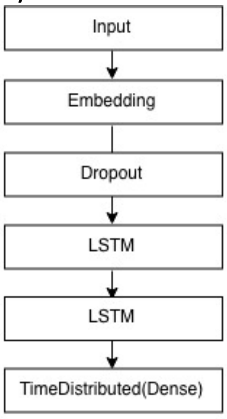
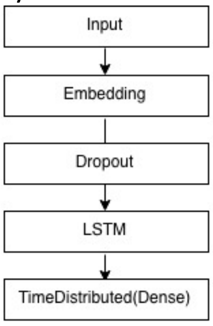
# COMP 4901K Project 3 Report

**Name: Yu-ning Huang**
**Student ID: 20213421**
**Scores on validation set: 1.6194 (Achieve Bonus baseline)**

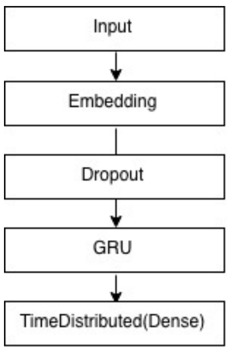
- **Algorithms/architecture used in this project**
  - Common architecture:
    - Output based on softmax activation
    - Optimizer: Adam
    - Model Checkpoint: I used ModelCheckpoint to monitor the validation loss and only saved the model parameters with the least validation loss to a h5 file.
    - Early Stopping: Although I set the number of epochs to run to be 100, it was almost impossible for any model to actually run 100 epochs because of EarlyStopping. EarlyStopping is a callback Keras provided to stop the training process if certain criteria is met. In my case, the training will be terminated if the validation loss did not decrease for 5 epochs.
  - 2-layer LSTM



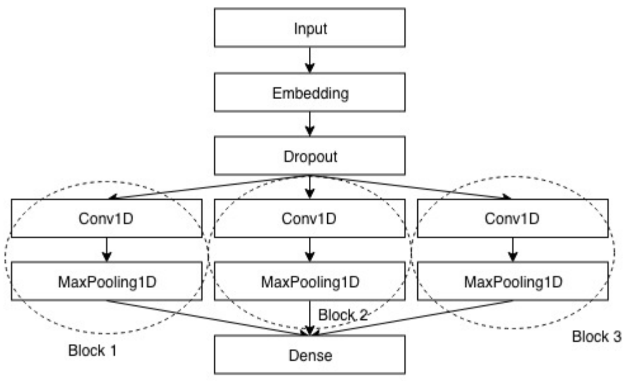
  - 1-layer LSTM



  - 1-layer GRU

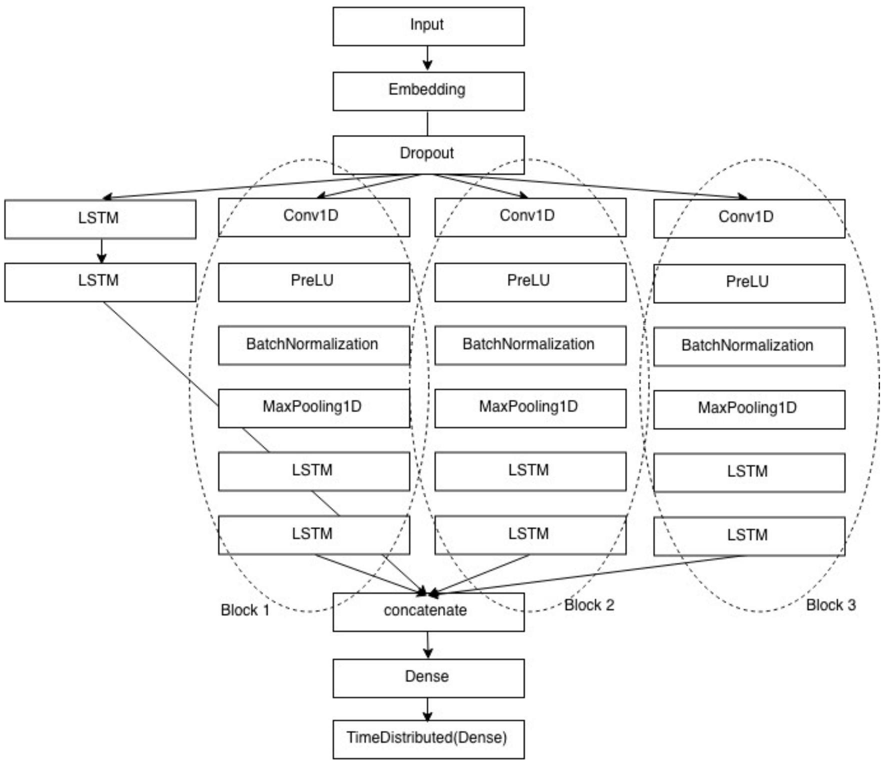


- CNN



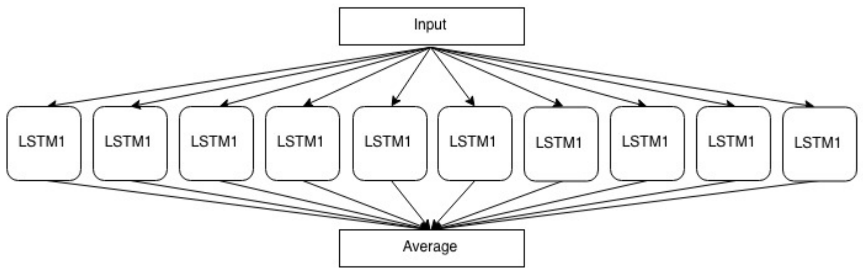
In Block 1, the kernel size in Conv1D layer was set to 3 to stimulate a 3-gram operation. The kernel size in Conv1D layer for block 2 was set to 4 and the kernel size in Conv1D layer for block 3 was set to 5.

