models
./build/bin/sherpa-onnx \
  --wenet-ctc-model=./model-streaming.int8.onnx
  --tokens=./tokens.txt \
  /path/to/some.wav
```

FAQs

sherpa-onnx/csrc/online-wenet-ctc-model.cc:Init:144 head does not exist in the metadata

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/online-wenet-ctc-model.
↪cc:Init:144 head does not exist in the metadata
```

To fix the above error, please check the following two items:

- Make sure you are using `model-streaming.onnx` or `model-streaing.int8.onnx`. The executable you are running requires a streaming model as input.
- Make sure you use the script from sherpa-onnx to export your model.

All models from WeNet

https://github.com/wenet-e2e/wenet/blob/main/docs/pretrained_models.en.md lists all pre-trained models from WeNet and we have converted all of them to sherpa-onnx using the following script:

https://github.com/k2-fsa/sherpa-onnx/blob/master/scripts/wenet/run.sh.

We have uploaded the exported models to huggingface and you can find them from the following figure:

To make it easier to copy the links, we list them below:

github.com/wenet-e2e/wenet/blob/main/docs/pretrained_models.md

main wenet / docs / pretrained_models.md

Preview Code Blame 28 lines (19 loc) · 3.45 KB Raw

The pretrained model in WeNet follows the license of it's corresponding dataset. For example, the pretrained model on LibriSpeech follows `CC BY 4.0`, since it is used as license of the LibriSpeech dataset, see http://openslr.org/12/.

Model List

Here is a list of the pretrained models on different datasets.

For non-Chinese users, please visit Pretrained Models(En) to download.

Datasets	Languages	Checkpoint Model	Runtime Model	Contributor	exported for sherpa-onnx
aishell	CN	Conformer	Conformer	出门问问	https://huggingface.co/csukuangfj/-sherpa-onnx-zh-wenet-aishell
aishell2	CN	Conformer	Conformer	出门问问	https://huggingface.co/csukuangfj/sherpa-onnx-zh-wenet-aishell2
gigaspeech	EN	Conformer	Conformer	出门问问	https://huggingface.co/csukuangfj/-sherpa-onnx-en-wenet-gigaspeech
librispeech	EN	Conformer	Conformer	出门问问	https://huggingface.co/csukuangfj/-sherpa-onnx-en-wenet-librispeech
multi_cn	CN	Conformer	Conformer	京东	https://huggingface.co/csukuangfj/-sherpa-onnx-zh-wenet-multi-cn
wenetspeech	CN	Conformer	Conformer	地平线 Horizon Robotics	https://huggingface.co/csukuangfj/-sherpa-onnx-zh-wenet-wenetspeech

Fig. 8.29: All pre-trained models from WeNet.

- https://huggingface.co/csukuangfj/sherpa-onnx-zh-wenet-aishell
- https://huggingface.co/csukuangfj/sherpa-onnx-zh-wenet-aishell2
- https://huggingface.co/csukuangfj/sherpa-onnx-en-wenet-gigaspeech
- https://huggingface.co/csukuangfj/sherpa-onnx-en-wenet-librispeech
- https://huggingface.co/csukuangfj/sherpa-onnx-zh-wenet-multi-cn
- https://huggingface.co/csukuangfj/sherpa-onnx-zh-wenet-wenetspeech

Colab

We provide a colab notebook for you to try the exported WeNet models with sherpa-onnx.

8.23.10 Small models

In this section, we list online/streaming models with fewer parameters that are suitable for resource constrained embedded systems.

Hint: You can use them as a first pass model in a two-pass system, where the second pass uses a non-streaming model.

Hint: If you are using Raspberry Pi 4, this section is not so helpful for you since all models in sherpa-onnx are able to run in real-time on it.

This page is especially useful for systems with less resource than Raspberry Pi 4.

- *csukuangfj/sherpa-onnx-streaming-zipformer-zh-14M-2023-02-23 (Chinese)*
- *csukuangfj/sherpa-onnx-streaming-zipformer-en-20M-2023-02-17 (English)*
- *sherpa-onnx-streaming-zipformer-small-bilingual-zh-en-2023-02-16 (Bilingual, Chinese + English)*

8.24 SenseVoice

This section describes how to use models from https://github.com/FunAudioLLM/SenseVoice.

A single model from SenseVoice supports the following languages

- Chinese (Mandarin,)
- Cantonese (,)
- English
- Japanese
- Korean

which is similar to what multilingual Whisper is doing.

We have converted SenseVoice to onnx and provided APIs for the following programming languages

- 1. C++
- 2. C

- 3. Python
- 4. C#
- 5. Go
- 6. Kotlin
- 7. Java
- 8. JavaScript (Support WebAssembly and Node)
- 9. Swift
- 10. Dart (Support Flutter)

Note that you can use SenseVoice with sherpa-onnx on the following platforms:

- Linux (x64, aarch64, arm, riscv64)
- macOS (x64, arm64)
- Windows (x64, x86, arm64)
- Android (arm64-v8a, armv7-eabi, x86, x86_64)
- iOS (arm64)

In the following, we describe how to download pre-trained SenseVoice models and use them in sherpa-onnx.

8.24.1 Huggingface space

You can try SenseVoice with sherpa-onnx with the following huggingface space

https://huggingface.co/spaces/k2-fsa/automatic-speech-recognition

Hint: You don't need to install anything. All you need is a browser.

You can even run it on your phone or tablet.

8.24.2 Export SenseVoice to sherpa-onnx

This page describes how to export SenseVoice to onnx so that you can use it with sherpa-onnx.

The code

Please refer to export-onnx.py

The entry point is run.sh

After executing run.sh, you should get the following files

- `model.onnx`, the float32 onnx model
- `model.int8.onnx`, the 8-bit quantized model
- `tokens.txt`, for converting integer token IDs to strings
- `test_wavs/zh.wav`, test wave for Chinese
- `test_wavs/en.wav`, test wave for English

Fig. 8.30: Try SenseVoice in our Huggingface space with sherpa-onnx

- `test_wavs/ko.wav`, test wave for Korean
- `test_wavs/ja.wav`, test wave for Japanese
- `test_wavs/yue.wav`, test wave for Cantonese

Test the exported model

You can use test.py to test the exported model.

Note that test.py does not depend on sherpa-onnx. It uses onnxruntime Python API.

Where to find exported models

You can find the exported SenseVoice models at

https://github.com/k2-fsa/sherpa-onnx/releases/tag/asr-models

The following is an example about how to download an exported SenseVoice model:

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2
```

(continues on next page)

(continued from previous page)

```
tar xvf sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2
rm sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2
```

To view the downloaded files, please use:

```
ls -lh sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17

total 1.1G
-rw-r--r-- 1 runner docker   71 Jul 18 13:06 LICENSE
-rw-r--r-- 1 runner docker  104 Jul 18 13:06 README.md
-rwxr-xr-x 1 runner docker 5.8K Jul 18 13:06 export-onnx.py
-rw-r--r-- 1 runner docker 229M Jul 18 13:06 model.int8.onnx
-rw-r--r-- 1 runner docker 895M Jul 18 13:06 model.onnx
drwxr-xr-x 2 runner docker 4.0K Jul 18 13:06 test_wavs
-rw-r--r-- 1 runner docker 309K Jul 18 13:06 tokens.txt

ls -lh sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs

total 940K
-rw-r--r-- 1 runner docker 224K Jul 18 13:06 en.wav
-rw-r--r-- 1 runner docker 226K Jul 18 13:06 ja.wav
-rw-r--r-- 1 runner docker 145K Jul 18 13:06 ko.wav
-rw-r--r-- 1 runner docker 161K Jul 18 13:06 yue.wav
-rw-r--r-- 1 runner docker 175K Jul 18 13:06 zh.wav
```

8.24.3 Pre-trained Models

This page describes how to download pre-trained SenseVoice models.

sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17

This model is converted from https://www.modelscope.cn/models/iic/SenseVoiceSmall using the script export-onnx.py.

It supports the following 5 languages:

- Chinese (Mandarin,)
- Cantonese (,)
- English
- Japanese
- Korean

In the following, we describe how to download it.

Download

Please use the following commands to download it:

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2

# For Chinese users, you can use the following mirror
# wget https://hub.nuaa.cf/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-onnx-
↪sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2

tar xvf sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2
rm sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2
```

After downloading, you should find the following files:

```
ls -lh sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17

total 1.1G
-rw-r--r-- 1 runner docker   71 Jul 18 13:06 LICENSE
-rw-r--r-- 1 runner docker  104 Jul 18 13:06 README.md
-rwxr-xr-x 1 runner docker 5.8K Jul 18 13:06 export-onnx.py
-rw-r--r-- 1 runner docker 229M Jul 18 13:06 model.int8.onnx
-rw-r--r-- 1 runner docker 895M Jul 18 13:06 model.onnx
drwxr-xr-x 2 runner docker 4.0K Jul 18 13:06 test_wavs
-rw-r--r-- 1 runner docker 309K Jul 18 13:06 tokens.txt

ls -lh sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs

total 940K
-rw-r--r-- 1 runner docker 224K Jul 18 13:06 en.wav
-rw-r--r-- 1 runner docker 226K Jul 18 13:06 ja.wav
-rw-r--r-- 1 runner docker 145K Jul 18 13:06 ko.wav
-rw-r--r-- 1 runner docker 161K Jul 18 13:06 yue.wav
-rw-r--r-- 1 runner docker 175K Jul 18 13:06 zh.wav
```

Decode a file with model.onnx

Without inverse text normalization

To decode a file without inverse text normalization, please use:

```
./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/tokens.txt \
  --sense-voice-model=/Users/fangjun/open-source/sherpa-onnx/scripts/sense-voice/model.
↪onnx \
  --num-threads=1 \
  --debug=0 \
  ./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs/zh.wav \
  ./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs/en.wav
```

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:375 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-
↪07-17/tokens.txt --sense-voice-model=/Users/fangjun/open-source/sherpa-onnx/scripts/
↪sense-voice/model.onnx --num-threads=1 --debug=0 ./sherpa-onnx-sense-voice-zh-en-ja-ko-
↪yue-2024-07-17/test_wavs/zh.wav ./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/
↪test_wavs/en.wav

OfflineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80, low_freq=20, high_freq=-400, dither=0), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), sense_
↪voice=OfflineSenseVoiceModelConfig(model="/Users/fangjun/open-source/sherpa-onnx/
↪scripts/sense-voice/model.onnx", language="auto", use_itn=False), telespeech_ctc="",␣
↪tokens="./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/tokens.txt", num_
↪threads=1, debug=False, provider="cpu", model_type="", modeling_unit="cjkchar", bpe_
↪vocab=""), lm_config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=0,␣
↪rule_fsts="", rule_fars="")
Creating recognizer ...
Started
Done!

