CS5242 Deep Learning and Neural Networks

Dynamic Malware Analysis – Group 12

**Group 12**

Gejng Wang (E0403433)

Hao Zhang (E0486551)

Huiqi Mao (E0431609)

Ningshuang Chen (E0403979)

# 

# Final Model

The final model is an ensemble of 3 RNN models by applying 0.5, 0.25 and 0.25 weightage respectively, which reduces the chance of overfitting.

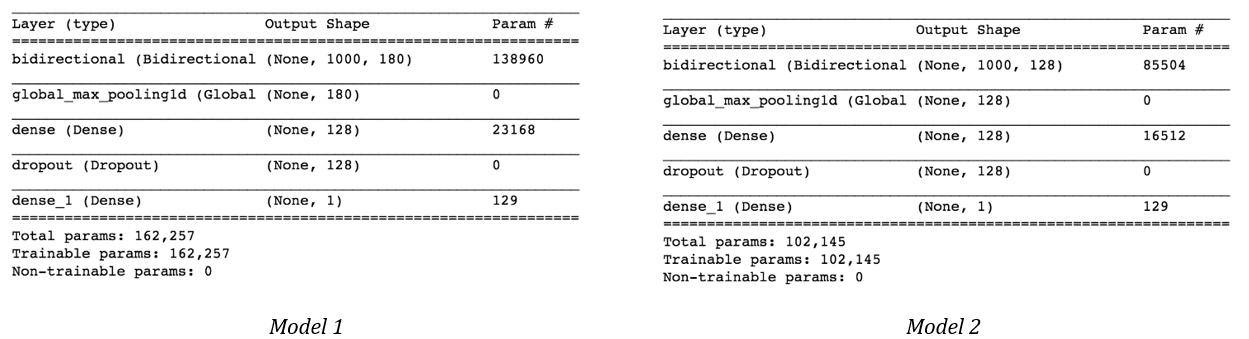
#### Model 1: 10-fold Cross Validation Model

As the training set is relatively small with only 18662 samples, this model applies the K-fold cross validation [1]. After splitting the training data into 10 sets (folds), 10 models are trained independently, where each model is trained on 9 out of the 10 sets and validated on the other 1 set. The final prediction is an average of the prediction results of the 10 models.

The architecture of Model 1 is shown in *Figure 1*. The input data is first passed to an RNN layer with LSTM units with output dimension size 128. After the LSTM layer, global max pooling would be applied before sending the data to a ReLU activation layer. Finally, a dropout layer with 0.2 dropout rate is applied before passing the result for binary classification.

#### Model 2: 8-fold Cross Validation Model

This model has identical layer architecture as Model 1, with differences in LSTM output size (180), global max pooling output size (180) and dropout rate (0.1). Also, 8-fold cross validation was applied to this model.



*Figure 1*

#### Model 3: Boosting Model

Model 3 has the same architecture and hyperparameters as Model 1. Similarly, it trains 10 different models and averages the result. In contrast, bagging and boosting techniques [2] are used in order to reduce the variance and validation loss. Every time before training a new model, the bagging algorithm samples the original training data into a bag, whose size is 60% of the original training data. Meanwhile, the boosting algorithm will also apply a higher probability to select samples that were poorly predicted by the models it trained in the earlier rounds. With the help of bagging and boosting, this model reduces the chance of overfitting.

# Experimental Study

#### Hyper-Parameter Setting

Optimizer related parameters were set for the 3 final models as *Table 1* by choosing parameters that yielded the best validation accuracy:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Optimizer | Learning Rate | Beta1 | Beta2 | LSTM Output Dim | Dense Layer Output Dim | Dropout Rate | Loss Function |
| Model 1 | Adam | 1 x 10-4 | 0.9 | 0.999 | 128 | 128 | 0.2 | Cross Entropy |
| Model 2 | Adam | 1 x 10-5 | 0.9 | 0.999 | 180 | 128 | 0.1 | Cross Entropy |
| Model 3 | Adam | 1 x 10-4 | 0.9 | 0.999 | 128 | 128 | 0.2 | Cross Entropy |

*Table 1*

#### Data Pre-Processing

We observed that the given input dataset has been pre-processed by doing hashing trick, which means no embedding is needed. However, each sample input data has varied length with a max of 1000. Thus, padding was done to make the input dimension (1000, 102) for all input data by adding zero vectors at the end of each sample.

