nemo_asr-Copy3

June 19, 2022

[16]:
```
import nemo
import nemo.collections.asr as nemo_asr

from jiwer import wer ##  libarry to compute WER
from statistics import mean  ## Used to compute mean WER
import pandas as pd #for IO

## change this base_path pointing to location of task_Data
base_path = '/home/manju/Desktop/assign/task_data/'

def readDatasetsForDecoding(csv_file):
    data = pd.read_csv(csv_file)
    wav_loc = data['path']
    ground_truth = data['transcription']
    #action = data['action']
    #object_category = data['object']
    #location = data['location']
    return wav_loc, ground_truth


def getAbsoluteWavPath(wav_loc):
    ##Add base path to wav_loc list to get absolute path
    files = []
    for fname in wav_loc:
        files.append(base_path + fname)
    #### files=files[0:2]
    return files


def decodeAndComputeWER(inputFileName, NemoModelName, outputFileName):
    print("*** Decoding wav_data of ", inputFileName, " and writing results to␣
 ↪", outputFileName, "**********")

    ## Reading the csv files
    wav_loc, ground_truth = readDatasetsForDecoding(inputFileName)
    files = getAbsoluteWavPath(wav_loc)
```

```
    index_cnt = 0
    wer_lst = []
    ## files=files[0:10]

    file_wr = open(outputFileName, 'w')
    #print(files)
    out = 'Wave-File-Name , ' + 'hypothesised-Transcription , ' +␣
 ↪'GroundTruth-Transcription' + 'WER' + '\n'
    file_wr.write(out)
    ## Transcribe validation data using NEMO model
    for fname, hypothesis in zip(files, NemoModelName.
 ↪transcribe(paths2audio_files=files)):
        #print(f"{fname},\t \t \"{decoded_transption}\",\t \t␣
 ↪\"{ground_truth[index_cnt]}\"")
        error = wer(ground_truth[index_cnt], hypothesis) ### compute WER
        wer_lst.append(error)
        ## append to write output
        out = '\"' + fname +'\", \"' + hypothesis + '\", \"' +␣
 ↪ground_truth[index_cnt] + '\", \"' +  str(error) + '\"\n'
        ## Write to file
        file_wr.write(out)
        index_cnt += 1 ##increase the indent


    print("Results of decoding written to ", outputFileName, " file.")
    ## Compute Overall WER
    overall_WER = mean(wer_lst) * 100
    print("Overall WER is: ", round(overall_WER, 3), "%")
    print("Total number of files decoded is: ", index_cnt)

    out = '\"Overall WER is: ' + str(round(overall_WER, 3)) + "%, " +'\"Total␣
 ↪number of files : "' + str(index_cnt) + '\n'
    file_wr.write(out)
    file_wr.close() ##close the file


## Read the data and put into different lists
def main(base_path):

    train_file = base_path + 'train_data.csv'
    valid_file = base_path + 'valid_data.csv'

    # Download various variants of Nemo models
    ## refer https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/main/
 ↪asr/results.html#english
```

```
    jasper_model = nemo_asr.models.EncDecCTCModel.
↪from_pretrained(model_name="stt_en_jasper10x5dr")
    quartznet_model = nemo_asr.models.EncDecCTCModel.
↪from_pretrained(model_name="QuartzNet15x5Base-En")
    ## conf_trandcr_model = nemo_asr.models.EncDecRNNTBPEModel.
↪from_pretrained(model_name="stt_en_conformer_transducer_xxlarge")
    ## asr_model = nemo_asr.models.EncDecCTCModelBPE.
↪from_pretrained(model_name="stt_en_conformer_ctc_xlarge")
    ## print(nemo_asr.models.EncDecCTCModelBPE.list_available_models())

    ## Derive hypothesis and compute WER
    ##for NemoModelName in jasper_model, quartznet_model:
    outputFileName = "jasper_validation_results.txt"
    decodeAndComputeWER(valid_file, jasper_model, outputFileName)

    outputFileName = "quartznet_validation_results.txt"
    decodeAndComputeWER(valid_file, quartznet_model, outputFileName)

    print("\n************** For reference : architecture of Jasper nemo model␣
↪************* \n")
    print(jasper_model)

    print("\n************** For reference : architecture of quartznet nemo␣
↪model ************* \n")
    print(quartznet_model)



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