./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs/zh.wav
{"text": "", "timestamps": [0.72, 0.96, 1.26, 1.44, 1.92, 2.10, 2.58, 2.82, 3.30, 3.90,␣
↪4.20, 4.56, 4.74], "tokens":["", "", "", "", "", "", "", "", "", "", "", "", ""],
↪"words": []}
----
./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs/en.wav
{"text": "the tribal chieftain called for the boy and presented him with fifty pieces of␣
↪gold", "timestamps": [0.90, 1.26, 1.56, 1.80, 2.16, 2.46, 2.76, 2.94, 3.12, 3.60, 3.96,
↪ 4.50, 4.74, 5.10, 5.52, 5.88, 6.18], "tokens":["the", " tri", "bal", " chief", "tain",
↪ " called", " for", " the", " boy", " and", " presented", " him", " with", " fifty", "␣
↪pieces", " of", " gold"], "words": []}
----
num threads: 1
decoding method: greedy_search
Elapsed seconds: 2.320 s
Real time factor (RTF): 2.320 / 12.744 = 0.182
```

With inverse text normalization

To decode a file with inverse text normalization, please use:

```
./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/tokens.txt \
  --sense-voice-model=/Users/fangjun/open-source/sherpa-onnx/scripts/sense-voice/model.
↪onnx \
  --num-threads=1 \
  --sense-voice-use-itn=1 \
  --debug=0 \
  ./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs/zh.wav \
  ./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs/en.wav
```

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:375 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-
↪07-17/tokens.txt --sense-voice-model=/Users/fangjun/open-source/sherpa-onnx/scripts/
↪sense-voice/model.onnx --num-threads=1 --sense-voice-use-itn=1 --debug=0 ./sherpa-onnx-
↪sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs/zh.wav ./sherpa-onnx-sense-voice-zh-
↪en-ja-ko-yue-2024-07-17/test_wavs/en.wav

OfflineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80, low_freq=20, high_freq=-400, dither=0), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), sense_
↪voice=OfflineSenseVoiceModelConfig(model="/Users/fangjun/open-source/sherpa-onnx/
↪scripts/sense-voice/model.onnx", language="auto", use_itn=True), telespeech_ctc="",␣
↪tokens="./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/tokens.txt", num_
↪threads=1, debug=False, provider="cpu", model_type="", modeling_unit="cjkchar", bpe_
↪vocab=""), lm_config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=0,␣
↪rule_fsts="", rule_fars="")
Creating recognizer ...
Started
Done!

./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs/zh.wav
{"text": "95", "timestamps": [0.72, 0.96, 1.26, 1.44, 1.92, 2.10, 2.58, 2.82, 3.30, 3.90,
↪ 4.20, 4.56, 4.74, 5.46], "tokens":["", "", "", "", "", "", "9", "", "", "", "", "5", "
↪", ""], "words": []}
----
./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs/en.wav
{"text": "The tribal chieftain called for the boy and presented him with 50 pieces of␣
↪gold.", "timestamps": [0.90, 1.26, 1.56, 1.80, 2.16, 2.46, 2.76, 2.94, 3.12, 3.60, 3.
↪96, 4.50, 4.74, 4.92, 5.10, 5.28, 5.52, 5.88, 6.18, 7.02], "tokens":["The", " tri",
↪"bal", " chief", "tain", " called", " for", " the", " boy", " and", " presented", " him
↪", " with", " ", "5", "0", " pieces", " of", " gold", "."], "words": []}
```

(continues on next page)

(continued from previous page)

```
----
num threads: 1
decoding method: greedy_search
Elapsed seconds: 1.543 s
Real time factor (RTF): 1.543 / 12.744 = 0.121
```

Hint: When inverse text normalziation is enabled, the results also punctuations.

Specify a language

If you don't provide a language when decoding, it uses `auto`.

To specify the language when decoding, please use:

```
./build/bin/sherpa-onnx-offline \
  --tokens=./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/tokens.txt \
  --sense-voice-model=/Users/fangjun/open-source/sherpa-onnx/scripts/sense-voice/model.
↪onnx \
  --num-threads=1 \
  --sense-voice-language=zh \
  --debug=0 \
  ./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs/zh.wav
```

You should see the following output:

```
/Users/fangjun/open-source/sherpa-onnx/sherpa-onnx/csrc/parse-options.cc:Read:375 ./
↪build/bin/sherpa-onnx-offline --tokens=./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-
↪07-17/tokens.txt --sense-voice-model=/Users/fangjun/open-source/sherpa-onnx/scripts/
↪sense-voice/model.onnx --num-threads=1 --sense-voice-language=zh --debug=0 ./sherpa-
↪onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs/zh.wav

OfflineRecognizerConfig(feat_config=FeatureExtractorConfig(sampling_rate=16000, feature_
↪dim=80, low_freq=20, high_freq=-400, dither=0), model_
↪config=OfflineModelConfig(transducer=OfflineTransducerModelConfig(encoder_filename="",␣
↪decoder_filename="", joiner_filename=""),␣
↪paraformer=OfflineParaformerModelConfig(model=""), nemo_
↪ctc=OfflineNemoEncDecCtcModelConfig(model=""),␣
↪whisper=OfflineWhisperModelConfig(encoder="", decoder="", language="", task="transcribe
↪", tail_paddings=-1), tdnn=OfflineTdnnModelConfig(model=""), zipformer_
↪ctc=OfflineZipformerCtcModelConfig(model=""), wenet_
↪ctc=OfflineWenetCtcModelConfig(model=""), sense_
↪voice=OfflineSenseVoiceModelConfig(model="/Users/fangjun/open-source/sherpa-onnx/
↪scripts/sense-voice/model.onnx", language="zh", use_itn=False), telespeech_ctc="",␣
↪tokens="./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/tokens.txt", num_
↪threads=1, debug=False, provider="cpu", model_type="", modeling_unit="cjkchar", bpe_
↪vocab=""), lm_config=OfflineLMConfig(model="", scale=0.5), ctc_fst_decoder_
↪config=OfflineCtcFstDecoderConfig(graph="", max_active=3000), decoding_method="greedy_
↪search", max_active_paths=4, hotwords_file="", hotwords_score=1.5, blank_penalty=0,␣
↪rule_fsts="", rule_fars="")
Creating recognizer ...
```

(continues on next page)

(continued from previous page)

```
Started
Done!

./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs/zh.wav
{"text": "", "timestamps": [0.72, 0.96, 1.26, 1.44, 1.92, 2.10, 2.58, 2.82, 3.30, 3.90,␣
↪4.20, 4.56, 4.74], "tokens":["", "", "", "", "", "", "", "", "", "", "", "", ""],
↪"words": []}
----
num threads: 1
decoding method: greedy_search
Elapsed seconds: 0.625 s
Real time factor (RTF): 0.625 / 5.592 = 0.112
```

Hint: Valid values for `--sense-voice-language` are `auto`, `zh`, `en`, `ko`, `ja`, and `yue`. where `zh` is for Chinese, `en` for English, `ko` for Korean, `ja` for Japanese, and `yue` for `Cantonese`.

8.24.4 C API for SenseVoice

This page describes how to use the C API for SenseVoice.

Please refer to *C API* for how to build sherpa-onnx.

The following is a very quick introduction for using the C API of sherpa-onnx in the form of shared libraries on macOS and Linux.

Hint: We do support static libraries and also support Windows.

If you copy, paste, and run the following commands in your terminal, you should be able to see the following recognition result:

```
Decoded text: The tribal chieftain called for the boy and presented him with 50 pieces␣
↪of gold.
```

```
cd /tmp

git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx

curl -SL -O https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-
↪onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2
tar xvf sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2
rm sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2


ls -lh sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17

echo "---"

ls -lh sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs
```

(continues on next page)

(continued from previous page)

```
mkdir build
cd build
cmake \
  -D CMAKE_BUILD_TYPE=Release \
  -D BUILD_SHARED_LIBS=ON \
  -D CMAKE_INSTALL_PREFIX=./install \
  -D SHERPA_ONNX_ENABLE_BINARY=OFF \
  ..

make -j2 install

ls -lh install/lib
ls -lh install/include

cd ..

gcc -o sense-voice-c-api ./c-api-examples/sense-voice-c-api.c \
  -I ./build/install/include \
  -L ./build/install/lib/ \
  -l sherpa-onnx-c-api \
  -l onnxruntime

ls -lh sense-voice-c-api

export LD_LIBRARY_PATH=$PWD/build/install/lib:$LD_LIBRARY_PATH
export DYLD_LIBRARY_PATH=$PWD/build/install/lib:$DYLD_LIBRARY_PATH

./sense-voice-c-api
```

Note that we have hard-coded the file paths inside sense-voice-c-api.c

Hint: Since we are using shared libraries in the above example, you have to set the environemnt variable `LD_LIBRARY_PATH` for Linux and `DYLD_LIBRARY_PATH` for macOS. Otherwise, you would get runtime errors when running `./sense-voice-c-api`.

Explanations

1. Download sherpa-onnx

```
cd /tmp

git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx
```

In this example, we download sherpa-onnx and place it inside the directory `/tmp/`. You can replace `/tmp/` with any directory you like.

Please always download the latest master of sherpa-onnx.

2. Download the model

```
curl -SL -O https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-
↪onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2
tar xvf sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2
rm sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2
```

Note that we have placed the model in the directory `/tmp/sherpa-onnx`.

3. Build sherpa-onnx

```
mkdir build
cd build
cmake \
  -D CMAKE_BUILD_TYPE=Release \
  -D BUILD_SHARED_LIBS=ON \
  -D CMAKE_INSTALL_PREFIX=./install \
  -D SHERPA_ONNX_ENABLE_BINARY=OFF \
  ..

make -j2 install
```

We build a Release version of sherpa-onnx. Also, we use shared libraries here. The header file `c-api.h` and shared libraries are installed into the directory `./build/install`.

If you are using Linux, you should see the following content:

```
Install the project...
-- Install configuration: "Release"
-- Installing: /tmp/sherpa-onnx/build/install/lib/libonnxruntime.so
-- Installing: /tmp/sherpa-onnx/build/install/./sherpa-onnx.pc
-- Installing: /tmp/sherpa-onnx/build/install/lib/libsherpa-onnx-c-api.so
-- Set non-toolchain portion of runtime path of "/tmp/sherpa-onnx/build/install/lib/
↪libsherpa-onnx-c-api.so" to "$ORIGIN"
-- Installing: /tmp/sherpa-onnx/build/install/include/sherpa-onnx/c-api/c-api.h
```

If you are using macOS, you should see:

```
Install the project...
-- Install configuration: "Release"
-- Installing: /tmp/sherpa-onnx/build/install/lib/libonnxruntime.1.17.1.dylib
-- Installing: /tmp/sherpa-onnx/build/install/lib/libonnxruntime.dylib
-- Installing: /tmp/sherpa-onnx/build/install/./sherpa-onnx.pc
-- Installing: /tmp/sherpa-onnx/build/install/lib/libsherpa-onnx-c-api.dylib
-- Installing: /tmp/sherpa-onnx/build/install/include/sherpa-onnx/c-api/c-api.h
```

4. View the build result

```
ls -lh install/lib
ls -lh install/include
```

If you are using Linux, you should see the following content:

```
total 19M
-rw-r--r-- 1 runner docker  15M Jul 22 08:47 libonnxruntime.so
-rw-r--r-- 1 runner docker 4.1M Jul 22 08:47 libsherpa-onnx-c-api.so
drwxr-xr-x 2 runner docker 4.0K Jul 22 08:47 pkgconfig
total 4.0K
drwxr-xr-x 3 runner docker 4.0K Jul 22 08:47 sherpa-onnx
```

If you are using macOS, you should see the following content:

```
total 53976
-rw-r--r--  1 runner  staff    23M Jul 22 08:48 libonnxruntime.1.17.1.dylib
lrwxr-xr-x  1 runner  staff    27B Jul 22 08:48 libonnxruntime.dylib -> libonnxruntime.1.
↪17.1.dylib
-rwxr-xr-x  1 runner  staff   3.5M Jul 22 08:48 libsherpa-onnx-c-api.dylib
drwxr-xr-x  3 runner  staff    96B Jul 22 08:48 pkgconfig
total 0
drwxr-xr-x  3 runner  staff    96B Jul 22 08:48 sherpa-onnx
```

5. Build the C API example

```
cd ..

gcc -o sense-voice-c-api ./c-api-examples/sense-voice-c-api.c \
  -I ./build/install/include \
  -L ./build/install/lib/ \
  -l sherpa-onnx-c-api

ls -lh sense-voice-c-api
```

Note that:

- `-I ./build/install/include` is to add the directory `./build/install/include` to the header search path so that `#include "sherpa-onnx/c-api/c-api.h` won't throw an error.
- `-L ./build/install/lib/` is to add the directory `./build/install/lib` to the library search path so that it can find `-l sherpa-onnx-c-api`
- `-l sherpa-onnx-c-api` is to link the library `libsherpa-onnx-c-api.so` for Linux and `libsherpa-onnx-c-api.dylib` for macOS.

