L635-Fall2025-Assignment 1: ESPnet Tutorial

In this tutorial, we will learn how to fine-tune a speech foundation model using ESPnet EZ.

Main references:

- ESPnet repository
- ESPnet documentation
- ESPnet-EZ repo

Important Notes

- Please submit PDF files of your completed notebooks to Canvas. You can print the notebook using `File -> Print` in the menu bar.

Acknowledgement

- This homework is adapted from the ESPnet online demos and tutorials.

Install ESPnet

- We are using a temporary version of ESPNET to avoid compatibility issues for this assignment. You may see some dependency errors. It should be safe for you to ignore them for now.

```
1 !wget https://github.com/Fhrozen/espnet/archive/refs/heads/pr-numpy.zip
2 !unzip pr-numpy.zip
```

```
Streaming output truncated to the last 5000 lines.
   creating: espnet-pr-numpy/espnet2/diar/separator/
 extracting: espnet-pr-numpy/espnet2/diar/separator/__init__.py
  inflating: espnet-pr-numpy/espnet2/diar/separator/tcn_separator_nomask.py
   creating: espnet-pr-numpy/espnet2/enh/
 extracting: espnet-pr-numpy/espnet2/enh/__init__.py
  inflating: espnet-pr-numpy/espnet2/enh/abs_enh.py
   creating: espnet-pr-numpy/espnet2/enh/decoder/
 extracting: espnet-pr-numpy/espnet2/enh/decoder/__init__.py
  inflating: espnet-pr-numpy/espnet2/enh/decoder/abs_decoder.py
  inflating: espnet-pr-numpy/espnet2/enh/decoder/conv_decoder.py
  inflating: espnet-pr-numpy/espnet2/enh/decoder/null_decoder.py
  inflating: espnet-pr-numpy/espnet2/enh/decoder/stft_decoder.py
   creating: espnet-pr-numpy/espnet2/enh/diffusion/
 extracting: espnet-pr-numpy/espnet2/enh/diffusion/__init__.py
  inflating: espnet-pr-numpy/espnet2/enh/diffusion/abs_diffusion.py
```

```
  creating: espnet-pr-numpy/espnet2/enh/diffusion/sampling/
 inflating: espnet-pr-numpy/espnet2/enh/diffusion/sampling/__init__.py
 inflating: espnet-pr-numpy/espnet2/enh/diffusion/sampling/correctors.py
 inflating: espnet-pr-numpy/espnet2/enh/diffusion/sampling/predictors.py
 inflating: espnet-pr-numpy/espnet2/enh/diffusion/score_based_diffusion.py
 inflating: espnet-pr-numpy/espnet2/enh/diffusion/sdes.py
 inflating: espnet-pr-numpy/espnet2/enh/diffusion_enh.py
  creating: espnet-pr-numpy/espnet2/enh/encoder/
extracting: espnet-pr-numpy/espnet2/enh/encoder/__init__.py
 inflating: espnet-pr-numpy/espnet2/enh/encoder/abs_encoder.py
 inflating: espnet-pr-numpy/espnet2/enh/encoder/conv_encoder.py
 inflating: espnet-pr-numpy/espnet2/enh/encoder/null_encoder.py
 inflating: espnet-pr-numpy/espnet2/enh/encoder/stft_encoder.py
 inflating: espnet-pr-numpy/espnet2/enh/espnet_enh_s2t_model.py
 inflating: espnet-pr-numpy/espnet2/enh/espnet_model.py
 inflating: espnet-pr-numpy/espnet2/enh/espnet_model_tse.py
  creating: espnet-pr-numpy/espnet2/enh/extractor/
extracting: espnet-pr-numpy/espnet2/enh/extractor/__init__.py
 inflating: espnet-pr-numpy/espnet2/enh/extractor/abs_extractor.py
 inflating: espnet-pr-numpy/espnet2/enh/extractor/td_speakerbeam_extractor.py
  creating: espnet-pr-numpy/espnet2/enh/layers/
extracting: espnet-pr-numpy/espnet2/enh/layers/__init__.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/adapt_layers.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/beamformer.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/beamformer_th.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/bsrnn.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/complex_utils.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/complexnn.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/conv_utils.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/dc_crn.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/dcunet.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/dnn_beamformer.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/dnn_wpe.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/dnsmos.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/dpmulcat.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/dprnn.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/dptnet.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/fasnet.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/ifasnet.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/mask_estimator.py
 inflating: espnet-pr-numpy/espnet2/enh/layers/ncsnpp.py
  creating: espnet-pr-numpy/espnet2/enh/layers/ncsnpp_utils/
```

```
1 !cd espnet-pr-numpy && pip install .
```

```
Processing /content/espnet-pr-numpy
  Preparing metadata (setup.py) ... done
Collecting g2p_en@ git+https://github.com/espnet/g2p.git@master (from espnet==202506)
  Cloning https://github.com/espnet/g2p.git (to revision master) to /tmp/pip-install-
  Running command git clone --filter=blob:none --quiet https://github.com/espnet/g2p.
  Resolved https://github.com/espnet/g2p.git to commit 053dfa94811efc6370bbdb0ff9777b
  Preparing metadata (setup.py) ... done
Collecting ctc-segmentation@ git+https://github.com/espnet/ctc-segmentation.git@9b9ea
  Cloning https://github.com/espnet/ctc-segmentation.git (to revision 9b9ea1d) to /tm
```

```
  Running command git clone --filter=blob:none --quiet https://github.com/espnet/ctc-
  WARNING: Did not find branch or tag '9b9ea1d', assuming revision or ref.
  Running command git checkout -q 9b9ea1d
  Resolved https://github.com/espnet/ctc-segmentation.git to commit 9b9ea1d
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting setuptools<74.0.0,>=38.5.1 (from espnet==202506)
  Downloading setuptools-73.0.1-py3-none-any.whl.metadata (6.6 kB)
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (
Collecting configargparse>=1.2.1 (from espnet==202506)
  Downloading configargparse-1.7.1-py3-none-any.whl.metadata (24 kB)
Requirement already satisfied: typeguard in /usr/local/lib/python3.12/dist-packages (
Collecting humanfriendly (from espnet==202506)
  Downloading humanfriendly-10.0-py2.py3-none-any.whl.metadata (9.2 kB)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: librosa>=0.10.2 in /usr/local/lib/python3.12/dist-pack
Collecting jamo==0.4.1 (from espnet==202506)
  Downloading jamo-0.4.1-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: PyYAML>=5.1.2 in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: soundfile>=0.10.2 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: h5py>=2.10.0 in /usr/local/lib/python3.12/dist-package
Collecting kaldiio>=2.18.0 (from espnet==202506)
  Downloading kaldiio-2.18.1-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.12/dist-packag
Collecting torch_complex (from espnet==202506)
  Downloading torch_complex-0.4.4-py3-none-any.whl.metadata (3.1 kB)
Requirement already satisfied: nltk>=3.4.5 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: numpy>=2.0.0 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: protobuf in /usr/local/lib/python3.12/dist-packages (f
Collecting hydra-core (from espnet==202506)
  Downloading hydra_core-1.3.2-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: opt-einsum in /usr/local/lib/python3.12/dist-packages
Collecting lightning (from espnet==202506)
  Downloading lightning-2.5.5-py3-none-any.whl.metadata (39 kB)
Collecting sentencepiece==0.2.0 (from espnet==202506)
  Downloading sentencepiece-0.2.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86
Collecting pyworld>=0.3.4 (from espnet==202506)
  Downloading pyworld-0.3.5.tar.gz (261 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 261.0/261.0 kB 25.2 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting pypinyin<=0.44.0 (from espnet==202506)
  Downloading pypinyin-0.44.0-py2.py3-none-any.whl.metadata (10 kB)
Collecting espnet_tts_frontend (from espnet==202506)
  Downloading espnet_tts_frontend-0.0.3-py3-none-any.whl.metadata (3.4 kB)
Collecting ci_sdr (from espnet==202506)
```

```
1 !pip install espnet-model-zoo # for downloading pre-trained models
2 !apt install ffmpeg # for audio file processing
3 !pip install ipywebrtc notebook # for real-time recording
4
```

```
Collecting espnet-model-zoo
  Downloading espnet_model_zoo-0.1.7-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (fro
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: espnet in /usr/local/lib/python3.12/dist-packages (fro
Requirement already satisfied: huggingface-hub in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (f
Collecting g2p-en@ git+https://github.com/espnet/g2p.git@master (from espnet->espnet-
  Cloning https://github.com/espnet/g2p.git (to revision master) to /tmp/pip-install-
  Running command git clone --filter=blob:none --quiet https://github.com/espnet/g2p.
  Resolved https://github.com/espnet/g2p.git to commit 053dfa94811efc6370bbdb0ff9777b
  Preparing metadata (setup.py) ... done
Collecting ctc-segmentation@ git+https://github.com/espnet/ctc-segmentation.git@9b9ea
  Cloning https://github.com/espnet/ctc-segmentation.git (to revision 9b9ea1d) to /tm
  Running command git clone --filter=blob:none --quiet https://github.com/espnet/ctc-
  WARNING: Did not find branch or tag '9b9ea1d', assuming revision or ref.
  Running command git checkout -q 9b9ea1d
  Resolved https://github.com/espnet/ctc-segmentation.git to commit 9b9ea1d
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: setuptools<74.0.0,>=38.5.1 in /usr/local/lib/python3.1
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: configargparse>=1.2.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: typeguard in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: humanfriendly in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: librosa>=0.10.2 in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: jamo==0.4.1 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: PyYAML>=5.1.2 in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: soundfile>=0.10.2 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: h5py>=2.10.0 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: kaldiio>=2.18.0 in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: torch-complex in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: nltk>=3.4.5 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: protobuf in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: hydra-core in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: opt-einsum in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: lightning in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: sentencepiece==0.2.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: pyworld>=0.3.4 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: pypinyin<=0.44.0 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: espnet-tts-frontend in /usr/local/lib/python3.12/dist-
Requirement already satisfied: ci-sdr in /usr/local/lib/python3.12/dist-packages (fro
Requirement already satisfied: fast-bss-eval==0.1.3 in /usr/local/lib/python3.12/dist
Requirement already satisfied: asteroid-filterbanks==0.4.0 in /usr/local/lib/python3.
Requirement already satisfied: editdistance in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: importlib-metadata<5.0 in /usr/local/lib/python3.12/di
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/di
```

```
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/
```

