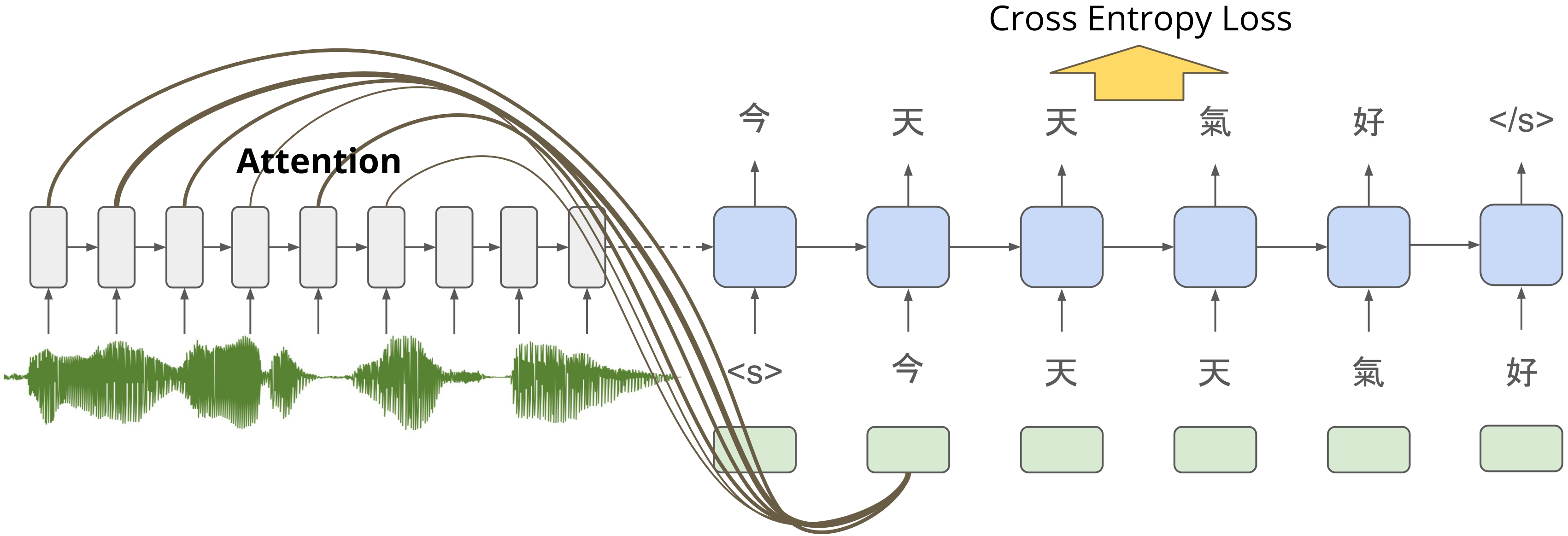
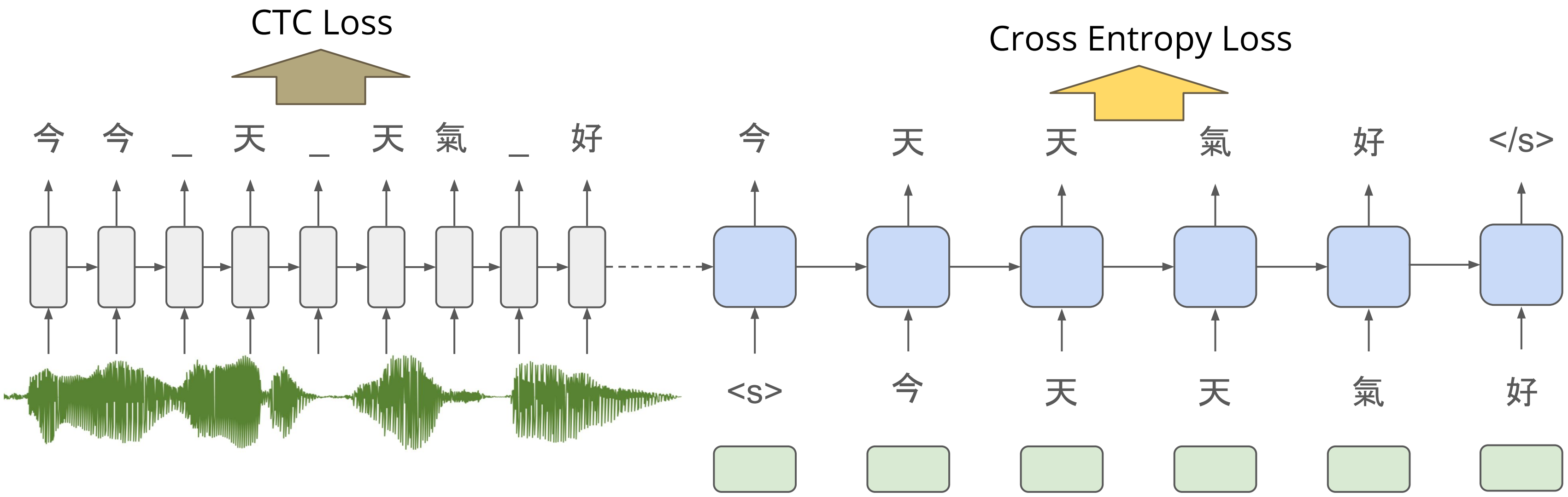
DLHLP - HW1
End-to-end Speech Recognition