```
[NeMo I 2022-06-19 08:24:12 cloud:56] Found existing object /home/manju/.cache/t
orch/NeMo/NeMo_1.0.0/stt_en_jasper10x5dr/856ae08d5c4bd78b5e27f696e96f7aab/stt_en
_jasper10x5dr.nemo.
[NeMo I 2022-06-19 08:24:12 cloud:62] Re-using file from: /home/manju/.cache/tor
ch/NeMo/NeMo_1.0.0/stt_en_jasper10x5dr/856ae08d5c4bd78b5e27f696e96f7aab/stt_en_j
asper10x5dr.nemo
[NeMo I 2022-06-19 08:24:12 common:675] Instantiating model from pre-trained
checkpoint

[NeMo W 2022-06-19 08:24:20 modelPT:137] If you intend to do training or fine-
tuning, please call the ModelPT.setup_training_data() method and provide a valid
configuration file to setup the train data loader.
    Train config :
    manifest_filepath: /data2/voices/train_1k.json
    sample_rate: 16000
    labels:
```

```
    - ' '
    - a
    - b
    - c
    - d
    - e
    - f
    - g
    - h
    - i
    - j
    - k
    - l
    - m
    - 'n'
    - o
    - p
    - q
    - r
    - s
    - t
    - u
    - v
    - w
    - x
    - 'y'
    - z
    - ''''
    batch_size: 32
    trim_silence: true
    max_duration: 16.7
    shuffle: true
    is_tarred: false
    tarred_audio_filepaths: null

[NeMo W 2022-06-19 08:24:20 modelPT:144] If you intend to do validation, please
call the ModelPT.setup_validation_data() or
ModelPT.setup_multiple_validation_data() method and provide a valid
configuration file to setup the validation data loader(s).
    Validation config :
    manifest_filepath: /data2/voices/train_1k_samp.json
    sample_rate: 16000
    labels:
    - ' '
    - a
    - b
    - c
    - d
```

```
    - e
    - f
    - g
    - h
    - i
    - j
    - k
    - l
    - m
    - 'n'
    - o
    - p
    - q
    - r
    - s
    - t
    - u
    - v
    - w
    - x
    - 'y'
    - z
    - ''''
    batch_size: 32
    shuffle: false


[NeMo I 2022-06-19 08:24:20 features:252] PADDING: 16
[NeMo I 2022-06-19 08:24:20 features:269] STFT using torch

[NeMo W 2022-06-19 08:24:20 nemo_logging:349]
/home/manju/anaconda3/lib/python3.9/site-
packages/nemo/collections/asr/parts/features.py:302: FutureWarning: Pass
sr=16000, n_fft=512 as keyword args. From version 0.10 passing these as
positional arguments will result in an error
      librosa.filters.mel(sample_rate, self.n_fft, n_mels=nfilt, fmin=lowfreq,
fmax=highfreq), dtype=torch.float


[NeMo I 2022-06-19 08:24:26 modelPT:434] Model EncDecCTCModel was successfully
restored from /home/manju/.cache/torch/NeMo/NeMo_1.0.0/stt_en_jasper10x5dr/856ae
08d5c4bd78b5e27f696e96f7aab/stt_en_jasper10x5dr.nemo.
[NeMo I 2022-06-19 08:24:26 cloud:56] Found existing object
/home/manju/.cache/torch/NeMo/NeMo_1.0.0/QuartzNet15x5Base-
En/2b066be39e9294d7100fb176ec817722/QuartzNet15x5Base-En.nemo.
[NeMo I 2022-06-19 08:24:26 cloud:62] Re-using file from:
/home/manju/.cache/torch/NeMo/NeMo_1.0.0/QuartzNet15x5Base-
En/2b066be39e9294d7100fb176ec817722/QuartzNet15x5Base-En.nemo
[NeMo I 2022-06-19 08:24:26 common:675] Instantiating model from pre-trained
```

