50.021 Artificial Intelligence Project: COVID-19 Retweet Prediction

Loh Zhi Xuan 1003389, Nurul Akhira Binte Zakaria 1003556,
Tan Qi Feng 1003290, Yu Wen Ling 1003893

9th August 2021

1 Introduction

Since the outbreak in December 2019, COVID-19 has caused drastic changes to lives around the world. The persistence of the disease also led to lasting effects of such changes, many even speculate that life will never resume to pre-COVID-19. One key measure to prevent the spread of the virus was to practice safe-distancing, causing us to shift our social life completely online. During the lockdown period, people around the world stay connected and continue bonding through online platforms such as Instagram, Facebook and Twitter. For our project, we will be tackling the CIKM2020 COVID-19 Retweet Prediction Challenge, focusing on modelling the retweet mechanism of Twitter.

Tweeting, a unique and key function of Twitter, is the act of a user posting a short text post of maximum 280 characters. This tweet can then be "retweeted" by the user's followers, or anyone that reads the tweet if the account is public, hence allowing information to be disseminated and amplified. With a Twitter community of more than 206 million users worldwide [3], inclusive of many influential celebrities and politicians, it is important to understand the mechanism as it reflects the movement of data across the world, the current trends and any possible ideology influence. In this report, we will attempt to understand and predict the retweet behaviour through several models.

2 Literature review

Before proceeding to developing our own models, the team first researched on the current methodologies employed by past year participants of the challenge.

Daichi Takehara [2] first generated features of three types: 1) numerical features, 2) categorical features, and 3) multi-hot categorical features, before flattening and concatenating them to be passed on into the neural network. Within the neural network, there are four fully-connected (FC) layers, with a descending number of nodes - from 2048 to 512, to 128, and finally to 1. Within each FC layer, the inputs would undergo batch normalization, ReLu and Dropout. The final FC layer will output a prediction on the number of retweets.

On the other hand, Piao and Huang [6], employed an ensemble of random forest models enhanced by linear regression, feed-forward neural networks and factorization machines.

Taking inspiration from the above two models, we approached the challenge with a neural network model (Section 4) and a random forest model (Section 3). More details on our own methodologies will be elaborated at the respective sections.

3 Data Cleaning and Preprocessing

The dataset used for this project is the same as the one utilised by the challenge, a publicly available dataset with more than 8 million tweets, spanning from October 2019 to April 2020. Due to the vast amount of data involved, the data was further split into 3 parts based on time period: 1) Oct 2019 to April 2020, 2) May 2020 to May 2020, 3) June 2020 to Dec 2020.

Each part consists of 12 columns, as listed in the table below.

Column	**Data Type**	**Description**
Tweet Id	String	ID generated for the tweet
Username	String	Encrypted username
Timestamp	String	Timestamp of the tweet, in the format of "EEE MMM dd HH:mm:ss Z yyyy"
#Followers	Integer	Number of followers
#Friends	Integer	Number of friends
#Retweets	Integer	Number of retweets
#Favorites	Integer	Number of favorites
Entities	String	Entities identified from the original text using the FEL library, in the form of "original_text:annotated_entity:score". If more than one entity is identified, each entity is separated by ";". If FEL did not identify any entity, "null;" value is stored.
#Sentiment	String	Positive and negative sentiment generated by SentiStrength, separated by a whitespace " ".
Mentions	String	The accounts mentioned in the tweet, with the "@" symbol removed and concatenated with whitespace " ". If there is no mention in the tweet, "null;" value is stored.
Hashtags	String	The hashtags mentioned in the tweet, with the "#" symbol removed and concatenated with whitespace " ". If there is no hashtag in the tweet, "null;" value is stored.
URLs	String	The URLs mentioned in the tweet, concatenated with ":-:". If there is no URL in the tweet, "null;" value is stored.

Table 1: 12 features for each tweet

Due to more familiarity with the language, R programming is used for the data processing. The first step to our data processing was to prune the data by removing invalid data points with "null;" as the value for either Tweet Id or Username.

Next, we went on to clean up the data in the columns for better analysis. This includes: 1) Extracting only the entity scores from the 'Entities' column, 2) Split the 'Sentiment' column into 'Positive sentiment' and 'Negative sentiment' columns. In order to achieve the former, a separate matrix containing values of the column 'Entities' is created, of which, all "null;" values were replaced by "NA". Then, the contents of the column were first split into individual columns by character ";", with each column consisting of one entity and its score. After which, the parse_number function of readr package is used to extract only the numeric values of the matrix. Hence returning a matrix with the columns as the maximum number of entities identified across the dataset and each cell containing the entity score for each data point. To achieve the latter, the separate function of the tidyr package was used to separate the column into two, named 'Positive_sentiment' and 'Negative_sentiment', by the whitespace. After which, both columns were converted to integer data type.

Lastly, we proceeded to generate more useful features from the current columns. This includes: 1) Converting timestamp into a cyclical format, 2) Counting the number of mentions for each data point in the 'Mention' column, 3) Counting the number of hashtags for each data point in the 'Hashtag' column, 4) Counting the number of times the username reappears within the dataset.

For the first objective, we aim to convert the timestamp into the cyclical format for both time and date to have a more accurate description of the difference in time. This will require us to encode both time and date using a sine and cosine transformation [2]. As the original timestamp column was of a string data type and of an unusual datetime format, it could not be directly converted to datetime data type through any package. Therefore, it was first split into multiple columns and rearranged into one column named "date_time" in the format of "DD MMM YYYY HH:MM:SS". The column is then converted to datetime data type through the dmy_hms function of the lubridate package. An additional "day_of_year" column is added, storing the day of the year of each date before performing the sine and cosine transformations. Upon converting the data into the correct format and type, we conducted sine and cosine transformation using the following formula, with x as the hour and y as the day of the year:

$$Hour\_sine = sin(\frac{2 \times \pi \times x}{23})$$

$$Hour\_cosine = cos(\frac{2 \times \pi \times x}{23})$$

$$Date\_sine = sin(\frac{2 \times \pi \times y}{365})$$

$$Date\_cosine = cos(\frac{2 \times \pi \times y}{365})$$

The new columns are added accordingly.

