College of Engineering & Applied Science
UNIVERSITY OF COLORADO BOULDER

University of Colorado Boulder **Bhuvanendra Putchala**
CSCI 5922: Neural Nets and Deep Learning **Lab3**
Prof. Danna Gurari **March 13, 2024**

Q1

Introduction

The Amazon Reviews is a collection created to advance text classification using deep learning methods. With the growth of e commerce there has been an increase, in customer reviews offering valuable insights into consumer opinions on products. These reviews encompass a range of sentiments from praises to critical feedback providing an array of perspectives that can be used to enhance product recommendations improve customer satisfaction and refine search algorithms.

This dataset focuses on classifying reviews into three sentiment categories; Negative, Neutral and Positive. This classification allows for a understanding of consumer emotions and viewpoints beyond just numerical ratings. Key features of the dataset include 'Sentiments indicating the sentiment category of each review; 'Cleaned Review' a preprocessed text column, for analysis; 'Cleaned Review Length' which gives the length of the review after cleaning; and 'Review Score' representing the original rating provided by the reviewer. These features are crucial for training models to accurately interpret and categorize feedback.The Dataset is loaded from kaggle website(Link to the dataset).

Data Preparation and Processing

By analyzing the target column which is 'sentiment', to understand the distribution of the target variable.

Sentiment	percentage
Positive	54.8 %
Neutral	36.35 %
Negative	8.85%

Data pre-processing

As the target column comes with imbalanced class distribution. It is advisable to re-sample the dataset such that the classes in the target column are equally balanced.

Train-Test Set

It is not an idea to rely on a train test split. There's a chance that all the positive samples could be grouped in the training set making it impossible to assess predictions accurately. That's why it's advisable to go for a train test split method. This approach guarantees that targets are evenly distributed between the training and test sets. So it's wise to go with a train test split, which ensures a balanced distribution of targets.Performed a train-test split of 75-25 % on given data. The size of the train test-split is as follows:

Train	3375
Test	1125

Feature transformation

Tokenizer converts the text into sequences of integers with each integer representing a word. It focuses on the words by setting a limit, on max features. This simplifies the dataset making it easier to analyze. Then the pad sequences function ensures that all sequences have the length by padding ones with zeros. This step is crucial for training networks that need input sizes.

Furthermore this process involves converting labels into a matrix using one hot encoding, which prepares the dataset for deep learning models. The complete pre-processing pipeline. Including tokenization, padding and encoding. Is vital for managing text data. It enables the creation of advanced models for tasks, like sentiment analysis.

Modeling

To assess the effectiveness of neural network designs, for text categorization we will develop three models distinguished by their core recurrent layers; one featuring a standard vanilla RNN, another utilizing an LSTM and a third employing a GRU. This method facilitates a comparison of how each design processes and learns from sequences of data.

The basic RNN model acts as a reference point handling sequences with a structure that captures relationships but struggles with retaining information over long periods due to gradient disappearance. The LSTM model overcomes these challenges with a structure incorporating memory cells and gates (input, forget and output) effectively managing dependencies by storing or discarding data as necessary. Conversely the GRU model offers a approach compared to LSTMs by merging input and forget gate features into one mechanism and simplifying the internal structure without compromising performance significantly.

Each model will be developed to ensure consistency in recurrent elements like the text representation embedding layer and output layer setup for fair evaluation. The primary objective is to assess each designs efficacy in handling data intricacies offering insights into their suitability, for text classification tasks.

To compare all three, we simply alter a single layer of a neural network while leaving everything else constant.