6. Run it

```
export LD_LIBRARY_PATH=$PWD/build/install/lib:$LD_LIBRARY_PATH
export DYLD_LIBRARY_PATH=$PWD/build/install/lib:$DYLD_LIBRARY_PATH

./sense-voice-c-api
```

Note that we have to use:

```
# For Linux
export LD_LIBRARY_PATH=$PWD/build/install/lib:$LD_LIBRARY_PATH
```

and:

```
# for macOS
export DYLD_LIBRARY_PATH=$PWD/build/install/lib:$DYLD_LIBRARY_PATH
```

Otherwise, it cannot find `libsherpa-onnx-c-api.so` for Linux and `libsherpa-onnx-c-api.dylib` at `runtime`.

7. Where to find sense-voice-c-api.c

You can find `sense-voice-c-api.c` at the following address:

https://github.com/k2-fsa/sherpa-onnx/blob/master/c-api-examples/sense-voice-c-api.c

8.24.5 Dart API for SenseVoice

This page describes how to use the Dart API to run SenseVoice models in sherpa-onnx

Note that we have published the package `sherpa_onnx` at https://pub.dev/packages/sherpa_onnx.

Fig. 8.31: Screenshot of sherpa-onnx on `pub.dev`.

Note that the package supports the following platforms:

- Android
- iOS
- Linux
- macOS
- Windows

In the following, we show how to use the pure Dart API to decode files with SenseVoice models.

```
cd /tmp

git clone http://github.com/k2-fsa/sherpa-onnx

cd sherpa-onnx
cd dart-api-examples
cd non-streaming-asr
dart pub get
./run-sense-voice.sh
```

You should see the following recognition result:

95

Explanations

1. Download the code

```
cd /tmp

git clone http://github.com/k2-fsa/sherpa-onnx
```

In this example, we download sherpa-onnx and place it inside the directory `/tmp/`. You can replace `/tmp/` with any directory you like.

2. Download the sherpa-onnx package

```
cd sherpa-onnx
cd dart-api-examples
cd non-streaming-asr
dart pub get
```

The command `dart pub get` will download the `sherpa_onnx` package automagically from `pub.dev`.

You should see something like below after running `dart pub get`:

```
(py38) fangjuns-MacBook-Pro:non-streaming-asr fangjun$ dart pub get
Resolving dependencies... (1.2s)
Downloading packages... (33.3s)
  collection 1.18.0 (1.19.0 available)
  lints 3.0.0 (4.0.0 available)
  material_color_utilities 0.8.0 (0.12.0 available)
```

(continues on next page)

(continued from previous page)

```
  meta 1.12.0 (1.15.0 available)
> sherpa_onnx 1.10.17 (was 1.9.29)
+ sherpa_onnx_android 1.10.17
+ sherpa_onnx_ios 1.10.17
+ sherpa_onnx_linux 1.10.17
+ sherpa_onnx_macos 1.10.17
+ sherpa_onnx_windows 1.10.17
Changed 6 dependencies!
4 packages have newer versions incompatible with dependency constraints.
Try `dart pub outdated` for more information.
```

3. Run it

```
./run-sense-voice.sh
```

The above script downloads models and run the code automatically.

You can find `run-sense-voice.sh` at the following address:

> https://github.com/k2-fsa/sherpa-onnx/blob/master/dart-api-examples/non-streaming-asr/run-sense-voice.sh

The Dart API example code can be found at:

> https://github.com/k2-fsa/sherpa-onnx/blob/master/dart-api-examples/non-streaming-asr/bin/sense-voice.dart

8.24.6 Python API for SenseVoice

This page describes how to use the Python API for SenseVoice.

Please refer to *Install the Python Package* for how to install the Python package of sherpa-onnx.

The following is a quick way to do that:

```
pip install sherpa-onnx
```

After installing the Python package, you can download the Python example code and run it with the following commands:

```
cd /tmp
git clone http://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx

curl -SL -O https://github.com/k2-fsa/sherpa-onnx/releases/download/asr-models/sherpa-
↪onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2
tar xvf sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2
rm sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17.tar.bz2

python3 ./python-api-examples/offline-sense-voice-ctc-decode-files.py
```

You should see something like below:

```
./sherpa-onnx-sense-voice-zh-en-ja-ko-yue-2024-07-17/test_wavs/zh.wav
{"text": "95", "timestamps": [0.72, 0.96, 1.26, 1.44, 1.92, 2.10, 2.58, 2.82, 3.30, 3.90,
↪ 4.20, 4.56, 4.74, 5.46], "tokens":["", "", "", "", "", "", "9", "", "", "", "", "5", "
↪", ""], "words": []}
(py38) fangjuns-MacBook-Pro:sherpa-onnx fangjun$ #python3 ./python-api-examples/offline-
↪sense-voice-ctc-decode-files.py
```

You can find `offline-sense-voice-ctc-decode-files.py` at the following address:

> https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/offline-sense-voice-ctc-decode-files.py

8.25 Speaker Identification

This page describes how to use sherpa-onnx for speaker identification.

Please first follow *Installation* and/or *Install the Python Package* to install sherpa-onnx before you continue.

Pre-trained models can be found at https://github.com/k2-fsa/sherpa-onnx/releases/tag/speaker-recongition-models

Hint: You can find Android APKs for each model at the following page

https://k2-fsa.github.io/sherpa/onnx/speaker-identification/apk.html

Please refer to https://github.com/k2-fsa/sherpa-onnx/tree/master/python-api-examples for usage examples.

8.26 Text-to-speech (TTS)

This page describes how to use sherpa-onnx for text-to-speech (TTS).

Please first follow *Installation* and/or *Install the Python Package* to install sherpa-onnx before you continue.

8.26.1 Huggingface space

We provide a huggingface space where you can try text-to-speech with sherpa-onnx from within your browser without installing anything.

Hint: We also have spaces using WebAssembly for text-to-speech. Please see *Huggingface Spaces (WebAssembly)*.

All you need is a browser, either running on your desk computer, your phone, or your iPad, etc.

Please visit

> https://huggingface.co/spaces/k2-fsa/text-to-speech

8.26.2 Pre-trained models

This page list pre-trained models for text-to-speech.

Hint: Please install git-lfs before you continue.

Otherwise, you will be SAD later.

vits

This page lists pre-trained vits models.

All models in a single table

The following table summarizes the information of all models in this page.

Note: Since there are more than `100` pre-trained models for over `40` languages, we don't list all of them on this page. Please find them at https://github.com/k2-fsa/sherpa-onnx/releases/tag/tts-models.

You can try all the models at the following huggingface space. https://huggingface.co/spaces/k2-fsa/text-to-speech.

Hint: You can find Android APKs for each model at the following page

https://k2-fsa.github.io/sherpa/onnx/tts/apk.html

Model	Language	# Speakers	Dataset	Model filesize (MB)	Sample rate (Hz)
csukuangfj/vits-zh-hf-fanchen-C (Chinese, 187 speakers)	Chinese	187	N/A	116	16000
csukuangfj/vits-zh-hf-fanchen-wnj (Chinese, 1 male)	Chinese	1	N/A	116	16000
csukuangfj/vits-zh-hf-theresa (Chinese, 804 speakers)	Chinese	804	N/A	117	22050
csukuangfj/vits-zh-hf-eula (Chinese, 804 speakers)	Chinese	804	N/A	117	22050
aishell3 (Chinese, multi-speaker, 174 speakers)	Chinese	174	aishell3	116	8000
ljspeech (English, single-speaker)	English (US)	1 (Female)	LJ Speech	109	22050
VCTK (English, multi-speaker, 109 speakers)	English	109	VCTK	116	22050
en_US-lessac-medium (English, single-speaker)	English (US)	1 (Male)	lessac_blizzard2013	61	22050

ljspeech (English, single-speaker)

This model is converted from pretrained_ljspeech.pth, which is trained by the vits author Jaehyeon Kim on the LJ Speech dataset. It supports only English and is a single-speaker model.

Note: If you are interested in how the model is converted, please see https://github.com/k2-fsa/sherpa-onnx/blob/master/scripts/vits/export-onnx-ljs.py

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/tts-models/vits-ljs.tar.bz2
tar xvf vits-ljs.tar.bz2
rm vits-ljs.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
-rw-r--r-- 1 1001 127 109M Apr 22 02:38 vits-ljs/vits-ljs.onnx
```

Generate speech with executable compiled from C++

```
 cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline-tts \
  --vits-model=./vits-ljs/vits-ljs.onnx \
  --vits-lexicon=./vits-ljs/lexicon.txt \
  --vits-tokens=./vits-ljs/tokens.txt \
  --output-filename=./liliana.wav \
  'liliana, the most beautiful and lovely assistant of our team!'
```

After running, it will generate a file `liliana.wav` in the current directory.

```
soxi ./liliana.wav

Input File     : './liliana.wav'
Channels       : 1
Sample Rate    : 22050
Precision      : 16-bit
Duration       : 00:00:04.39 = 96768 samples ~ 329.143 CDDA sectors
File Size      : 194k
Bit Rate       : 353k
Sample Encoding: 16-bit Signed Integer PCM
```

Generate speech with Python script

```
cd /path/to/sherpa-onnx

python3 ./python-api-examples/offline-tts.py \
 --vits-model=./vits-ljs/vits-ljs.onnx \
 --vits-lexicon=./vits-ljs/lexicon.txt \
 --vits-tokens=./vits-ljs/tokens.txt \
 --output-filename=./armstrong.wav \
 "That's one small step for a man, a giant leap for mankind."
```

After running, it will generate a file `armstrong.wav` in the current directory.

```
soxi ./armstrong.wav

Input File     : './armstrong.wav'
Channels       : 1
Sample Rate    : 22050
Precision      : 16-bit
Duration       : 00:00:04.81 = 105984 samples ~ 360.49 CDDA sectors
File Size      : 212k
Bit Rate       : 353k
Sample Encoding: 16-bit Signed Integer PCM
```

VCTK (English, multi-speaker, 109 speakers)

This model is converted from pretrained_vctk.pth, which is trained by the vits author Jaehyeon Kim on the VCTK dataset. It supports only English and is a multi-speaker model. It contains 109 speakers.

Note: If you are interested in how the model is converted, please see https://github.com/k2-fsa/sherpa-onnx/blob/master/scripts/vits/export-onnx-vctk.py

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/tts-models/vits-vctk.tar.bz2
tar xvf vits-vctk.tar.bz2
rm vits-vctk.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
vits-vctk fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff    37M Oct 16 10:57 vits-vctk.int8.onnx
-rw-r--r--  1 fangjun  staff   116M Oct 16 10:57 vits-vctk.onnx
```

Generate speech with executable compiled from C++

Since there are 109 speakers available, we can choose a speaker from 0 to 198. The default speaker ID is 0.

