NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4042 Neural Network and Deep Learning

SentEx

Assignment Report

Group Members	
Name	Matriculation No.
Teo Jun Yi, Austin	U2022547L
Gupta, Suhana	U1923230B
Foo Chuan Ann	U1922138G

Table of Contents

Declaration of Original Work

I hereby declare to have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work.

We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work. In addition, disciplinary actions may be taken.

Name	Course	Lab Group	Signature /Date
Teo Jun Yi, Austin	CZ4042	SC4	Austin 09/11/22
Gupta, Suhana	CZ4042	SC4	Suhana 09/11/22
Foo Chuan Ann	CZ4042	SC4	Chuan Ann 09/11/22

1. Project Objective

This project aims to develop machine learning models, through deep learning techniques, that are capable of sentiment analysis (pertaining to text). With the aid of a Recurrent Neural Network (RNN) trained model and a Convolutional Neural Network (CNN) trained model, we attempt to determine the advantages and disadvantages of each.

The dataset that we are using to train the models is the IMDB Movie Review Dataset. (https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews)

Our final models will be

- Trained with the constraint of a small dataset
- Comparing CNNs and RNNs

2. Development Methodology

2.1 Development Environment

Keras, a popular, tested-proven API written in Python is used for our developmental needs. The flexible yet robust nature of Keras' implementation allows for deep learning projects to be written in a clear and readable fashion.

2.2 Data Pre-processing

The dataset consists of 50,000 movie reviews of which 50% is of a positive sentiment while 50% is of a negative sentiment. As IMDB reviews are rated in a score range of 1 to 10, reviews that score ≤ 4 out of 10 are classified as negative while reviews that score ≥ 7 out of 10 are classified as positive by the dataset authors. Neutral reviews are not included in the dataset, hence we are able to expect word tokens that are strongly veered towards the extremities of the two (negative and positive) spectrum.

2.2.1 Cleaning of Data (Tokenization and Lemmatization)

Due to the source of data being user-generated input, it must be sanitized and standardized before being trained on. Regular expressions are used to filter out improper words or any text that would otherwise not be considered to be part of the English natural language.

The data is then tokenized and lemmatized which removes any words that would otherwise be considered meaningless by itself in English. This leaves us with root words that have meaning and are impactful in conveying contextual semantics which we will be able to train our model on.

Some words in the English dictionary are inherently perceived as negative while others are perceived as positive. With this, we are able to construct a dictionary of words that can be used to achieve a probabilistic sentiment analysis by occurrence of certain keywords.

2.2.2 Data Segmentation into Sets (Train, Valid, Test)

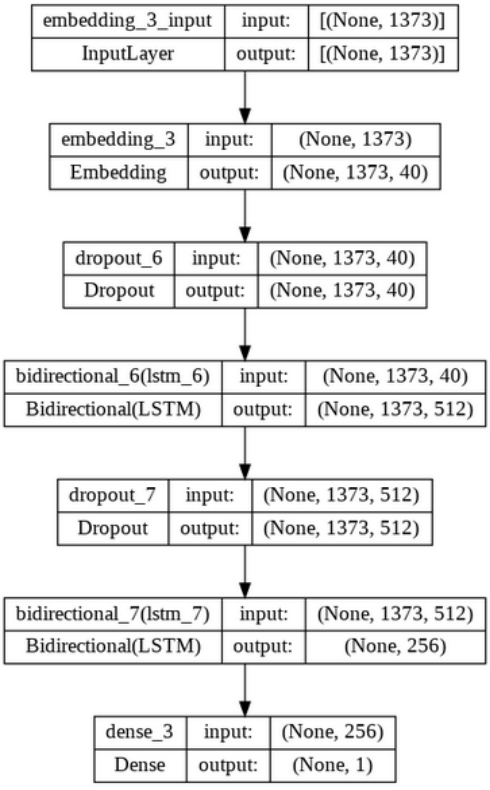
The data is segmented into 3 distinct, non-overlapping sets with a makeup of **30%**, **52.5%**, **17.5%** for Train, Valid, Test sets respectively.
This is equivalent to **15000** training samples, **26250** validation samples, and **8750** evaluation (test) samples.

2.3 Proposed Architecture

We have explored using both Recurrent Neural Networks (RNN) and Convolutional Neural Network (CNN) to achieve our target objective.

2.3.1 Bi-Long Short-Term Memory (LSTM)+Attention

We are using a bi-directional LSTM model, that is an RNN modified to suit our needs. The model is trained for **50** epochs, using the **Adamax optimizer** with a learning rate set to **0.01**, with a batch size of **256.**



2.3.3.1 Concept

LSTM (Long Short Term Memory) is a type of recurrent neural network (RNN) that has been widely used for sequential prediction problems. It is made up of neurons, otherwise known as memory cells, that contain weights and three types of gates: input, forget and output gate. It was developed in order to solve the disadvantages of RNN, hence LSTM is capable of handling long-term dependencies and improves the problem of vanishing gradients (loss with respect to a particular set of weights). LSTM also has an excellent time complexity of O(1) to update each weight. As a result of the property of selectively remembering patterns for long periods of time, we include more regulating parameters and gates that govern the flow and mixing of inputs based on training weights using LSTM, allowing for greater flexibility in managing the outputs.

2.3.1.2 Implementation

Bidirectional LSTM (Bi-LSTM) enables neural networks to have both backward and forward information about the sequence at each time step. Using bidirectional will run inputs in two directions, one from the past to the future and one from the future to the past, maintaining information from both the past and the future at every point in time. Using knowledge from the future will help the network comprehend what the next word is and hence better forecast the outputs. We required such a network since predicting the word in a phrase coupled with its context made predicting its class easier and helped improve the accuracy.

Furthermore, Keras's attention layer is based on the notion of cognitive attention. This effect allows the neural network to concentrate on relevant bits of the input data while fading out the remainder. The idea behind this method is to draw the focus and attention to a tiny yet significant portion of the data. Because they store information in a memory, LSTMs allow RNN to remember inputs over time. Because of this trait, it is critical to emphasize significant sections of the phrase when deciding so that we may shift our focus to vital elements and efficiently employ computer resources across each sentence. The attention class accepts a layer and then performs the necessary functions, building the layer by adding the necessary weights and biases based on the input shape, and then obtaining the required outputs.

To create the final architecture, we introduced an embedding layer and then passed the Bidirectional LSTM layer to the attention class.

2.3.2 Basic RNN

For RNNs, we tried a total of four different basic architectures:

1-Layer GRU

An embedding input layer is used, followed by a GRU layer with 100 neurons. The output of the GRU layer is then passed to a dropout layer with a dropout chance of 50% to prevent overfitting. Lastly it is passed to an output layer with sigmoid activation.

2-Layer GRU

An embedding input layer is used, followed by a GRU layer with 100 neurons with return_sequences enabled. The output of the GRU layer is then passed to a dropout layer with a dropout chance of 50% to prevent overfitting. It is then passed on to another GRU layer which does not have return_sequences enabled, similarly having its output passed to a dropout layer of 50% chance. Lastly it is passed to an output layer with sigmoid activation.

1- Layer LSTM

An embedding input layer is used, followed by a LSTM layer with 100 neurons. The output of the LSTM layer is then passed to a dropout layer with a dropout chance of 50% to prevent overfitting. Lastly it is passed to an output layer with sigmoid activation.

2- Layer LSTM

An embedding input layer is used, followed by a LSTM layer with 100 neurons with return_sequences enabled. The output of the LSTM layer is then passed to a dropout layer with a dropout chance of 50% to prevent overfitting. It is then passed on to another LSTM layer which does not have return_sequences enabled, similarly having its output passed to a dropout layer of 50% chance. Lastly it is passed to an output layer with sigmoid activation.

We trained all four models over **80** epochs, using **Adam optimizer** with a learning rate set to **0.01**, and a batch size of **512**.

2.3.3 Basic CNN

For CNNs, we tried a total of three different basic architectures:

2-Layer CNN

The 2-Layer CNN contains 2 Conv1D layers that are connected with dropout layers of 30% in between them. It is then connected to a relu activation layer with 20 neurons before a dropout layer of 50% and then outputted to a sigmoid activation layer

3-Layer CNN

The 3-Layer CNN contains 3 Conv1D layers that are connected with dropout layers of varying dropout percentages in between them. It is then connected to a relu activation layer with 100 neurons before a dropout layer of 50% and then outputted to a sigmoid activation layer

4-Layer CNN

The 4-Layer CNN contains 4 Conv1D layers that are connected with dropout layers of varying dropout percentages in between them. It is then connected to a relu activation layer with 100 neurons before a dropout layer of 50% and then outputted to a sigmoid activation layer

The CNN models were trained over **150** epochs, using **Adam optimizer** with a learning rate set to **0.01**, and a batch size of **512**, to study the behavior of each architecture.

3. Experimentation and Results

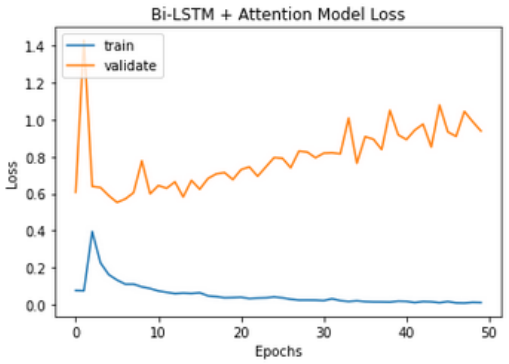
Omitted results

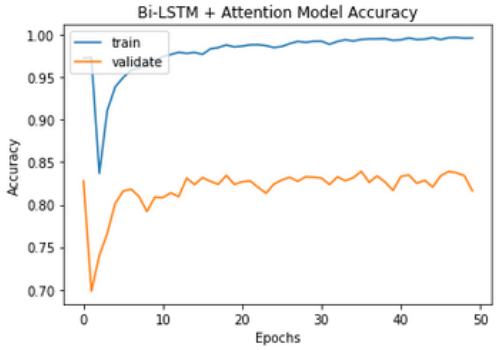
Results for 2-Layer GRU (RNN), 2-Layer LSTM (RNN), 3-Layer CNN, 4-Layer CNN were omitted due to unsatisfactory results. Only the best performing models for each category were included.

The following evaluation sample mentioned in this section refers to the evaluation set that was segmented from the IMDB data, which is of the same domain as training data, unless otherwise stated.

3.1 Bi-Long Short-Term Memory (LSTM)+Attention

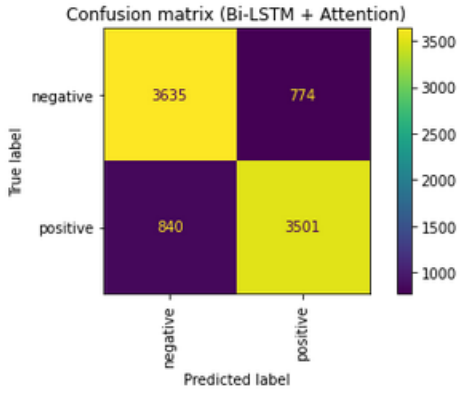
Evaluation metrics:





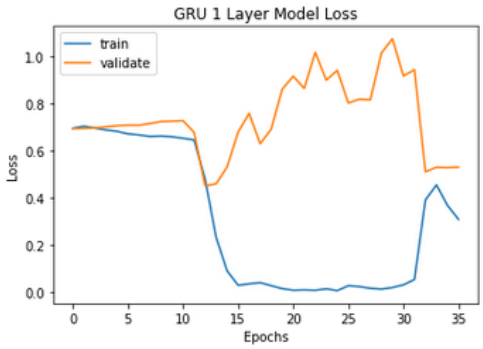
	precision	recall	f1-score	support
0	0.8123	0.8244	0.8183	4409
1	0.8189	0.8065	0.8127	4341
accuracy			0.8155	8750
macro avg	0.8156	0.8155	0.8155	8750
weighted avg	0.8156	0.8155	0.8155	8750

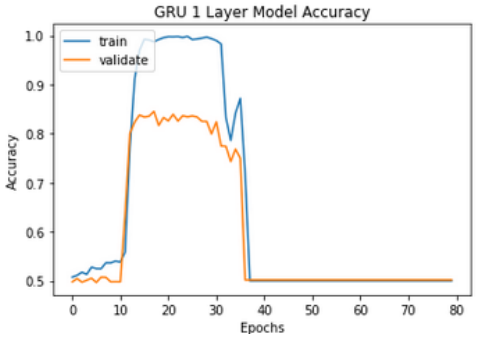
Micro F1 Score: 0.8155428571428572
Macro F1 Score: 0.8154995867504287
Accuracy Score: 0.8155428571428571



3.2 Basic RNN

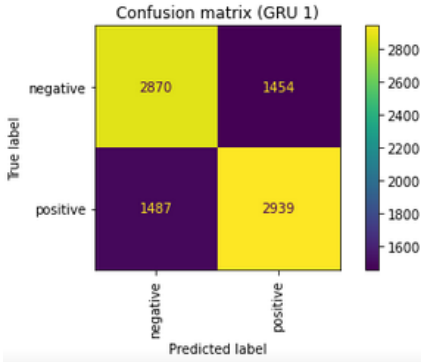
1-Layer GRU



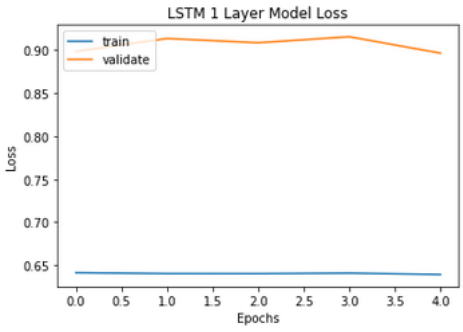


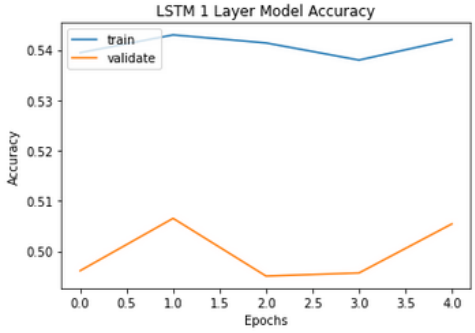
	precision	recall	f1-score	support
0.0	0.6587	0.6637	0.6612	4324
1.0	0.6690	0.6640	0.6665	4426
accuracy			0.6639	8750
macro avg	0.6639	0.6639	0.6639	8750
weighted avg	0.6639	0.6639	0.6639	8750

Micro F1 Score: 0.6638857142857143
Macro F1 Score: 0.6638648118905393
Accuracy Score: 0.6638857142857143



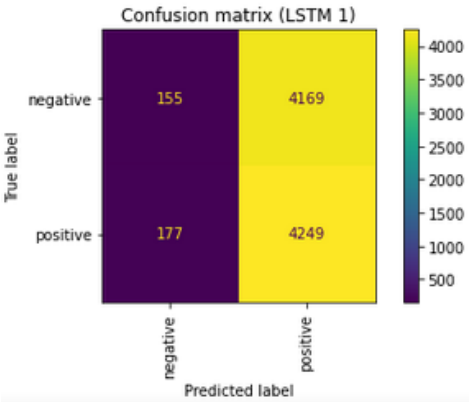
1- Layer LSTM





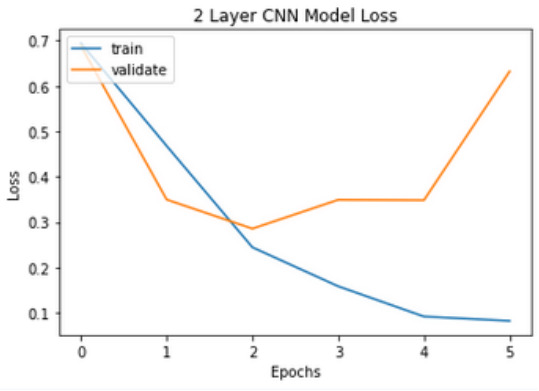
	precision	recall	f1-score	support
0.0	0.4669	0.0358	0.0666	4324
1.0	0.5048	0.9600	0.6616	4426
accuracy			0.5033	8750
macro avg	0.4858	0.4979	0.3641	8750
weighted avg	0.4860	0.5033	0.3676	8750

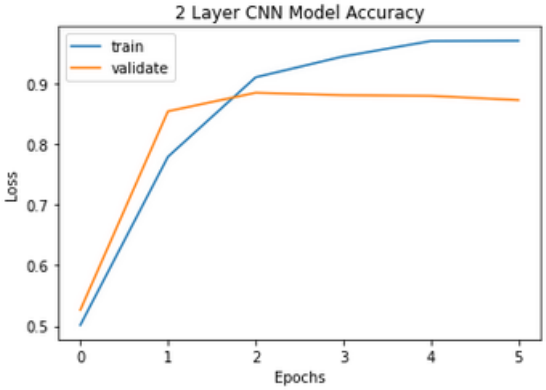
Micro F1 Score: 0.5033142857142857
Macro F1 Score: 0.3641063231952877
Accuracy Score: 0.5033142857142857



3.3 Basic CNN

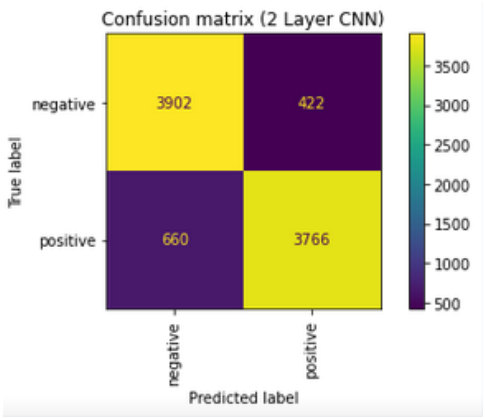
2-Layer CNN





	precision	recall	f1-score	support
0.0	0.8553	0.9024	0.8782	4324
1.0	0.8992	0.8509	0.8744	4426
accuracy			0.8763	8750
macro avg	0.8773	0.8766	0.8763	8750
weighted avg	0.8775	0.8763	0.8763	8750

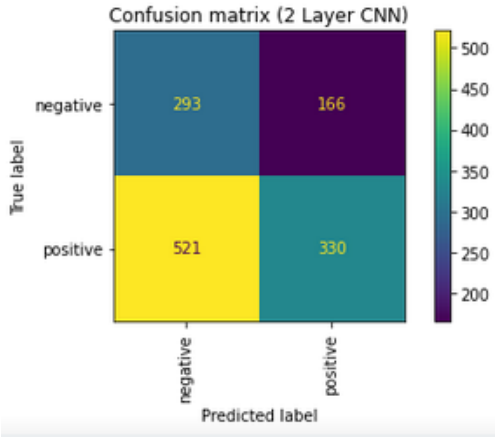
Micro F1 Score: 0.8763428571428571
Macro F1 Score: 0.8763129767813035
Accuracy Score: 0.8763428571428571



3.4 Domain Adaptation

	precision	recall	f1-score	support
0.0	0.3600	0.6383	0.4603	459
1.0	0.6653	0.3878	0.4900	851
accuracy			0.4756	1310
macro avg	0.5126	0.5131	0.4752	1310
weighted avg	0.5583	0.4756	0.4796	1310

Micro F1 Score: 0.47557251908396947
Macro F1 Score: 0.4751538287929711
Accuracy Score: 0.47557251908396947



Our model attempts to predict video game reviews from https://www.cs.jhu.edu/~mdredze/datasets/sentiment/.

Unfortunately it seems like there is too large of a disparity in the domains of the trained model. Upon manual review of the dataset used, we observe that it was largely because there is a large sample count of reviews that are objectively criticizing the game instead of negatively commenting on aspects of the game. The reviews are not as emotional as the ones in the IMDB dataset. In lieu of the objective criticisms (pointing out which aspect of the game is not polished etc.), our model has had trouble making a prediction since there was not much it could go on as reviews were just pointing out pros and cons of the games.

4. Conclusion

The Bi-directional LSTM + Attention model managed to achieve **81.5%** accuracy when evaluated with the evaluation sample. The model is able to predict the classes with relatively equivalent success rates. However, a simple 2-Layer CNN was able to outperform the best performing RNN model, with a classification accuracy of **87.6%** when evaluated against the evaluation samples. Despite having a high loss rate, the 2-Layer CNN was still able to achieve the highest accuracy amongst the different architectures we have experimented with.

However, we hypothesize that we might be able to have more performance uplift by utilizing a hybrid of CNN and RNN to be able to extract the best of both worlds. However, due to time constraints, we are unable to perform any additional research and experimentations into the feasibility of utilizing CRNNs.

4.1 Training with Small Sample Counts

Despite having a dataset of 50,000 samples, we have only utilized **30% (15,000 samples)** of the samples for training purposes. Within the same domain (IMDB dataset, without cross contamination of data), the 2-Layer CNN and Bi-LSTM + Attention model did well in making accurate predictions.

4.2 Domain Adaptation

Unfortunately we are unable to find good datasets of a different but relevant domain. Many of the datasets that were available online were derived mostly from the same source. Hence, resorting to utilizing our model to make predictions of video game reviews with the intentions of limit testing our trained model.