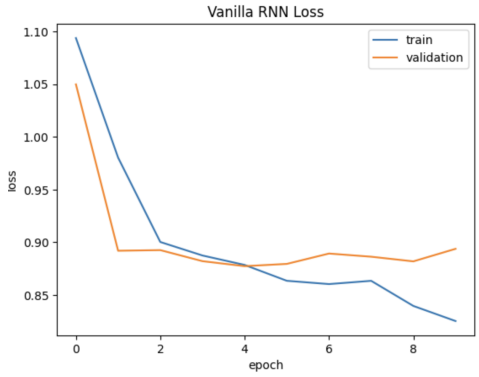
Some of the constant parameters used during training are:

- loss: cross-entropy loss
- batch size: 256
- epochs: 10

1) Model 1 - Vanilla RNN

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, 200, 32)           6400

 simple_rnn_2 (SimpleRNN)    (None, 64)                6208

 dense_12 (Dense)            (None, 64)                4160

 dropout_8 (Dropout)         (None, 64)                0

 dense_13 (Dense)            (None, 64)                4160

 dropout_9 (Dropout)         (None, 64)                0

 dense_14 (Dense)            (None, 3)                 195

=================================================================
Total params: 21123 (82.51 KB)
Trainable params: 21123 (82.51 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

The loss/learning curve is as follows:



The test accuracy and confusion curve is as follows:

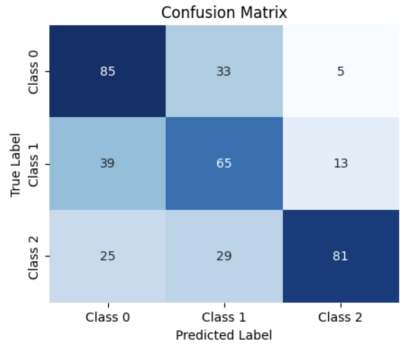
```
Vanilla RNN Model Evaluation:
36/36 [==============================] - 1s 24ms/step - loss: 0.8490 - accuracy: 0.6222
Test Accuracy: 0.6222222447395325
36/36 [==============================] - 1s 25ms/step
Confusion Matrix:
 [[267 109  26]
 [120 210  38]
 [ 58  74 223]]
```

Another interesting exercise was performed to evaluate model performance based on the length of test sentences, because rnns can consume a variable number of tokens/words and gradients

propagate correspondingly. To do this, the test set was broken into three subsets by sorting the data by length and dividing it into three equal portions, which were then tested on the previously trained model.

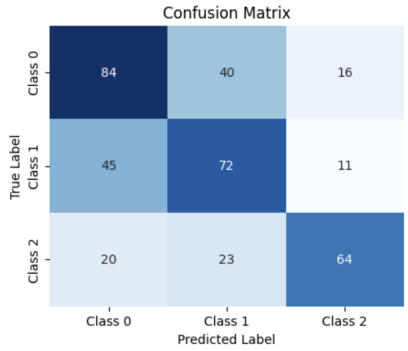
The accuracy and confusion matrix for RNN short is as follows:

```
Vanilla RNN - Short Subset Accuracy: 61.60%
12/12 [==============================] - 0s 21ms/step
[[85 33  5]
 [39 65 13]
 [25 29 81]]
Vanilla RNN - Short Subset Confusion Matrix:
```



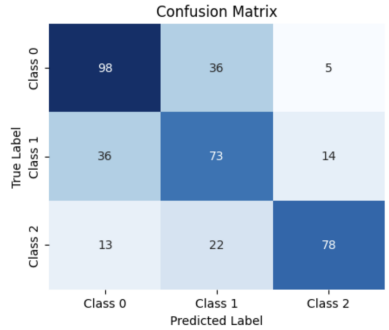
The accuracy and confusion matrix for RNN medium is as follows

```
Vanilla RNN - Medium Subset Accuracy: 58.67%
12/12 [==============================] - 0s 18ms/step
[[84 40 16]
 [45 72 11]
 [20 23 64]]
Vanilla RNN - Medium Subset Confusion Matrix:
```



The accuracy and confusion matrix for RNN long is as follows:

```
Vanilla RNN - Long Subset Accuracy: 66.40%
12/12 [==============================] - 0s 17ms/step
[[98 36  5]
 [36 73 14]
 [13 22 78]]
Vanilla RNN - Long Subset Confusion Matrix:
```