TA: 莊永松、柯上優
dlhlp.ta@gmail.com

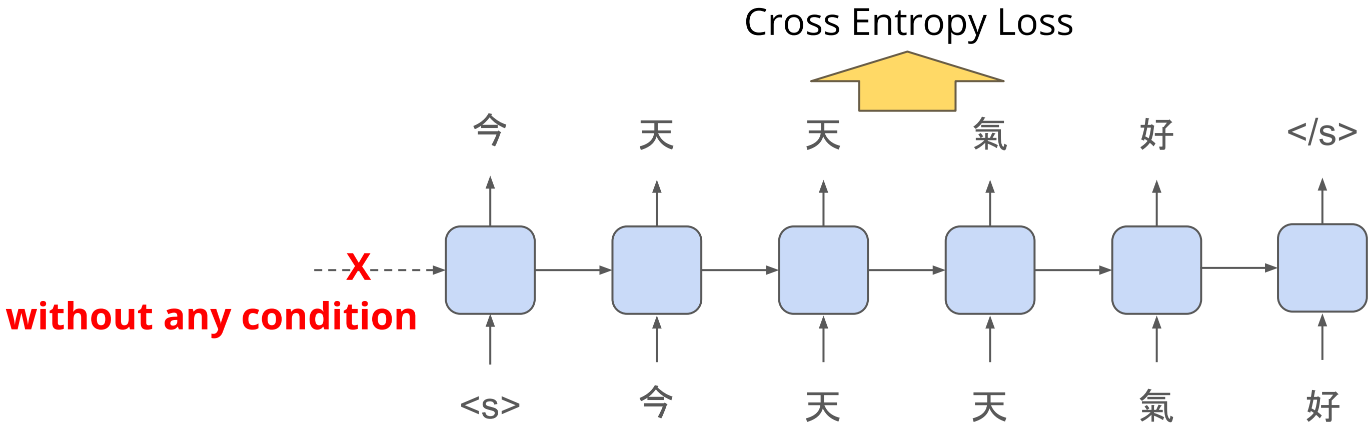
Seq2Seq attention-based ASR



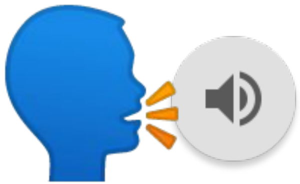
Joint CTC-attention based ASR



RNN-LM (for re-scoring)



Dataset



To make HW1 easier...

- We use a small 10-hours Mandarin Chinese audio corpus
- All from a single speaker (no speaker varient problem)
- We transform text to ㄅㄆㄇㄈ to make vocabulary size smaller and help the model to converge faster
- You would not need to train a “real-world” ASR model
- About **2~4 hours** to converge on K80, while real-world ASR may need weeks to converge

Download Data

- Download Data(2.88G)
    - https://drive.google.com/file/d/1BPR3IfAEOFOQzsU4vDs2fezDyEsU7bvQ/view?usp=sharing
    - train: 8000, dev: 1000, test: 1000
    - the transcript file in testing set is not removed for convenience, but all the answer is replaced with ㄅㄆㄇㄈ...

```
/DLHLP
├ /train
│ ├ 000001.wav
│ ├ ......
│ ├ 008000.wav
│ └ bopomo.trans.txt
├ /dev
│ ├ 008001.wav
│ ├ ......
│ ├ 009000.wav
│ └ bopomo.trans.txt
├ /test
│ ├ 009001.wav
│ ├ ......
│ ├ 010000.wav
│ └ bopomo.trans.txt
└ text-data.txt
```

Hint: In Linux, use `bash get_dataset.sh`
https://github.com/DLHLP2020/hw1-speech-recognition/blob/master/get_dataset.sh

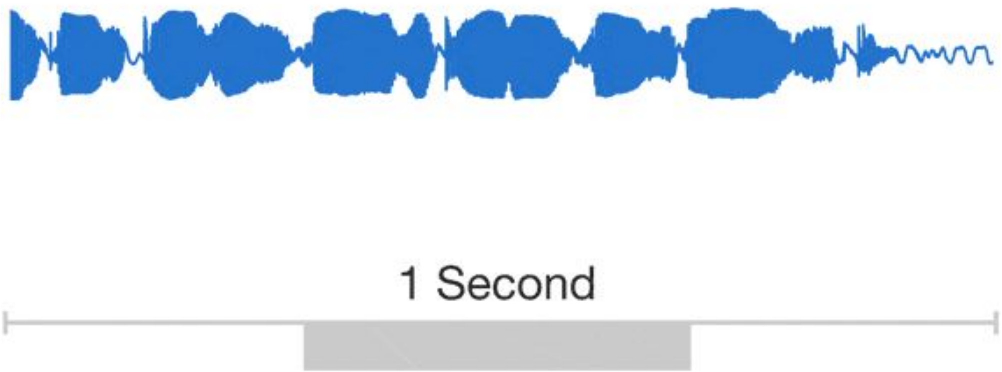
Extract features from audio file w/ torchaudio

Step 1. Load WAV File

```
waveform, sample_freq = torchaudio.load("000001.wav")
```

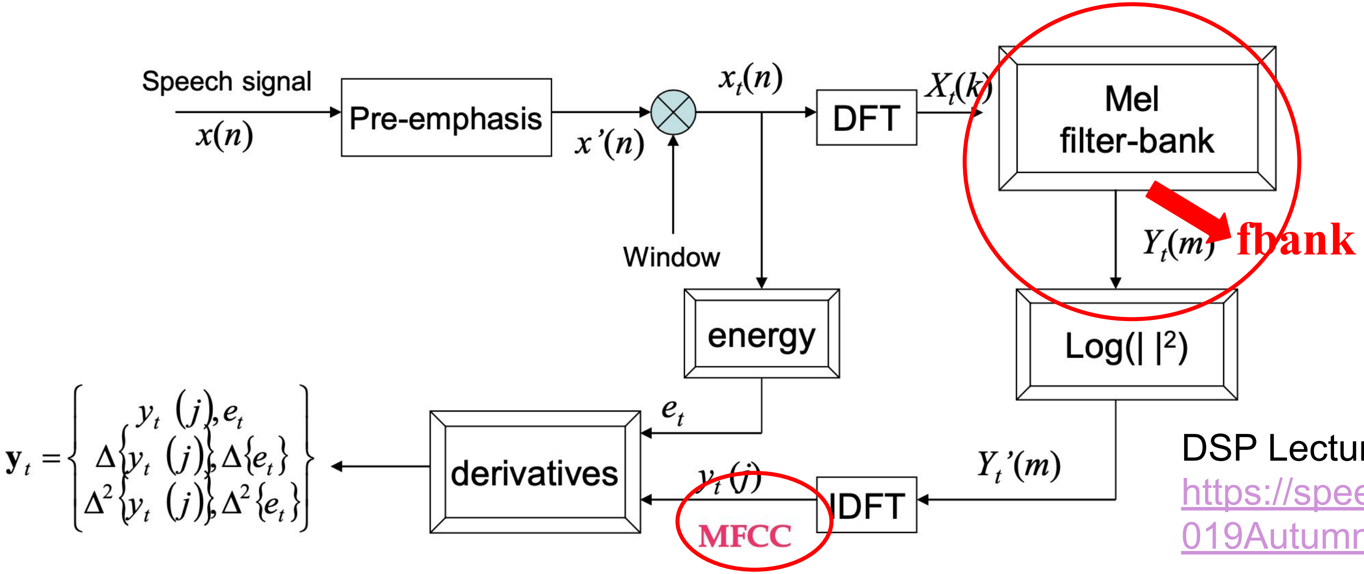
waveform.shape: (1, 116400)