We have also explored data normalization. Two approaches have been explored: **1)** normalize by dividing input data with global maximum value and **2)** normalize each feature by dividing maximum feature value respectively. However, it turns out model without data normalization had better validation accuracy, so data normalization was not used in our final model.

#### Result Comparison and Analysis

We explored a few different model architectures: **1)** Bidirectional LSTM; **2)** Stacked LSTM; **3)** CNN + LSTM; **4)** Bidirectional LSTM + Global Max Pooling, whose validation accuracy and loss are shown in *Table 2*:

|  |  |  |
| --- | --- | --- |
| Model | Validation Loss | Validation Accuracy |
| 1) Bidirectional LSTM | 0.2084 | 0.933 |
| 2) Stacked LSTM | 0.2045 | 0.9324 |
| 3) CNN + LSTM | 0.32 | 0.87 |
| 4) LSTM + Global Max Pooling | 0.1763 | 0.9498 |

*Table 2*

It was observed that complex models such as **2)** and **3)** had obvious convergence issues (spikes and high variance) and overfitting, which led us to use a relatively simple model architecture **4)** as base model. It was also observed that adding global max pooling layer has massively improved model stability.

However, even with our best model **4)**, the validation loss could not be reduced to below 0.15 despite various hyper-parameter tuning. We then set our eyes on ensemble methods (bagging and boosting) that are believed to reduce the variance and bias of the prediction [1]. With ensemble methods, we managed to improve Kaggle public score from 0.97993 to 0.98963. Finally, we explored K-fold cross validation [2] to make full use of the training data to further decrease the variance. With a weighted average of our best few submissions, we obtained our final predicted result that scored 0.99035 on public dataset and 0.99130 on private dataset.

We observed that we are one of the very few teams whose Kaggle private score is higher than the public score. This observation may be a result of our extensive use of ensemble methods, which led to lower variance in the prediction.

# Workload distribution

* Model Training:
  + Initial model: all team members
  + Research: all team members
  + Data normalization: Gejing Wang, Ningshuang Chen
  + Ensemble Methods: Hao Zhang
  + Model training: Hao Zhang, Huiqi Mao
* Presentation: Hao Zhang, Huiqi Mao
* Report Writing: Gejing Wang, Ningshuang Chen
* Code Submission Script: Hao Zhang