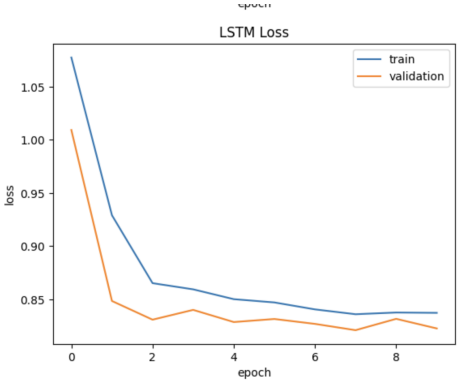
2) Model 2 - LSTM

```
Model: "sequential_3"
```

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, 200, 32)	6400
lstm_1 (LSTM)	(None, 64)	24832
dense_9 (Dense)	(None, 64)	4160
dropout_6 (Dropout)	(None, 64)	0
dense_10 (Dense)	(None, 64)	4160
dropout_7 (Dropout)	(None, 64)	0
dense_11 (Dense)	(None, 3)	195

```
Total params: 39747 (155.26 KB)
Trainable params: 39747 (155.26 KB)
Non-trainable params: 0 (0.00 Byte)
```

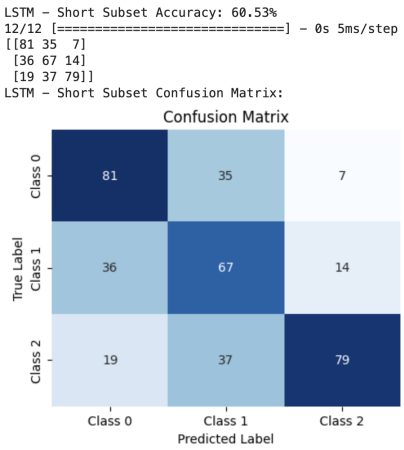
The loss/learning curve is as follows:



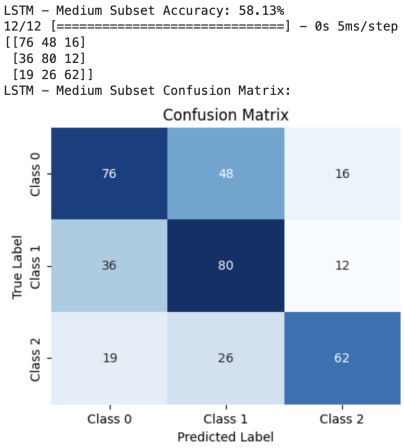
The test accuracy and confusion matrix is as follows:

```
LSTM Model Evaluation:
36/36 [==============================] - 0s 8ms/step - loss: 0.8363 - accuracy: 0.6142
Test Accuracy: 0.6142222285270691
36/36 [==============================] - 1s 5ms/step
Confusion Matrix:
 [[244 127  31]
 [104 231  33]
 [ 47  92 216]]
```

The accuracy and confusion matrix for LSTM short is as follows:

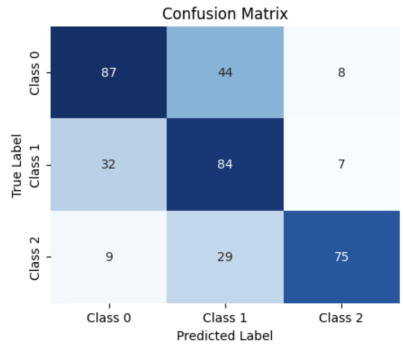


The accuracy and confusion matrix for LSTM medium is as follows:



The accuracy and confusion matrix for LSTM long is as follows:

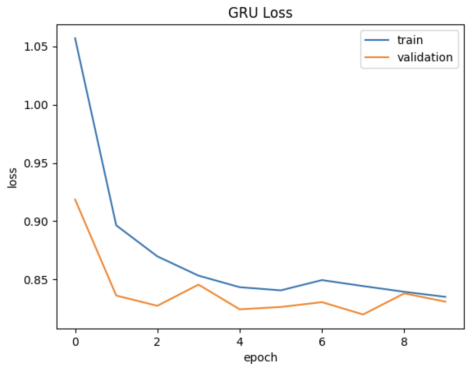
```
LSTM - Long Subset Accuracy: 65.60%
12/12 [==============================] - 0s 5ms/step
[[87 44  8]
 [32 84  7]
 [ 9 29 75]]
LSTM - Long Subset Confusion Matrix:
```



3) Model 3 - GRU

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (None, 200, 32)           6400

 gru_1 (GRU)                 (None, 64)                18816

 dense_15 (Dense)            (None, 64)                4160

 dropout_10 (Dropout)        (None, 64)                0

 dense_16 (Dense)            (None, 64)                4160

 dropout_11 (Dropout)        (None, 64)                0

 dense_17 (Dense)            (None, 3)                 195

=================================================================
Total params: 33731 (131.76 KB)
Trainable params: 33731 (131.76 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

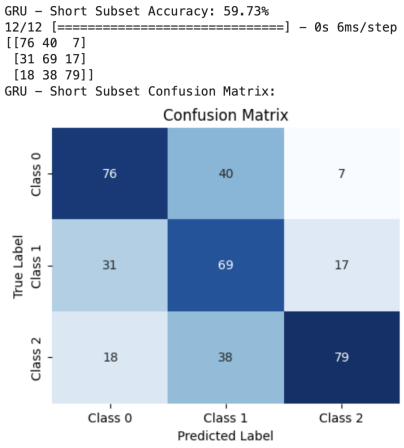
The loss curve is as follows:



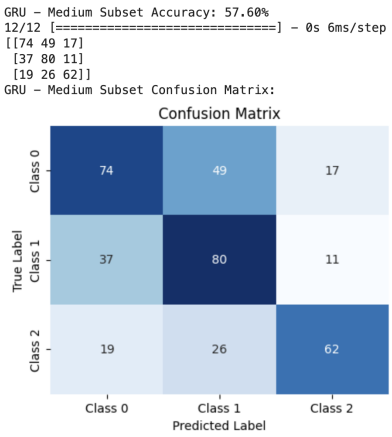
The test accuracy and confusion matrix is as follows:

```
GRU Model Evaluation:
36/36 [==============================] - 0s 6ms/step - loss: 0.8415 - accuracy: 0.6098
Test Accuracy: 0.6097777485847473
36/36 [==============================] - 0s 4ms/step
Confusion Matrix:
 [[236 134  32]
 [ 98 234  36]
 [ 47  92 216]]
```

The accuracy and confusion matrix for GRU short is as follows:

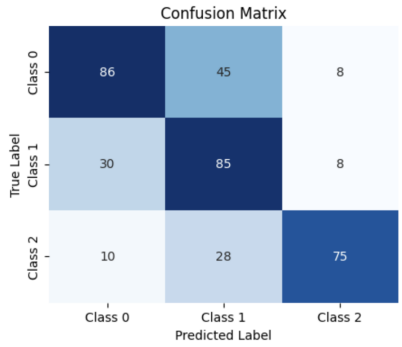


The accuracy and confusion matrix for GRU medium is as follows:



The accuracy and confusion matrix for GRU long is as follows:

```
GRU - Long Subset Accuracy: 65.60%
12/12 [==============================] - 0s 6ms/step
[[86 45  8]
 [30 85  8]
 [10 28 75]]
GRU - Long Subset Confusion Matrix:
```



Observations/Patterns

Comparsion of all the test sentence length is as follows:

Model	Test sentence length	accuracy
RNN	short	61.6
RNN	medium	58.67
RNN	long	66.4
LSTM	short	60.53
LSTM	medium	58.13
LSTM	long	65.6
GRU	short	59.73
GRU	medium	57.6
GRU	long	65.6

If we compare the time taken for training the models, in general, we can see that the Adam optimizer takes a little longer when compared to the Rmsprop optimizer, and regularization techniques like batch normalization and l1- norm take more time than without any regularization and drop out, because extra computation is happening there. However, the difference is minimal and can be overlooked for better performance in terms of accuracy.

While comparing accuracies, a model with batch normalization has the highest accuracy for both Adam and rmsprop optimizers, but we can see higher fluctuations in the loss curve and accuracy curve when compared to other models.

The loss for l1-norm is in general higher when compared to other models since we explicitly add penality terms of modulus of learnable parameters.

Q2

Introduction

The Amazon Reviews collection was created to advance text classification through deep learning algorithms. With the rise of e commerce, there has been an increase in customer evaluations, which provide significant insights into consumer attitudes toward products. These evaluations include a wide spectrum of sentiments, from praise to negative feedback, and provide a variety of perspectives that can be used to improve product suggestions, increase customer satisfaction, and refine search engines.The Dataset is loaded from kaggle website(Link to the dataset).

Train-Test Set

It is not advisable to rely on a train test split. There is a potential that all of the positive samples will be pooled in the training set, making it impossible to reliably evaluate predictions. That is why it is recommended to use the train test split method. This method ensures that targets are spread evenly across the training and testing sets. So it's best to go with a train test split, which ensures a fair distribution of goals.Ran a 75-25 percent train-test split on the provided data. The train test-split size is as follows:

Train	3375
Test	1125

Data Processing

As the target column comes with imbalanced class distribution. It is advisable to re-sample the dataset such that the classes in the target column are equally balanced.

Modeling

We use two types of GloVe embeddings, GloVe 50D and GloVe 100D, as dense representations in a neural network architecture trained using the same hyperparameters and optimization method. We begin by loading pre-trained word vectors into memory to prepare the embeddings, as per a defined approach. Then we create a neural network architecture with an embedding layer that converts input words into dense vector representations based on the pre-trained embeddings. Both models are trained using consistent hyperparameters and optimization procedures, ensuring a fair comparison. Post-training evaluation involves evaluating model performance on a separate test dataset, generally using metrics such as accuracy.

1) Model 1 - Word Embedding with Glove 50 is as follows:

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_12 (Embedding)    (None, None, 50)          245950

 simple_rnn_3 (SimpleRNN)    (None, 64)                7360

 dense_33 (Dense)            (None, 64)                4160

 dropout_21 (Dropout)        (None, 64)                0

 dense_34 (Dense)            (None, 3)                 195

=================================================================
Total params: 257665 (1006.50 KB)
Trainable params: 11715 (45.76 KB)
Non-trainable params: 245950 (960.74 KB)
_________________________________________________________________
```

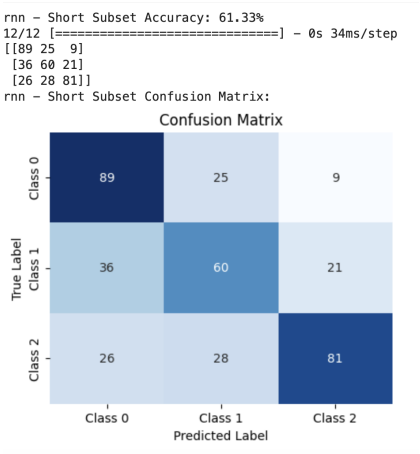
The accuracy as follows:

```
36/36 [==============================] - 1s 32ms/step
Accuracy for model_50 after fitting: 0.6106666666666667
```

The confusion matrix is as follows:

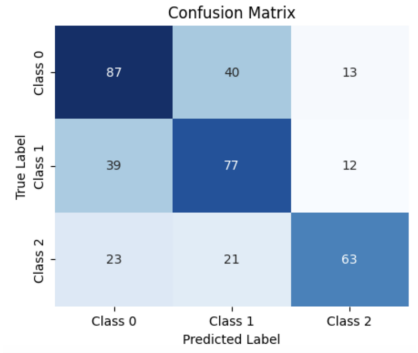
```
Confusion Matrix (50D):
[[323  60  19]
 [191 153  24]
 [107  47 201]]
```

The accuracy and confusion matrix for RNN short with GLove 50 is as follows:



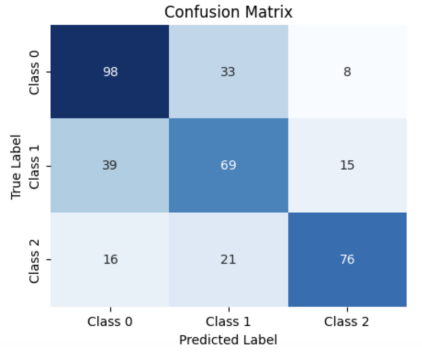
The accuracy and confusion matrix for RNN medium with Glove 50 is as follows

```
rnn – Medium Subset Accuracy: 60.53%
12/12 [==============================] – 0s 23ms/step
[[87 40 13]
 [39 77 12]
 [23 21 63]]
rnn – Medium Subset Confusion Matrix:
```



The accuracy and confusion matrix for RNN long with Glove 50 is as follows:

```
rnn – Long Subset Accuracy: 64.80%
12/12 [==============================] – 0s 17ms/step
[[98 33  8]
 [39 69 15]
 [16 21 76]]
rnn – Long Subset Confusion Matrix:
```



2) Model 2 - Word Embedding with Glove 100 is as follows:

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_13 (Embedding)    (None, None, 100)         491900

 simple_rnn_4 (SimpleRNN)    (None, 64)                10560

 dense_35 (Dense)            (None, 64)                4160

 dropout_22 (Dropout)        (None, 64)                0

 dense_36 (Dense)            (None, 3)                 195

=================================================================
Total params: 506815 (1.93 MB)
Trainable params: 14915 (58.26 KB)
Non-trainable params: 491900 (1.88 MB)
_________________________________________________________________
```

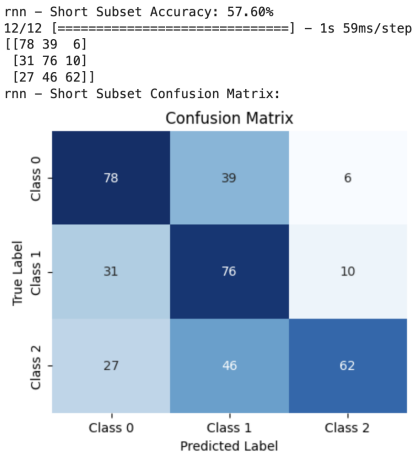
The accuracy as follows:

```
36/36 [==============================] - 1s 32ms/step
Accuracy for model_100 after fitting: 0.4951111111111111
```

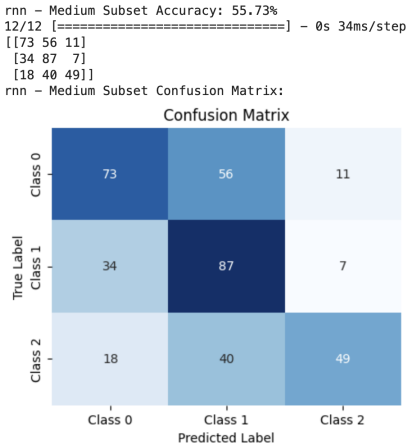
The confusion matrix is as follows:

```
Confusion Matrix (100D):
[[345  48   9]
 [269  89  10]
 [196  36 123]]
```

The accuracy and confusion matrix for RNN short with GLove 100 is as follows:

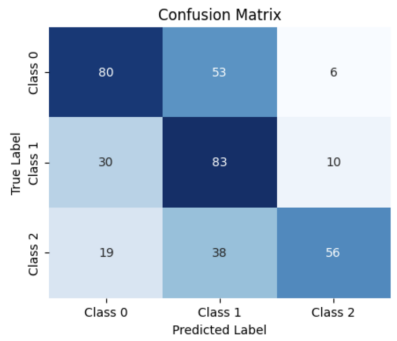


The accuracy and confusion matrix for RNN medium with Glove 100 is as follows



The accuracy and confusion matrix for RNN long with Glove 100 is as follows:

```
rnn – Long Subset Accuracy: 58.40%
12/12 [==============================] – 0s 33ms/step
[[80 53  6]
 [30 83 10]
 [19 38 56]]
rnn – Long Subset Confusion Matrix:
```



Observations/Patterns

According to my understanding, for this dataset the pre-trained models weren't a huge plus, because we are seeing lower accuracies of around 10-20 %, this might be because the current dataset is very different from the dataset the pre-trained model was trained on. However, the pre-trained model might have helped in extracting relevant features in the initial epochs itself, and had there not been a pre-trained model, initial epoch accuracies might have been much lower.