```
checkpoint
[NeMo I 2022-06-19 08:24:26 features:252] PADDING: 16
[NeMo I 2022-06-19 08:24:26 features:269] STFT using torch
[NeMo I 2022-06-19 08:24:27 modelPT:434] Model EncDecCTCModel was successfully
restored from /home/manju/.cache/torch/NeMo/NeMo_1.0.0/QuartzNet15x5Base-
En/2b066be39e9294d7100fb176ec817722/QuartzNet15x5Base-En.nemo.
*** Decoding wav_data of  /home/manju/Desktop/assign/task_data/valid_data.csv
and writing results to  jasper_validation_results.txt *********

Transcribing:   0%|          | 0/780 [00:00<?, ?it/s]

[NeMo W 2022-06-19 08:24:27 nemo_logging:349]
/home/manju/anaconda3/lib/python3.9/site-
packages/nemo/collections/asr/parts/segment.py:67: FutureWarning: Pass
orig_sr=8000, target_sr=16000 as keyword args. From version 0.10 passing these
as positional arguments will result in an error
      samples = librosa.core.resample(samples, sample_rate, target_sr)

[NeMo W 2022-06-19 08:24:27 nemo_logging:349]
/home/manju/anaconda3/lib/python3.9/site-
packages/nemo/collections/asr/parts/segment.py:70: FutureWarning: Pass top_db=60
as keyword args. From version 0.10 passing these as positional arguments will
result in an error
      samples, _ = librosa.effects.trim(samples, trim_db)

[NeMo W 2022-06-19 08:24:27 nemo_logging:349]
/home/manju/anaconda3/lib/python3.9/site-packages/torch/autocast_mode.py:162:
UserWarning: User provided device_type of 'cuda', but CUDA is not available.
Disabling
      warnings.warn('User provided device_type of \'cuda\', but CUDA is not
available. Disabling')

[NeMo W 2022-06-19 08:24:27 nemo_logging:349]
/home/manju/anaconda3/lib/python3.9/site-
packages/nemo/collections/asr/parts/jasper.py:303: UserWarning: __floordiv__ is
deprecated, and its behavior will change in a future version of pytorch. It
currently rounds toward 0 (like the 'trunc' function NOT 'floor'). This results
in incorrect rounding for negative values. To keep the current behavior, use
torch.div(a, b, rounding_mode='trunc'), or for actual floor division, use
torch.div(a, b, rounding_mode='floor').
      return (


Results of decoding written to  jasper_validation_results.txt  file.
Overall WER is:  47.805 %
Total number of files decoded is:  3118
*** Decoding wav_data of  /home/manju/Desktop/assign/task_data/valid_data.csv
and writing results to  quartznet_validation_results.txt *********

Transcribing:   0%|          | 0/780 [00:00<?, ?it/s]
```

```
[NeMo W 2022-06-19 08:45:32 nemo_logging:349]
/home/manju/anaconda3/lib/python3.9/site-
packages/nemo/collections/asr/parts/segment.py:67: FutureWarning: Pass
orig_sr=8000, target_sr=16000 as keyword args. From version 0.10 passing these
as positional arguments will result in an error
      samples = librosa.core.resample(samples, sample_rate, target_sr)

[NeMo W 2022-06-19 08:45:32 nemo_logging:349]
/home/manju/anaconda3/lib/python3.9/site-
packages/nemo/collections/asr/parts/segment.py:70: FutureWarning: Pass top_db=60
as keyword args. From version 0.10 passing these as positional arguments will
result in an error
      samples, _ = librosa.effects.trim(samples, trim_db)

[NeMo W 2022-06-19 08:45:32 nemo_logging:349]
/home/manju/anaconda3/lib/python3.9/site-packages/torch/autocast_mode.py:162:
UserWarning: User provided device_type of 'cuda', but CUDA is not available.
Disabling
      warnings.warn('User provided device_type of \'cuda\', but CUDA is not
available. Disabling')

[NeMo W 2022-06-19 08:45:32 nemo_logging:349]
/home/manju/anaconda3/lib/python3.9/site-
packages/nemo/collections/asr/parts/jasper.py:303: UserWarning: __floordiv__ is
deprecated, and its behavior will change in a future version of pytorch. It
currently rounds toward 0 (like the 'trunc' function NOT 'floor'). This results
in incorrect rounding for negative values. To keep the current behavior, use
torch.div(a, b, rounding_mode='trunc'), or for actual floor division, use
torch.div(a, b, rounding_mode='floor').
      return (