- CNN LSTM



Due to the bad performance of CNN (see experiment results below), I tried to combine LSTM and CNN. The kernel size settings were the same as block 1 to block 3 in CNN.

- Ensemble



The ensemble model was based on 10 one-layer LSTM models with the best parameters I tuned (i.e. embedding dimension = 750 and hidden size = 500, the row colored in blue). Each model was set with different seeds to ensure they had different random weights initialization. The outputs from the 10 models were average to generate the final prediction.

- **Parameter tuning methodology and all the parameters and results tried**

The parameter tuning process was mainly based on the validation loss. I made changes to the model parameters one at a time. If the validation loss decreased, I kept the change. Otherwise, I reversed it.
(Notation: val = validation)

- 2-layer LSTM

| | parameters | val loss | val baseline |
|---|---|---|---|
| 1 | -batch_size 32 -embedding_dim 300 -hidden_size 300 -drop 0.5 | 1.7528 | 90 |
| 2 | -batch_size 32 -embedding_dim 500 -hidden_size 300 -drop 0.5 | 1.7400 | 90 |
| 3 | -batch_size 32 -embedding_dim 500 -hidden_size 500 -drop 0.5 | 1.7249 | 100 |

I started my experiment with randomly chosen parameters (i.e. 300 for both embedding dimension and hidden size).

- 1-layer LSTM

| | parameters | val loss | val baseline |
|---|---|---|---|
| 1 | -batch_size 32 -embedding_dim 500 -hidden_size 500 -drop 0.5 | 1.7258 | 100 |
| 2 | -batch_size 32 -embedding_dim 750 -hidden_size 500 -drop 0.5 | 1.7104 | 100 |
| 3 | -batch_size 32 -embedding_dim 750 -hidden_size 750 -drop 0.5 | 1.7168 | 100 |
| 4 | -batch_size 32 -embedding_dim 1000 -hidden_size 500 -drop 0.5 | 1.7501 | 90 |

The first set of parameters (1) were chosen based on the best model I got in 2-layer LSTM. The validation loss was slightly worse than 2-layer LSTM but the training time was much faster. Therefore, I decided to continue tuning based on 1-layer LSTM.

- 1-layer GRU

| | parameters | val loss | val baseline |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 1 | -batch_size 32 -embedding_dim 750 -hidden_size 500 -drop 0.5 | 1.7236 | 100 |

With the same parameter setting as the best model in 1-layer LSTM models (embedding dimension = 750, hidden size = 500), GRU model did not outperform LSTM model. Thus, I decided to continue using LSTM-based models.

- CNN

| | parameters | val loss | val baseline |
|---|---|---|---|
| 1 | -batch_size 32 -embedding_dim 500 -drop 0.5 -filter 128 | 28.3939 | 0 |

The performance of CNN was significantly worse than any LSTM-based model. Thus, I decided not to pursue with CNN.

- CNN LSTM

| | parameters | val loss | val baseline |
|---|---|---|---|
| 1 | -batch_size 32 -embedding_dim 500 -hidden_size 500 -drop 0.5 -filter 128 | 33.6680 | 0 |

The CNN LSTM combined model still performance poorly comparing to LSTM-only models. Hence, I decided not to pursue this model anymore.

- Ensemble

| | parameters | val loss | val baseline |
|---|---|---|---|
| 1 | Average 10 models | 1.6194 | bonus |

Ensemble model hits bonus baseline!

- **How to run the code?**
  - Run single model
    python main.py -mode train -saved_model <path to save the newly-built model> -student_id <student id> -batch_size 32 -embedding_dim <embedding dimension> -hidden_size <number of hidden layer> -drop <drop out rate> -filter <number of filters> -model <type of model>
    *Note:*
    1. *Valid type of model arguments:*
       a. *lstm1: 1-layer LSTLM*
       b. *lstm2: 2-layer LSTM*
       c. *gru1: 1-layer GRU*
       d. *cnn: CNN*
       e. *cnn_lstm: CNN LSTM*
    2. *Not all the arguments are needed for each model. Choose the arguments based on the model you are using. For example: while CNN has -filter, LSTM does not.*
  - Run ensemble model
    python main.py -mode ensemble -saved_model <path to save the newly-built model> -student_id <student id> -gpu <gpu you are using>
    *Note: The models that you would want to be used in the ensemble should be put in models/ensemble directory.*
  - Score validation result
    python scorer.py -submission <student id>_valid_result.csv

- **Third-party libraries**

- Keras: deep learning library to build LSTM, GRU, CNN and ensemble models.
- Numpy: perform matrix calculate and set random seeds for
- Sklearn: calculate log loss