sample_freq: 48000



Step 2. Get fbank or MFCC from waveform

- **Mel-Frequency Cepstral Coefficients (MFCC)**
  - Most widely used in the speech recognition
  - Has generally obtained a better accuracy at relatively low computational complexity
  - The process of MFCC extraction :



DSP Lecture 7:
https://speech.ee.ntu.edu.tw/DSP2019Autumn/Slides/7.0.pptx

Step 2. Get Fbank or MFCC from waveform

MFCC

```
feature = torchaudio.compliance.kaldi.mfcc(waveform,
                                           sample_frequency=sample_freq)
```

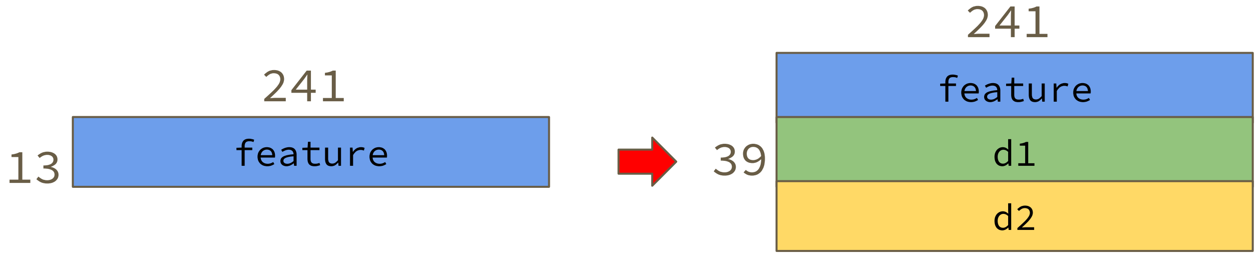
feature.shape: (241, 13)

Fbank

```
feature = torchaudio.compliance.kaldi.fbank(waveform,
                                            sample_frequency=sample_freq,
                                            num_mel_bins=40)
```

feature.shape: (241, 40)

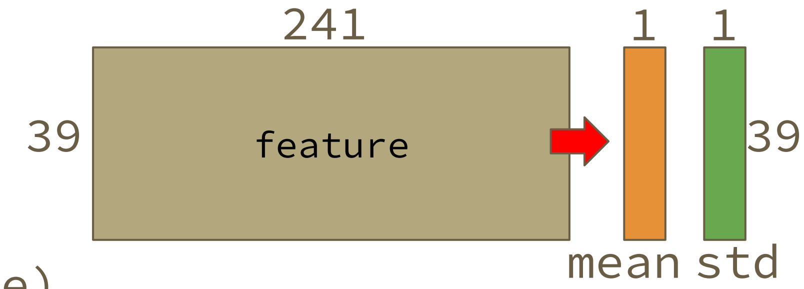
Step 3. Add Deltas




```
d1 = torchaudio.functional.compute_deltas(feature)
d2 = torchaudio.functional.compute_deltas(d1)
feature = torch.cat([feature, d1, d2], dim=-1)
```

<u>`feature.shape: (241, 39) or (241, 120)`</u>

Step 4. CMVN (Normalization)




```
eps = 1e-10
mean = feature.mean(0, keepdim=True)
std = feature.std(0, keepdim=True)
feature = (feature - mean) / (std + eps)
```

Sample code (strongly recommended)

https://github.com/Alexander-H-Liu/End-to-end-ASR-Pytorch

A completed & stable ASR implementation (Thanks to 劉浩然學長!)

Please read README before using it!

Alexander-H-Liu / **End-to-end-ASR-Pytorch**

Watch 29 | Unstar 472 | Fork 143

Code | Issues 11 | Pull requests 1 | Actions | Projects 0 | Wiki | Security | Insights

This is an open source project (formerly named Listen, Attend and Spell - PyTorch Implementation) for end-to-end ASR implemented with Pytorch, the well known deep learning toolkit.

73 commits | **3** branches | **0** packages | **0** releases | **3** contributors | MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

Alexander-H-Liu fix adadelta init for testing & char decoder repeat issue — Latest commit `fa6683b` 28 days ago

bin	fix adadelta init for testing & char decoder repeat issue	28 days ago
config	hybrid typo fixed	2 months ago
corpus	Fixed scheduler, corpus loading assertion, defualt lm	4 months ago
src	fix adadelta init for testing & char decoder repeat issue	28 days ago
tests	rename subword 16k	4 months ago
util	readme draft, libri example	4 months ago
.gitignore	add *.swp	7 months ago
LICENSE	Rebase to clear history	12 months ago

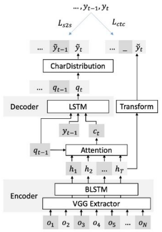
End-to-end Automatic Speech Recognition Systems - PyTorch Implementation

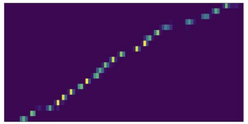
This is an open source project (formerly named **Listen, Attend and Spell - PyTorch Implementation**) for end-to-end ASR by Tzu-Wei Sung and me. Implementation was mostly done with Pytorch, the well known deep learning toolkit.

The end-to-end ASR was based on Listen, Attend and Spell[1]. Multiple techniques proposed recently were also implemented, serving as additional plug-ins for better performance. For the list of techniques implemented, please refer to the highlights, configuration and references.

Feel free to use/modify them, any bug report or improvement suggestion will be appreciated. If you find this project helpful for your research, please do consider to cite our paper, thanks!

Highlights





TODO: Add new Dataset

- See the example corpus/librispeech.py
- Create a copy corpus/dlhlp.py. Replace all “`Libri`” to “`Dlhlp`” in the file.
    1. open each ***.flac** files → open each ***.wav** files
    2. def read_text()
        - origin: src_file = '-'.join(file.split('-')[:-1])+'.trans.txt'
        - ours: src_file = file.rsplit('/', 1)[0]+'/bopomo.trans.txt'
        - Base on the format of our dataset

```python
class DlhlpDataset(Dataset):
    def __init__(self, path, split, tokenizer, bucket_size, ascending=False):
        # Setup
        self.path = path
        self.bucket_size = bucket_size

        # List all wave files
        file_list = []
        for s in split:
            split_list = list(Path(join(path, s)).rglob("*.wav"))
            assert len(split_list) > 0, "No data found @ {}".format(join(path,s))
            file_list += split_list
```

```python
def read_text(file):
    '''Get transcription of target wave file,
       it's somewhat redundant for accessing each txt multiplt times,
       but it works fine with multi-thread'''
    src_file = file.rsplit('/', 1)[0]+'/bopomo.trans.txt'
    idx = file.split('/')[-1].split('.')[0]

    with open(src_file, 'r') as fp:
        for line in fp:
            if idx == line.split(' ')[0]:
                return line[:-1].split(' ', 1)[1]
```

TODO: Import new Dataset

- In the src/data.py
  - def create_dataset
  - def create_textset

```python
def create_dataset(tokenizer, ascending, name, path, bucketing, batch_size,
                   train_split=None, dev_split=None, test_split=None):
    ''' Interface for creating all kinds of dataset'''

    # Recognize corpus
    if name.lower() == "librispeech":
        from corpus.librispeech import LibriDataset as Dataset
    elif name.lower() == "dlhlp":
        from corpus.dlhlp import DlhlpDataset as Dataset
    else:
        raise NotImplementedError
```

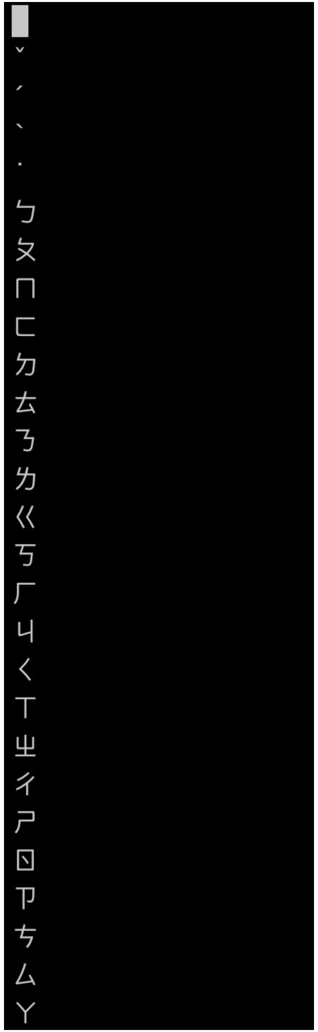
```python
def create_textset(tokenizer, train_split, dev_split, name, path, buck
    ''' Interface for creating all kinds of text dataset'''
    msg_list = []

    # Recognize corpus
    if name.lower() == "librispeech":
        from corpus.librispeech import LibriTextDataset as Dataset
    elif name.lower() == "dlhlp":
        from corpus.dlhlp import DlhlpTextDataset as Dataset
    else:
        raise NotImplementedError
```

TODO: Prepare vocab-count file

```
$ python util/generate_vocab_file.py
                 --input_file <input_text_file>
                 --mode character
                 --output_file <output_vocab_file>
```

we provide “text-data.txt” in the dataset as input_text file.



TODO: Write your own config file (1/3)

- See example config/libri/asr_example.yaml
- Create copys as you want to train a new model (e.g. asr_dlhlp.yaml)
  - `path`: where you unzip the data (e.g. ‘`data/DLHLP`’)
  - `train/dev_split`: dir name for train/dev under ‘`path`’
  - audio feature setting: follow the original setting
  - `mode`: ‘character’
  - `vocab_file`: the vocab file your have prepared

```yaml
data:
  corpus:
    name: 'Dlhlp'
    path: 'data/DLHLP'
    train_split: ['train']
    dev_split: ['dev']
    bucketing: True
    batch_size: 16
  audio:
    feat_type: 'fbank'
    feat_dim:  40
    frame_length: 25
    frame_shift: 10
    dither: 0
    apply_cmvn: True
    delta_order: 2
    delta_window_size: 2
  text:
    mode: 'character'
    vocab_file: 'bopomo_vocab_file'
```

TODO: Write your own config file (2/3)

- Training Hyperparams
  - set valid_step to 500 (step for one epoch)
  - max_step: 12k step is enough actually
  - teather forcing: always use it
    => tf_start: 1.0, tf_end:1.0
  - optimizer and lr and eps:
    just follow the original settings

```
hparas:
  valid_step: 500
  max_step: 12001
  tf_start: 1.0
  tf_end: 1.0
  tf_step: 500000
  optimizer: 'Adadelta'
  lr: 1.0
  eps: 0.00000001
  lr_scheduler: 'fixed'
  curriculum: 0
```

TODO: Write your own config file (3/3)

- Model Architecture:
    - A thinner model (1~2 layer LSTM) is enough
    - ctc_weight:
        - set to 0.0 to train seq2seq without CTC
        - set between 0.0~1.0 to jointly optimize for CTC + seq2seq
    - Other settings: just follow the original settings

```yaml
model:
  ctc_weight: 0.0
  encoder:
    prenet: 'vgg'
    # vgg: True
    module: 'LSTM'
    bidirection: True
    dim: [512,512]
    dropout: [0,0]
    layer_norm: [False,False]
    proj: [True,True]
    sample_rate: [1,1]
    sample_style: 'drop'
  attention:
    mode: 'loc'
    dim: 300
    num_head: 1
    v_proj: False
    temperature: 0.5
    loc_kernel_size: 100
    loc_kernel_num: 10
  decoder:
    module: 'LSTM'
    dim: 512
    layer: 1
    dropout: 0
```

TODO: Write your own config file for LM

- See example config/libri/lm_example.yaml
- Need to modify:
  - path
  - train/dev_spit
  - vocab_file
- You can also prepare your own ㄅㄆㄇㄈcorpus much larger than training set!
  - just replace `['train']` with `['xxx.txt']` where `xxx.txt` is your collected corpus

```yaml
data:
  corpus:
    # The following depends on corpus
    name: 'dlhlp'
    path: 'data/DLHLP'
    train_split: ['train']
    dev_split: ['dev']
    bucketing: True
    batch_size: 32
  text:
    mode: 'character'
    vocab_file: 'bopomo_vocab_file'
```

Train model!

ASR:

```
$ python3 main.py --config config/dlhlp/asr_dlhlp.yaml
```

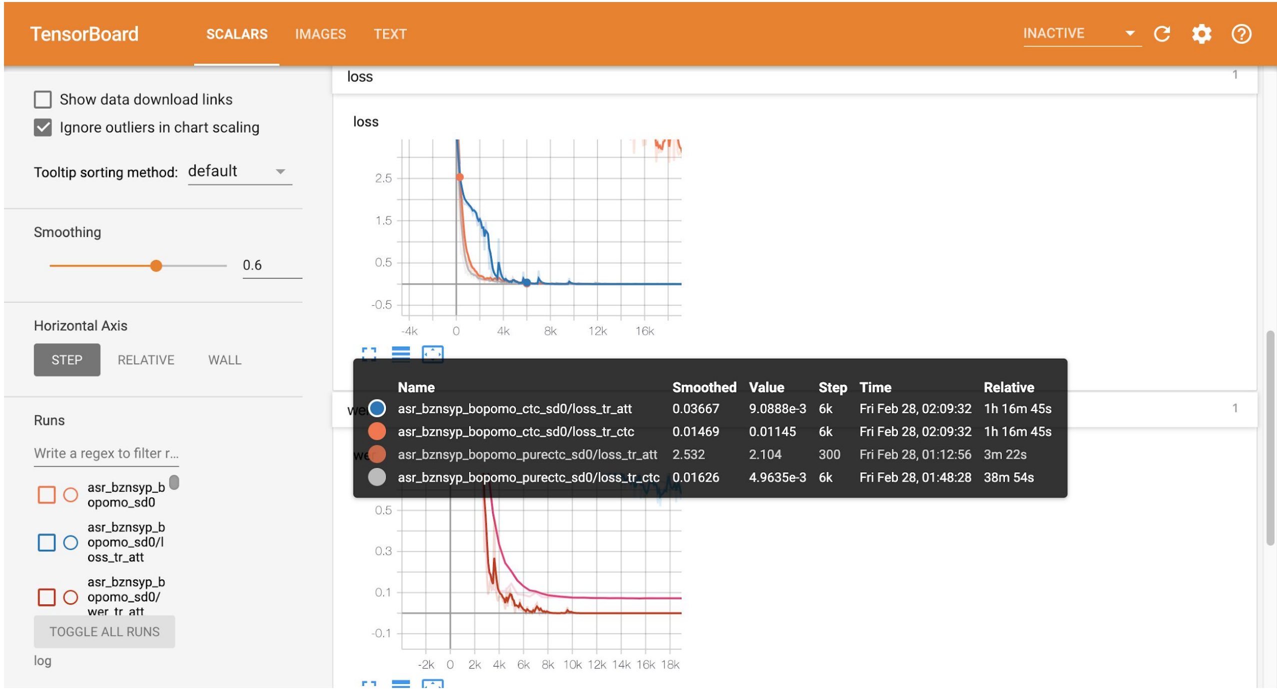
LM:

```
$ python3 main.py --config config/dlhlp/lm_dlhlp.yaml --lm
```

Tensorboard

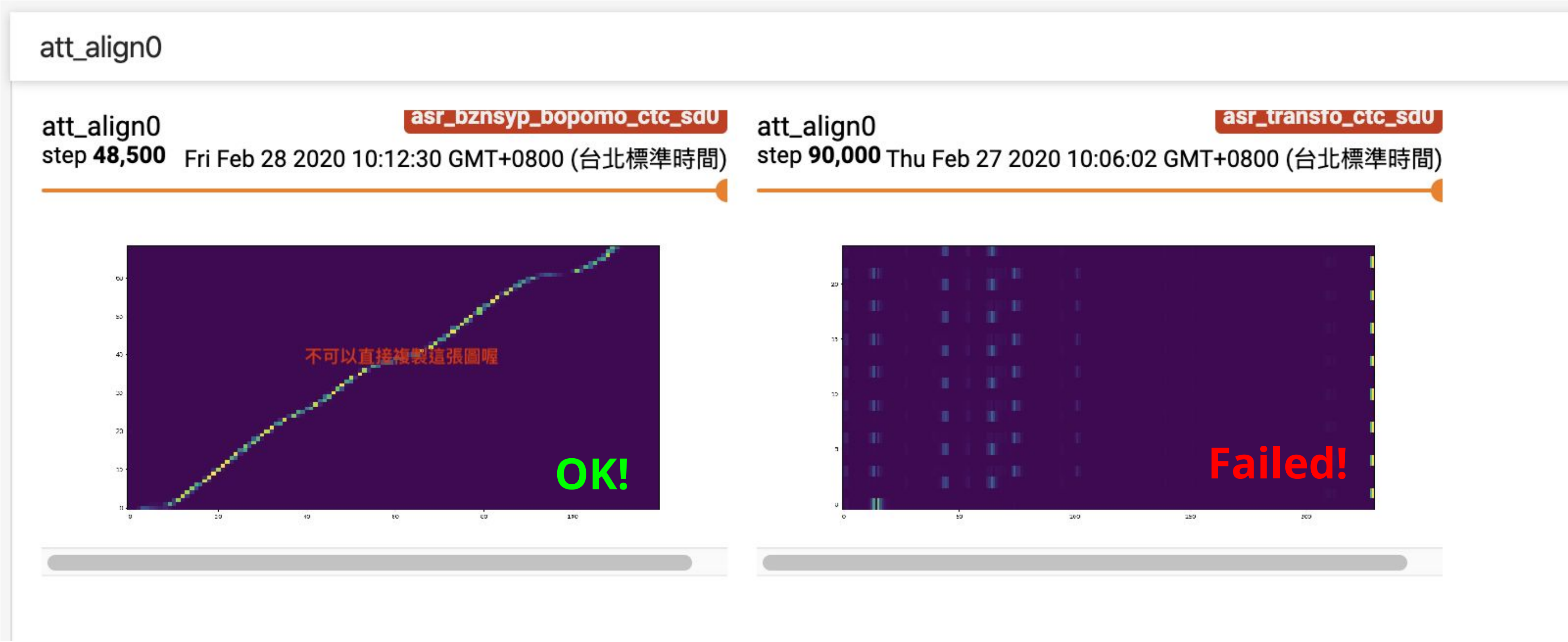
$ tensorboard --logdir log/ --port <port_you_want>

Training curve
(loss &WER / train&dev)

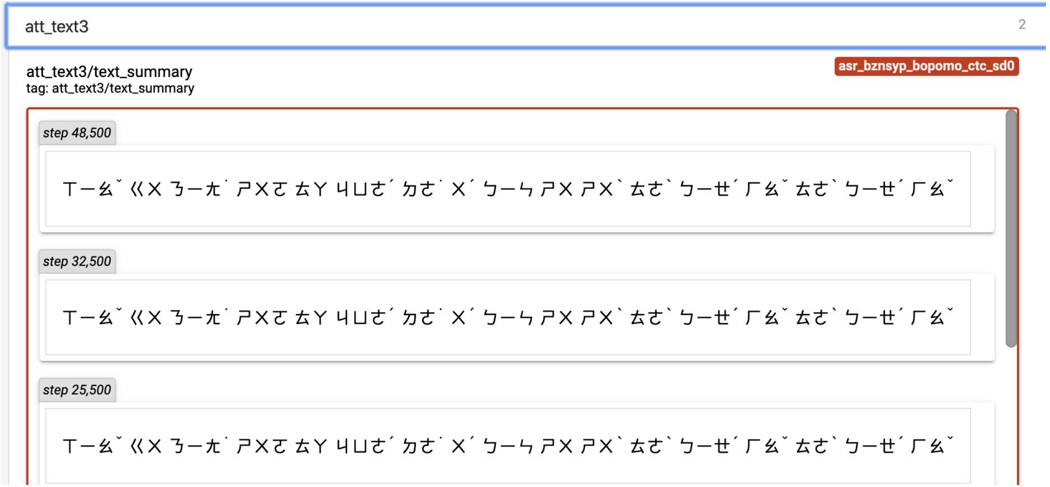


Alignment


SCALARS
IMAGES
TEXT


att_align0
att_align0
asr_bznsyp_bopomo_ctc_sd0
step 48,500 Fri Feb 28 2020 10:12:30 GMT+0800 (台北標準時間)
不可以直接複製這張圖喔
OK!
att_align0
asr_transfo_ctc_sd0
step 90,000 Thu Feb 27 2020 10:06:02 GMT+0800 (台北標準時間)
Failed!

Output example



Test your model - seq2seq

- See example config/libri/decode_example.yaml
- set max_len_ratio: 0.30
- set ctc_weight: 0.0
- set beam size 2~20 (1 not support)

If you want to use external LM:

- set lm_config&lm_path
- set lm_weight>0.0

```yaml
# Most of the parameters will be imported from th
src:
  ckpt: 'ckpt/asr_dlhlp_bopomo_sd0/best_att.pth'
  config: 'config/dlhlp/asr_dlnlp_bopomo.yaml'
data:
  corpus:
    name: 'Dlhlp'
    dev_split: ['dev']
    test_split: ['test']
decode:
  beam_size: 2
  min_len_ratio: 0.01
  max_len_ratio: 0.30
  lm_path: 'lm_dlhlp_sd0/best_ppx.pth'
  lm_config: 'config/dlhlp/lm_dlhlp.yaml'
  lm_weight: 0.3
  ctc_weight: 0.0
```

Test your model - CTC

- Original code has some problems for jointly decode by seq2seq&CTC
- Add few lines of code to support decode only by CTC (works for ctc_weight=1.0)
- src/decode.py line 101

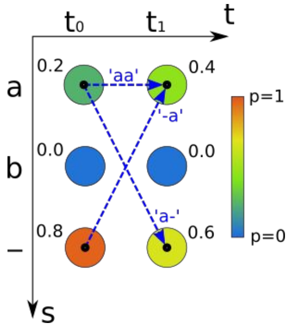
```
95          # CTC decoding
96          if self.apply_ctc:
97              ctc_output = F.log_softmax(
98                  self.asr.ctc_layer(encode_feature), dim=-1)
99              ctc_prefix = CTCPrefixScore(ctc_output)
100             ctc_state = ctc_prefix.init_state()
101             if self.ctc_w == 1.0:
102                 output_seq = ctc_output[0].argmax(dim=-1)
103                 hypothesis = [Hypothesis(decoder_state=dec_state, output_seq=output_seq,
104                                          output_scores=[0]*len(output_seq), lm_state=None, ctc_prob=0,
105                                          ctc_state=ctc_state, att_map=None)]
106                 return hypothesis
```

Patch for pure CTC decode (add to src/decode.py line 101)

Simply Greedy Decode!

```
    if self.ctc_w == 1.0:
        output_seq = ctc_output[0].argmax(dim=-1)
        hypothesis = [Hypothesis(decoder_state=dec_state, output_seq=output_seq,
                        output_scores=[0]*len(output_seq), lm_state=None, ctc_prob=0,
                        ctc_state=ctc_state, att_map=None)]
        return hypothesis
```