For the second objective, we wish to count the number of accounts mentioned in each tweet. To achieve this, the "null;" values in the "Mentions" column are first replaced with "NA" and then split into individual strings via the whitespace, hence forming a list. The number of strings for each data point is then returned through an if-else statement and the lengths function, which returns the length of each element of the list. The same approach is applied to the "Hashtag" column.

Finally, for the last objective, we aimed to count the number of times each username reappeared within the dataset to examine the tweeting frequency. This is achieved through grouping the data points by username

and filter for groups with size greater than 1. From there, an additional column "n" is created to store the group size, where group size of 1 is recorded as "NA" as there is no duplicate.

With the cleaned data and converted features, we removed the relevant original columns, giving us the final columns as follows:

Column	**Data Type**	**Description**
Tweet Id	String	ID generated for the tweet
Username	String	Encrypted username
hour_sine	Numeric	Sine transformation of hour of timestamp
hour_cosine	Numeric	Cosine transformation of hour of timestamp
Date_sine	Numeric	Sine transformation of day of year of timestamp
Date_cosine	Numeric	Cosine transformation of day of year of timestamp
#Followers	Integer	Number of followers
#Friends	Integer	Number of friends
#Retweets	Integer	Number of retweets
#Favorites	Integer	Number of favorites
Positive_sentiment	Integer	Positive sentiment generated by SentiStrength.
Negative_sentiment	Integer	Negative sentiment generated by SentiStrength.
#Mentions	Numeric	Number of accounts mentioned in the tweet.
#Hashtags	Numeric	Number of hashtags mentioned in the tweet.
n	Numeric	Number of times the Username reappears within the dataset.
1	Numeric	Entity score of annotated entity identified within the data point.
2	Numeric	Entity score of annotated entity identified within the data point.

Table 2: Processed features for each tweet

4 Model: Random Forest

Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. They are trained based on the entropy and information gain of the nodes. A higher information entropy implies a better model [1].

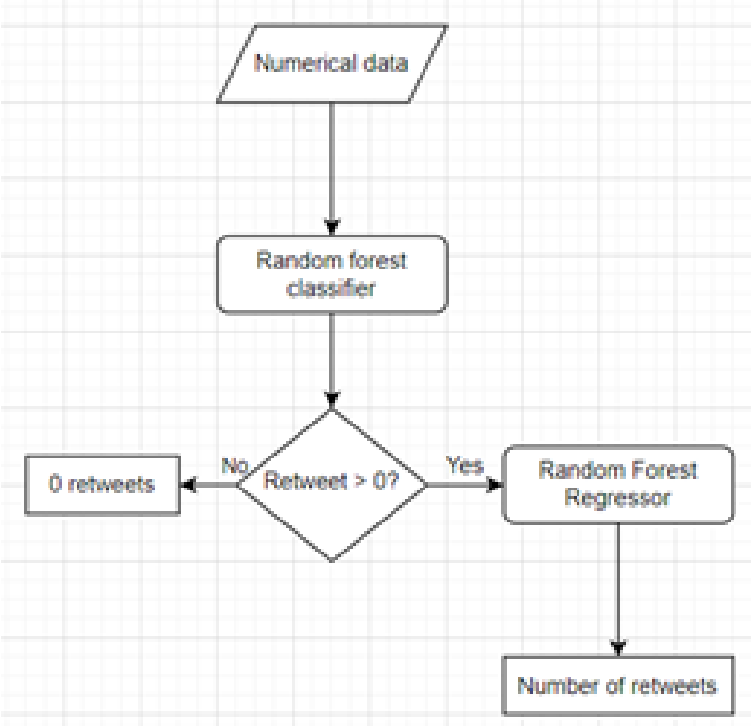


Figure 1: Random Forest model flow

4.1 Training

Training is done in Python's sklearn module [5]. For this model, all the numerical features were used. The number of entities, URLs and hashtags were used in place of the original columns and the parameters listed in the resulting table were used for training both a random forest classifier and regressor.

Category	Details	Remarks
Experiment settings	• Train-test split ratio of 3:1. • Train data used for hyper-parameter tuning	-
Hyperparameter settings	• Max depth: 2 • min_samples_leaf: 1 • min_weight_fraction_leaf: 0 • Allow unlimited leaves	Applies to both random forest regressor and classifier
Target variable	Due to the left-skewed nature of the number of retweets, log(Retweets) was used as the target variable to improve performance	See figure 2
MSLE	1.62905	

Table 3: Implemented Random Forest

The regressor is trained with log(retweets) instead of retweets but at the evaluation, the original number of retweets is recovered before calculating MSE. Regarding the evaluation metric, our model tries to predict log(Retweets) and the result is tested using MSE, which effectively tests the model using MSLE.

