6-WeNet部署

本文介绍使用Triton Infernecr Server部署WeNet提供语音识别服务的方法

1、非流式部署

1.1 转换预训练模型为onnx格式

```
# FP16转换，同时保存FP32和FP16模型
python export_onnx_gpu.py --config=trt_model/train.yaml --
checkpoint=trt_model/final.pt --cmvn_file=trt_model/global_cmvn --
ctc_weight=0.5 --output_onnx_dir=onnx_trt_model --fp16
```

1.2 tritonserver2.20.0-jetpack5.0启动triton服务（Jetson Xavier NX）

```
# wenet模型部署
/home/lzl/tritonserver2.20.0-jetpack5.0/bin/tritonserver --model-
repository=/home/lzl/wenet/runtime/gpu/model_repo/ --backend-
directory=/home/lzl/lzl/tritonserver2.20.0-jetpack5.0/backends
```

1.3 客户端识别（Ubuntu 虚拟机）

```
cd clients/wenet/
# 测试脚本
python3 client.py --audio_file=test.wav --url=10.26.35.149:8001
# 测试scp音频列表以及计算CER
python3 client.py --wavscp=wav.scp --trans=refer.txt --
url=10.26.35.149:8001
```

1.4 本地onnx模型识别

```
path_dir=/home/lzl/wenet/wenet/bin
onnx_dir=/home/lzl/model/trt_model
test_data_dir=/home/lzl/clients/wenet/
python3 $path_dir/recognize_onnx.py --config=$onnx_dir/train.yaml --
test_data=$test_data_dir/wav.list --dict=$onnx_dir/units.txt --
encoder_onnx=$onnx_dir/encoder_fp16.onnx --
decoder_onnx=$onnx_dir/decoder_fp16.onnx --result_file=results.txt --
gpu=0
```

1.5 本地pt模型识别

```
path_dir=/home/lzl/wenet/wenet/bin
model_dir=/home/lzl/model/pretrained_model
test_data_dir=/home/lzl/clients/wenet/
python3 $path_dir/recognize.py --config=$model_dir/train.yaml --
dict=$model_dir/units.txt --checkpoint=$model_dir/final.pt --
test_data=$test_data_dir/wav.list --result_file=results.txt --gpu=0
```

2、流式部署

2.1 转换预训练模型为onnx格式

导出用于流式推理的流式模型（推理 by chunks））

```
python wenet/bin/export_onnx_gpu.py --
config=20211025_conformer_exp/train.yaml --
checkpoint=quantized/final.pt --
cmvn_file=20211025_conformer_exp/global_cmvn  --ctc_weight=0.5 --
output_onnx_dir=onnx_model_dir --fp16
```

2.2 填写config.pbtxt内容

```
onnx_model_dir=/home/lzl/model/onnx_streaming/
model_repo=/home/lzl/wenet/runtime/gpu/model_repo_stateful/
python3 /home/lzl/wenet/runtime/gpu/scripts/convert.py --
config=$onnx_model_dir/train.yaml --vocab=$onnx_model_dir/units.txt -
-model_repo=$model_repo --onnx_model_dir=$onnx_model_dir
```

2.3 tritonserver2.20.0-jetpack5.0启动triton服务（Jetson Xavier NX）

```
# wenet模型部署
/home/lzl/tritonserver2.20.0-jetpack5.0/bin/tritonserver --model-
repository=/home/lzl/wenet/runtime/gpu/model_repo_stateful/ --
backend-directory=/home/lzl/tritonserver2.20.0-jetpack5.0/backends
```

2.4 客户端识别推理

- 本地识别

```
cd clients/wenet/
python3 client.py --audio_file=test.wav --model_name=streaming_wenet
--streaming --url=localhost:8001
```

- 局域网识别

```
cd clients/wenet/
# 单个音频文件识别
python3 client.py --audio_file=test.wav --model_name=streaming_wenet
--streaming --url=10.26.35.149:8001

# 测试scp音频列表以及计算CER
python3 client.py --wavscp=wav.scp --model_name=streaming_wenet --
streaming --trans=refer.txt --url=10.26.35.149:8001
```

3、实验细节

3.1 Scripts—填写config.pbtxt文件内容（更换模型的时候需要填写）

```
onnx_model_dir=/home/lzl/model/trt_model/
model_repo=/home/lzl/wenet/runtime/gpu/model_repo/
python3 /home/lzl/wenet/runtime/gpu/scripts/convert.py --config=$onnx_model_dir/train.yaml --vocab=$onnx_model_dir/units.txt --model_repo=$model_repo --onnx_model_dir=$onnx_model_dir
```

3.2 将音频文件数据集制作成wav音频格式列表

```
find wav -iname '*.wav' > wav.scp.temp
sed -i 's/\/Session0//g' wav.scp.temp
cat  wav.scp.temp | awk -F '/' '{printf("%s_%s\n",$(NF-1),$NF)}' | sed 's/.wav//' | sed 's/Speaker/Speaker_/' > wav_id
paste -d' ' wav_id wav.scp.temp> wav.scp
rm wav.scp.temp wav_id
```

3.3 Kaldifeat源码安装

```
git clone https://github.com/csukuangfj/kaldifeat
cd kaldifeat
python3 setup.py install
```

3.4 python_backends安装

```
mkdir build
cd build
cmake -DTRITON_ENABLE_GPU=ON DCMAKE_INSTALL_PREFIX:PATH=`pwd`/install ..
make install

cmake -DTRITON_ENABLE_GPU=ON -DTRITON_BACKEND_REPO_TAG=r22.03 -DTRITON_COMMON_REPO_TAG=r22.03 -DTRITON_CORE_REPO_TAG=r22.03 -DCMAKE_INSTALL_PREFIX:PATH=`pwd`/install ..
```

3.5 Dockerfile.server安装内容

```
sudo apt-get update && sudo apt-get -y install swig && sudo apt-get install python3-dev
pip3 install -v kaldifeat
pip3 install pyyaml onnx
# 安装特定版本的cmake
wget https://github.com/Kitware/CMake/releases/download/v3.18.2/cmake-3.18.2-Linux-aarch64.tar.gz
gitclone https://github.com/Slyne/ctc_decoder.git && cd ctc_decoder/swig && sudo bash setup.sh
```

3.6 Dockerfile.client安装内容

```
sudo apt-get update && sudo apt-get install -y libsndfile1
pip3 install soundfile
pip3 install --upgrade clients/python/tritonclient-2.20.0-py3-none-manylinux2014_aarch64.whl[all]
```

4、性能测试

- 离线输入模型

```
python3 generate_perf_input.py --audio_file=test.wav
# offline_input.json generated
perf_analyzer -m attention_rescoring -b 2 --concurrency-range 2 -i gRPC --input-data=offline_input.json -u localhost:8001

perf_analyzer -m attention_rescoring -b 2 --concurrency-range 2 -i gRPC --input-data=offline_input.json -u 10.24.83.22:30323
```

- 流式输入模型

```
python3 generate_perf_input.py --audio_file=test1.wav --streaming

perf_analyzer -u "localhost:8001" -i gRPC --streaming --input-data=online_input.json -m streaming_wenet -b 1 -p 10000 --concurrency-range 16
```