Actually, pure CTC decode also needs beam search.
It could be a good bonus topic for you to implement it.
https://towardsdatascience.com/beam-search-decoding-in-ctc-trained-neural-networks-5a889a3d85a7



Test your model - CTC

- set ctc_weight = 1.0
- external LM is not supported to jointly decode with CTC in our code (*But you can implement it as bonus!*)

```yaml
# Most of the parameters will be imported from the
src:
  ckpt: 'ckpt/asr_dlhlp_bopomo_sd0/best_ctc.pth'
  config: 'config/dlhlp/asr_dlnlp_bopomo.yaml'
data:
  corpus:
    name: 'Dlhlp'
    dev_split: ['dev']
    test_split: ['test']
decode:
  beam_size: 2
  min_len_ratio: 0.01
  max_len_ratio: 0.30
  lm_path: 
  lm_config: 
  lm_weight: 0.0
  ctc_weight: 1.0
```

Test!

```
$ python3 main.py --config <config file> --test --njobs 8
```

- result will be produced at:
    - result/<config file>_dev_output.csv
    - result/<config file>_dev_beam.csv 用不到
    - result/<config file>_test_output.csv
    - result/<config file>_test_beam.csv 用不到
- The format of output file is `<id> <predicted seq> <truth seq>` (line by line)
- The truth sequence of testing set was replaced with ㄅㄆㄇㄈ.
- `--njobs` decides the number of threads used for decoding, very important in terms of efficiency. You can set it higher as your machine have more cores.

Process the output from CTC

CTC has repeated tokens
Process it by yourself!

```
008721  ㄨㄛㄛˊˊ ㄏㄣㄣˇ  ㄓㄨˋ  ㄧˋ  ㄧㄝˇ  ㄇㄧㄢㄢˋ      ㄉㄉㄚ˙˙     ㄨㄛˊ ㄏㄣˇ ㄓㄨˋ ㄧˋ ㄧㄢˊ ㄇㄧㄢˋ ㄉㄚ
008722  ㄨˊˊ  ㄒㄧㄡˋ ㄅㄛㄛ  ㄉㄠˋ ㄉㄧˇ  ㄕˋˋ       ㄕㄨㄨㄟˊˊ        ㄨˊ ㄒㄧㄡˋ ㄅㄛ ㄉㄠˋ ㄉㄧˇ ㄕˋ ㄕㄨㄟˊ
008725  ㄐㄧㄠㄠˋ ㄋㄧˊ  ㄐㄧㄝˊ ㄨㄛˇ ㄍㄍㄢˋ       ㄇㄚㄚˊˊ   ㄐㄧㄠˋ ㄋㄧˊ ㄐㄧㄝˊ ㄨㄛˇ ㄍㄢˋ ㄇㄚˊ
008693  ㄊㄚㄚ  ㄇㄣ˙ ㄏㄞˊ ㄗㄗㄞˋ  ㄩˋ ㄇㄣㄣ˙       ㄋㄋㄜ˙˙   ㄊㄚ ㄇㄣ˙ ㄏㄞˊ ㄗㄞˋ ㄩˋ ㄇㄣˋ ㄋㄜ˙
008706  ㄋㄧㄧˇ  ㄓㄨㄜˋ ㄕㄕˋˋ ㄅㄅㄨˊ  ㄞㄞˋ  ㄨㄛㄛˇ      ㄉㄉㄜ˙˙˙    ㄋㄧˇ ㄓㄜˋ ㄕˋ ㄅㄨˊ ㄞˋ ㄨㄛˇ ㄉㄜ˙
008460  ㄔㄣˊ  ㄑㄧㄝˋˋ ㄗㄨㄛˋ ㄅㄨˊ ㄉㄠˋˋ         ㄚ˙˙˙        ㄔㄣˊ ㄑㄧㄝˋ ㄗㄨㄛˋ ㄅㄨˊ ㄉㄠˋ ㄚ˙
008713  ㄇㄥㄥˋ ㄕㄕㄨㄨ  ㄩㄩㄣˊ  ㄉㄉㄤ   ㄔㄤㄤˇ  ㄩㄣㄣ          ㄉㄉㄠㄠˇˇ   ㄇㄥˋ ㄕㄨ ㄩㄣˊ ㄉㄤ ㄔㄤˇ ㄩㄣ ㄉ
ㄠˇ
```

Or you can add additional code to the patch in page 25 to process the 'output_seq' directly before it is written to file.

Eval your output (dev set)

```
$ python3 eval.py --file result/<config_file>_dev_output.csv
```

hint: python2 is not supported here (unicode would be treat as two char)

```
============  Result of result/decode_bopomofo_beam2_lm0.3_dev_output.csv ============
 ----------------------------------------------------------------------
| Statics               | Truth           | Prediction     | Abs. Diff.    |
 ----------------------------------------------------------------------
| Avg. # of chars       | 66.99           | 66.96          |  0.37         |
| Avg. # of words       | 17.14           | 17.12          |  0.02         |
 ----------------------------------------------------------------------
 --------------------------------------------------------------
| Error Rate (%)| Mean            | Std.            | Min./Max.     |
 --------------------------------------------------------------
| Character     | 2.1157          | 2.43            | 0.00/33.33    |
| Word          | 7.0030          | 7.48            | 0.00/100.00   |
 --------------------------------------------------------------
Note : If the text unit is phoneme, WER = PER and CER is meaningless.
```

report this!

Submit your result (test set)

- Extract the result in kaggle format
- Sample script: https://github.com/DLHLP2020/hw1-speech-recognition/blob/master/format.py
- usage: `python3 format.py result/<config>_test_output.csv answer.csv`
- upload answer.csv to kaggle

Kaggle rules

- website: https://www.kaggle.com/c/dlhlp2020spring-asr
- Your team name should be in [team_github_id]_[any_string]
  e.g. daikin_大金
- 5 submission per team & per day
- Using any extra kaggle account to submit is cheating!

Kaggle Evaluation Metric

Mean **Levenshtein Distance** calculated for each sentence in char-level.