Results of decoding written to  quartznet_validation_results.txt  file.
Overall WER is:  50.412 %
Total number of files decoded is:  3118

************** For reference : architecture of Jasper nemo model *************

EncDecCTCModel(
  (preprocessor): AudioToMelSpectrogramPreprocessor(
    (featurizer): FilterbankFeatures()
  )
  (encoder): ConvASREncoder(
    (encoder): Sequential(
      (0): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(64, 256, kernel_size=(11,), stride=(2,),
```

```
padding=(5,), bias=False)
          )
          (1): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.2, inplace=False)
        )
      )
      (1): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(11,), stride=(1,),
padding=(5,), bias=False)
          )
          (1): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Dropout(p=0.2, inplace=False)
          (4): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(11,), stride=(1,),
padding=(5,), bias=False)
          )
          (5): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (6): ReLU(inplace=True)
          (7): Dropout(p=0.2, inplace=False)
          (8): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(11,), stride=(1,),
padding=(5,), bias=False)
          )
          (9): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (10): ReLU(inplace=True)
          (11): Dropout(p=0.2, inplace=False)
          (12): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(11,), stride=(1,),
padding=(5,), bias=False)
          )
          (13): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (14): ReLU(inplace=True)
          (15): Dropout(p=0.2, inplace=False)
          (16): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(11,), stride=(1,),
padding=(5,), bias=False)
          )
```

```
          (17): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.2, inplace=False)
        )
      )
      (2): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(11,), stride=(1,),
padding=(5,), bias=False)
          )
          (1): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Dropout(p=0.2, inplace=False)
          (4): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(11,), stride=(1,),
padding=(5,), bias=False)
          )
          (5): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (6): ReLU(inplace=True)
          (7): Dropout(p=0.2, inplace=False)
          (8): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(11,), stride=(1,),
padding=(5,), bias=False)
          )
          (9): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (10): ReLU(inplace=True)
          (11): Dropout(p=0.2, inplace=False)
          (12): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(11,), stride=(1,),
padding=(5,), bias=False)
          )
```

```
          (13): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (14): ReLU(inplace=True)
          (15): Dropout(p=0.2, inplace=False)
          (16): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(11,), stride=(1,),
padding=(5,), bias=False)
          )
          (17): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (1): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.2, inplace=False)
        )
      )
      (3): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(256, 384, kernel_size=(13,), stride=(1,),
padding=(6,), bias=False)
          )
          (1): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Dropout(p=0.2, inplace=False)
          (4): MaskedConv1d(
            (conv): Conv1d(384, 384, kernel_size=(13,), stride=(1,),
padding=(6,), bias=False)
          )
```

```
          (5): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (6): ReLU(inplace=True)
          (7): Dropout(p=0.2, inplace=False)
          (8): MaskedConv1d(
            (conv): Conv1d(384, 384, kernel_size=(13,), stride=(1,),
padding=(6,), bias=False)
          )
          (9): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (10): ReLU(inplace=True)
          (11): Dropout(p=0.2, inplace=False)
          (12): MaskedConv1d(
            (conv): Conv1d(384, 384, kernel_size=(13,), stride=(1,),
padding=(6,), bias=False)
          )
          (13): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (14): ReLU(inplace=True)
          (15): Dropout(p=0.2, inplace=False)
          (16): MaskedConv1d(
            (conv): Conv1d(384, 384, kernel_size=(13,), stride=(1,),
padding=(6,), bias=False)
          )
          (17): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 384, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (1): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 384, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (2): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 384, kernel_size=(1,), stride=(1,),
bias=False)
```

