IE Project3 Report

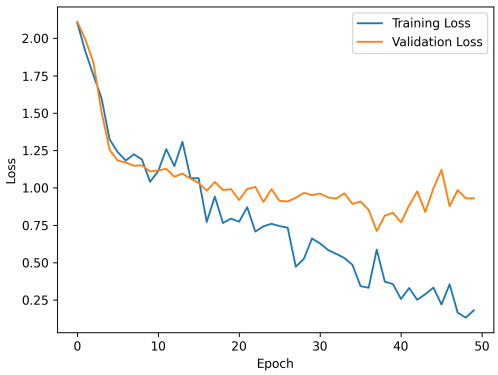
Yining Lu

May 2023

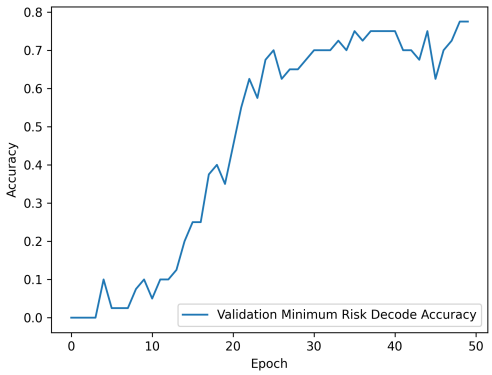
Primary System

Letter pad token `<pad>` was created with id 25 and feature label pad token was created as well with id 256. I use a two-layer LSTM with an embedding size 40 and a hidden size 256. A validation dataset of size 40 was created from the training dataset. The model finally achieved 77.5% accuracy on the validation dataset using a minimum CTC loss decoding strategy (figure (b)). I also tried using Greedy Search and Beam Search with beam size 3 for decoding. On the training dataset, both the final beam search and greedy search accuracies are 49.6%. On the validation dataset, however, they drop significantly to 7.5% (figure (c)).

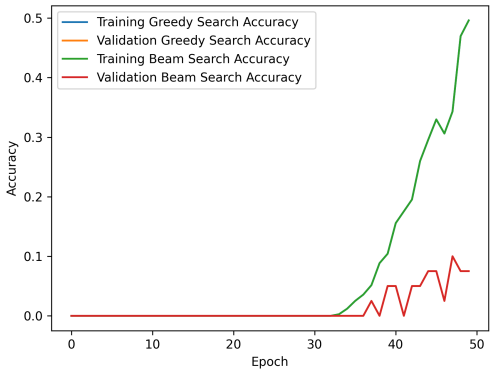
Plot



(a) Training and Validating Loss over Epochs



(b) Validation Accuracy by Minimum CTC Decoder



(c) Training and Validation Accuracy by Beam and Greedy Search

Prediction

393 testing results with associated confidence scores are stored in the file `discrete_test_result.json`. Here is one example


```
{
"Test Id":  0,
"Beam Search Decode":  "een",
"Beam Search Forward Log Prob":  -5.01182746887207,
"Greedy Search Decode":  "een",
"Greedy Search Forward Log Prob":  -5.011830806732178,
"Minimum Risk Decode":  "even",
"Minimum Risk CTC Loss":  0.9165775775909424
}
```


`test Id` is used for index items in the testing dataset. Beam search and greedy search decoder use forward log probability (summation of log probability of each time step) as the confidence score. A higher score is expected to indicate a better result. On the other hand, the Minimum CTC loss decoder uses CTC loss as its confidence score, and we want it to be as small as possible. In this example, Beam and Greedy search output `een` which is very close to `even`

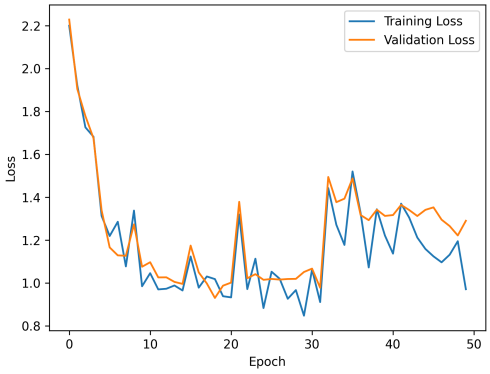
Source Code

Training and evaluating the model: `python main.py -v -b 16 -f discrete -e 50`. Loading trained model and evaluating: `python evaluate.py`. For more information, please refer to `README.md`

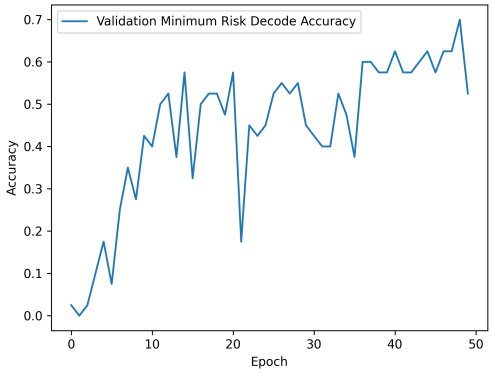
Contrastive System

Contrastive system almost has the same setting as the primary system except for the input features. For more model information, please refer to `README.md`.
The final validation accuracy using Minimum CTC loss is 52.5% and the highest is 70% which is slightly less than the primary system (figure (b)). However, beam and greedy search perform poorly on MFCC features (figure (c)).

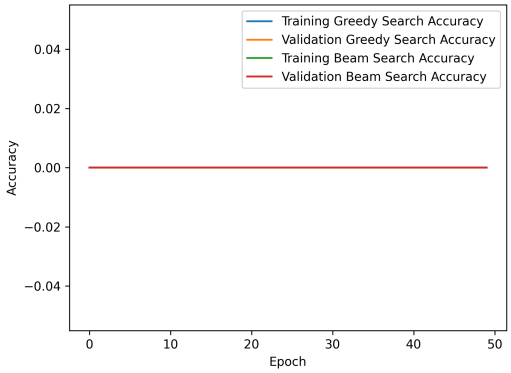
Plot



(a) Training and Validating Loss over Epochs



(b) Validation Accuracy by Minimum CTC Decoder



(c) Training and Validation Accuracy by Beam and Greedy Search

Prediction

393 testing results are stored in the file `mfcc_test_result.json`. And three decoders use the same confidence score as in `discrete_test_result.json`. Here is one example:
{

```
"Test Id":  0,
"Beam Search Decode":  "",
"Beam Search Forward Log Prob":  -8.803388595581055,
"Greedy Search Decode":  "",
"Greedy Search Forward Log Prob":  -8.803389549255371,
"Minimum Risk Decode":  "even",
"Minimum Risk CTC Loss":  0.9819819927215576
}
```


Source Code

Training and evaluating the model: `python main.py -v -b 16 -f mfcc -e 50`. Loading trained model and evaluating: `python evaluate.py`. For more information, please refer to `README.md`