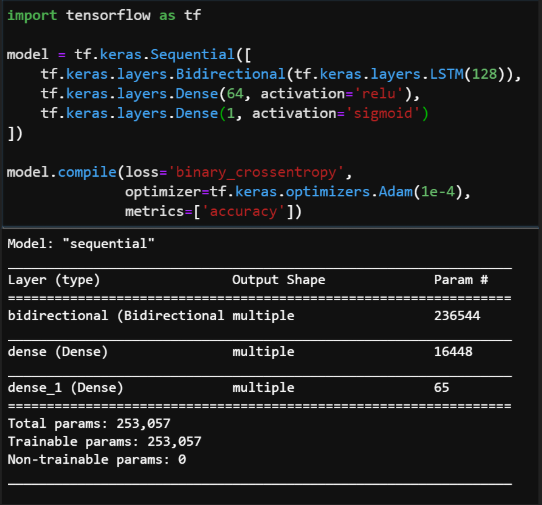
# Reference

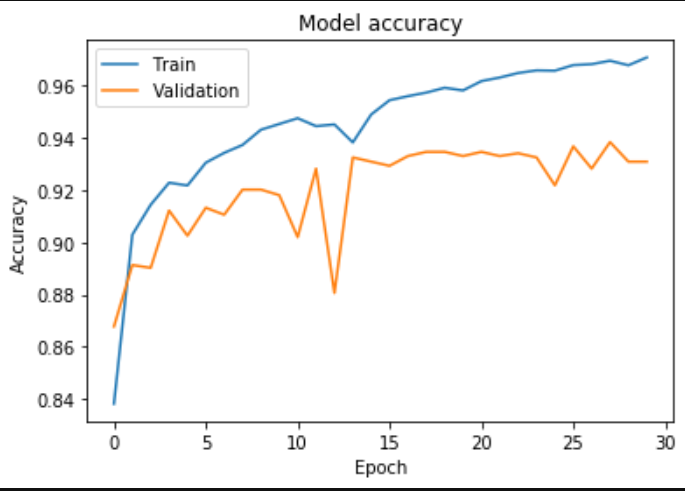
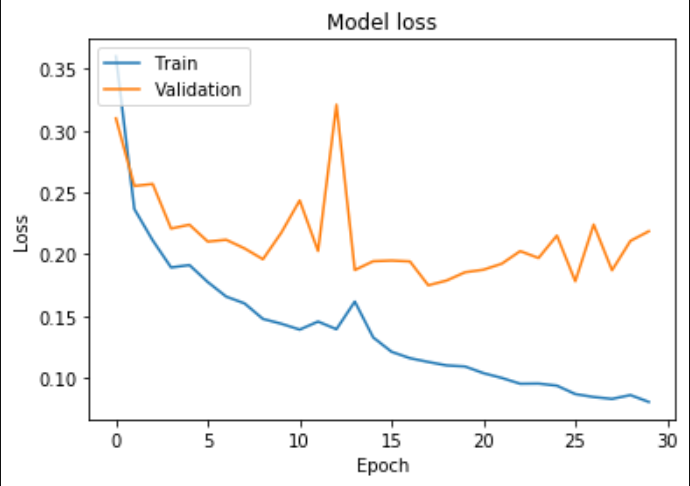
[1] Ng, A. (2019, August). Regularization and model selection. CS229. Retrieved from <http://cs229.stanford.edu/notes/cs229-notes5.pdf>

[2] Townshend, R. J. L. (2019, November). Ensembling Methods. CS299. Retrieved from <http://cs229.stanford.edu/notes/cs229-notes-ensemble.pdf>

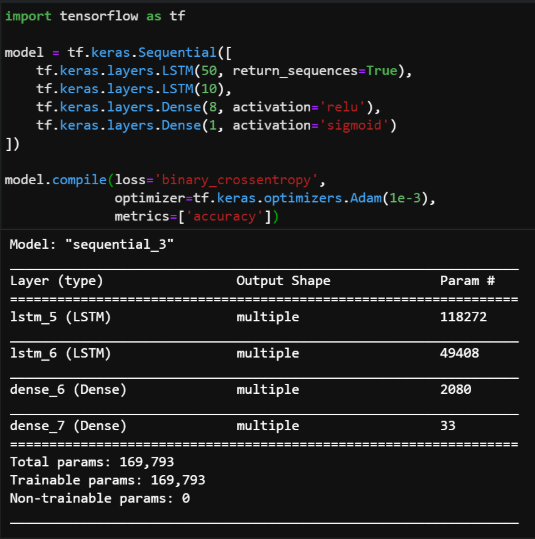
# Appendix

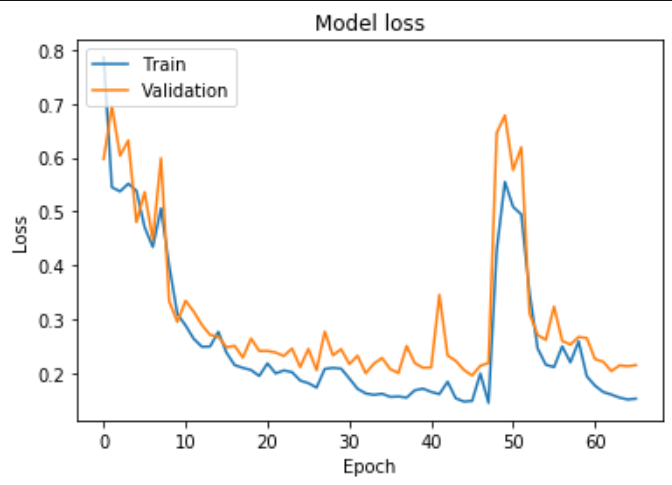
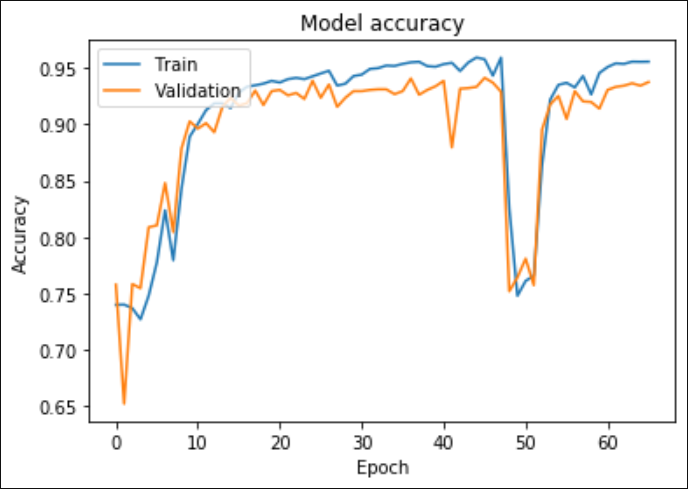
#### Bidirectional LSTM