```
            )
            (1): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.2, inplace=False)
        )
      )
      (4): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(384, 384, kernel_size=(13,), stride=(1,),
padding=(6,), bias=False)
          )
          (1): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Dropout(p=0.2, inplace=False)
          (4): MaskedConv1d(
            (conv): Conv1d(384, 384, kernel_size=(13,), stride=(1,),
padding=(6,), bias=False)
          )
          (5): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (6): ReLU(inplace=True)
          (7): Dropout(p=0.2, inplace=False)
          (8): MaskedConv1d(
            (conv): Conv1d(384, 384, kernel_size=(13,), stride=(1,),
padding=(6,), bias=False)
          )
          (9): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (10): ReLU(inplace=True)
          (11): Dropout(p=0.2, inplace=False)
          (12): MaskedConv1d(
            (conv): Conv1d(384, 384, kernel_size=(13,), stride=(1,),
padding=(6,), bias=False)
          )
          (13): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (14): ReLU(inplace=True)
          (15): Dropout(p=0.2, inplace=False)
          (16): MaskedConv1d(
            (conv): Conv1d(384, 384, kernel_size=(13,), stride=(1,),
padding=(6,), bias=False)
          )
```

```
          (17): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 384, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (1): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 384, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (2): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 384, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (3): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(384, 384, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(384, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.2, inplace=False)
        )
      )
      (5): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(384, 512, kernel_size=(17,), stride=(1,),
padding=(8,), bias=False)
          )
```

```
          (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Dropout(p=0.2, inplace=False)
          (4): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(17,), stride=(1,),
padding=(8,), bias=False)
          )
          (5): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (6): ReLU(inplace=True)
          (7): Dropout(p=0.2, inplace=False)
          (8): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(17,), stride=(1,),
padding=(8,), bias=False)
          )
          (9): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (10): ReLU(inplace=True)
          (11): Dropout(p=0.2, inplace=False)
          (12): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(17,), stride=(1,),
padding=(8,), bias=False)
          )
          (13): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (14): ReLU(inplace=True)
          (15): Dropout(p=0.2, inplace=False)
          (16): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(17,), stride=(1,),
padding=(8,), bias=False)
          )
          (17): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (1): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 512, kernel_size=(1,), stride=(1,),
bias=False)
```

```
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (2): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (3): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(384, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (4): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(384, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.2, inplace=False)
        )
      )
      (6): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(17,), stride=(1,),
padding=(8,), bias=False)
          )
          (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Dropout(p=0.2, inplace=False)
          (4): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(17,), stride=(1,),
padding=(8,), bias=False)
          )
```

```
          (5): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (6): ReLU(inplace=True)
          (7): Dropout(p=0.2, inplace=False)
          (8): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(17,), stride=(1,),
padding=(8,), bias=False)
          )
          (9): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (10): ReLU(inplace=True)
          (11): Dropout(p=0.2, inplace=False)
          (12): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(17,), stride=(1,),
padding=(8,), bias=False)
          )
          (13): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (14): ReLU(inplace=True)
          (15): Dropout(p=0.2, inplace=False)
          (16): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(17,), stride=(1,),
padding=(8,), bias=False)
          )
          (17): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (1): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (2): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 512, kernel_size=(1,), stride=(1,),
bias=False)
```

```
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (3): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(384, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (4): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(384, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (5): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.2, inplace=False)
        )
      )
      (7): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(512, 640, kernel_size=(21,), stride=(1,),
padding=(10,), bias=False)
          )
          (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Dropout(p=0.3, inplace=False)
          (4): MaskedConv1d(
            (conv): Conv1d(640, 640, kernel_size=(21,), stride=(1,),
padding=(10,), bias=False)
          )
```

```
          (5): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (6): ReLU(inplace=True)
          (7): Dropout(p=0.3, inplace=False)
          (8): MaskedConv1d(
            (conv): Conv1d(640, 640, kernel_size=(21,), stride=(1,),
padding=(10,), bias=False)
          )
          (9): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (10): ReLU(inplace=True)
          (11): Dropout(p=0.3, inplace=False)
          (12): MaskedConv1d(
            (conv): Conv1d(640, 640, kernel_size=(21,), stride=(1,),
padding=(10,), bias=False)
          )
          (13): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (14): ReLU(inplace=True)
          (15): Dropout(p=0.3, inplace=False)
          (16): MaskedConv1d(
            (conv): Conv1d(640, 640, kernel_size=(21,), stride=(1,),
padding=(10,), bias=False)
          )
          (17): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 640, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (1): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 640, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (2): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 640, kernel_size=(1,), stride=(1,),
bias=False)
```

