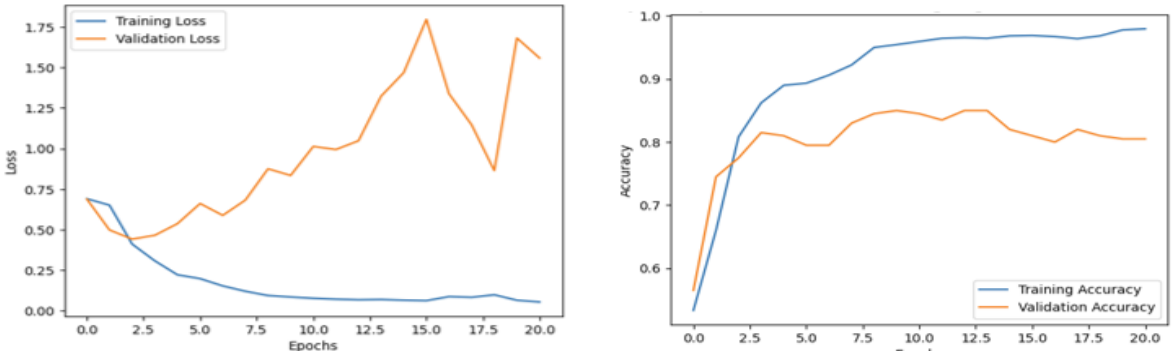
Assignment 2 – CSC3066 Deep Learning (Fake News Detection)

Student Name: Ryan McKee
Student Number: 40294886
Date: 07/04/2024

The implementation and evaluation of various Artificial Neural Network (ANN) models aimed to address the challenge of automatically classifying Twitter posts as True or Fake based on their veracity. Throughout Task One, several techniques and settings were experimented with to enhance the performance of each model architecture. This report provides a comprehensive analysis of the outcomes achieved, justifying the selection of techniques, discussing their impact on the models, and drawing conclusions from the observed results. Initially, hyperparameter and model architecture decisions were made based on intuition due to time constraints. The choice of ReLU activation function efficiently captures complex textual patterns, which are crucial for our task, while leveraging GloVe word embeddings of 300 dimensions per token. These embeddings provide a dense representation of words, enabling the model to train efficiently while retaining the semantic meaning of words learned from a large text corpus. The diverse nature of the corpus enables our model to capture general semantic information, enhancing its generalizability and adaptability to different types of sentences. In the output layer, we employ the sigmoid activation function to ensure outputs between 0 and 1, which is ideal for binary classification tasks such as identifying fake information. This is complemented using cross-entropy loss. To accurately assess errors and performance on unseen data, we utilize 10-fold cross-validation. This approach was chosen due to the relatively small size of our training set (2000 samples). Further experimentation will focus on optimizing parameters and exploring baseline settings to achieve the best solution.

MLP with Single Flattened Word Embedding Vector:

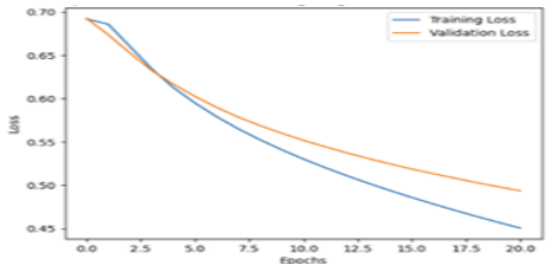
Initial experimentation with this model revealed promising results with 078 accuracy on testing data and took only 8.4 seconds per fold to train.

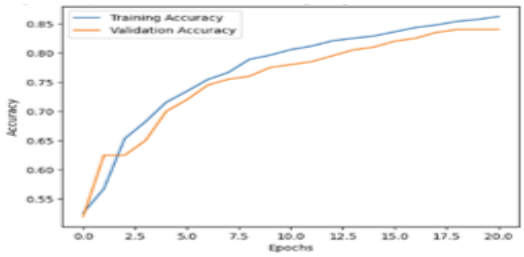


However, I encountered stability issues observed in the learning curve attributed to potentially high learning rates and noisy training data. Lowering of the learning rate from originally 0.05 to 0.0001 improved the stability and made a better learning curve as it allowed for the model at each increment to make smaller parameter updates therefore reducing the likelihood of large weight updates that cause the model to overshoot or diverge from optimal solutions and it makes it easier for the adam optimization to escape from local saddle points which may be holding the model back. However there was still quite significant overfitting to overcome this issue I experimented with lowering neurons in hidden layer from 5 to 4 in an attempt to improve the generalizability of the model as model complexity was making it learn the training data too well and reducing batch size from 128 to 64 for the purpose of introducing more variability into the optimization process updating mitigating the risk of overfitting finally regularization using Lasso to encourage sparsity and perform feature selection to automatically select the most relevant features and shrinking the less important

features of the samples to zero for improving training, and data preprocessing steps including stop word removal and lemmatization were employed. These adjustments led to a significant enhancement in stability, reduced overfitting, and improved performance on the test set, resulting in a viable model for classifying false information with a final accuracy of 0.81 and model training time was reduced to 6.4 seconds per fold with a good, fitted learning curve.

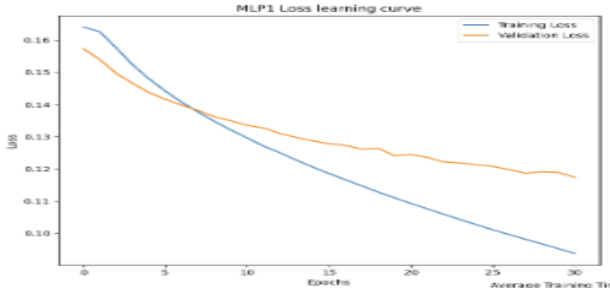
Final Model Testing stats	Predicted negative	Predicted positive
True negative	203	22
True Positive	72	203
Accuracy	0.812	
Precision	0.902	
Recall	0.738	
F1 Score	0.812	

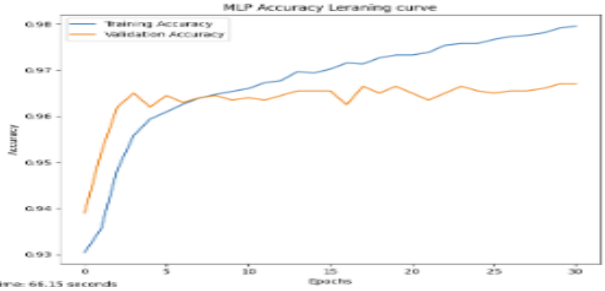




MLP with Keras Embedding Layer:

Building upon the previous MLP model, this iteration utilized the Kera's Embedding layer to capture semantic relationships and contextual information more effectively and after experimentation it was found the most optimum performance occurred when using the model architecture- pre-processing and hyperparameters from the previous model. However, overfitting was observed due to the complexity of the model being too high for the smaller dimension inputs captured by the Kera's embedding layer which inherently seeks more meaningful sample representation. Reduction in the number of neurons from 5 to 4 helped mitigate overfitting while improving performance metrics to the previous MLP model increasing testing accuracy to 0.85 from 0.81. Despite increased training time to 16.4 seconds due to the additional model layer, this model leveraging Kera's embedding layer will be easier to use by the client as they won't have to manually embed their samples in glove embeddings instead the model will have a layer to do this for them, the final optimized Kera's embedding layer model is also an overall performance improvement on the original therefore making it the better of the two models and highlighting the usefulness of kera's embedding over manually embedding

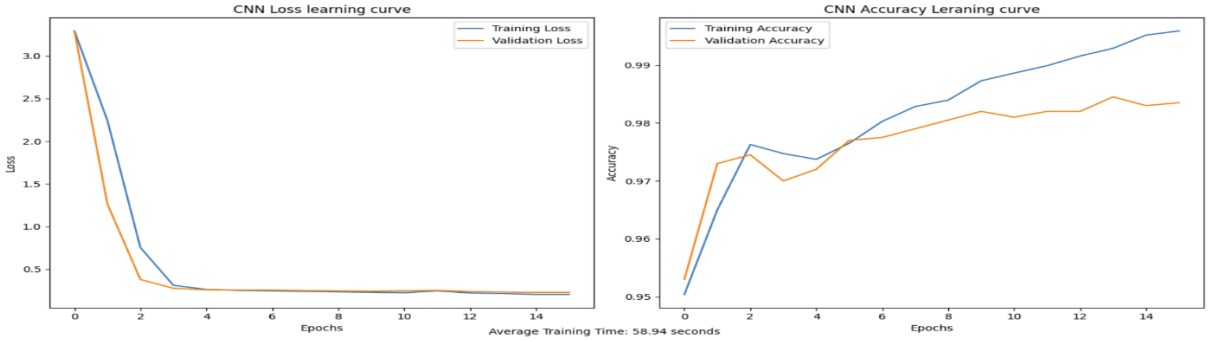




Final Model Testing stats	Predicted negative	Predicted positive
True negative	203	22
True Positive	54	221
Accuracy	0.85	
Precision	0.91	
Recall	0.80	
F1 Score	0.85	

Convolutional Neural Network (CNN) Model:

CNN architecture aimed to capture spatial dependencies within sentences for improved text classification with accuracy of 0.788. Experimentation with CNN revealed significant overfitting initially with the filter number of 128 and filter size of 4, This was address by a reduction of the number of filters to 64 and an increase in the filter size to 5 which increased generalisability this was aided by reducing the number of hidden layer neurons from 5 initially to 4. These modifications alongside the pre-processing data cleaning resulted in the best-performing in terms of classification accuracy at 0.87 model among the tested it takes significantly longer to train when including the convolutional and pooling layer on top of the embeddings taking 74.56 seconds to train it also exhibits some instability during training even after finding the most optimized version of this model I tried dropout layers and lasso regularisation which did improve the model much with dropout actually decrementing performance likely because there being only 4 neurons in the hidden layer making this redundant or even actually decrementing to performance.

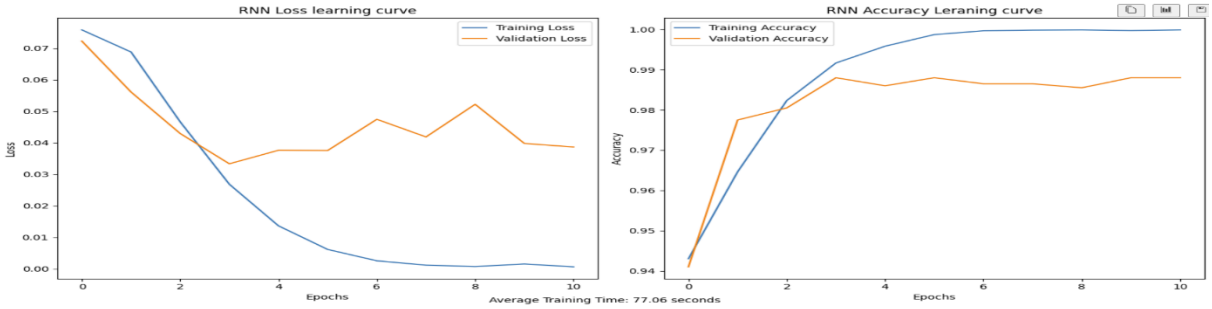


Final Model Testing stats	Predicted negative	Predicted positive
True negative	182	43
True Positive	23	252
Accuracy	0.87	
Precision	0.85	
Recall	0.92	
F1 Score	0.88	

Recurrent Neural Network (RNN) Model:

The RNN model, designed to capture long-range dependencies and contextual information crucial for text analysis, was optimized with various hyperparameters. Surprisingly, a simple architecture featuring a non-complex LSTM recurrent model with just 4 units and ReLU activation emerged as highly effective. This streamlined approach achieved rapid convergence within 2 epochs and yielded an impressive accuracy of 0.88. However, training time averaged 77 seconds per fold, warranting consideration in time-sensitive applications. Nonetheless, the model's notable accuracy highlights its potential utility.

Final Model Testing stats	Predicted negative	Predicted positive
True negative	200	25
True Positive	37	238
Accuracy	0.88	
Precision	0.90	
Recall	0.87	
F1 Score	0.88	



Model Choice

Among the array of models investigated, the Multi-Layer Perceptron (MLP) with the Keras Embedding layer emerges as a compelling choice for several compelling reasons. Despite its marginally lower accuracy compared to its counterparts, the Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) models, the MLP boasts significant advantages in terms of training efficiency and stability. The integration of the Keras Embedding layer streamlines the input process, facilitating the seamless incorporation of new training data without necessitating intricate embedding requirements. Its swifter training times and consistent learning curves make it particularly suitable for real-time applications or environments with constrained computational resources. Achieving an accuracy rate of 0.85, with potential for further enhancements with expanded datasets, underscores the MLP's viability as a solution for the classification task at hand. Moreover, its interpretability and user-friendly nature render it well-suited for deployment in practical settings, prioritizing accessibility, and ease of use.

Conversely, while both the CNN and RNN models yielded superior accuracy scores, they entail notable trade-offs in terms of training duration, complexity, and resource consumption. Despite the CNN model's impressive accuracy of 0.87, its lengthier training times attributable to convolutional and pooling layers represent a significant drawback. Similarly, the RNN model, although boasting an accuracy of 0.88, exhibits extended training durations and heightened computational expenses. While these models may potentially offer superior performance with increased data and complexity, their practical utility may be hindered by hardware constraints and considerations regarding training duration.

However, it is crucial to acknowledge the challenges faced during the project, such as hardware limitations impacting computation due to GPU constraints. Reliance on CPU for model training resulted in prolonged training times, limiting the exploration of various architecture combinations, and leaving potential for further optimization untapped. To address these limitations and enhance model performance, implementing a grid search algorithm for parameter tuning and leveraging faster hardware or cloud computing for model training are recommended. Additionally, augmenting the training dataset beyond the existing 2000 samples and exploring advanced preprocessing techniques, such as custom embeddings for URLs and numbers, could further bolster classification accuracy. Presently, while the proposed MLP model may not suffice as a standalone fake content detector due to the risk of false positives, as indicated by its precision score of 0.91, it could serve as a valuable supportive tool for manual checkers. Further experiments aimed at establishing confidence thresholds for automated classifications are warranted to ensure high accuracy while preserving human oversight for ambiguous cases.