

Assignment 2

CZ4042 Neural Network & Deep Learning

U1721316F

12th Nov 2020

Table of Contents

GPU	3
Google Colab	3
Kaggle	3
Part A Introduction	4
Objective.....	4
Model Architecture	5
Experiments and Results	5
Question 1	5
Question 2	8
Question 3	9
a. Momentum.....	9
b. RMSProp Optimizer	10
c. Adam Optimizer	10
d. 2 Dropout Layers	11
e. 1 Dropout Layer (Additional)	12
Discussion of Results	13
Question 4	13
Part 1	13
Part 2	13
Part 3	13
Improving Performance (Additional).....	16
Conclusion	17
Part B: Introduction	18
Objective.....	18
Experiments and Results	18
Question 1	18
Question 2	19
Question 3	20
Question 4	21
Question 5	22
Question 6	23
Character RNN	24
Word RNN.....	25
Conclusion	26
Appendix.....	27

GPU

Google Colab

The free Google Colab GPU was used to carry out the experiments in this assignment. Upon carrying out the experiments, I discovered that the results are not reproducible due to the inherent randomness accumulated by [GPU parallelism](#) in Google Colab. This is unavoidable even after fixing the TensorFlow and NumPy random seed.



👁 Name (5 visualized)	Created ▾	Runtime	End Time	best_val_loss	loss	val_accuracy	val_loss
👁  Q1a V5	6h ago	2m 15s	2020-11-12T09:40:40.000Z	1.032	0.2476	0.6643	1.319
👁  Q1a V4	6h ago	2m 16s	2020-11-12T09:35:33.000Z	1.026	0.2524	0.67	1.281
👁  Q1a V3	7h ago	5m 32s	2020-11-12T08:11:48.000Z	1.025	0.2711	0.6557	1.318
👁  Q1a V2	8h ago	3m 29s	2020-11-12T08:06:00.000Z	1.035	0.2548	0.6586	1.338
👁  Sanity Check Q1a	8h ago	2m 36s	2020-11-12T08:02:04.000Z	1.038	0.2581	0.6571	1.365

Figure 1: Part B Q1a results over 5 runs

As such, some of the results may not actively reflect the best accuracy but avoids the worst performing parameters. To avoid this issue, when the test accuracy is marginally close, I decided to repeat the experiment and take the average of 3 runs.

Kaggle

The Kaggle GPU was also used for training the full dataset for CIFAR-10 as it does not require the user to upload the dataset in the set directory unlike Google Colab.

Part A: Object Recognition

Part A Introduction

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10,000 images. The test batch contains exactly 1000 randomly selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. The CIFAR-10 dataset consists of 60,000 32x32 colour images in 10 classes, with 6000 images per class. There are 50,000 training images and 10000 test images.

For the purpose of this assignment, the training sample has been reduced to 10000 training samples and 2000 test samples.

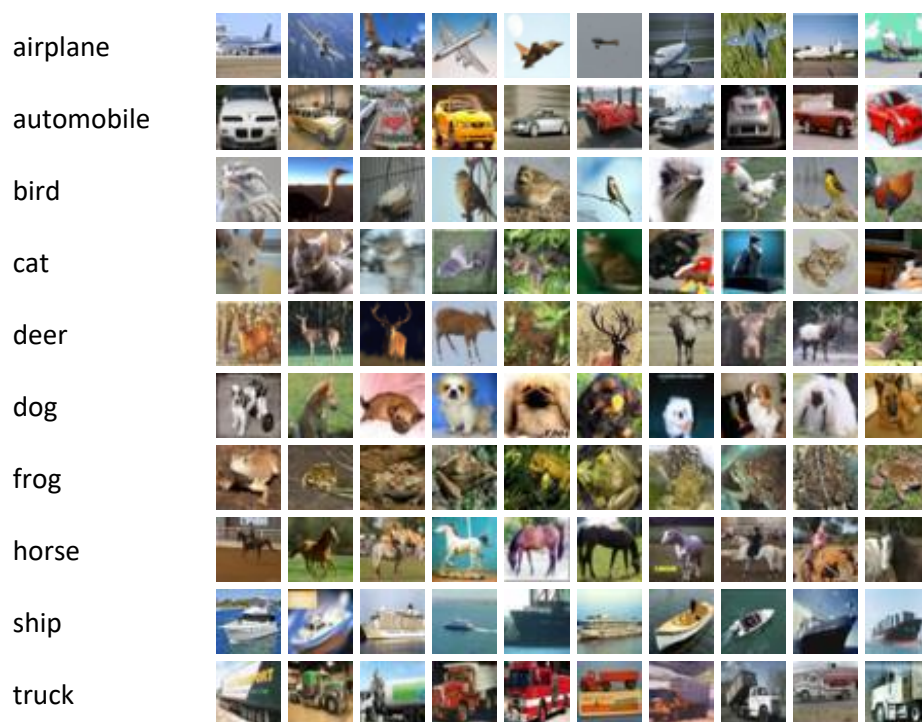


Figure 1: Classes in the CIFAR-10 dataset

Objective

The purpose of this project is to perform image recognition to predict the label of the image.