```
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (3): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(384, 640, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (4): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(384, 640, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (5): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 640, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (6): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 640, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.3, inplace=False)
        )
      )
      (8): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(640, 640, kernel_size=(21,), stride=(1,),
padding=(10,), bias=False)
          )
```

```
          (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Dropout(p=0.3, inplace=False)
          (4): MaskedConv1d(
            (conv): Conv1d(640, 640, kernel_size=(21,), stride=(1,),
padding=(10,), bias=False)
          )
          (5): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (6): ReLU(inplace=True)
          (7): Dropout(p=0.3, inplace=False)
          (8): MaskedConv1d(
            (conv): Conv1d(640, 640, kernel_size=(21,), stride=(1,),
padding=(10,), bias=False)
          )
          (9): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (10): ReLU(inplace=True)
          (11): Dropout(p=0.3, inplace=False)
          (12): MaskedConv1d(
            (conv): Conv1d(640, 640, kernel_size=(21,), stride=(1,),
padding=(10,), bias=False)
          )
          (13): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (14): ReLU(inplace=True)
          (15): Dropout(p=0.3, inplace=False)
          (16): MaskedConv1d(
            (conv): Conv1d(640, 640, kernel_size=(21,), stride=(1,),
padding=(10,), bias=False)
          )
          (17): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 640, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (1): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 640, kernel_size=(1,), stride=(1,),
bias=False)
```

```
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (2): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 640, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (3): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(384, 640, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (4): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(384, 640, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (5): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 640, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (6): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 640, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (7): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(640, 640, kernel_size=(1,), stride=(1,),
bias=False)
```

```
            )
            (1): BatchNorm1d(640, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.3, inplace=False)
        )
      )
      (9): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(640, 768, kernel_size=(25,), stride=(1,),
padding=(12,), bias=False)
          )
          (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Dropout(p=0.3, inplace=False)
          (4): MaskedConv1d(
            (conv): Conv1d(768, 768, kernel_size=(25,), stride=(1,),
padding=(12,), bias=False)
          )
          (5): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (6): ReLU(inplace=True)
          (7): Dropout(p=0.3, inplace=False)
          (8): MaskedConv1d(
            (conv): Conv1d(768, 768, kernel_size=(25,), stride=(1,),
padding=(12,), bias=False)
          )
          (9): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (10): ReLU(inplace=True)
          (11): Dropout(p=0.3, inplace=False)
          (12): MaskedConv1d(
            (conv): Conv1d(768, 768, kernel_size=(25,), stride=(1,),
padding=(12,), bias=False)
          )
          (13): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (14): ReLU(inplace=True)
          (15): Dropout(p=0.3, inplace=False)
          (16): MaskedConv1d(
            (conv): Conv1d(768, 768, kernel_size=(25,), stride=(1,),
padding=(12,), bias=False)
          )
```

```
          (17): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (1): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (2): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (3): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(384, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (4): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(384, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (5): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 768, kernel_size=(1,), stride=(1,),
bias=False)
```

```
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (6): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (7): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(640, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (8): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(640, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.3, inplace=False)
        )
      )
      (10): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(768, 768, kernel_size=(25,), stride=(1,),
padding=(12,), bias=False)
          )
          (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Dropout(p=0.3, inplace=False)
          (4): MaskedConv1d(
            (conv): Conv1d(768, 768, kernel_size=(25,), stride=(1,),
padding=(12,), bias=False)
          )
```

```
          (5): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (6): ReLU(inplace=True)
          (7): Dropout(p=0.3, inplace=False)
          (8): MaskedConv1d(
            (conv): Conv1d(768, 768, kernel_size=(25,), stride=(1,),
padding=(12,), bias=False)
          )
          (9): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (10): ReLU(inplace=True)
          (11): Dropout(p=0.3, inplace=False)
          (12): MaskedConv1d(
            (conv): Conv1d(768, 768, kernel_size=(25,), stride=(1,),
padding=(12,), bias=False)
          )
          (13): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (14): ReLU(inplace=True)
          (15): Dropout(p=0.3, inplace=False)
          (16): MaskedConv1d(
            (conv): Conv1d(768, 768, kernel_size=(25,), stride=(1,),
padding=(12,), bias=False)
          )
          (17): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (1): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (2): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 768, kernel_size=(1,), stride=(1,),
bias=False)
```