We use speaker ID 0, 10, and 108 below to generate audio for the same text.

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline-tts \
  --vits-model=./vits-vctk/vits-vctk.onnx \
  --vits-lexicon=./vits-vctk/lexicon.txt \
  --vits-tokens=./vits-vctk/tokens.txt \
```

(continues on next page)

(continued from previous page)

```
  --sid=0 \
  --output-filename=./kennedy-0.wav \
  'Ask not what your country can do for you; ask what you can do for your country.'

./build/bin/sherpa-onnx-offline-tts \
  --vits-model=./vits-vctk/vits-vctk.onnx \
  --vits-lexicon=./vits-vctk/lexicon.txt \
  --vits-tokens=./vits-vctk/tokens.txt \
  --sid=10 \
  --output-filename=./kennedy-10.wav \
  'Ask not what your country can do for you; ask what you can do for your country.'

./build/bin/sherpa-onnx-offline-tts \
  --vits-model=./vits-vctk/vits-vctk.onnx \
  --vits-lexicon=./vits-vctk/lexicon.txt \
  --vits-tokens=./vits-vctk/tokens.txt \
  --sid=108 \
  --output-filename=./kennedy-108.wav \
  'Ask not what your country can do for you; ask what you can do for your country.'
```

It will generate 3 files: `kennedy-0.wav`, `kennedy-10.wav`, and `kennedy-108.wav`.

Generate speech with Python script

We use speaker ID 30, 66, and 99 below to generate audio for different transcripts.

```
cd /path/to/sherpa-onnx

python3 ./python-api-examples/offline-tts.py \
 --vits-model=./vits-vctk/vits-vctk.onnx \
 --vits-lexicon=./vits-vctk/lexicon.txt \
 --vits-tokens=./vits-vctk/tokens.txt \
 --sid=30 \
 --output-filename=./einstein-30.wav \
 "Life is like riding a bicycle. To keep your balance, you must keep moving."

python3 ./python-api-examples/offline-tts.py \
 --vits-model=./vits-vctk/vits-vctk.onnx \
 --vits-lexicon=./vits-vctk/lexicon.txt \
 --vits-tokens=./vits-vctk/tokens.txt \
 --sid=66 \
 --output-filename=./franklin-66.wav \
 "Three can keep a secret, if two of them are dead."

python3 ./python-api-examples/offline-tts.py \
 --vits-model=./vits-vctk/vits-vctk.onnx \
 --vits-lexicon=./vits-vctk/lexicon.txt \
 --vits-tokens=./vits-vctk/tokens.txt \
 --sid=99 \
 --output-filename=./martin-99.wav \
 "Darkness cannot drive out darkness: only light can do that. Hate cannot drive out␣
↪hate: only love can do that"
```

(continues on next page)

(continued from previous page)

It will generate 3 files: `einstein-30.wav`, `franklin-66.wav`, and `martin-99.wav`.

csukuangfj/vits-zh-hf-fanchen-C (Chinese, 187 speakers)

You can download the model using the following commands:

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/tts-models/vits-zh-hf-
↪fanchen-C.tar.bz2
tar xvf vits-zh-hf-fanchen-C.tar.bz2
rm vits-zh-hf-fanchen-C.tar.bz2
```

Hint: This model is converted from https://huggingface.co/spaces/lkz99/tts_model/tree/main/zh

```
# information about model files

total 291M
-rw-r--r-- 1 1001 127  58K Apr 21 05:40 date.fst
drwxr-xr-x 3 1001 127 4.0K Apr 19 12:42 dict
-rwxr-xr-x 1 1001 127 4.0K Apr 21 05:40 export-onnx-zh-hf-fanchen-models.py
-rwxr-xr-x 1 1001 127 2.5K Apr 21 05:40 generate-lexicon-zh-hf-fanchen-models.py
-rw-r--r-- 1 1001 127 2.4M Apr 21 05:40 lexicon.txt
-rw-r--r-- 1 1001 127  22K Apr 21 05:40 new_heteronym.fst
-rw-r--r-- 1 1001 127  63K Apr 21 05:40 number.fst
-rw-r--r-- 1 1001 127  87K Apr 21 05:40 phone.fst
-rw-r--r-- 1 1001 127 173M Apr 21 05:40 rule.far
-rw-r--r-- 1 1001 127  331 Apr 21 05:40 tokens.txt
-rw-r--r-- 1 1001 127 116M Apr 21 05:40 vits-zh-hf-fanchen-C.onnx
-rwxr-xr-x 1 1001 127 2.0K Apr 21 05:40 vits-zh-hf-fanchen-models.sh
```

usage:

```
sherpa-onnx-offline-tts \
  --vits-model=./vits-zh-hf-fanchen-C/vits-zh-hf-fanchen-C.onnx \
  --vits-dict-dir=./vits-zh-hf-fanchen-C/dict \
  --vits-lexicon=./vits-zh-hf-fanchen-C/lexicon.txt \
  --vits-tokens=./vits-zh-hf-fanchen-C/tokens.txt \
  --vits-length-scale=0.5 \
  --output-filename="./value-2x.wav" \
  ""


sherpa-onnx-offline-tts \
  --vits-model=./vits-zh-hf-fanchen-C/vits-zh-hf-fanchen-C.onnx \
  --vits-dict-dir=./vits-zh-hf-fanchen-C/dict \
  --vits-lexicon=./vits-zh-hf-fanchen-C/lexicon.txt \
  --vits-tokens=./vits-zh-hf-fanchen-C/tokens.txt \
  --vits-length-scale=1.0 \
  --tts-rule-fsts=./vits-zh-hf-fanchen-C/number.fst \
```

(continues on next page)

(continued from previous page)

```
  --output-filename="./numbers.wav" \
  "14"

sherpa-onnx-offline-tts \
  --sid=100 \
  --vits-model=./vits-zh-hf-fanchen-C/vits-zh-hf-fanchen-C.onnx \
  --vits-dict-dir=./vits-zh-hf-fanchen-C/dict \
  --vits-lexicon=./vits-zh-hf-fanchen-C/lexicon.txt \
  --vits-tokens=./vits-zh-hf-fanchen-C/tokens.txt \
  --vits-length-scale=1.0 \
  --tts-rule-fsts=./vits-zh-hf-fanchen-C/phone.fst,./vits-zh-hf-fanchen-C/number.fst \
  --output-filename="./numbers-100.wav" \
  "110 18601200909"

sherpa-onnx-offline-tts \
  --sid=14 \
  --vits-model=./vits-zh-hf-fanchen-C/vits-zh-hf-fanchen-C.onnx \
  --vits-dict-dir=./vits-zh-hf-fanchen-C/dict \
  --vits-lexicon=./vits-zh-hf-fanchen-C/lexicon.txt \
  --vits-tokens=./vits-zh-hf-fanchen-C/tokens.txt \
  --vits-length-scale=1.0 \
  --output-filename="./wo-mi-14.wav" \
  ""

sherpa-onnx-offline-tts \
  --sid=102 \
  --vits-model=./vits-zh-hf-fanchen-C/vits-zh-hf-fanchen-C.onnx \
  --vits-dict-dir=./vits-zh-hf-fanchen-C/dict \
  --vits-lexicon=./vits-zh-hf-fanchen-C/lexicon.txt \
  --vits-tokens=./vits-zh-hf-fanchen-C/tokens.txt \
  --tts-rule-fsts=./vits-zh-hf-fanchen-C/number.fst \
  --vits-length-scale=1.0 \
  --output-filename="./heteronym-102.wav" \
  "35, 91"
```

csukuangfj/vits-zh-hf-fanchen-wnj (Chinese, 1 male)

You can download the model using the following commands:

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/tts-models/vits-zh-hf-
↪fanchen-wnj.tar.bz2
tar xvf vits-zh-hf-fanchen-wnj.tar.bz2
rm vits-zh-hf-fanchen-wnj.tar.bz2
```

Hint: This model is converted from https://huggingface.co/spaces/lkz99/tts_model/blob/main/G_wnj_latest.pth

```
# information about model files
total 594760
-rw-r--r--  1 fangjun  staff    58K Apr 21 13:40 date.fst
```

(continues on next page)

(continued from previous page)

```
drwxr-xr-x  9 fangjun  staff   288B Apr 19 20:42 dict
-rwxr-xr-x  1 fangjun  staff   3.9K Apr 21 13:40 export-onnx-zh-hf-fanchen-models.py
-rwxr-xr-x  1 fangjun  staff   2.4K Apr 21 13:40 generate-lexicon-zh-hf-fanchen-models.py
-rw-r--r--  1 fangjun  staff   2.3M Apr 21 13:40 lexicon.txt
-rw-r--r--  1 fangjun  staff    21K Apr 21 13:40 new_heteronym.fst
-rw-r--r--  1 fangjun  staff    63K Apr 21 13:40 number.fst
-rw-r--r--  1 fangjun  staff    87K Apr 21 13:40 phone.fst
-rw-r--r--  1 fangjun  staff   172M Apr 21 13:40 rule.far
-rw-r--r--  1 fangjun  staff   331B Apr 21 13:40 tokens.txt
-rwxr-xr-x  1 fangjun  staff   1.9K Apr 21 13:40 vits-zh-hf-fanchen-models.sh
-rw-r--r--  1 fangjun  staff   115M Apr 21 13:40 vits-zh-hf-fanchen-wnj.onnx
```

usage:

```
sherpa-onnx-offline-tts \
  --vits-model=./vits-zh-hf-fanchen-wnj/vits-zh-hf-fanchen-wnj.onnx \
  --vits-dict-dir=./vits-zh-hf-fanchen-wnj/dict \
  --vits-lexicon=./vits-zh-hf-fanchen-wnj/lexicon.txt \
  --vits-tokens=./vits-zh-hf-fanchen-wnj/tokens.txt \
  --output-filename="./kuayue.wav" \
  ""

sherpa-onnx-offline-tts \
  --vits-model=./vits-zh-hf-fanchen-wnj/vits-zh-hf-fanchen-wnj.onnx \
  --vits-dict-dir=./vits-zh-hf-fanchen-wnj/dict \
  --vits-lexicon=./vits-zh-hf-fanchen-wnj/lexicon.txt \
  --vits-tokens=./vits-zh-hf-fanchen-wnj/tokens.txt \
  --tts-rule-fsts=./vits-zh-hf-fanchen-wnj/number.fst \
  --output-filename="./os.wav" \
  "14"
```

csukuangfj/vits-zh-hf-theresa (Chinese, 804 speakers)

You can download the model with the following commands:

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/tts-models/vits-zh-hf-
↪theresa.tar.bz2
tar xvf vits-zh-hf-theresa.tar.bz2
rm vits-zh-hf-theresa.tar.bz2
```

Hint: This model is converted from https://huggingface.co/spaces/zomehwh/vits-models-genshin-bh3/tree/main/pretrained_models/theresa

```
# information about model files

total 596992
-rw-r--r--  1 fangjun  staff    58K Apr 21 13:39 date.fst
drwxr-xr-x  9 fangjun  staff   288B Apr 19 20:42 dict
-rw-r--r--  1 fangjun  staff   2.6M Apr 21 13:39 lexicon.txt
```