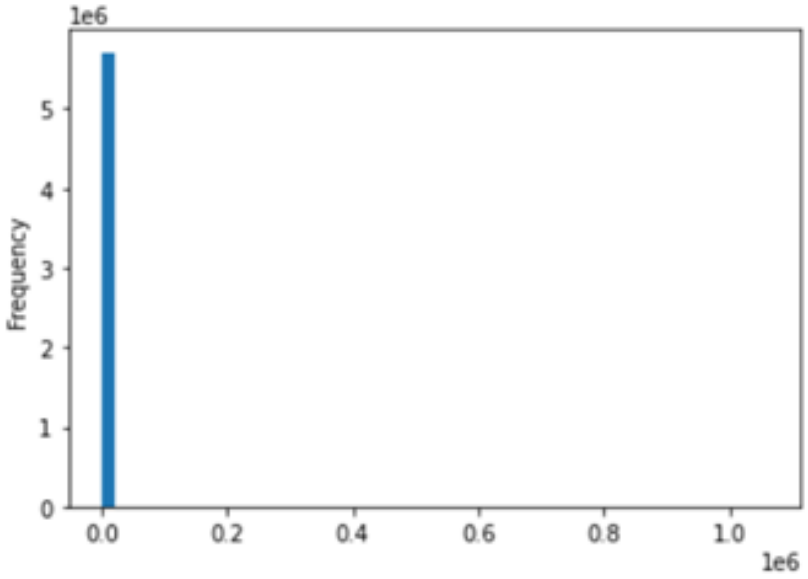


Figure 2: Frequency

4.2 Evaluation

Results

When evaluating, the log(Retweets) are passed through the exponential function to recover the predicted number of retweets before they are compared with the expected output. MSLE was used to evaluate the results and the MSLE value obtained is 1.62905. A graph was plotted to find a suitable training set size and there is a generally positive relationship between the training set size and the MSLE as shown in the log graph below.

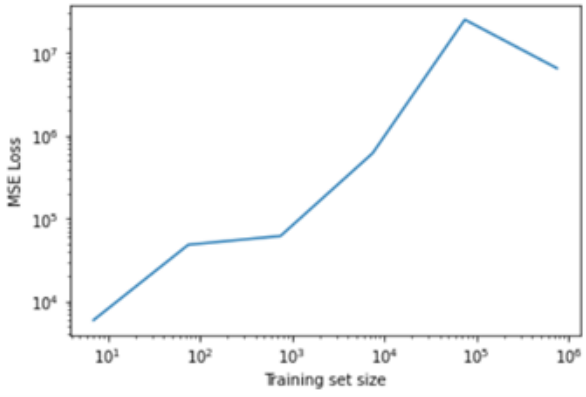


Figure 3: MSEloss plotted again training set size

Comparison with the literature

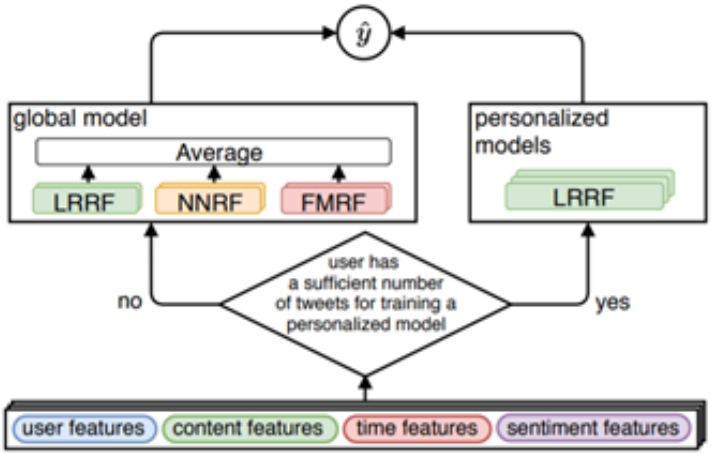


Figure 4: Overview of the approach based on LRRF (Linear Regression-enhanced Random Forest), NNRF (Neural Networks-enhanced Random Forest), and FMRF (Factorization Machine-enhanced Random Forest) [6]

The model from the literature has 2 main groups. 1 for global users and 1 for personalized users. Global users are users who have never appeared in the evaluation dataset whereas personalized users appear in

both training and evaluation sets. The global model uses a mixture of random forests enhanced with linear regression, neural networks and factorization machines, while the personalized model uses linear regression enhanced random forest. Whether an observation is processed by a global or personalized model depends on whether a user has a sufficient number of tweets for training a personalized model. The model ranked 4th overall with a MSLE value of 0.149997.

User (Team)	MSLE
vinayaka (BIAS)	0.120551 (1)
mc-aida (MC-AIDA)	0.121094 (2)
myaunraitau	0.136239 (3)
parklize (PH)	0.149997 (4)
JimmyChang (GrandMasters)	0.156876 (5)
Thomary	0.169047 (6)
⋮	⋮

Figure 5: Leaderboard of the top few models in COVID19 Retweet challenge [6]

5 Model: Neural Network

To improve the MSLE value, we attempted a neural network model that follows a similar architecture as mentioned in the paper by Daichi Takehara [2], as shown in Figure 6. However, for this model, the categorical features and multi-hot categorical features were excluded, and only numerical features were included.