```
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (3): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(384, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (4): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(384, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (5): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (6): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (7): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(640, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (8): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(640, 768, kernel_size=(1,), stride=(1,),
bias=False)
```

```
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (9): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(768, 768, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(768, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.3, inplace=False)
        )
      )
      (11): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(768, 896, kernel_size=(29,), stride=(1,),
padding=(28,), dilation=(2,), bias=False)
          )
          (1): BatchNorm1d(896, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.4, inplace=False)
        )
      )
      (12): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(896, 1024, kernel_size=(1,), stride=(1,), bias=False)
          )
          (1): BatchNorm1d(1024, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.4, inplace=False)
        )
      )
    )
  )
  (decoder): ConvASRDecoder(
```

```
    (decoder_layers): Sequential(
      (0): Conv1d(1024, 29, kernel_size=(1,), stride=(1,))
    )
  )
  (loss): CTCLoss()
  (spec_augmentation): SpectrogramAugmentation(
    (spec_cutout): SpecCutout()
  )
  (_wer): WER()
)

************** For reference : architecture of quartznet nemo model
*************

EncDecCTCModel(
  (preprocessor): AudioToMelSpectrogramPreprocessor(
    (featurizer): FilterbankFeatures()
  )
  (encoder): ConvASREncoder(
    (encoder): Sequential(
      (0): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(64, 64, kernel_size=(33,), stride=(2,),
padding=(16,), groups=64, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(64, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (1): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
```

```
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
          (5): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (6): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (7): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (8): ReLU(inplace=True)
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (16): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (17): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
          (20): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
```

```
              (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (2): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
          (5): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (6): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (7): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (8): ReLU(inplace=True)
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
```

```
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (16): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (17): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
          (20): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (3): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
```

```
          (5): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (6): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (7): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (8): ReLU(inplace=True)
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (16): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (17): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
          (20): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(33,), stride=(1,),
padding=(16,), groups=256, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,),
bias=False)
```

```
            )
            (1): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (4): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
padding=(19,), groups=256, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
          (5): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
padding=(19,), groups=256, bias=False)
          )
          (6): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (7): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (8): ReLU(inplace=True)
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
padding=(19,), groups=256, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
padding=(19,), groups=256, bias=False)
```

```
          )
          (16): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (17): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
          (20): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
padding=(19,), groups=256, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (5): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
padding=(19,), groups=256, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
          (5): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
```

```
padding=(19,), groups=256, bias=False)
          )
          (6): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (7): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (8): ReLU(inplace=True)
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
padding=(19,), groups=256, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
padding=(19,), groups=256, bias=False)
          )
          (16): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (17): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
          (20): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
padding=(19,), groups=256, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
```

```
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (6): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
padding=(19,), groups=256, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
          (5): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
padding=(19,), groups=256, bias=False)
          )
          (6): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (7): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (8): ReLU(inplace=True)
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
padding=(19,), groups=256, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
padding=(19,), groups=256, bias=False)
          )
          (16): MaskedConv1d(
```

```
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (17): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
          (20): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(39,), stride=(1,),
padding=(19,), groups=256, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 256, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(256, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (7): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(256, 256, kernel_size=(51,), stride=(1,),
padding=(25,), groups=256, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(256, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
          (5): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(51,), stride=(1,),
padding=(25,), groups=512, bias=False)
          )
```

```
          (6): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (7): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (8): ReLU(inplace=True)
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(51,), stride=(1,),
padding=(25,), groups=512, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(51,), stride=(1,),
padding=(25,), groups=512, bias=False)
          )
          (16): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (17): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
          (20): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(51,), stride=(1,),
padding=(25,), groups=512, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(256, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
```

```
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (8): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(51,), stride=(1,),
padding=(25,), groups=512, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
          (5): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(51,), stride=(1,),
padding=(25,), groups=512, bias=False)
          )
          (6): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (7): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (8): ReLU(inplace=True)
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(51,), stride=(1,),
padding=(25,), groups=512, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(51,), stride=(1,),
padding=(25,), groups=512, bias=False)
          )
          (16): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
```