Import the dependencies and check the state of installation

```
!pip install datasets==3.6.0
```

```
Collecting datasets==3.6.0
  Downloading datasets-3.6.0-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (fro
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: xxhash in /usr/local/lib/python3.12/dist-packages (fro
Requirement already satisfied: multiprocess<0.70.17 in /usr/local/lib/python3.12/dist
Requirement already satisfied: fsspec<=2025.3.0,>=2023.1.0 in /usr/local/lib/python3.
Requirement already satisfied: huggingface-hub>=0.24.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python3.1
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.1
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/di
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: aiohappyeyeballs>=2.5.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: aiosignal>=1.4.0 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (f
Downloading datasets-3.6.0-py3-none-any.whl (491 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 491.5/491.5 kB 31.2 MB/s eta 0:00:00
Installing collected packages: datasets
  Attempting uninstall: datasets
    Found existing installation: datasets 4.0.0
    Uninstalling datasets-4.0.0:
      Successfully uninstalled datasets-4.0.0
Successfully installed datasets-3.6.0
```

```
import torch
import datasets
```

```
import espnetez as ez # ESPnet wrapper that simplifies integration. If you get an erro
import numpy as np
import librosa
from espnet2.bin.s2t_inference import Speech2Text # Core ESPnet module for pre-trained

print("Installation success!")
```

```
Failed to import Flash Attention, using ESPnet default: No module named 'flash_attn'
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
[nltk_data] Downloading package cmudict to /root/nltk_data...
[nltk_data]   Unzipping corpora/cmudict.zip.
/usr/local/lib/python3.12/dist-packages/espnet2/enh/encoder/stft_encoder.py:79: Futur
  @torch.cuda.amp.autocast(enabled=False)
/usr/local/lib/python3.12/dist-packages/espnet2/enh/layers/uses2_swin.py:329: FutureW
  @torch.cuda.amp.autocast(enabled=False)
Installation success!
```

Data Processing

For this tutorial, we will use the FLEURS dataset from HuggingFace: https://huggingface.co/datasets/google/fleurs .

FLEURS is a 102-language multilingual speech dataset, supporting tasks such as Automatic Speech Recognition (ASR), Speech Translation (ST), and Language Identification (LID).

While the total size of FLEURS is relatively large at ~1000 hours of training data, each individual language only has 7-10 hours of audio.

For this tutorial, we will focus on monolingual ASR for one of the 102 languages.

Data Downloading

We will first download the data for one language of FLEURS. FLEURS organizes the languages by its ISO2 language code and locale. For example, American English is `en_us` .

We will use English for the first fine-tuning experiment. You will have the opportunity to try a different language later on in the assignment.

If you want to download the data for another language, you can map the language name to the ISO2 code using Table 9 in the FLEURS paper: https://arxiv.org/pdf/2205.12446. Then, you can use that to identify the language+region combination using the HuggingFace data previewer: https://huggingface.co/datasets/google/fleurs .

(Please select y for the prompt of running custom code to download the data)

```
1 fleurs_language = 'en_us' # fleurs language codes are in the <language code>_<region
2 fleurs_hf = datasets.load_dataset("google/fleurs", fleurs_language)
```

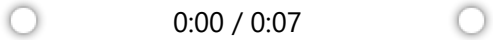
```
README.md:        13.3k/? [00:00<00:00, 1.40MB/s]

fleurs.py:        12.5k/? [00:00<00:00, 1.13MB/s]

The repository for google/fleurs contains custom code which must be executed to corre
You can avoid this prompt in future by passing the argument `trust_remote_code=True`.

Do you wish to run the custom code? [y/N] y

data/en_us/audio/                                                  1.38G/1.38G [00:20<00:00, 81.4MB/
train.tar.gz: 100%                                                                              s]

data/en_us/audio/                                                    171M/171M [00:02<00:00, 100MB/
dev.tar.gz: 100%                                                                                s]

data/en_us/audio/                                                    290M/290M [00:02<00:00, 138MB/
test.tar.gz: 100%                                                                               s]

train.tsv:        1.41M/? [00:00<00:00, 55.4MB/s]

dev.tsv:          213k/? [00:00<00:00, 12.3MB/s]

test.tsv:         368k/? [00:00<00:00, 15.4MB/s]

Generating train split:        2602/0 [00:37<00:00, 179.60 examples/s]
```

Inspect the data

```
1 fleurs_hf['train'][0]
```

```
{'id': 903,
 'num_samples': 108800,
 'path': '/root/.cache/huggingface/datasets/downloads/extracted/
be467d88ba270014363a9d0aaae3893b4701a2710e0a55c6091a6d4fa56a9d84/10004088536354799741
 'audio': {'path': 'train/10004088536354799741.wav',
  'array': array([ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
         -3.15904617e-06, -3.03983688e-06, -3.27825546e-06]),
  'sampling_rate': 16000},
 'transcription': 'a tornado is a spinning column of very low-pressure air which
sucks the surrounding air inward and upward',
 'raw_transcription': 'A tornado is a spinning column of very low-pressure air,
which sucks the surrounding air inward and upward.',
 'gender': 1,
 'lang_id': 19,
 'language': 'English',
 'lang_group_id': 0}
```

```
from IPython.display import Audio, display
display(Audio(fleurs_hf['train'][0]['audio']['array'], rate=16000))
print(fleurs_hf['train'][0]['transcription'])
```




```
a tornado is a spinning column of very low-pressure air which sucks the surrounding a
```

Pretrained Model

In low-resource settings, training a model from scratch is unlikely to lead to good results. So instead, we will fine-tune a pre-trained foundation model.

We will use the base version of OWSM 3.1, an open-source speech foundation model trained on 180K hours of multilingual ASR and ST.

Downloading

Since it needs to support many language varieties, OWSM uses ISO3 for the language IDs. The ISO3 code for your language of choice can also be found in Table 9 in the FLEURS paper: https://arxiv.org/pdf/2205.12446

```
FINETUNE_MODEL="espnet/owsm_v3.1_ebf_base"
owsm_language="eng" # language code in ISO3
```

```
pretrained_model = Speech2Text.from_pretrained(
    FINETUNE_MODEL,
    lang_sym=f"<{owsm_language}>",
    beam_size=1,
    device='cuda'
)
torch.save(pretrained_model.s2t_model.state_dict(), 'original.pth')
pretrain_config = vars(pretrained_model.s2t_train_args)
tokenizer = pretrained_model.tokenizer
converter = pretrained_model.converter
```

```
Fetching 29 files: 100% 29/29 [00:14<00:00, 1.60it/s]
data/token_list/bpe_unigram50000/ 1.04M/1.04M [00:02<00:00, 387kB/
bpe.mod(…): 100% s]
exp/s2t_stats_raw_bpe50000/train/ 1.40k/1.40k [00:02<00:00, 526B/
feats_s(…): 100% s]
```

README.md: 4.69k/? [00:00<00:00, 71.4kB/s]

RESULTS.md: 100% 366/366 [00:00<00:00, 11.9kB/s]

.gitattributes: 1.52k/? [00:00<00:00, 26.5kB/s]

acc.png: 100% 33.1k/33.1k [00:00<00:00, 585kB/s]

backward_time.png: 100% 42.2k/42.2k [00:00<00:00, 579kB/s]

config.yaml: 490k/? [00:00<00:00, 1.42MB/s]

cer_ctc.png: 100% 26.6k/26.6k [00:00<00:00, 2.96MB/s]

forward_time.png: 100% 48.7k/48.7k [00:00<00:00, 2.29MB/s]

gpu_max_cached_mem_GB.png: 100% 36.9k/36.9k [00:00<00:00, 846kB/s]

cer.png: 100% 28.5k/28.5k [00:00<00:00, 505kB/s]

clip.png: 100% 29.1k/29.1k [00:00<00:00, 946kB/s]

grad_norm.png: 100% 26.8k/26.8k [00:00<00:00, 2.82MB/s]

iter_time.png: 100% 41.8k/41.8k [00:00<00:00, 4.82MB/s]

loss_att.png: 100% 31.6k/31.6k [00:00<00:00, 2.15MB/s]

loss_ctc.png: 100% 34.5k/34.5k [00:00<00:00, 1.32MB/s]

loss.png: 100% 31.4k/31.4k [00:00<00:00, 1.41MB/s]

loss_scale.png: 100% 32.8k/32.8k [00:00<00:00, 3.96MB/s]

optim0_lr0.png: 100% 30.6k/30.6k [00:00<00:00, 2.94MB/s]

Setup Training

We first need to convert the HuggingFace data into a format that ESPnet can read. This can be easily done by defining a `data_info` dictionary that maps each field required for OWSM fine-tuning to a column in our dataset.

```
'''
pretrained_model -> the pre-trained model we downloaded earlier
tokenizer -> Tokenizes raw text into subwords
converter -> Converts subwords into integer IDs for model input
'''

def tokenize(text):
    return np.array(converter.tokens2ids(tokenizer.text2tokens(text)))
data_info = {
    "speech": lambda d: d['audio']['array'].astype(np.float32), # 1-D raw waveform
    "text": lambda d: tokenize(f"<{owsm_language}><asr><notimestamps> {d['transcripti
    "text_prev": lambda d: tokenize("<na>"), # tokenized text of previous utterance f
    "text_ctc": lambda d: tokenize(d['transcription']), # tokenized text mapped to in
}
test_data_info = {
    "speech": lambda d: d['audio']['array'].astype(np.float32),
    "text": lambda d: tokenize(f"<{owsm_language}><asr><notimestamps> {d['transcripti
    "text_prev": lambda d: tokenize("<na>"),
    "text_ctc": lambda d: tokenize(d['transcription']),
    "text_raw": lambda d: d['transcription'], # raw untokenized text as the reference
}
train_dataset = ez.dataset.ESPnetEZDataset(fleurs_hf['train'], data_info=data_info)
valid_dataset = ez.dataset.ESPnetEZDataset(fleurs_hf['validation'], data_info=data_in
test_dataset = ez.dataset.ESPnetEZDataset(fleurs_hf['test'], data_info=test_data_info
```

Next we need to define a function that will pass our pre-trained model to ESPnet. This function here doesn't do much since our setup is simple, but its required for more complex settings (such as LoRA fine-tuning).

```
# define model loading function
def count_parameters(model):
    return sum(p.numel() for p in model.parameters() if p.requires_grad)

def build_model_fn(args):
  model = pretrained_model.s2t_model
  model.train()
  print(f'Trainable parameters: {count_parameters(model)}')
  return model
```

Training

Training requires tuning many hyper-parameters. Here is an initial config to start you off.

```
!gdown 1Hp4hgtgdt84i4Qd99hW5ZM5aV743xN1U
```

```
!mkdir config
!mv finetune.yaml config/finetune.yaml
```

```
Downloading...
From: https://drive.google.com/uc?id=1Hp4hgtgdt84i4Qd99hW5ZM5aV743xN1U
To: /content/finetune.yaml
100% 987/987 [00:00<00:00, 6.22MB/s]
```

Before we begin training, we need to define where our model files and logs will be saved. We also need to override some of the settings used to pre-train the foundation model with our own settings.

```
EXP_DIR = f"./exp/finetune"
STATS_DIR = f"./exp/stats_finetune"
finetune_config = ez.config.update_finetune_config(
    's2t',
    pretrain_config,
    f"./config/finetune.yaml"
)

# You can edit your config by changing the finetune.yaml file directly (but make sure
# You can also change it programatically like this
finetune_config['max_epoch'] = 1
finetune_config['num_iters_per_epoch'] = 500
```

Finally, we just need to pass our model, data, and configs to a trainer.

```
trainer = ez.Trainer(
    task='s2t',
    train_config=finetune_config,
    train_dataset=train_dataset,
    valid_dataset=valid_dataset,
    build_model_fn=build_model_fn, # provide the pre-trained model
    data_info=data_info,
    output_dir=EXP_DIR,
    stats_dir=STATS_DIR,
    ngpu=1
)
```

```
trainer.collect_stats() # collect audio/text length information to construct batches
```

```
/usr/bin/python3 /usr/local/lib/python3.12/dist-packages/colab_kernel_launcher.py -f
Trainable parameters: 101182628
```

```
trainer.train() # every 100 steps takes ~1 min
```

```
/usr/bin/python3 /usr/local/lib/python3.12/dist-packages/colab_kernel_launcher.py -f
Trainable parameters: 101182628
/usr/local/lib/python3.12/dist-packages/espnet2/train/trainer.py:638: FutureWarning:
  with autocast(
/usr/local/lib/python3.12/dist-packages/espnet2/s2t/espnet_model.py:279: FutureWarnin
  with autocast(False):
/usr/local/lib/python3.12/dist-packages/espnet2/train/trainer.py:856: FutureWarning:
  with autocast(
```

Inference

Here is a demo of how to perform inference, and how to load checkpoints.

```
id, sample_test_utterance = test_dataset.__getitem__(0)
```

```
pretrained_model.s2t_model.cuda()
pretrained_model.device = 'cuda'

d = torch.load("original.pth")
pretrained_model.s2t_model.load_state_dict(d)
pred = pretrained_model(sample_test_utterance['speech'])
print('PREDICTED: ' + pred[0][0])
print('REFERENCE: ' + sample_test_utterance['text_raw'])
```

```
PREDICTED: <eng><asr><notimestamps> However, due to the slow communication channels,
REFERENCE: however due to the slow communication channels styles in the west could la
```

Inference with fine-tuned model

```
d = torch.load("./exp/finetune/1epoch.pth")
pretrained_model.s2t_model.load_state_dict(d)
pred = pretrained_model(sample_test_utterance['speech'])
print('PREDICTED: ' + pred[0][0])
print('REFERENCE: ' + sample_test_utterance['text_raw'])
```

```
PREDICTED: <eng><asr><notimestamps> however due to the slow communication channels st
REFERENCE: however due to the slow communication channels styles in the west could la
```

✅ Task 1

Now that you have performed inference with both the pre-trained model and your fine-tuned model, provide some qualitative analyses of the results. How does the output between the two models differ? Do you observe any stylistic differences in the transcriptions?

TASK 1 ANSWER:

Comparing the pre-trained and fine-tuned models: PT: PREDICTED: However, due to the slow communication channels, styles in the west could lag behind by twenty five to thirty years. REFERENCE: however due to the slow communication channels styles in the west could lag behind by 25 to 30 year FT: PREDICTED: however due to the slow communication channels styles in the west could lag behind by 25 to 30 years REFERENCE: however due to the slow communication channels styles in the west could lag behind by 25 to 30 year

Analysis: A comparison of the pre-trained (PT) and fine-tuned (FT) model outputs shows stylistic improvements aligned with the reference data. The PT model produced a more formal output, capitalizing the first word 'However' and writing out numbers 'twenty five to thirty years'. The FT model, on the other hand successfully adapted to the stylistic conventions of the FLEURS dataset, which are closer to raw speech transcription. It used lowercase 'however' and, most importantly, utilized numerals '25 to 30', which exactly matched the reference. Interestingly, both models correctly pluralized 'years' at the end, whereas the reference text contained the error 'year', suggesting the audio clearly contained the plural form.

✅ Task 2

A good way to understand the capabilities of neural models is to play with them yourself. Record a brief audio clip of yourself speaking. How accurately does the model transcribe your speech?

Then, utter the same sentence, but vary some characteristics of your speech. For example, you can try speaking faster, in a different pitch, or in a different accent. How does the resulting transcription change?