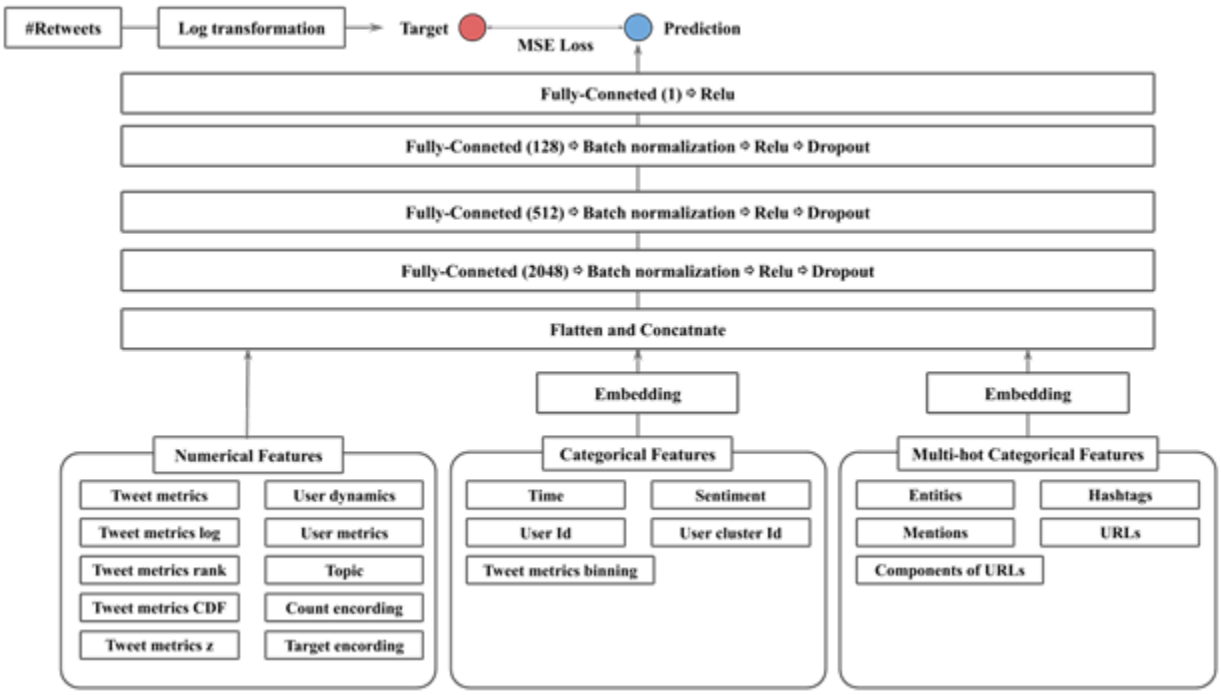


Figure 6: Model architecture by Daichi Takehara [2]

The model consisted of 4 fully connected layers. The layers had a descending number of nodes (2048, 512, 128, 1). Mean Square Loss was used to evaluate the predictions with the log transformed labels.

The model was trained on an ASUS UX430UNR notebook. It has an Intel i5-8250U CPU @ 1.6 GHz with 8GB RAM, and NVIDIA GeForce MX150 GPU.

While training the model, we tested several different values of the parameters and the following lists the results and configuration of each attempt.

5.1 Attempt 1

Configurations

- Data: COVID DATASET 1
- Total Samples: 8M samples
- Data splits: 70% for training, 30% for training
- Selected data Columns: Followers, Friends, Favorites, Timeseg, Weekend, Entity_count, Hashtag_count, Mention_count, Url_count, Ratio_fav_followers, Day_of_week, Sentiment_p, Sentiment_n, Sentiment_ppn
- Learning rate: 0.1
- Batch size: 2500
- Epochs: 100
- Optimizer: Adam
- Loss function: MSLE

Results

Final trainig loss was 0.9939596652984619, and the test loss was 1.93565762042999327. Figure 7 shows the training loss graph recorded during the training process.

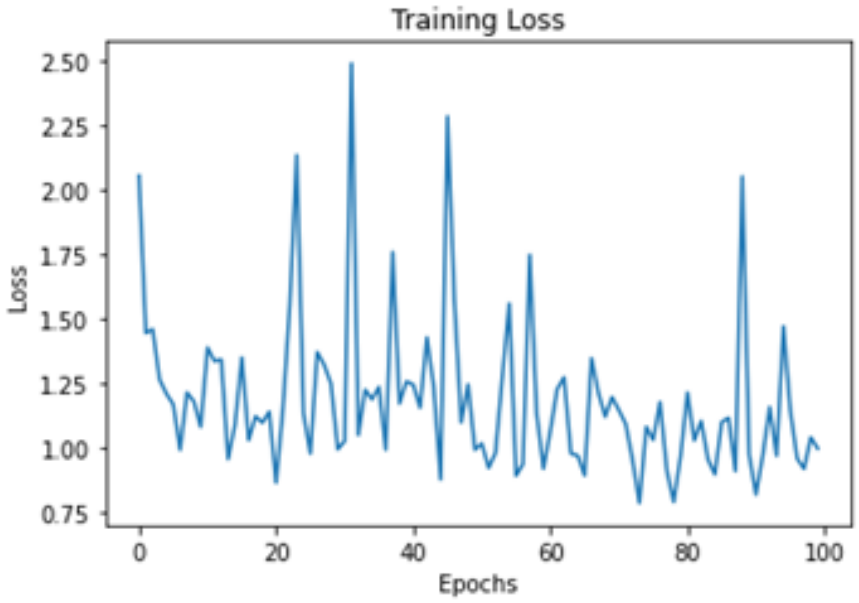


Figure 7: Attempt 1 training loss

The loss graph seems to show that the training process did not improve. The model did not perform well. The possible reason for this could be that the model has been overfitting from the beginning and stayed at that stage throughout the training process. Thus, for the next attempt, we lowered the learning rate and decreased the number of epochs.

The model was saved with the name model_20210803_043938.pt.

5.2 Attempt 2

Configurations

- Data: COVID DATASET 1
- Total Samples: 8M samples
- Data splits: 70% for training, 30% for training
- Selected data Columns: Followers, Friends, Favorites, Timeseg, Weekend, Entity_count, Hashtag_count, Mention_count, Url_count, Ratio_fav_followers, Day_of_week, Sentiment_p, Sentiment_n, Sentiment_ppn
- Learning rate: 0.001
- Batch size: 2500
- Epochs: 20
- Optimizer: Adam
- Loss function: MSLE

Results

Final training loss was 0.7118250727653503, and the test loss was 5.058868885040283. Figure 8 shows the training loss graph recorded during the training process.