```
          (17): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
          (20): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(51,), stride=(1,),
padding=(25,), groups=512, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (9): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(51,), stride=(1,),
padding=(25,), groups=512, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
          (5): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(51,), stride=(1,),
padding=(25,), groups=512, bias=False)
          )
          (6): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
```

```
          )
          (7): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (8): ReLU(inplace=True)
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(51,), stride=(1,),
padding=(25,), groups=512, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(51,), stride=(1,),
padding=(25,), groups=512, bias=False)
          )
          (16): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (17): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
          (20): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(51,), stride=(1,),
padding=(25,), groups=512, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
```

```
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (10): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
          (5): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (6): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (7): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (8): ReLU(inplace=True)
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (16): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (17): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
```

```
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
          (20): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (11): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
          (5): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (6): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (7): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
```

```
track_running_stats=True)
          (8): ReLU(inplace=True)
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (16): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (17): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
          (20): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
```

```
        )
      )
      (12): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
          (5): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (6): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (7): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (8): ReLU(inplace=True)
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (16): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (17): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
```

```
          (20): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(63,), stride=(1,),
padding=(31,), groups=512, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (13): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
padding=(37,), groups=512, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
          (5): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
padding=(37,), groups=512, bias=False)
          )
          (6): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (7): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (8): ReLU(inplace=True)
```

```
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
padding=(37,), groups=512, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
padding=(37,), groups=512, bias=False)
          )
          (16): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (17): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
          (20): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
padding=(37,), groups=512, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
```

```
      (14): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
padding=(37,), groups=512, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
          (5): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
padding=(37,), groups=512, bias=False)
          )
          (6): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (7): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (8): ReLU(inplace=True)
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
padding=(37,), groups=512, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
padding=(37,), groups=512, bias=False)
          )
          (16): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (17): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
          (20): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
```

```
padding=(37,), groups=512, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (15): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
padding=(37,), groups=512, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (3): ReLU(inplace=True)
          (4): Dropout(p=0.0, inplace=False)
          (5): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
padding=(37,), groups=512, bias=False)
          )
          (6): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (7): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (8): ReLU(inplace=True)
          (9): Dropout(p=0.0, inplace=False)
          (10): MaskedConv1d(
```

```
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
padding=(37,), groups=512, bias=False)
          )
          (11): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (12): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (13): ReLU(inplace=True)
          (14): Dropout(p=0.0, inplace=False)
          (15): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
padding=(37,), groups=512, bias=False)
          )
          (16): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (17): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          (18): ReLU(inplace=True)
          (19): Dropout(p=0.0, inplace=False)
          (20): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(75,), stride=(1,),
padding=(37,), groups=512, bias=False)
          )
          (21): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (22): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (res): ModuleList(
          (0): ModuleList(
            (0): MaskedConv1d(
              (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,),
bias=False)
            )
            (1): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (16): JasperBlock(
        (mconv): ModuleList(
```

```
          (0): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(87,), stride=(1,),
padding=(86,), dilation=(2,), groups=512, bias=False)
          )
          (1): MaskedConv1d(
            (conv): Conv1d(512, 512, kernel_size=(1,), stride=(1,), bias=False)
          )
          (2): BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
      (17): JasperBlock(
        (mconv): ModuleList(
          (0): MaskedConv1d(
            (conv): Conv1d(512, 1024, kernel_size=(1,), stride=(1,), bias=False)
          )
          (1): BatchNorm1d(1024, eps=0.001, momentum=0.1, affine=True,
track_running_stats=True)
        )
        (mout): Sequential(
          (0): ReLU(inplace=True)
          (1): Dropout(p=0.0, inplace=False)
        )
      )
    )
  )
  (decoder): ConvASRDecoder(
    (decoder_layers): Sequential(
      (0): Conv1d(1024, 29, kernel_size=(1,), stride=(1,))
    )
  )
  (loss): CTCLoss()
  (spec_augmentation): SpectrogramAugmentation(
    (spec_cutout): SpecCutout()
  )
  (_wer): WER()
)
```

[ ]: