# Fake News Detection Using Recurrent Neural Network and LSTM

# COMP 4211 Programming Assignment 3

**Lee Jae Yeol (20308109) | jyleeag@connect.ust.hk**

## Introduction

For programming assignment 3, 12375 texts are provided for the train dataset and 200 for the test set. One label column is provided for each train and validation dataset, whether the news is fake (1) or real (0). The main objective of pa3 is to be familiar with text data loader such as Tabular Dataset and Bucket Iterator, and to implement a Recurrent neural network as baseline model for fake news detection. After the dataset and data-loader classes are configured in the first part, the baseline of the RNN model and the train function are built. Upon successful implementation, other variants such as Long Short Term Memory model and Gated Recurrent Unit model are built. Then, the model undergoes several enhancement techniques, ones that are chosen for assessment are model weight initialization and model ensemble technique.

## Experiment Environment

| CPU | Memory | Python | PyTorch |
|---|---|---|---|
| Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz | 8 GB | 3.7.9 | 1.8.0+cu101 |

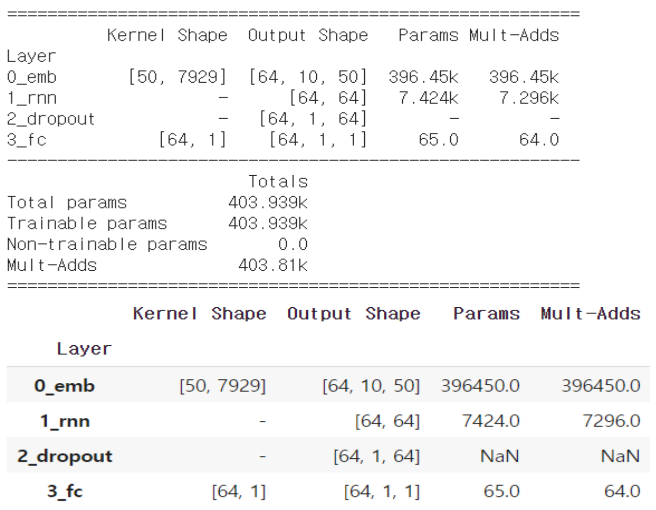
## Part 1: Dataset and DataLoader

### [Q1]

Out of Vocabulary refers to words in the training or validation dataset that aren't present in the vocabulary list, which in our case is built solely on subsets of training data. Torchtext library from Pytorch handles OOV issues by replacing unseen words with the token <unk>.

## Part 2: Recurrent Neural Network Baseline Model

### [Q2]

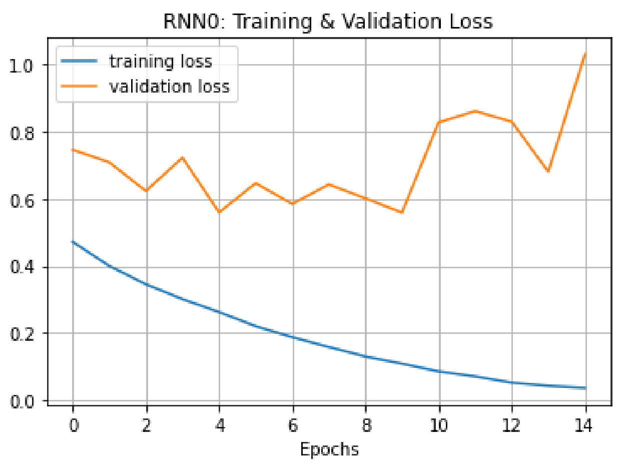


```
==========================================================
          Kernel Shape  Output Shape   Params Mult-Adds
Layer
0_emb       [50, 7929]  [64, 10, 50]  396.45k   396.45k
1_rnn                -      [64, 64]   7.424k    7.296k
2_dropout            -   [64, 1, 64]        -         -
3_fc           [64, 1]    [64, 1, 1]     65.0      64.0
----------------------------------------------------------
                      Totals
Total params        403.939k
Trainable params    403.939k
Non-trainable params     0.0
Mult-Adds            403.81k
==========================================================
```

| Layer | Kernel Shape | Output Shape | Params | Mult-Adds |
|---|---|---|---|---|
| **0_emb** | [50, 7929] | [64, 10, 50] | 396450.0 | 396450.0 |
| **1_rnn** | - | [64, 64] | 7424.0 | 7296.0 |
| **2_dropout** | - | [64, 1, 64] | NaN | NaN |
| **3_fc** | [64, 1] | [64, 1, 1] | 65.0 | 64.0 |

## Part 3: Training and Validation

### [Q3]



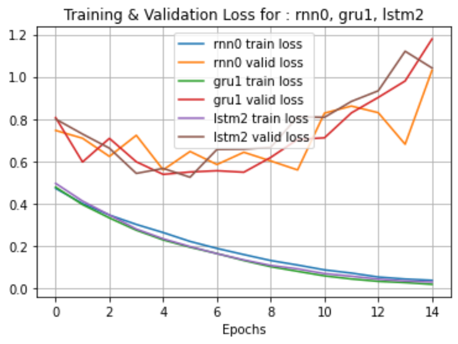
Best Validation Accuracy: 0.4846875

## Part 4: Empirical Study

### [Q4]

The crucial difference between RNN model and LSTM/GRU is that the posterior models have a 'memory cell' that holds the information from the past layers of the model. Therefore, LSTM/GRU models have access to long-term information, much more efficient than RNN model when the past information is important or when the sequential input is large. Another advantage 'memory cell' has is that it makes the model less vulnerable to vanishing gradient problems, with the same reason as above.

## [Q5]



Best Validation Accuracy

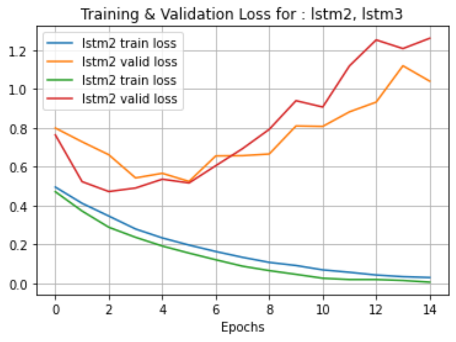
RNN0: 0.4846875

GRU1: 0.4890625

LSTM2: 0.483125

As expected, the GRU and LSTM models output a higher validation accuracy score compared to the baseline RNN model. The two model shows faster convergence, meaning that with the same number of epochs, it might lead to overfitting.

## [Q6]



Best Validation Accuracy

LSTM2: 0.483125

LSTM3: 0.5

By setting bidirectional to True in LSTM model, not only the information from the past is held, but the additional information from current state can be passed to the previous layer cell, resulting in better validation loss and accuracy score.

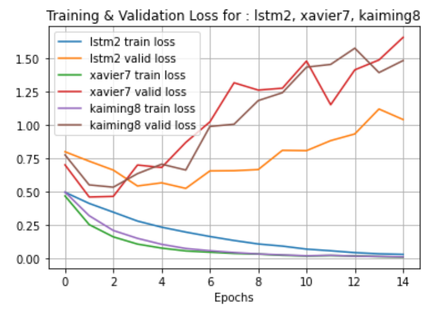
## Part 5: Improving Model Training

### B. Model Weight Initialization

### [Q7b]

Weight initialization is as it sounds, initializing the model neural networks' weights. By manually setting them, it prevents the model from facing exploding or vanishing gradient problems. Selecting an appropriate initial weight gives a higher chance for the model to converge quickly.

### [Q8b]



Best Validation Accuracy

LSTM2: 0.483125

Xavier7: 0.481875

Kaiming8: 0.4790625

### [Q9b]

As observed from the above graph, xavier7 and kaiming8 model converges relatively quicker than lstm2 model. However, the best validation accuracy hasn't shown improvement. It is plausible that the weights have been moderately set for lstm2 model, making weight initialization for these models impact less.

## D. Model Ensemble

### [Q7d]

Ensemble method combines several models of choice, producing an output that takes into account all the results of the chosen models. Ensembling models reduce the variance of the predictions, hence increasing the accuracy of the composite models. There are several different variants of ensemble learning, and in pa3, average ensemble and weighted average ensemble techniques will be implemented.

### [Q8d]

The first intuitive method will be to brute search the weights for each model (sum of weights = 1) and utilize the weight set with the lowest validation loss.

Out of many other possible methods, one plausible method will be to weight them with respect to their ranks (i.e. best validation score). The normalized validation score will be the precise weights for each model.

### [Q9d]

Best Validation Accuracy

Model12 <lstm3>: 0.5

Model13 <gru1>: 0.4890625

Model14 <rnn0>: 0.4846875

Model15 <average ensemble>: 0.5009375

Model16 <weighted average ensemble>: 0.4940625

There is a slight increase in the validation accuracy score for the model 15 – average ensemble on models. However, the improvement is minor. It might be due to the fact that bidirectional lstm3 already contains the necessary manipulation for classification, so the ensemble model doesn't show much of an improvement. For weighted average ensemble model, implementing a brute force method to find the optimal weight settings might lead to enhancement of the model.