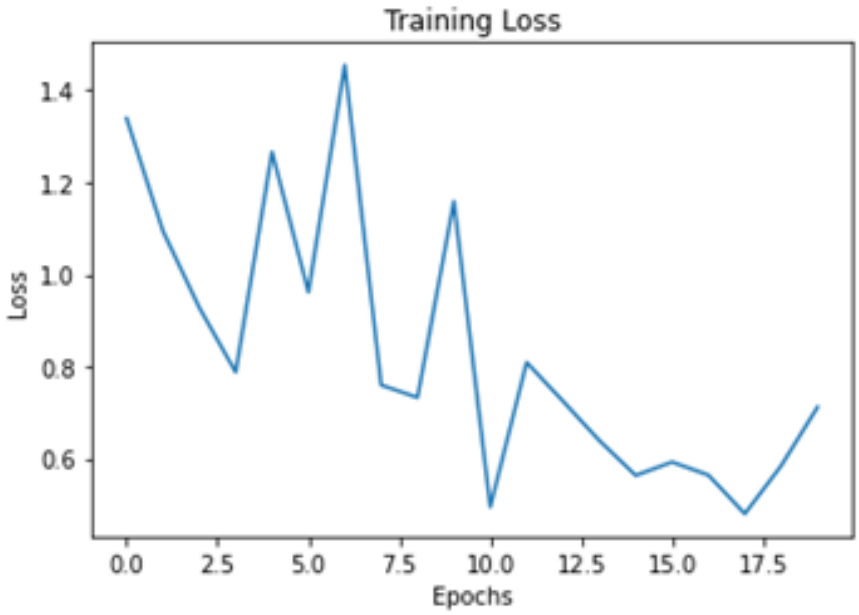


Figure 8: Attempt 2 training loss

The training loss did improve; however, the overall test loss was worse then the previous attempt, even after lowering the learning rate and the number of epochs. This meant that the model is far more overfitted to the training dataset. To fix this, we added more features, trained the model using a larger dataset and decreased the number of epochs. We also removed the batch size, hence the weights will only be updated after the model has run through one whole csv file (maximum 10000 samples). The optimiser is too changed to AdamW.

The model was saved with the name model_20210803_103312.pt.

5.3 Attempt 3

Configurations

- Data: COVID DATASET 1, COVID DATASET 2, COVID DATASET 3
- Total Samples: 20M samples
- Data splits: 70% for training, 30% for training
- Selected data Columns: Followers, Friends, Favorites, Weekend, Entity_count, Hashtag_count, Mention_count, Url_count, tlen, Ratio_fav_followers, Time_importance, day_of_week, Sentiment_ppn, Sine_hour, Cosine_hour, Sine_day, Cosine_day, Sine_day_of_week, Cosine_day_of_week
- Learning rate: 0.001
- Epochs: 5

- Optimizer: AdamW with AMSGrad
- Loss function: MSLE

Results

Final training loss was 0.49519190192222595, and the test loss was 0.9292153011668812. Figure 9 shows the training loss graph recorded during the training process. FIgure 10a and figure 10b show the comparison between the actual and predicted number of retweets.

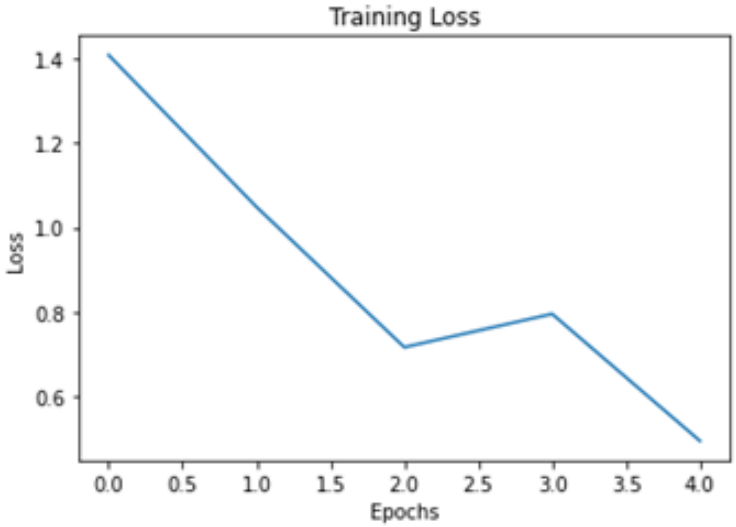
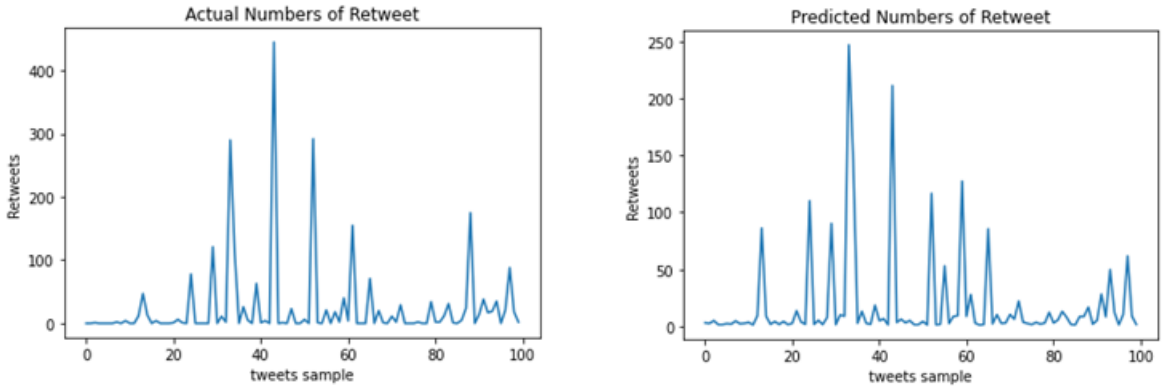


Figure 9: Attempt 3 training loss



(a) Actual

(b) Predicted

Figure 10: Actual vs Predicted retweets

The results are far better than previous attempts. The training loss graph appears to be improving with a promising trend and the test loss improved as well. This is likely due to the lower frequency of weight updates, the change of optimizer and the additional features. Full details of why the AdamW optimiser was

used in replacement of Adam can be found in the section "AdamW with AMSGrad". The addition of the cyclical representation of time and day seem to have played a vital role in the retweet trend hence improving the model's prediction accuracy.

The model was saved with the name Model_20210804_144105.pt.

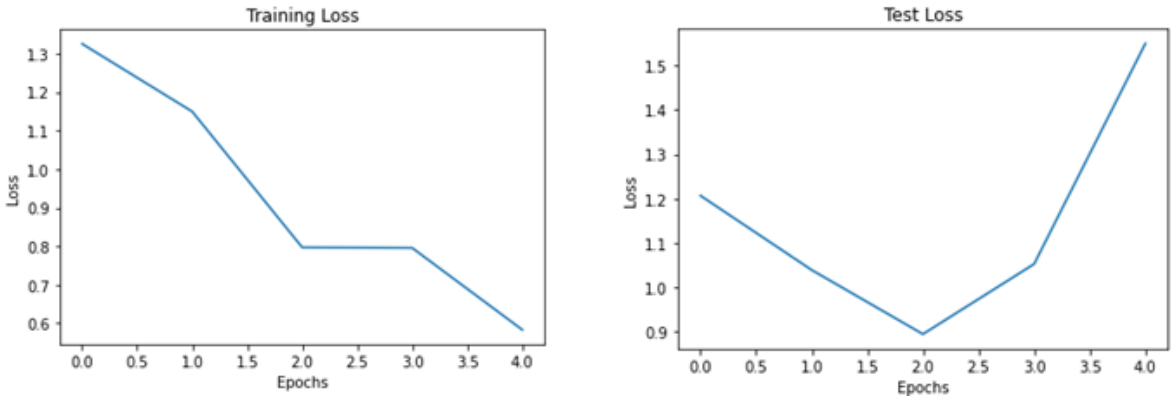
5.4 Attempt 4

Configurations

- Data: COVID DATASET 1, COVID DATASET 2, COVID DATASET 3
- Total Samples: 20M samples
- Data splits: 70% for training, 30% for training
- Selected data Columns: Followers, Friends, Favorites, Weekend, Entity_count, Hashtag_count, Mention_count, Url_count, tlen, Ratio_fav_followers, Time_importance, day_of_week, Sentiment_ppn, Sine_hour, Cosine_hour, Sine_day, Cosine_day, Sine_day_of_week, Cosine_day_of_week
- Learning rate: 0.001
- Epochs: 5
- Optimizer: AdamW with AMSGrad
- Loss function: MSLE

Results

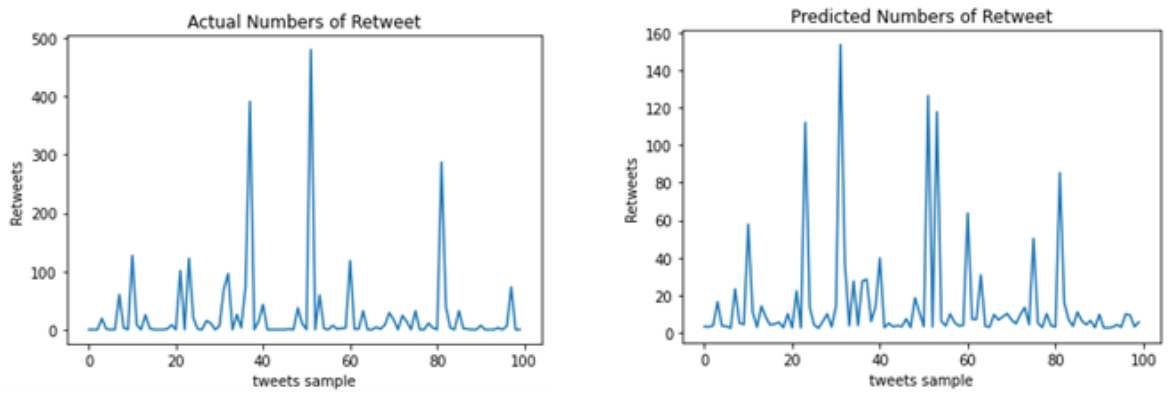
For this attempt, we rerun the model with the same configuration to check if the results are replicable. However, this time the final training loss was 0.583283007144928, and the test loss was 1.5500206781812935. Figure 11a and figure 11b shows the training loss graph and the test loss graph recorded during the training process. Figure 12a and Figure 12b show the comparison between the actual and predicted number of retweets.



(a) Training loss

(b) Test loss

Figure 11: Loss Graphs



(a) Actual

(b) Predicted

Figure 12: Actual vs Predicted retweets

The training loss graph similarly showed a promising trend. However, the test loss graph showed a negative trend. This results could have been caused by the low number of epochs and the high learning rate, even at 0.001. This shows that the model was unstable, hence, the number of epochs have to be increase and learning rate have to be further decreased.

The model was saved with the name Model_20210808_151123.pt

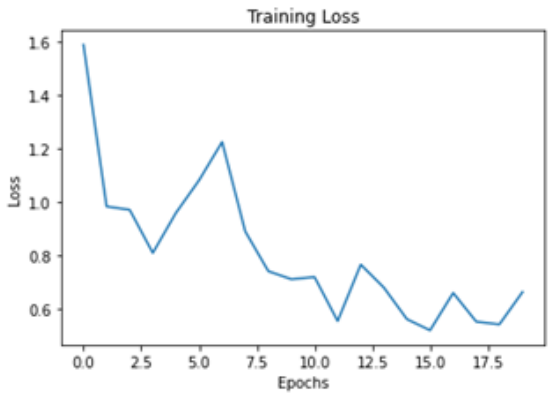
5.5 Attempt 5

Configurations

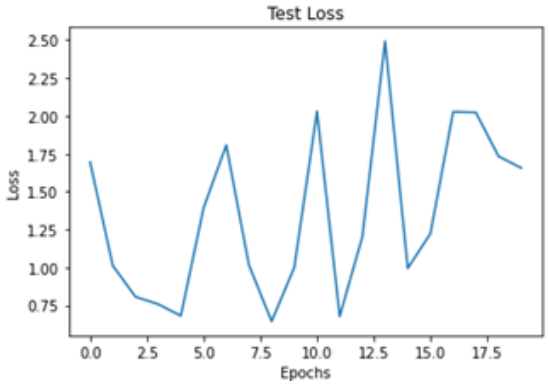
- Data: COVID DATASET 1, COVID DATASET 2, COVID DATASET 3
- Total Samples: 20M samples
- Data splits: 70% for training, 30% for training
- Selected data Columns: Followers, Friends, Favorites, Weekend, Entity_count, Hashtag_count, Mention_count, Url_count, tlen, Ratio_fav_followers, Time_importance, day_of_week, Sentiment_ppn, Sine_hour, Cosine_hour, Sine_day, Cosine_day, Sine_day_of_week, Cosine_day_of_week
- Learning rate: 0.00001
- Epochs: 20
- Optimizer: AdamW with AMSGrad
- Loss function: MSLE

Results

Final training loss was 0.6629476547241211, and the test loss was 1.6563515639502155. Figure 13a and Figure 13b shows the training loss graph and the test loss graph recorded during the training process. Figure 14a and Figure 14b show the comparison between the actual and predicted number of retweets.

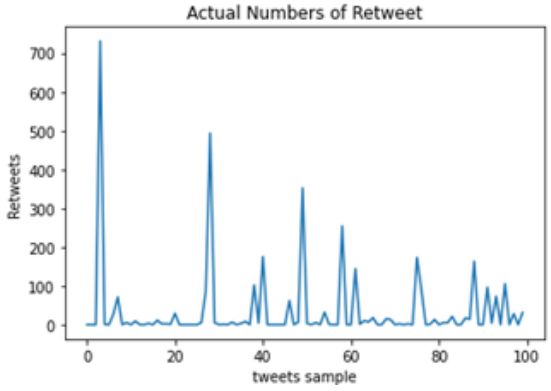


(a) Training loss

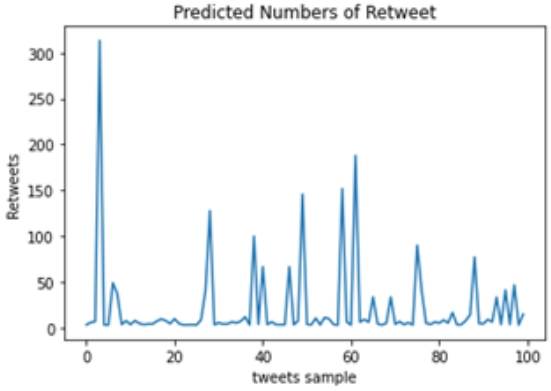


(b) Test loss

Figure 13: Loss Graphs



(a) Actual



(b) Predicted

Figure 14: Actual vs Predicted retweets

The model seems to have overfitted after the 5th epoch. There might also be multiple minimum points, which might explain the results of having multiple point where the loss dipped. To reduce overfitting, we further reduced the learning rate. But to obtain a more stable test results, we attempted to increase the number of epochs.

The model was saved with the name Model_20210809_004242.pt

5.6 Attempt 6

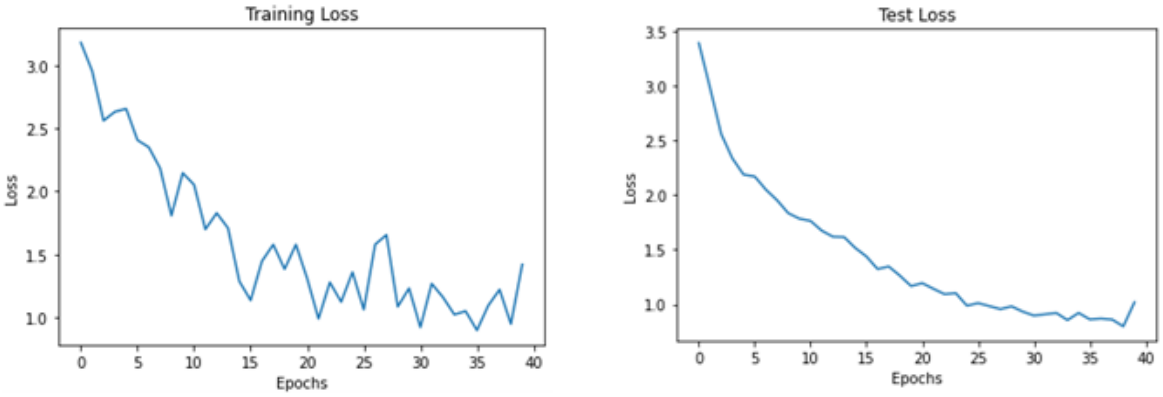
Configurations

- Data: COVID DATASET 1, COVID DATASET 2, COVID DATASET 3
- Total Samples: 20M samples
- Data splits: 70% for training, 30% for training

- Selected data Columns: Followers, Friends, Favorites, Weekend, Entity_count, Hashtag_count, Mention_count, Url_count, tlen, Ratio_fav_followers, Time_importance, day_of_week, Sentiment_ppn, Sine_hour, Cosine_hour, Sine_day, Cosine_day, Sine_day_of_week, Cosine_day_of_week
- Learning rate: 0.000001
- Epochs: 40
- Optimizer: AdamW with AMSGrad
- Loss function: MSLE

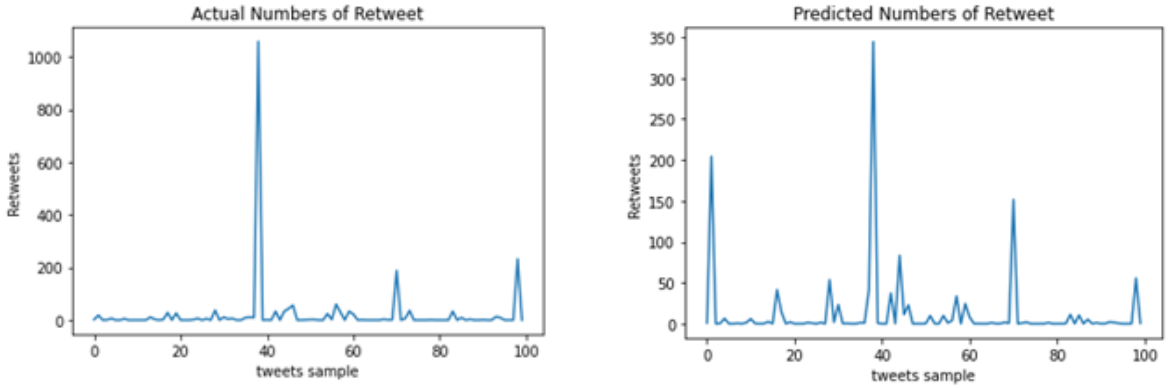
Results

Final training loss was 0.8983719348907471, and the test loss was 0.8625897141527539. Figure 15a and Figure 15b shows the training loss graph and the test loss graph recorded during the training process. Figure 16a and Figure 16b show the comparison between the actual and predicted number of retweets.



(a) Training loss (b) Test loss

Figure 15: Loss Graphs



(a) Actual (b) Predicted

Figure 16: Actual vs Predicted retweets

The training and test loss graphs showed promising trends. The lower learning rate did help in preventing

overfitting. The increase in number of epochs might have improved the model's stability in prediction too. However, the general predictions for number of retweets are lower than the actual retweets.

The model was saved with the name Model_20210809_032813_35.pt

6 AdamW with AMSGrad

According to Reddi et al. (2018) as mentioned in the article by Sebastian Ruder [7], the AMSGrad version of the Adam optimizer prevents the adaptive learning rate from becoming infinitely small and diminishing training performance. With an infinitely small learning rate,the model stops learning and might lead to poor convergence. Such behaviour is often observed in Adam and Adagrad optimizers when dealing with big datasets, where the adaptive learning rate diminishes to infinitely small, and the model ceases to learn. Hence, this might help in preventing both overfitting and underfitting as the datasets used in our problem is also huge: there are total of 20 million tweet samples.

7 Comparison of MSLE

Model	**MSLE**
Random Forest	1.62905
NN attempt 1	1.93565762042999327
NN attempt 2	5.058868885040283
NN attempt 3	0.9292153011668812
NN attempt 4	1.5500206781812935
NN attempt 5	1.6563515639502155
NN attempt 6	0.8625897141527539

Table 4: Comparison of MSLE

8 Conclusion

In conclusion, the neural network model saved at the 6th attempt yield the most preferrable results based of the MSLE Loss. The graph of predicted retweets for each tweet is almost similar to the graph of actual retweets. The random forest model performed worse possibly due to our lack of understanding on how to optimize the model. However, the neural network model still can be improved as the general predictions for number of retweets are lower than the actual retweets. Possible ways to improve the model might be to further lower the learning rate and increase the number of the epochs to train, select more suitable features, and to introduce embedding of the categorical features as suggested by the paper by Daichi Takehara [2].

9 GitHub links to code

Random Forest Model: https://github.com/beazt123/AI-project-2021-SUTD.git

Neural Network Model: https://github.com/Qiftan/AI-Project2021SUTD

GUI: https://github.com/beazt123/AI-Term-8-GUI.git

Bibliography

[1] L. Breiman, *Random forests*, Statistics Department University of California Berkeley, CA 94720, January 2001. [Online]. Available: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf, (Accessed: 9 August 2021).

[2] haradai1262, *Cikmanalyticup-2020 covid-19 retweet prediction challenge*, Github, 26 November 2020. [Online]. Available: https://github.com/haradai1262/CIKM2020-AnalytiCup, (Accessed: 9 August 2021).

[3] *Leading countries based on number of twitter users as of july 2021*, Statista, July 2021. [Online]. Available: https://www.statista.com/statistics/242606/number-of-active-twitter-users-in-selected-countries/, (Accessed: 9 August 2021).

[4] parklize, *Team ph at the covid-19 retweet prediction challenge at cikm2020 analyticup*, Github, 14 June 2020. [Online]. Available: https://github.com/parklize/cikm2020-analyticup, (Accessed: 9 August 2021).

[5] F. Pedregosa, G. Varoquaux, A. Gramfort *et al.*, *Scikit-learn: Machine learning in python*, Journal of Machine Learning Research, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html, (Accessed: 9 August 2021).

[6] G. Piao and W. Huang, *Regression-enhanced random forests with personalized patching for covid-19 retweet prediction*, ResearchGate, October 2020. [Online]. Available: https://www.researchgate.net/publication/344587686_Regression-enhanced_Random_Forests_with_Personalized_Patching_for_COVID-19_Retweet_Prediction, (Accessed: 9 August 2021).

[7] S. RUDER, *An overview of gradient descent optimization algorithms*, Ruder, 19 January 2016. [Online]. Available: https://ruder.io/optimizing-gradient-descent/, (Accessed: 9 August 2021).

[8] A. van Wyk, *Encoding cyclical features for deep learning*, Kaggle, 2018. [Online]. Available: https://www.kaggle.com/avanwyk/encoding-cyclical-features-for-deep-learning, (Accessed: 9 August 2021).