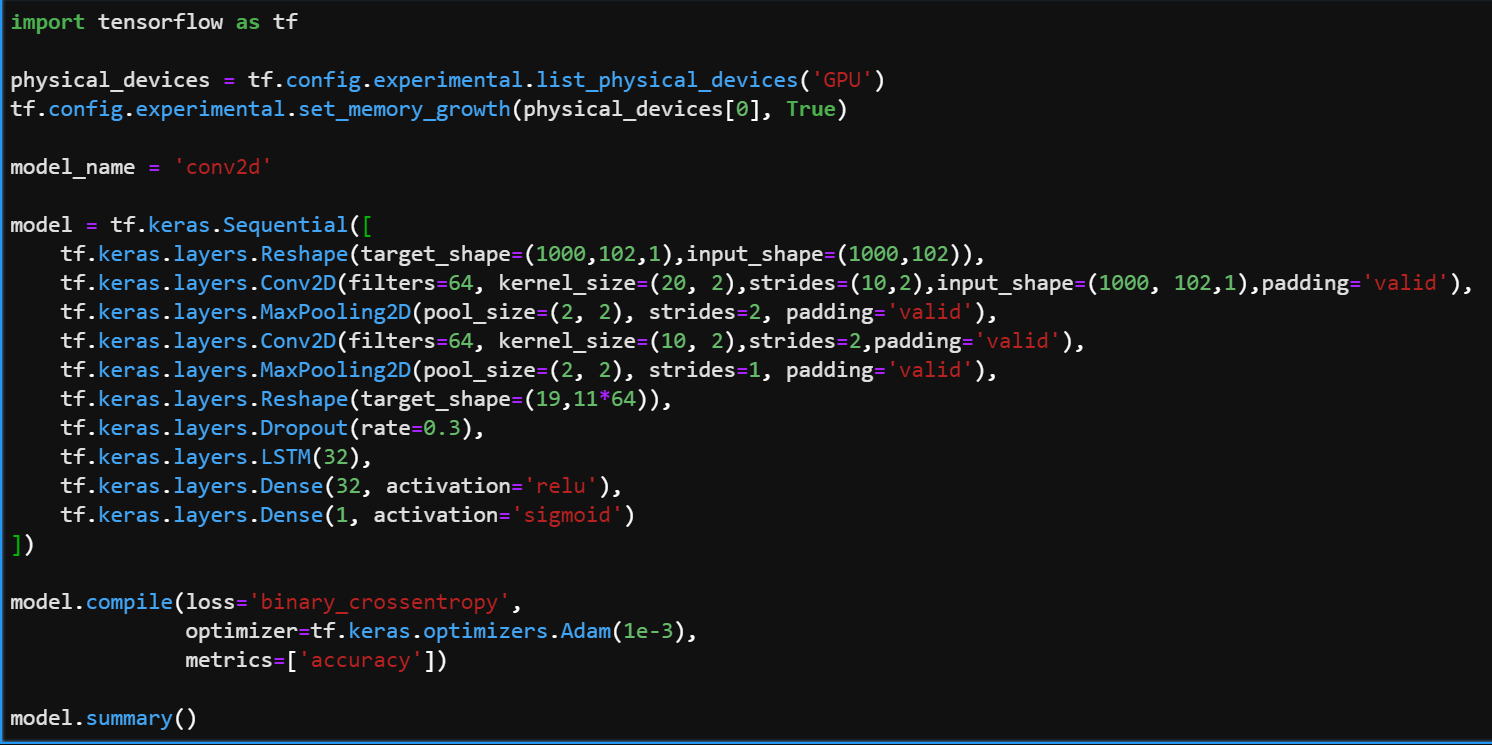
 

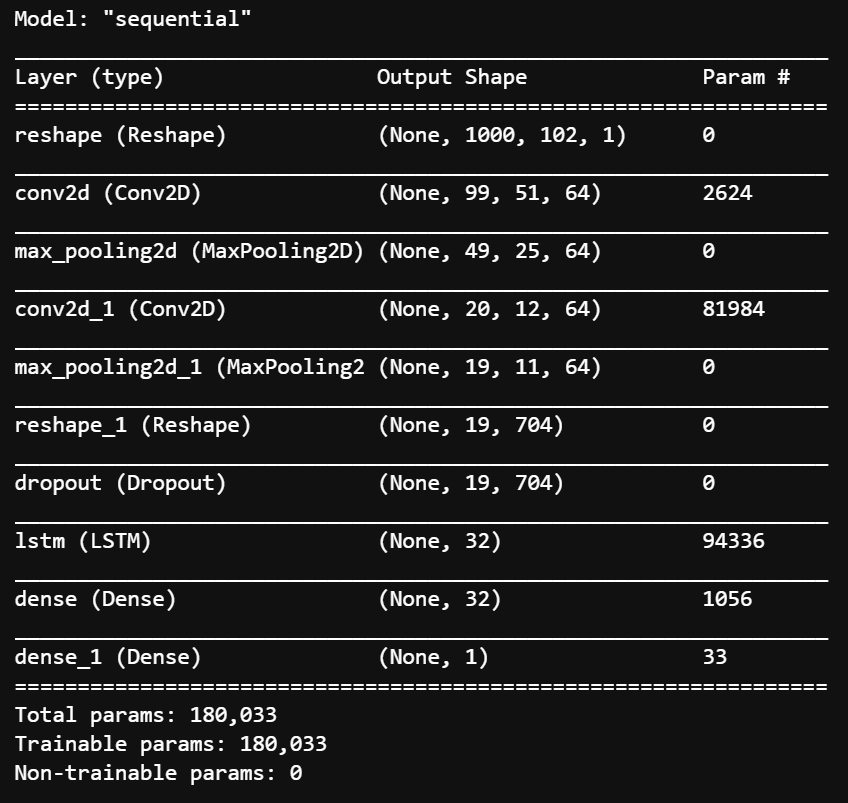
#### Stacked LSTM

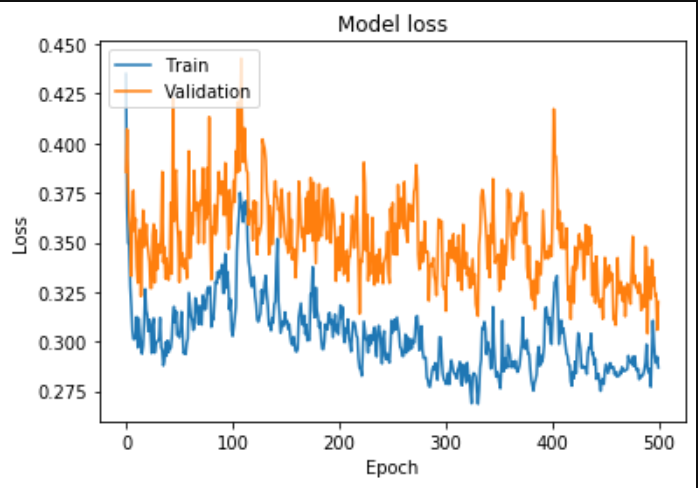
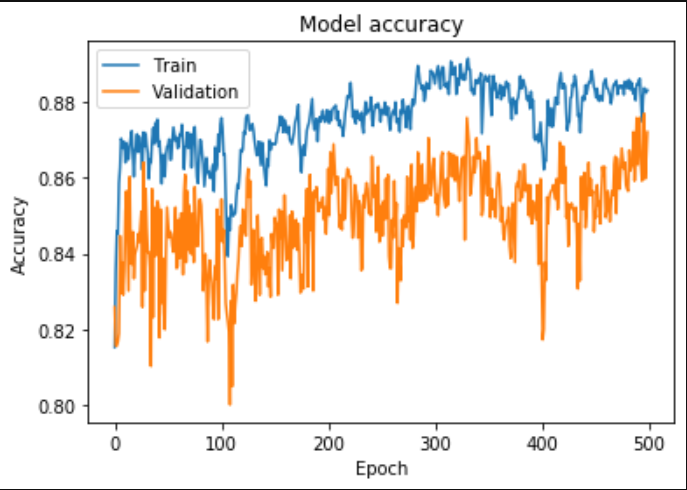




#### CNN + LSTM







#### Bidirectional LSTM + Global Max Pooling (Final Base Model)