(continues on next page)

(continued from previous page)

```
-rw-r--r--  1 fangjun  staff    21K Apr 21 13:39 new_heteronym.fst
-rw-r--r--  1 fangjun  staff    63K Apr 21 13:39 number.fst
-rw-r--r--  1 fangjun  staff    87K Apr 21 13:39 phone.fst
-rw-r--r--  1 fangjun  staff   172M Apr 21 13:39 rule.far
-rw-r--r--  1 fangjun  staff   116M Apr 21 13:39 theresa.onnx
-rw-r--r--  1 fangjun  staff   268B Apr 21 13:39 tokens.txt
-rwxr-xr-x  1 fangjun  staff   5.3K Apr 21 13:39 vits-zh-hf-models.py
-rwxr-xr-x  1 fangjun  staff   571B Apr 21 13:39 vits-zh-hf-models.sh
```

usage:

```
sherpa-onnx-offline-tts \
  --vits-model=./vits-zh-hf-theresa/theresa.onnx \
  --vits-dict-dir=./vits-zh-hf-theresa/dict \
  --vits-lexicon=./vits-zh-hf-theresa/lexicon.txt \
  --vits-tokens=./vits-zh-hf-theresa/tokens.txt \
  --sid=0 \
  --output-filename="./reai-0.wav" \
  ""

sherpa-onnx-offline-tts \
  --vits-model=./vits-zh-hf-theresa/theresa.onnx \
  --vits-dict-dir=./vits-zh-hf-theresa/dict \
  --vits-lexicon=./vits-zh-hf-theresa/lexicon.txt \
  --vits-tokens=./vits-zh-hf-theresa/tokens.txt \
  --tts-rule-fsts=./vits-zh-hf-theresa/number.fst \
  --debug=1 \
  --sid=88 \
  --output-filename="./mi14-88.wav" \
  "141000000"
```

csukuangfj/vits-zh-hf-eula (Chinese, 804 speakers)

You can download the model using the following commands:

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/tts-models/vits-zh-hf-eula.
↪tar.bz2
tar xvf vits-zh-hf-eula.tar.bz2
rm vits-zh-hf-eula.tar.bz2
```

Hint: This model is converted from https://huggingface.co/spaces/zomehwh/vits-models-genshin-bh3/tree/main/pretrained_models/eula

```
# information about model files

total 596992
-rw-r--r--  1 fangjun  staff    58K Apr 21 13:39 date.fst
drwxr-xr-x  9 fangjun  staff   288B Apr 19 20:42 dict
-rw-r--r--  1 fangjun  staff   116M Apr 21 13:39 eula.onnx
```

(continues on next page)

(continued from previous page)

```
-rw-r--r--  1 fangjun  staff   2.6M Apr 21 13:39 lexicon.txt
-rw-r--r--  1 fangjun  staff    21K Apr 21 13:39 new_heteronym.fst
-rw-r--r--  1 fangjun  staff    63K Apr 21 13:39 number.fst
-rw-r--r--  1 fangjun  staff    87K Apr 21 13:39 phone.fst
-rw-r--r--  1 fangjun  staff   172M Apr 21 13:39 rule.far
-rw-r--r--  1 fangjun  staff   268B Apr 21 13:39 tokens.txt
-rwxr-xr-x  1 fangjun  staff   5.3K Apr 21 13:39 vits-zh-hf-models.py
-rwxr-xr-x  1 fangjun  staff   571B Apr 21 13:39 vits-zh-hf-models.sh
```

usage:

```
sherpa-onnx-offline-tts \
  --vits-model=./vits-zh-hf-eula/eula.onnx \
  --vits-dict-dir=./vits-zh-hf-eula/dict \
  --vits-lexicon=./vits-zh-hf-eula/lexicon.txt \
  --vits-tokens=./vits-zh-hf-eula/tokens.txt \
  --debug=1 \
  --sid=666 \
  --output-filename="./news-666.wav" \
  ""

sherpa-onnx-offline-tts \
  --vits-model=./vits-zh-hf-eula/eula.onnx \
  --vits-dict-dir=./vits-zh-hf-eula/dict \
  --vits-lexicon=./vits-zh-hf-eula/lexicon.txt \
  --vits-tokens=./vits-zh-hf-eula/tokens.txt \
  --tts-rule-fsts=./vits-zh-hf-eula/number.fst \
  --sid=99 \
  --output-filename="./news-99.wav" \
  "925"
```

aishell3 (Chinese, multi-speaker, 174 speakers)

This model is trained on the aishell3 dataset using icefall.

It supports only Chinese and it's a multi-speaker model and contains 174 speakers.

Hint: You can download the Android APK for this model at

https://k2-fsa.github.io/sherpa/onnx/tts/apk-engine.html

(Please search for `vits-icefall-zh-aishell3` in the above Android APK page)

Note: If you are interested in how the model is converted, please see the documentation of icefall.

If you are interested in training your own model, please also refer to icefall.

icefall is also developed by us.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/tts-models/vits-icefall-zh-
↪aishell3.tar.bz2
tar xvf vits-icefall-zh-aishell3.tar.bz2
rm vits-icefall-zh-aishell3.tar.bz2
```

Please check that the file sizes of the pre-trained models are correct. See the file sizes of `*.onnx` files below.

```
vits-icefall-zh-aishell3 fangjun$ ls -lh *.onnx
-rw-r--r--  1 fangjun  staff    29M Mar 20 22:50 model.onnx
```

Generate speech with executable compiled from C++

Since there are 174 speakers available, we can choose a speaker from 0 to 173. The default speaker ID is 0.

We use speaker ID 10, 33, and 99 below to generate audio for the same text.

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline-tts \
  --vits-model=./vits-icefall-zh-aishell3/model.onnx \
  --vits-lexicon=./vits-icefall-zh-aishell3/lexicon.txt \
  --vits-tokens=./vits-icefall-zh-aishell3/tokens.txt \
  --tts-rule-fsts=./vits-icefall-zh-aishell3/phone.fst,./vits-icefall-zh-aishell3/date.
↪fst,./vits-icefall-zh-aishell3/number.fst \
  --sid=10 \
  --output-filename=./liliana-10.wav \
  ""

./build/bin/sherpa-onnx-offline-tts \
  --vits-model=./vits-icefall-zh-aishell3/model.onnx \
  --vits-lexicon=./vits-icefall-zh-aishell3/lexicon.txt \
  --vits-tokens=./vits-icefall-zh-aishell3/tokens.txt \
  --tts-rule-fsts=./vits-icefall-zh-aishell3/phone.fst,./vits-icefall-zh-aishell3/date.
↪fst,./vits-icefall-zh-aishell3/number.fst \
  --sid=33 \
  --output-filename=./liliana-33.wav \
  ""

./build/bin/sherpa-onnx-offline-tts \
  --vits-model=./vits-icefall-zh-aishell3/model.onnx \
  --vits-lexicon=./vits-icefall-zh-aishell3/lexicon.txt \
  --vits-tokens=./vits-icefall-zh-aishell3/tokens.txt \
  --tts-rule-fsts=./vits-icefall-zh-aishell3/phone.fst,./vits-icefall-zh-aishell3/date.
↪fst,./vits-icefall-zh-aishell3/number.fst \
  --sid=99 \
  --output-filename=./liliana-99.wav \
  ""
```

It will generate 3 files: `liliana-10.wav`, `liliana-33.wav`, and `liliana-99.wav`.

We also support rule-based text normalization, which is implemented with OpenFst. Currently, only number normalization is supported.

Hint: We will support other normalization rules later.

The following is an example:

```
./build/bin/sherpa-onnx-offline-tts \
  --vits-model=./vits-icefall-zh-aishell3/model.onnx \
  --vits-lexicon=./vits-icefall-zh-aishell3/lexicon.txt \
  --vits-tokens=./vits-icefall-zh-aishell3/tokens.txt \
  --tts-rule-fsts=./vits-icefall-zh-aishell3/phone.fst,./vits-icefall-zh-aishell3/date.
↪fst,./vits-icefall-zh-aishell3/number.fst \
  --sid=66 \
  --output-filename=./rule-66.wav \
  "35, 91"
```

Generate speech with Python script

We use speaker ID 21, 41, and 45 below to generate audio for different transcripts.

```
cd /path/to/sherpa-onnx

python3 ./python-api-examples/offline-tts.py \
 --vits-model=./vits-icefall-zh-aishell3/model.onnx \
 --vits-lexicon=./vits-icefall-zh-aishell3/lexicon.txt \
 --vits-tokens=./vits-icefall-zh-aishell3/tokens.txt \
 --tts-rule-fsts=./vits-icefall-zh-aishell3/phone.fst,./vits-icefall-zh-aishell3/date.
↪fst,./vits-icefall-zh-aishell3/number.fst \
 --sid=21 \
 --output-filename=./liubei-21.wav \
 ""

python3 ./python-api-examples/offline-tts.py \
 --vits-model=./vits-icefall-zh-aishell3/model.onnx \
 --vits-lexicon=./vits-icefall-zh-aishell3/lexicon.txt \
 --vits-tokens=./vits-icefall-zh-aishell3/tokens.txt \
 --tts-rule-fsts=./vits-icefall-zh-aishell3/phone.fst,./vits-icefall-zh-aishell3/date.
↪fst,./vits-icefall-zh-aishell3/number.fst \
 --sid=41 \
 --output-filename=./demokelite-41.wav \
 ""

python3 ./python-api-examples/offline-tts.py \
 --vits-model=./vits-icefall-zh-aishell3/model.onnx \
 --vits-lexicon=./vits-icefall-zh-aishell3/lexicon.txt \
 --vits-tokens=./vits-icefall-zh-aishell3/tokens.txt \
 --tts-rule-fsts=./vits-icefall-zh-aishell3/phone.fst,./vits-icefall-zh-aishell3/date.
↪fst,./vits-icefall-zh-aishell3/number.fst \
 --sid=45 \
```

(continues on next page)

(continued from previous page)

```
 --output-filename=./zhugeliang-45.wav \
 """
```

It will generate 3 files: `liubei-21.wav`, `demokelite-41.wav`, and `zhugeliang-45.wav`.

The Python script also supports rule-based text normalization.

```
python3 ./python-api-examples/offline-tts.py \
 --vits-model=./vits-icefall-zh-aishell3/model.onnx \
 --vits-lexicon=./vits-icefall-zh-aishell3/lexicon.txt \
 --vits-tokens=./vits-icefall-zh-aishell3/tokens.txt \
 --tts-rule-fsts=./vits-icefall-zh-aishell3/phone.fst,./vits-icefall-zh-aishell3/date.
↪fst,./vits-icefall-zh-aishell3/number.fst \
 --sid=103 \
 --output-filename=./rule-103.wav \
 "71443497378311411778724110138123456 78"
```

en_US-lessac-medium (English, single-speaker)

This model is converted from https://huggingface.co/rhasspy/piper-voices/tree/main/en/en_US/lessac/medium.

The dataset used to train the model is lessac_blizzard2013.

Hint: The model is from piper.

In the following, we describe how to download it and use it with sherpa-onnx.

Download the model

Please use the following commands to download it.

```
cd /path/to/sherpa-onnx

wget https://github.com/k2-fsa/sherpa-onnx/releases/download/tts-models/vits-piper-en_US-
↪lessac-medium.tar.bz2
tar xf vits-piper-en_US-lessac-medium.tar.bz2
```

Hint: You can find a lot of pre-trained models for over 40 languages at <*https://github.com/k2-fsa/sherpa-onnx/releases/tag/tts-models*>.

Generate speech with executable compiled from C++

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline-tts \
  --vits-model=./vits-piper-en_US-lessac-medium/en_US-lessac-medium.onnx \
  --vits-data-dir=./vits-piper-en_US-lessac-medium/espeak-ng-data \
  --vits-tokens=./vits-piper-en_US-lessac-medium/tokens.txt \
  --output-filename=./liliana-piper-en_US-lessac-medium.wav \
  'liliana, the most beautiful and lovely assistant of our team!'
```

Hint: You can also use

```
cd /path/to/sherpa-onnx

./build/bin/sherpa-onnx-offline-tts-play \
  --vits-model=./vits-piper-en_US-lessac-medium/en_US-lessac-medium.onnx \
  --vits-data-dir=./vits-piper-en_US-lessac-medium/espeak-ng-data \
  --vits-tokens=./vits-piper-en_US-lessac-medium/tokens.txt \
  --output-filename=./liliana-piper-en_US-lessac-medium.wav \
  'liliana, the most beautiful and lovely assistant of our team!'
```

which will play the audio as it is generating.

After running, it will generate a file `liliana-piper.wav` in the current directory.

```
soxi ./liliana-piper-en_US-lessac-medium.wav

Input File     : './liliana-piper-en_US-lessac-medium.wav'
Channels       : 1
Sample Rate    : 22050
Precision      : 16-bit
Duration       : 00:00:03.48 = 76800 samples ~ 261.224 CDDA sectors
File Size      : 154k
Bit Rate       : 353k
Sample Encoding: 16-bit Signed Integer PCM
```

Generate speech with Python script

```
cd /path/to/sherpa-onnx

python3 ./python-api-examples/offline-tts.py \
 --vits-model=./vits-piper-en_US-lessac-medium/en_US-lessac-medium.onnx \
 --vits-data-dir=./vits-piper-en_US-lessac-medium/espeak-ng-data \
 --vits-tokens=./vits-piper-en_US-lessac-medium/tokens.txt \
 --output-filename=./armstrong-piper-en_US-lessac-medium.wav \
 "That's one small step for a man, a giant leap for mankind."
```

Hint: You can also use

```
cd /path/to/sherpa-onnx

python3 ./python-api-examples/offline-tts-play.py \
  --vits-model=./vits-piper-en_US-lessac-medium/en_US-lessac-medium.onnx \
  --vits-data-dir=./vits-piper-en_US-lessac-medium/espeak-ng-data \
  --vits-tokens=./vits-piper-en_US-lessac-medium/tokens.txt \
  --output-filename=./armstrong-piper-en_US-lessac-medium.wav \
  "That's one small step for a man, a giant leap for mankind."
```

which will play the audio as it is generating.

After running, it will generate a file `armstrong-piper-en_US-lessac-medium.wav` in the current directory.

```
soxi ./armstrong-piper-en_US-lessac-medium.wav

Input File     : './armstrong-piper-en_US-lessac-medium.wav'
Channels       : 1
Sample Rate    : 22050
Precision      : 16-bit
Duration       : 00:00:03.74 = 82432 samples ~ 280.381 CDDA sectors
File Size      : 165k
Bit Rate       : 353k
Sample Encoding: 16-bit Signed Integer PCM
```

8.26.3 WebAssembly

In this section, we describe how to build text-to-speech from sherpa-onnx for WebAssembly so that you can run text-to-speech with WebAssembly.

Please follow the steps below to build and run sherpa-onnx for WebAssembly.

Hint: We provide a colab notebook for you to try this section step by step.

If you are using Windows or you don't want to setup your local environment to build WebAssembly support, please use the above colab notebook.

Install Emscripten

We need to compile the C/C++ files in sherpa-onnx with the help of emscripten.

Please refer to https://emscripten.org/docs/getting_started/downloads for detailed installation instructions.

The following is an example to show you how to install it on Linux/macOS.

```
git clone https://github.com/emscripten-core/emsdk.git
cd emsdk
git pull
./emsdk install latest
./emsdk activate latest
source ./emsdk_env.sh
```

To check that you have installed emscripten successfully, please run:

```
emcc -v
```

The above command should print something like below:

```
emcc (Emscripten gcc/clang-like replacement + linker emulating GNU ld) 3.1.48␣
↪(e967e20b4727956a30592165a3c1cde5c67fa0a8)
shared:INFO: (Emscripten: Running sanity checks)
(py38) fangjuns-MacBook-Pro:open-source fangjun$ emcc -v
emcc (Emscripten gcc/clang-like replacement + linker emulating GNU ld) 3.1.48␣
↪(e967e20b4727956a30592165a3c1cde5c67fa0a8)
clang version 18.0.0 (https://github.com/llvm/llvm-project␣
↪a54545ba6514802178cf7cf1c1dd9f7efbf3cde7)
Target: wasm32-unknown-emscripten
Thread model: posix
InstalledDir: /Users/fangjun/open-source/emsdk/upstream/bin
```

Congratulations! You have successfully installed emscripten.

Build

After installing emscripten, we can build text-to-speech from sherpa-onnx for WebAssembly now.

Please use the following command to build it:

```
git clone https://github.com/k2-fsa/sherpa-onnx
cd sherpa-onnx

cd wasm/tts/assets

wget -q https://github.com/k2-fsa/sherpa-onnx/releases/download/tts-models/vits-piper-en_
↪US-libritts_r-medium.tar.bz2
tar xf vits-piper-en_US-libritts_r-medium.tar.bz2
rm vits-piper-en_US-libritts_r-medium.tar.bz2
mv vits-piper-en_US-libritts_r-medium/en_US-libritts_r-medium.onnx ./model.onnx
mv vits-piper-en_US-libritts_r-medium/tokens.txt ./
mv vits-piper-en_US-libritts_r-medium/espeak-ng-data ./
rm -rf vits-piper-en_US-libritts_r-medium

cd ../../..

./build-wasm-simd-tts.sh
```

Hint: You can visit https://github.com/k2-fsa/sherpa-onnx/releases/tag/tts-models to download a different model.

After building, you should see the following output:

```
Install the project...
-- Install configuration: "Release"
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/lib/
↪libkaldi-native-fbank-core.a
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/lib/
↪libkaldi-decoder-core.a
```

(continues on next page)

(continued from previous page)

```
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/lib/
↪libsherpa-onnx-kaldifst-core.a
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/lib/
↪libsherpa-onnx-fst.a
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/lib/
↪libonnxruntime.a
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/lib/
↪libespeak-ng.a
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/lib/
↪libucd.a
-- Up-to-date: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/lib/
↪libucd.a
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/lib/
↪libpiper_phonemize.a
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/./
↪sherpa-onnx.pc
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/lib/
↪pkgconfig/espeak-ng.pc
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/share/
↪vim/vimfiles/ftdetect
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/share/
↪vim/vimfiles/ftdetect/espeakfiletype.vim
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/share/
↪vim/vimfiles/syntax
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/share/
↪vim/vimfiles/syntax/espeakrules.vim
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/share/
↪vim/vimfiles/syntax/espeaklist.vim
-- Up-to-date: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/lib/
↪libucd.a
-- Up-to-date: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/lib/
↪libespeak-ng.a
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/lib/
↪libsherpa-onnx-core.a
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/lib/
↪libsherpa-onnx-c-api.a
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/
↪include/sherpa-onnx/c-api/c-api.h
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/bin/
↪wasm/tts/sherpa-onnx-wasm-main.js
-- Up-to-date: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/bin/
↪wasm/tts/sherpa-onnx-wasm-main.js
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/bin/
↪wasm/tts/index.html
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/bin/
↪wasm/tts/sherpa-onnx.js
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/bin/
↪wasm/tts/app.js
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/bin/
↪wasm/tts/sherpa-onnx-wasm-main.wasm
-- Installing: /Users/fangjun/open-source/sherpa-onnx/build-wasm-simd-tts/install/bin/
↪wasm/tts/sherpa-onnx-wasm-main.data
```

(continues on next page)

(continued from previous page)

```
+ ls -lh install/bin/wasm/tts
total 211248
-rw-r--r--  1 fangjun  staff   5.3K Feb 22 09:18 app.js
-rw-r--r--  1 fangjun  staff   1.3K Feb 22 09:18 index.html
-rw-r--r--  1 fangjun  staff    92M Feb 22 10:35 sherpa-onnx-wasm-main.data
-rw-r--r--  1 fangjun  staff   117K Feb 22 10:39 sherpa-onnx-wasm-main.js
-rw-r--r--  1 fangjun  staff    11M Feb 22 10:39 sherpa-onnx-wasm-main.wasm
-rw-r--r--  1 fangjun  staff   4.5K Feb 22 09:18 sherpa-onnx.js
```

Now you can use the following command to run it:

```
cd build-wasm-simd-tts/install/bin/wasm/tts
python3 -m http.server 6008
```

Start your browser and visit http://localhost:6008/; you should see the following page:

Now you can enter some text and click `Generate`

A screenshot is given below:

Congratulations! You have successfully run text-to-speech with WebAssembly in your browser.

Huggingface Spaces (WebAssembly)

We provide two Huggingface spaces so that you can try text-to-speech with WebAssembly in your browser.

English TTS

https://huggingface.co/spaces/k2-fsa/web-assembly-tts-sherpa-onnx-en

Hint: If you don't have access to Huggingface, please visit the following mirror:

https://modelscope.cn/studios/k2-fsa/web-assembly-tts-sherpa-onnx-en/summary

Next-gen Kaldi WebAssembly
localhost:6008
Next-gen Kaldi + WebAssembly
Text-to-speech Demo with sherpa-onnx
Initialized! Please enter text and click the Generate button.
Speaker ID (0 - 903): 0
Speed: 1
this is a text-to-speech application running with WebAssembly.
Generate
0:00 / 0:03
1 this is a text-to-speech application running with WebAssembly. ...
Delete

huggingface.co/spaces/k2-fsa/web-assembly-tts-sherpa-onnx-en
Spaces
k2-fsa / web-assembly-tts-sherpa-onnx-en
like 0
Running
App
Files
Community
Settings
Next-gen Kaldi + WebAssembly
Text-to-speech Demo with sherpa-onnx
Loading model ... ...
Speaker ID: 0
Speed: 1
Please enter your text here and click the Generate button
Generate

Note: The script for building this space can be found at https://github.com/k2-fsa/sherpa-onnx/blob/master/.github/workflows/wasm-simd-hf-space-en-tts.yaml

German TTS

https://huggingface.co/spaces/k2-fsa/web-assembly-tts-sherpa-onnx-de

Hint: If you don't have access to Huggingface, please visit the following mirror:

https://modelscope.cn/studios/k2-fsa/web-assembly-tts-sherpa-onnx-de/summary

Note: The script for building this space can be found at https://github.com/k2-fsa/sherpa-onnx/blob/master/.github/workflows/wasm-simd-hf-space-de-tts.yaml

8.26.4 Piper

In this section, we describe how to convert piper pre-trained models from https://huggingface.co/rhasspy/piper-voices.

Hint:

You can find `all` of the converted models from piper in the following address:

https://github.com/k2-fsa/sherpa-onnx/releases/tag/tts-models

If you want to convert your own pre-trained piper models or if you want to learn how the conversion works, please read on.

Otherwise, you only need to download the converted models from the above link.

Note that there are pre-trained models for over 30 languages from piper. All models share the same converting method, so we use an American English model in this section as an example.

Install dependencies

```
pip install onnx onnxruntime
```

Hint: We suggest that you always use the latest version of onnxruntime.

Find the pre-trained model from piper

All American English models from piper can be found at https://huggingface.co/rhasspy/piper-voices/tree/main/en/en_US.

We use https://huggingface.co/rhasspy/piper-voices/tree/main/en/en_US/amy/low as an example in this section.

Download the pre-trained model

We need to download two files for each model:

```
wget https://huggingface.co/rhasspy/piper-voices/resolve/main/en/en_US/amy/low/en_US-amy-
↪low.onnx
wget https://huggingface.co/rhasspy/piper-voices/resolve/main/en/en_US/amy/low/en_US-amy-
↪low.onnx.json
```

Add meta data to the onnx model

Please use the following code to add meta data to the downloaded onnx model.

```
#!/usr/bin/env python3

import json
import os
from typing import Any, Dict

import onnx


def add_meta_data(filename: str, meta_data: Dict[str, Any]):
    """Add meta data to an ONNX model. It is changed in-place.

    Args:
      filename:
        Filename of the ONNX model to be changed.
      meta_data:
        Key-value pairs.
    """
    model = onnx.load(filename)
    for key, value in meta_data.items():
        meta = model.metadata_props.add()
        meta.key = key
        meta.value = str(value)
```

(continues on next page)

(continued from previous page)

```
    onnx.save(model, filename)


def load_config(model):
    with open(f"{model}.json", "r") as file:
        config = json.load(file)
    return config


def generate_tokens(config):
    id_map = config["phoneme_id_map"]
    with open("tokens.txt", "w", encoding="utf-8") as f:
        for s, i in id_map.items():
            f.write(f"{s} {i[0]}\n")
    print("Generated tokens.txt")


def main():
    # Caution: Please change the filename
    filename = "en_US-amy-low.onnx"

    # The rest of the file should not be changed.
    # You only need to change the above filename = "xxx.onxx" in this file

    config = load_config(filename)

    print("generate tokens")
    generate_tokens(config)

    print("add model metadata")
    meta_data = {
        "model_type": "vits",
        "comment": "piper",  # must be piper for models from piper
        "language": config["language"]["name_english"],
        "voice": config["espeak"]["voice"],  # e.g., en-us
        "has_espeak": 1,
        "n_speakers": config["num_speakers"],
        "sample_rate": config["audio"]["sample_rate"],
    }
    print(meta_data)
    add_meta_data(filename, meta_data)


main()
```

After running the above script, your `en_US-amy-low.onnx` is updated with meta data and it also generates a new file `tokens.txt`.

From now on, you don't need the config json file `en_US-amy-low.onnx.json` any longer.

Download espeak-ng-data

```
wget https://github.com/k2-fsa/sherpa-onnx/releases/download/tts-models/espeak-ng-data.
↪tar.bz2
tar xf espeak-ng-data.tar.bz2
```

Note that `espeak-ng-data.tar.bz2` is shared by all models from piper, no matter which language your are using for your model.

Test your converted model

To have a quick test of your converted model, you can use

```
pip install sherpa-onnx
```

to install sherpa-onnx and then use the following commands to test your model:

```
# The command "pip install sherpa-onnx" will install several binaries,
# including the following one

which sherpa-onnx-offline-tts

sherpa-onnx-offline-tts \
  --vits-model=./en_US-amy-low.onnx \
  --vits-tokens=./tokens.txt \
  --vits-data-dir=./espeak-ng-data \
  --output-filename=./test.wav \
  "How are you doing? This is a text-to-speech application using next generation Kaldi."
```

The above command should generate a wave file `test.wav`.

Congratulations! You have successfully converted a model from piper and run it with sherpa-onnx.

8.26.5 MMS

This section describes how to convert models from https://huggingface.co/facebook/mms-tts/tree/main to sherpa-onnx.

Note that facebook/mms-tts supports more than 1000 languages. You can try models from facebook/mms-tts at the huggingface space https://huggingface.co/spaces/mms-meta/MMS.

You can try the converted models by visiting https://huggingface.co/spaces/k2-fsa/text-to-speech. To download the converted models, please visit https://github.com/k2-fsa/sherpa-onnx/releases/tag/tts-models. If a filename contains `vits-mms`, it means the model is from facebook/mms-tts.

Install dependencies

```
pip install -qq onnx scipy Cython
pip install -qq torch==1.13.0+cpu -f https://download.pytorch.org/whl/torch_stable.html
```

Download the model file

Suppose that we want to convert the English model, we need to use the following commands to download the model:

```
name=eng
wget -q https://huggingface.co/facebook/mms-tts/resolve/main/models/$name/G_100000.pth
wget -q https://huggingface.co/facebook/mms-tts/resolve/main/models/$name/config.json
wget -q https://huggingface.co/facebook/mms-tts/resolve/main/models/$name/vocab.txt
```

Download MMS source code

```
git clone https://huggingface.co/spaces/mms-meta/MMS
export PYTHONPATH=$PWD/MMS:$PYTHONPATH
export PYTHONPATH=$PWD/MMS/vits:$PYTHONPATH

pushd MMS/vits/monotonic_align

python3 setup.py build

ls -lh build/
ls -lh build/lib*/
ls -lh build/lib*/*/

cp build/lib*/vits/monotonic_align/core*.so .

sed -i.bak s/.monotonic_align.core/.core/g ./__init__.py
popd
```

Convert the model

Please save the following code into a file with name `./vits-mms.py`:

```
#!/usr/bin/env python3

import collections
import os
from typing import Any, Dict

import onnx
import torch
from vits import commons, utils
from vits.models import SynthesizerTrn
```

(continues on next page)

(continued from previous page)

```
class OnnxModel(torch.nn.Module):
    def __init__(self, model: SynthesizerTrn):
        super().__init__()
        self.model = model

    def forward(
        self,
        x,
        x_lengths,
        noise_scale=0.667,
        length_scale=1.0,
        noise_scale_w=0.8,
    ):
        return self.model.infer(
            x=x,
            x_lengths=x_lengths,
            noise_scale=noise_scale,
            length_scale=length_scale,
            noise_scale_w=noise_scale_w,
        )[0]


def add_meta_data(filename: str, meta_data: Dict[str, Any]):
    """Add meta data to an ONNX model. It is changed in-place.

    Args:
      filename:
        Filename of the ONNX model to be changed.
      meta_data:
        Key-value pairs.
    """
    model = onnx.load(filename)
    for key, value in meta_data.items():
        meta = model.metadata_props.add()
        meta.key = key
        meta.value = str(value)

    onnx.save(model, filename)


def load_vocab():
    return [
        x.replace("\n", "") for x in open("vocab.txt", encoding="utf-8").readlines()
    ]


@torch.no_grad()
def main():
    hps = utils.get_hparams_from_file("config.json")
    is_uroman = hps.data.training_files.split(".")[-1] == "uroman"
    if is_uroman:
        raise ValueError("We don't support uroman!")
```

(continues on next page)

(continued from previous page)

```
symbols = load_vocab()

# Now generate tokens.txt
all_upper_tokens = [i.upper() for i in symbols]
duplicate = set(
    [
        item
        for item, count in collections.Counter(all_upper_tokens).items()
        if count > 1
    ]
)

print("generate tokens.txt")

with open("tokens.txt", "w", encoding="utf-8") as f:
    for idx, token in enumerate(symbols):
        f.write(f"{token} {idx}\n")

        # both upper case and lower case correspond to the same ID
        if (
            token.lower() != token.upper()
            and len(token.upper()) == 1
            and token.upper() not in duplicate
        ):
            f.write(f"{token.upper()} {idx}\n")

net_g = SynthesizerTrn(
    len(symbols),
    hps.data.filter_length // 2 + 1,
    hps.train.segment_size // hps.data.hop_length,
    **hps.model,
)
net_g.cpu()
_ = net_g.eval()

_ = utils.load_checkpoint("G_100000.pth", net_g, None)

model = OnnxModel(net_g)

x = torch.randint(low=1, high=10, size=(50,), dtype=torch.int64)
x = x.unsqueeze(0)

x_length = torch.tensor([x.shape[1]], dtype=torch.int64)
noise_scale = torch.tensor([1], dtype=torch.float32)
length_scale = torch.tensor([1], dtype=torch.float32)
noise_scale_w = torch.tensor([1], dtype=torch.float32)

opset_version = 13

filename = "model.onnx"
```

(continues on next page)

(continued from previous page)

```
    torch.onnx.export(
        model,
        (x, x_length, noise_scale, length_scale, noise_scale_w),
        filename,
        opset_version=opset_version,
        input_names=[
            "x",
            "x_length",
            "noise_scale",
            "length_scale",
            "noise_scale_w",
        ],
        output_names=["y"],
        dynamic_axes={
            "x": {0: "N", 1: "L"},  # n_audio is also known as batch_size
            "x_length": {0: "N"},
            "y": {0: "N", 2: "L"},
        },
    )
    meta_data = {
        "model_type": "vits",
        "comment": "mms",
        "url": "https://huggingface.co/facebook/mms-tts/tree/main",
        "add_blank": int(hps.data.add_blank),
        "language": os.environ.get("language", "unknown"),
        "frontend": "characters",
        "n_speakers": int(hps.data.n_speakers),
        "sample_rate": hps.data.sampling_rate,
    }
    print("meta_data", meta_data)
    add_meta_data(filename=filename, meta_data=meta_data)


main()
```

The you can run it with:

```
export PYTHONPATH=$PWD/MMS:$PYTHONPATH
export PYTHONPATH=$PWD/MMS/vits:$PYTHONPATH
export lang=eng
python3 ./vits-mms.py
```

It will generate the following two files:

- `model.onnx`
- `tokens.txt`

Use the converted model

We can use the converted model with the following command after installing sherpa-onnx.

```
./build/bin/sherpa-onnx-offline-tts \
  --vits-model=./model.onnx \
  --vits-tokens=./tokens.txt \
  --debug=1 \
  --output-filename=./mms-eng.wav \
  "How are you doing today? This is a text-to-speech application using models from␣
↪facebook with next generation Kaldi"
```

The above command should generate a wave file `mms-eng.wav`.

Congratulations! You have successfully converted a model from MMS and run it with sherpa-onnx.

We are using `eng` in this section as an example, you can replace it with other languages, such as `deu` for German, `fra` for French, etc.

8.26.6 Frequently Asked Question (FAQs)

Is there a colab notebook

Yes, we have one. Please see

https://github.com/k2-fsa/colab/blob/master/sherpa-onnx/Text_to_speech_with_sherpa_onnx.ipynb

It shows you

- How to install sherpa-onnx
- How to download pre-trained text-to-speech (TTS) models
- How to use sherpa-onnx with pre-trained models for TTS

How to enable UTF-8 on Windows

For Chinese Users: issue

Please see win11 cmdutf-8

How to install sherpa-onnx for TTS

For Python users

The fastest way to install sherpa-onnx for TTS is:

```
pip install sherpa-onnx
```

The above command does NOT require you to install a C++ compiler and it supports a variety of platforms, such as:

- Linux
  - x64
  - arm, e.g., 32-bit Raspberry Pi
  - arm64, e.g., 64-bit Raspberry Pi
- Windows
  - x64, e.g., 64-bit Windows
  - x86, e.g., 32-bit Windows

- macOS
  - x64
  - arm64, e.g., M1 and M2 chips

If you want to build the sherpa-onnx Python package from source, please refer to *Install the Python Package*.

After installation, please refer to https://github.com/k2-fsa/sherpa-onnx/blob/master/python-api-examples/offline-tts.py for example usage.

Hint: `pip install sherpa-onnx` also installs an executable `sherpa-onnx-offline-tts`. The directory where it is installed should be already on your `PATH` after you activate your Python virtual environment.

You can run

```
sherpa-onnx-offline-tts --help
```

in your terminal to get the help information about it.

Build from source

Please refer to *Installation*.

Where to get pre-trained TTS models

Please refer to *Pre-trained models*.

How to handle OOVs

Please add them to `lexicon.txt`.

CHAPTER

NINE

TRITON

Nvidia Triton Inference Server provides a cloud and edge inferencing solution optimized for both CPUs and GPUs.

The following content describes how to deploy ASR models trained by icefall using Triton.

9.1 Installation

We prepare a dockerfile based on official triton docker containers. The customized dockerfile intergrates Triton-server, Triton-client and sherpa-related requirements into a single image. You need to install Docker first before starting installation.

Hint: For your production environment, you could build triton manually to reduce the size of container.

9.1.1 Build Triton Image

```
git clone https://github.com/k2-fsa/sherpa
cd sherpa/triton
docker build . -f Dockerfile/Dockerfile.server -t sherpa_triton_server:latest
```

Note: It may take a lot of time since we build k2 from source. If you only need to use greedy search scorer, you could comment k2-related lines.

9.1.2 Launch a inference container

```
docker run --gpus all --name sherpa_server --net host --shm-size=1g -it sherpa_triton_
↪server:latest
```

Now, you should enter into the container successfully.

9.2 Triton-server

This page gives serveral examples to deploy streaming and offline ASR pretrained models with Triton server.

9.2.1 Deploy streaming ASR models with Onnx

First, we need to export pretrained models with Onnx.

```
export SHERPA_SRC=./sherpa
export ICEFALL_SRC=/workspace/icefall
# copy essentials
cp $SHERPA_SRC/triton/scripts/*onnx*.py $ICEFALL_DIR/egs/wenetspeech/ASR/pruned_
↪stateless_transducer5/
cd $ICEFALL_SRC/egs/wenetspeech/ASR/
# download pretrained models
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/luomingshuang/icefall_asr_
↪wenetspeech_pruned_transducer_stateless5_streaming
cd ./icefall_asr_wenetspeech_pruned_transducer_stateless5_streaming
git lfs pull --include "exp/pretrained_epoch_7_avg_1.pt"
cd -
# export to onnx fp16
ln -s ./icefall_asr_wenetspeech_pruned_transducer_stateless5_streaming/exp/pretrained_
↪epoch_7_avg_1.pt ./icefall_asr_wenetspeech_pruned_transducer_stateless5_streaming/exp/
↪epoch-999.pt
./pruned_transducer_stateless5/export_onnx.py \
   --exp-dir ./icefall_asr_wenetspeech_pruned_transducer_stateless5_streaming/exp \
   --tokenizer-file ./icefall_asr_wenetspeech_pruned_transducer_stateless5_streaming/
↪data/lang_char \
   --epoch 999 \
   --avg 1 \
   --streaming-model 1 \
   --causal-convolution 1 \
   --onnx 1 \
   --left-context 64 \
   --right-context 4 \
   --fp16
```

Note: For Chinese models, `--tokenizer-file` points to `<pretrained_dir>/data/lang_char`. While for English models, it points to `<pretrained_dir>/data/lang_bpe_500/bpe.model` file.

Then, in the docker container, you could start the service with:

```
cd sherpa/triton/
bash scripts/start_streaming_server.sh
```

9.2.2 Deploy offline ASR models with torchscript

Caution: Currently, we only support FP32 offline ASR inference for torchscript backend. Streaming ASR and FP16 inference are not supported.

First, we need to export pretrained models using jit.

```
export SHERPA_SRC=./sherpa
export ICEFALL_SRC=/workspace/icefall
# Download pretrained models
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/csukuangfj/icefall-asr-
↪librispeech-pruned-transducer-stateless3-2022-04-29 $ICEFALL_DIR/egs/librispeech/ASR/
↪pruned_stateless_transducer3/
cd icefall-asr-librispeech-pruned-transducer-stateless3-2022-04-29
git lfs pull --include "exp/pretrained-epoch-25-avg-7.pt"
# export them to three jit models: encoder_jit.pt, decoder_jit.pt, joiner_jit.pt
cp $SHERPA_SRC/triton/scripts/conformer_triton.py $ICEFALL_DIR/egs/librispeech/ASR/
↪pruned_stateless_transducer3/
cp $SHERPA_SRC/triton/scripts/export_jit.py $ICEFALL_DIR/egs/librispeech/ASR/pruned_
↪stateless_transducer3/
cd $ICEFALL_DIR/egs/librispeech/ASR/pruned_stateless_transducer3
python3 export_jit.py \
        --pretrained-model $ICEFALL_DIR/egs/librispeech/ASR/pruned_stateless_transducer3/
↪icefall-asr-librispeech-pruned-transducer-stateless3-2022-04-29 \
        --output-dir <jit_model_dir> --bpe-model <bpe_model_path>
# copy bpe file to <jit_model_dir>, later we would mount <jit_model_dir> to the triton␣
↪docker container
cp <bpe_model_path> <jit_model_dir>
```

Note: If you export models outside the docker container, you could mount the exported `<jit_model_dir>` with `-v <host_dir>:<container_dir>` when lauching the container.

Then, in the docker container, you could start the service with:

```
cd sherpa/triton/
bash scripts/start_offline_server_jit.sh
```

9.3 Triton-client

9.3.1 Send requests using client

In the docker container, run the client script to do ASR inference:

```
cd sherpa/triton/client
# Test one audio using offline ASR
python3 client.py --audio_file=./test_wavs/1089-134686-0001.wav --url=localhost:8001
```

(continues on next page)

(continued from previous page)

```
# Test one audio using streaming ASR
python3 client.py --audio_file=./test_wavs/1089-134686-0001.wav --url=localhost:8001 --
↪streaming
```

The above command sends a single audio `1089-134686-0001.wav` to the server and get the result. `--url` option specifies the IP and port of the server, in this example, we set the server and client on the same machine, therefore IP is `localhost`, and we use port `8001` since it is the default port for gRPC in Triton.

You can also test a bunch of audios together with the client. Just specify the path of `wav.scp` with `--wavscp` option, set the path of test set directory with `--data_dir` option, and set the path of ground-truth transcript file with `--trans` option, the client will infer all the audios in test set and calculate the WER upon the test set.

9.3.2 Decode manifests

You could also decode a whole dataset to benchmark metrics e.g. RTF, WER.

Caution: Decode manifests in simulation streaming mode would be supported in the future.

```
cd sherpa/triton/client

# For aishell manifests:
git lfs install
git clone https://huggingface.co/csukuangfj/aishell-test-dev-manifests
sudo mkdir -p /root/fangjun/open-source/icefall-aishell/egs/aishell/ASR/download/aishell
tar xf ./aishell-test-dev-manifests/data_aishell.tar.gz -C /root/fangjun/open-source/
↪icefall-aishell/egs/aishell/ASR/download/aishell/
# dev set: ./aishell-test-dev-manifests/data/fbank/aishell_cuts_test.jsonl.gz
# test set: ./aishell-test-dev-manifests/data/fbank/aishell_cuts_test.jsonl.gz

python3 decode_manifest_triton.py \
   --server-addr localhost \
   --num-tasks 300 \
   --log-interval 20 \
   --model-name transducer \
   --manifest-filename ./aishell-test-dev-manifests/data/fbank/aishell_cuts_test.jsonl.
↪gz \
   --compute-cer
```

9.4 Perf Analyzer

We can use perf_analyzer provided by Triton to test the performance of the service.

9.4.1 Generate Input Data from Audio Files

For offline ASR server:

```
cd sherpa/triton/client
# en
python3 generate_perf_input.py --audio_file=test_wavs/1089-134686-0001.wav
# zh
python3 generate_perf_input.py --audio_file=test_wavs/zh/mid.wav
```

It will generate a `offline_input.json` file `sherpa/triton/client`.

For streaming ASR server, you need to add a `--streaming` option:

```
python3 generate_perf_input.py --audio_file=test_wavs/1089-134686-0001.wav --streaming
```

A `online_input.json` file would be generated.

9.4.2 Test Throughput using Perf Analyzer

```
# Offline ASR Test with grpc
perf_analyzer -m transducer -b 1 -a -p 20000 --concurrency-range 100:200:50 -i gRPC --
↪input-data=offline_input.json  -u localhost:8001

# Streaming ASR Test with grpc
perf_analyzer -m transducer -b 1 -a -p 20000 --concurrency-range 100:200:50 -i gRPC --
↪input-data=online_input.json  -u localhost:8001 --streaming
```

You could save the below results with a `-f log.txt` option.

Concurrency	Inferences/Second	Client Send	Network+Server Send/Recv	Server Queue	Server Compute Input	Server Compute Infer	Server Compute Output	Client Recv	p50 latency	p90 latency	p95 latency	p99 latency
300	226.24	109	230434	1	9314	1068792	14512	1	1254206	1616224	1958245	2551406

Note: Please refer to https://github.com/triton-inference-server/server/blob/main/docs/user_guide/perf_analyzer.md for advanced usuage.

9.5 TensorRT acceleration

This page shows how to use TensorRT engine to accelerate inference speed for K2 models

9.5.1 Preparation

First of all, you have to install the TensorRT. Here we suggest you to use docker container to run TRT. Just run the following command:

```
docker run --gpus '"device=0"' -it --rm --net host -v $PWD/:/k2 nvcr.io/nvidia/
↪tensorrt:22.12-py3
```

You can also see here to build TRT on your machine.

Note: Please pay attention that, the TRT version must have to >= 8.5.3!!!

If your TRT version is < 8.5.3, you can download the desired TRT version and then run the following command inside the docker container to use the TRT you just download:

```
# inside the container
bash tools/install.sh
```

9.5.2 Model export

You have to prepare the ONNX model by referring here to export your models into ONNX format. Assume you have put your ONNX model in the `$model_dir` directory. Then, just run the command:

```
bash tools/build.sh $model_dir
cp $model_dir/encoder.trt model_repo_offline_fast_beam_trt/encoder/1
```

The generated TRT model will be saved into `$model_dir/encoder.trt`. We also give an example of `model_repo` of TRT model. You can follow the same procedure as described here to deploy the pipeline using triton.

9.5.3 Benchmark for Conformer TRT encoder vs ONNX

Model	Batch size	Avg latency(ms)	QPS
ONNX	1	7.44	134.48
	8	14.92	536.09
	16	22.84	700.67
	32	41.62	768.84
	64	80.48	795.27
	128	171.97	744.32
TRT	1	5.21834	193.93
	8	11.7826	703.49
	16	20.4444	815.79
	32	37.583	893.56
	64	69.8312	965.40
	128	139.702	964.57