Recep Oğuz Araz
Prof. Engin Erzin
24.11.2020
ELEC 390

Report V: Training the Model

Hardware Implementation

Implementation of the PyTorch's CTC Loss on Windows 10 (Home x64).

Utilizing CUDA on Jupyter Notebook (Windows 10) and Google Colab.

Constructing a Data Loader object for increasing feature extraction speed.

- Do not let CPU idle while GPU working, make up for transfer time.
- Assign 2 workers for each thread in CPU. (Does not work now. (Windows 10))

Feature extraction

1. Amplitude Spectrogram vs. Power Spectrogram.
2. Normalization vs. Whitening for input layer.
3. Spectrogram vs Mel-Spectrogram.

Training

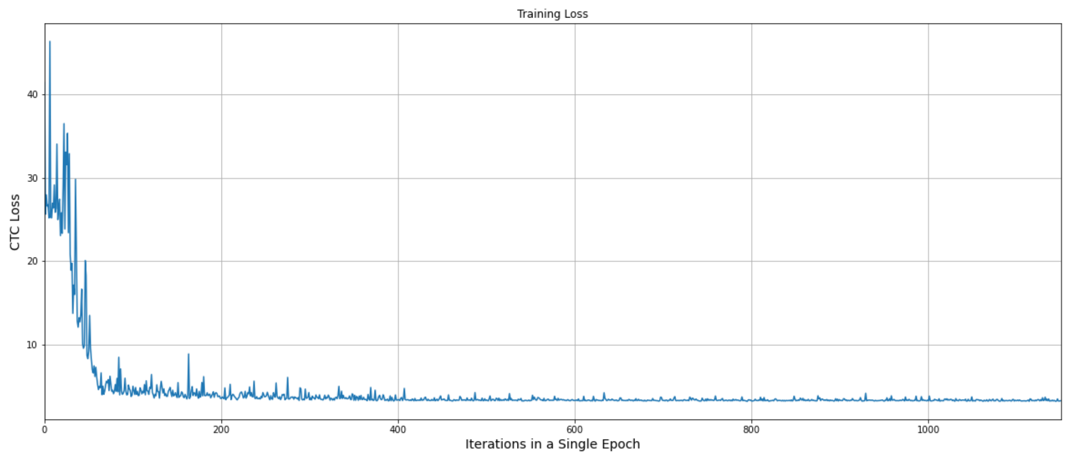
Observing Loss decrease over the train set, global(?) minimum is reached.

Model parameter, hyperparameter tuning.

For optimization SortaGrad does not exist but Adam works fine. (Forums and Graves et. Al.)

First Training

- Single Epoch, no validation set (predictions bad).
- Learning rate between $(10^{-6}, 10^{-2})$
- $N_{FFT} = 1024, 512 (Graves\ et. al)$
- $N_{Alphabet} = 38$
- With and without Convolutional layer.
- $N_{hidden} = 128$
- $FC1_{out} = 128, FC2_{out} = 128, FC3_{out} = N_{Alphabet} = 38$



Recep Oğuz Araz
Prof. Engin Erzin
24.11.2020
ELEC 390

Problems with PyTorch’s CTC Loss.

- All blank outputs on first 3 epochs. (many other users report similar problems.)

Possible solutions:

1. Many epochs. ➔ Faster GPU or longer runtime needed.
2. Baidu CTC-Warp. (not available for PyTorch on Windows 10)
3. BatchNorm
4. Adaptive Learning

Changes

Removal of Convolutional layer (2D, single). For now.

- Clamp causing problems (Data non-familiar.)
- No apparent effect until now (Makes sense as FFT has windows that smooth transition in time).

Question

What to expect from CTC?
How to choose $FC1_{out}, FC2_{out}, N\_hidden$?

To Do

Implement the original BatchNorm to each layer input. (Baidu version for Very Large Datasets.)

Bring baseline model to its potential.

Utilize Ku HPC for experiments.