- the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other.

$$LevDistance = N_{ins} + N_{del} + N_{sub}$$

Word Error Rate:

$$WER = \frac{N_{ins} + N_{del} + N_{sub}}{N_{target-length}}$$

Baselines (5 points)

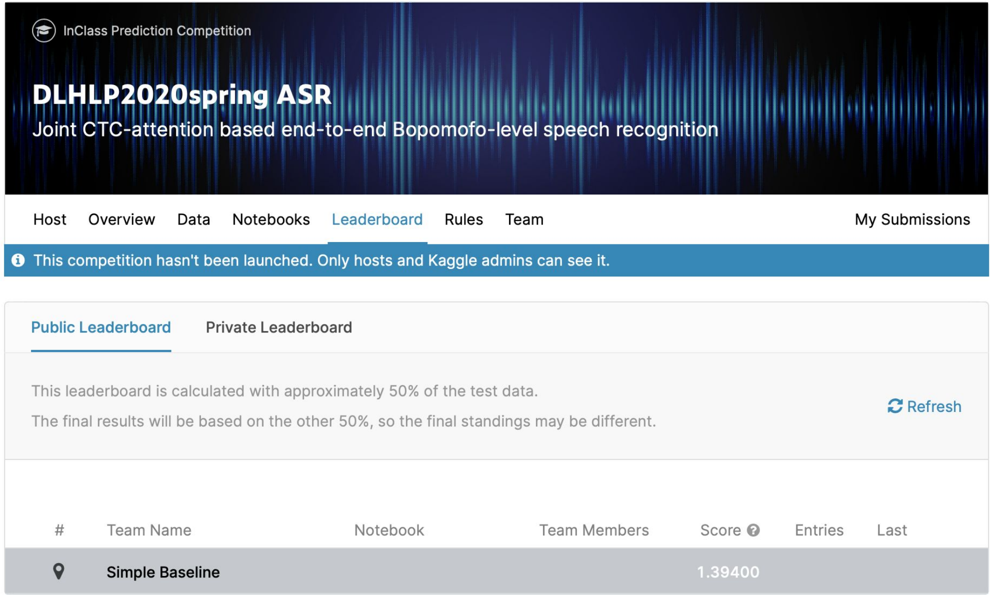
Public Simple Baseline: 1.394 (2 pt)
Public Strong Baseline: 1.278 (1 pt)

Private Simple Baseline: (?)   (1 pt)
Private Strong Baseline: (?)   (1 pt)

p.s. private score will be shown after kaggle deadline

Both would be easy to beat~

https://www.kaggle.com/c/dlhlp2020spring-asr



Report Questions (1/2)

In 1. ~ 3., just decode without language model.
In 1. ~ 4., set beam size = 2 for speeding.

1. Train a **seq2seq attention-based** ASR model. Paste the learning curve and alignment plot from tensorboard. Report the CER/WER of dev set and kaggle score of testing set. (2 points)

2. Repeat 1. by training a **joint CTC-attention** ASR model (decoding with seq2seq decoder). Which model converges faster? Explain why. (2 points)

3. Use the model in 2. to **decode only in CTC** (ctc_weight=1.0). Report the CER/WER of dev set and kaggle score of testing set. Which model performs better in 1. 2. 3.? Explain why. (2 points)

Report Questions (2/2)

4. Train an **external language model.** Use it to help the model in 1. to decode. Report the CER/WER of dev set and kaggle score of testing set. (2 points)
5. Try decoding the model in 4. with **different beam size** (e.g. 2, 5, 10, 20). Which beam size is the best? (2 points)

Bonus. Other Improvement and Innovation

- If you don't have anything to share, you can just say 'Nothing'. It will be fine.
- Guidiance
  - Read new papers
  - Browse github
  - Apply some cool tips

Scoring: Submission

- Kaggle Deadline: 3/22(Sun) 23:59
- Github submission Deadline: 3/25(Wed) before class
- Create a folder ‘hw1’ under your team Github repo
- ‘hw1/’ contains:
    - report.pdf
    - reproduce.sh
    - other files and directories
- Report Template: https://docs.google.com/document/d/1NyIgXrlai9Lgysplh9p742zgn2j3cE3B7epqiLDAesE/
- We restrict Python version 3.6.8 and must compatible with these package
- Scoring
    - report 10pts
    - kaggle 5pts (over baseline + successfully reproduce)

Github maximum capacity

- within 100MB
- use Dropbox to put your model, use ‘wget’ to download
- Dropbox Tutorial: https://docs.google.com/presentation/d/1SsIeIij9ZOEN_TGdbAS1oWcI6bT1uSTI6b5__u2wdDc/edit?usp=sharing
- your shell script files should be able to download the model automatically

Scoring: Reproduce

[03/23更新] reproduce.sh內若需要指定--njobs xx 的時候
請統一指定成 --njobs $NJOBS
方便助教直接從外面調整 $NJOBS
(這邊完全是方便助教改作業，弄錯不會扣分)

- `bash reproduce.sh $1 $2`
  - `$1` is the audio testing set directory (e.g. data/DLHLP/test)
  - `$2` is name of the output prediction csv file (e.g. ans.csv)
- This script should produce the same result (in kaggle submission format) as your **best submission on public leaderboard.**
- Hint: use `sed` to modify the data path (e.g. 'data/DLHLP') in your config
- It HAVE to automatically download EVERYTHING you want to wget
- If your code reproduce fail, your **CANNOT** get kaggle score (5%)
- Prepare it carefully!

Scoring: Bonus

- Presentation in class
- Selection criteria
    1. Innovation
    2. Different ways compare to LAS
        - Exiting github is valid, but you have to understand and explain it.
    3. Good performance
- 1 extra pts
- The team quota and the presentation time will be announced based on the time we have.

Late submission policy

- You can submit file until 3 days after deadline
- The score will be calculated as:

$$\text{score}_{\text{final}}(\text{hr}) = \begin{cases} \text{score}_{\text{original}} \times 0.985^{\text{hr}} & , \text{hr} \leqslant 72 \\ 0 & , \text{hr} > 72 \end{cases}$$

- Late submision form would be anounced after deadline

FB Group:

Deep Learning for Human Language Processing (2020,Spring)

Q&A

dlhlp.ta@gmail.com