Finally, if the model made any mistakes in its transcriptions, why do you think these errors occured?

```
# You may need to run this cell twice before it will work the first time
!rm recording.webm
!rm recording.wav
from ipywebrtc import AudioRecorder, CameraStream
from google.colab import output
output.enable_custom_widget_manager()
mic = AudioRecorder(stream=CameraStream(constraints={'audio': True,'video':False}))
mic # press the circle to record, press it again to stop recording
```

```
rm: cannot remove 'recording.webm': No such file or directory
rm: cannot remove 'recording.wav': No such file or directory
```


```
1 with open('recording.webm', 'wb') as out_f:
2     out_f.write(mic.audio.value)
3 audio, sr = librosa.load('recording.webm', sr=16000) # ASR typically uses 16kHz audio
4 pretrained_model(audio)[0][0]
```

```
/usr/local/lib/python3.12/dist-packages/librosa/core/audio.py:184: FutureWarning: lib
    Deprecated as of librosa version 0.10.0.
    It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
/usr/local/lib/python3.12/dist-packages/espnet2/s2t/espnet_model.py:279: FutureWarnin
  with autocast(False):
'<eng><asr><notimestamps> This is a sentence with my normal accent.'
```

```
1 # You may need to run this cell twice before it will work the first time
2 !rm recording.webm
3 !rm recording.wav
4 from ipywebrtc import AudioRecorder, CameraStream
5 from google.colab import output
6 output.enable_custom_widget_manager()
7 mic = AudioRecorder(stream=CameraStream(constraints={'audio': True,'video':False}))
8 mic # press the circle to record, press it again to stop recording
```

```
rm: cannot remove 'recording.wav': No such file or directory
```

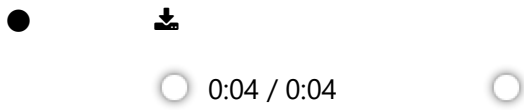

```
1 with open('recording.webm', 'wb') as out_f:
2     out_f.write(mic.audio.value)
3 audio, sr = librosa.load('recording.webm', sr=16000) # ASR typically uses 16kHz audio
4 pretrained_model(audio)[0][0]
```

```
/usr/local/lib/python3.12/dist-packages/librosa/core/audio.py:184: FutureWarning: lib
    Deprecated as of librosa version 0.10.0.
    It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
/usr/local/lib/python3.12/dist-packages/espnet2/s2t/espnet_model.py:279: FutureWarnin
  with autocast(False):
'<eng><asr><notimestamps> This is a sentence with my terrible eastern European accen
t.'
```

```
1 # You may need to run this cell twice before it will work the first time
```

```
!rm recording.webm
!rm recording.wav
from ipywebrtc import AudioRecorder, CameraStream
from google.colab import output
output.enable_custom_widget_manager()
mic = AudioRecorder(stream=CameraStream(constraints={'audio': True,'video':False}))
mic # press the circle to record, press it again to stop recording
```

```
rm: cannot remove 'recording.wav': No such file or directory
```

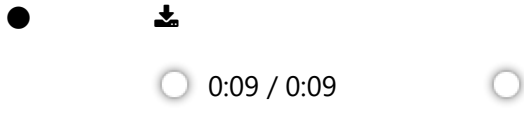



```
with open('recording.webm', 'wb') as out_f:
    out_f.write(mic.audio.value)
audio, sr = librosa.load('recording.webm', sr=16000) # ASR typically uses 16kHz audio
pretrained_model(audio)[0][0]
```

```
/usr/local/lib/python3.12/dist-packages/librosa/core/audio.py:184: FutureWarning: lib
    Deprecated as of librosa version 0.10.0.
    It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
/usr/local/lib/python3.12/dist-packages/espnet2/s2t/espnet_model.py:279: FutureWarnin
  with autocast(False):
'<eng><asr><notimestamps> They did a sentence with a terrible European accent speaki
ng faster.'
```

```
# You may need to run this cell twice before it will work the first time
!rm recording.webm
!rm recording.wav
from ipywebrtc import AudioRecorder, CameraStream
from google.colab import output
output.enable_custom_widget_manager()
mic = AudioRecorder(stream=CameraStream(constraints={'audio': True,'video':False}))
mic # press the circle to record, press it again to stop recording
```

```
rm: cannot remove 'recording.wav': No such file or directory
```



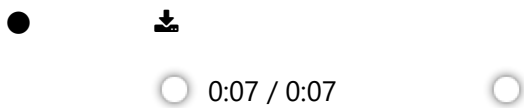

```
with open('recording.webm', 'wb') as out_f:
    out_f.write(mic.audio.value)
audio, sr = librosa.load('recording.webm', sr=16000) # ASR typically uses 16kHz audio
pretrained_model(audio)[0][0]
```

```
/usr/local/lib/python3.12/dist-packages/librosa/core/audio.py:184: FutureWarning: lib
```

```
    Deprecated as of librosa version 0.10.0.
    It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
/usr/local/lib/python3.12/dist-packages/espnet2/s2t/espnet_model.py:279: FutureWarnin
  with autocast(False):
'<eng><asr><notimestamps> Sí, alara, enqueyoastoi, acustombrado, no me intendrias pa
ionada.'
```

```
# You may need to run this cell twice before it will work the first time
!rm recording.webm
!rm recording.wav
from ipywebrtc import AudioRecorder, CameraStream
from google.colab import output
output.enable_custom_widget_manager()
mic = AudioRecorder(stream=CameraStream(constraints={'audio': True,'video':False}))
mic # press the circle to record, press it again to stop recording
```

```
rm: cannot remove 'recording.webm': No such file or directory
rm: cannot remove 'recording.wav': No such file or directory
```




```
with open('recording.webm', 'wb') as out_f:
    out_f.write(mic.audio.value)
audio, sr = librosa.load('recording.webm', sr=16000) # ASR typically uses 16kHz audio
pretrained_model(audio)[0][0]
```

```
/usr/local/lib/python3.12/dist-packages/librosa/core/audio.py:184: FutureWarning: lib
    Deprecated as of librosa version 0.10.0.
    It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
/usr/local/lib/python3.12/dist-packages/espnet2/s2t/espnet_model.py:279: FutureWarnin
  with autocast(False):
'<eng><asr><notimestamps> The endillusion accident, since you're larry and manenerly
that he can cause you, especially consummbrado no mentan edrics for the nave.'
```

TASK 2 ANSWER:

Because the second test also performed pretty well, I tested a few extra sentences.

1. Baseline Performance (Normal Speech): Input: "This is a sentence with my normal accent." Output: 'This is a sentence with my normal accent.' The model's transcription was perfect. It captured the lexical content and correctly inferred the sentence-ending punctuation. The model performs with high accuracy.
2. Investigating Robustness Through Speech Variations: Input: "This is a sentence with my terrible eastern European accent." Modification: The same sentence was spoken with a

terrible eastern European accent." Modification: The same sentence was spoken with a contrived Eastern European accent. Output: "This is a sentence with my terrible eastern European accent.`" The model was robust. It accurately transcribed the sentence, including the self-referential description of the accent itself "terrible eastern European." This indicates the ability to generalize across a range of pronunciations without a loss in precision.

3. Accent + Increased Speech Rate: Input: "This is a sentence with my terrible eastern European accent speaking faster." Modification: The accented speech was delivered at a faster pace. Output: "They did a sentence with a terrible European accent speaking faster." This combination introduced the first material errors. The model substituted the phrase "This is" with "They did," a 'homophone error' likely caused by the blurred acoustic signal of rapid, accented speech. The word "eastern" was dropped, probably due to reduced articulation and the model's bias towards selecting more common or expected word sequences; "European accent" is a more frequent collocation than "eastern European accent" in this context.
4. Cross-Lingual Challenge - Spanish Input Input: "Si hablara en la manera ne que estoy acostumbrado, no me entenderias." Modification: The input language was switched to Spanish, first spoken clearly (a) and then with a thick, fast accent (b). Output a (Clear Spanish): "Sí, alara, enqueyoastoi, acustombrado, no me intendrias paionada." Output b (Fast, Accented Spanish): "since you're larry and manenerly that he can cause you, especially consummbrado no mentan edrics for the nave." As an English-trained model, its primary objective is to map audio to English text. When presented with Spanish, it does not fail silently but instead hallucinates and generates semantically nonsensical yet syntactically plausible English text. The model forcibly maps non-English phonemes to the closest-sounding English words it knows. The more distorted the Spanish input (b), the more prolonged and complex the hallucinated output becomes, showing the model prioritizes generating coherent English structures over accurately representing the input signal.

Acoustic-Phonetic Mismatch:** The model's performance is highly dependent on the input audio conforming to the acoustic profiles of English phonemes in its training data. Deviations caused by speed, heavy accent, or a different language degrade the signal, leading to substitutions and, ultimately, a complete breakdown into hallucination.

```
1 Start coding or generate with AI.
```