Model Architecture

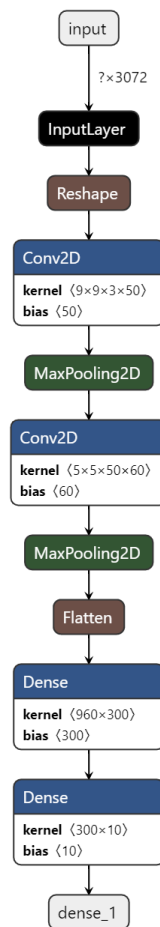


Figure 2: Baseline Model Architecture

The above image shows the baseline model architecture derived from the question requirements.

Experiments and Results

Question 1

1. Train the network using mini-batch gradient descent learning for 1000 epochs. Set the batch size to 128, and learning rate $\alpha = 0.001$.

1a. Plot the (1) training cost, (2) test cost, (3) training accuracy, and (4) test accuracy against learning epochs. One plot for the costs and one plot for the accuracies.

Source Code: Part_A_Object_Recognition -> obj_recognition_Q1a

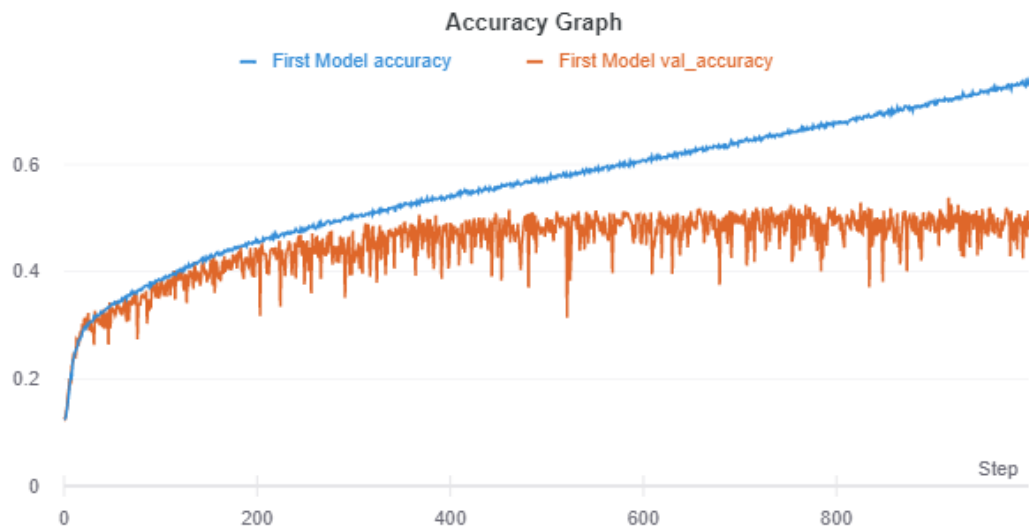


Figure 3: Accuracy against epochs for baseline model



Figure 4: Loss against epochs for baseline model

1b. For the first two test images, plot the feature maps at both convolution layers ($C1$ and $C2$) and pooling layers ($S1$ and $S2$) along with the test images. (In total one image and four feature maps)

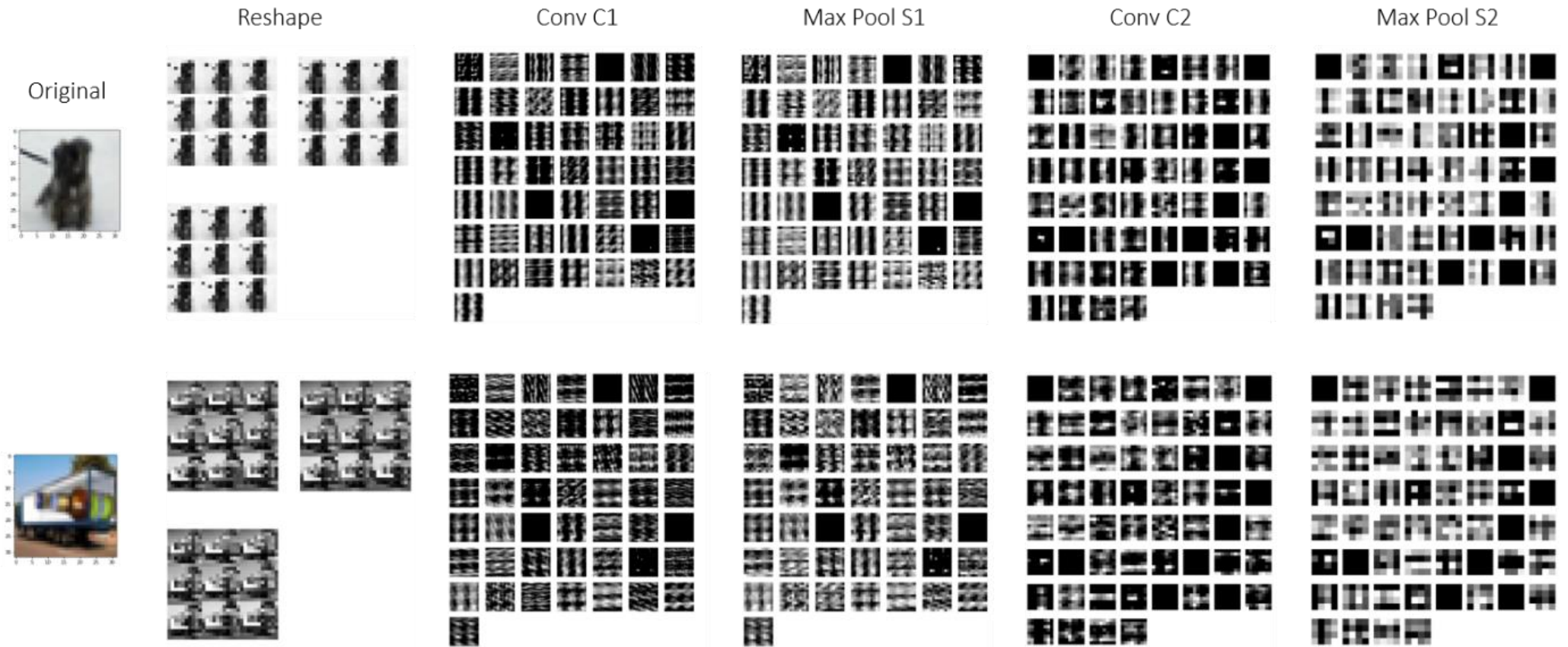


Figure 5: Feature Map for each layer

Full Images: Part_A_Object_Recognition -> Feature Map Viz -> Feature Maps

Source Code: Part_A_Object_Recognition -> Feature Map Viz -> obj_recognition_viz.ipynb

The original image can be seen on the left. The first image is an image of a dog. The second image is an image of a truck. The 32 by 32 image is first resized to 32,32,3. It is then passed into a convolution layers and max pooling layers. As we can observe from Figure 4, there is a reduction in the number of features as the batch is passed from 1 layer to another.

Question 2

2. Use a grid search ($C1=\{10,30,50,70,90\}$, $C2 = \{20,40,60,80,100\}$, in total 25 combinations) to find the optimal combination of the numbers of channels at the convolution layers. Use the test accuracy to determine the optimal combination. Report all 25 accuracies.

File: Part_A_Object_Recognition -> obj_recognition_Q2.ipynb




















































num_ch_c1	num_ch_c2	<input type="checkbox"/>  Name (25 visualized)	Runtime	accuracy	best_epoch	best_val	loss	val_loss	val_accuracy ▼
90	40	  Conv (90, 40)	10m 45s	0.7429	716	1.384	0.7773	1.506	0.504
70	40	  Conv (70, 40)	9m 43s	0.734	705	1.379	0.802	1.505	0.497
30	60	  Conv (30, 60)	7m 12s	0.7255	732	1.405	0.8074	1.583	0.49
90	20	  Conv (90, 20)	10m 9s	0.6963	753	1.399	0.8976	1.533	0.488
50	60	  Conv (50, 60)	8m 47s	0.7566	724	1.404	0.75	1.626	0.484
10	40	  Conv (10, 40)	7m 5s	0.6319	753	1.431	1.062	1.515	0.483
50	20	  Conv (50, 20)	7m 57s	0.6321	833	1.436	1.066	1.499	0.4825
50	80	  Conv (50, 80)	9m 21s	0.7719	717	1.381	0.7112	1.665	0.482
30	20	  Conv (30, 20)	7m 25s	0.6276	825	1.411	1.057	1.526	0.4805
30	100	  Conv (30, 100)	8m 5s	0.7574	732	1.41	0.7398	1.663	0.476
10	80	  Conv (10, 80)	7m 25s	0.6983	647	1.451	0.887	1.591	0.475
50	40	  Conv (50, 40)	8m 22s	0.7185	717	1.407	0.8331	1.608	0.471
90	60	  Conv (90, 60)	11m 5s	0.7949	596	1.402	0.6608	1.726	0.4675
70	80	  Conv (70, 80)	10m 40s	0.7807	717	1.391	0.6923	1.714	0.463
90	100	  Conv (90, 100)	11m 58s	0.8267	717	1.378	0.5837	1.752	0.463
10	20	  Conv (10, 20)	6m 57s	0.5898	716	1.478	1.174	1.581	0.459
10	60	  Conv (10, 60)	7m 9s	0.6928	747	1.434	0.9009	1.682	0.4575
70	60	  Conv (70, 60)	10m 7s	0.7701	649	1.398	0.7197	1.748	0.455
50	100	  Conv (50, 100)	9m 40s	0.777	717	1.391	0.6941	1.71	0.4545
70	20	  Conv (70, 20)	9m 16s	0.6545	916	1.408	1.01	1.633	0.449
90	80	  Conv (90, 80)	11m 32s	0.8081	747	1.38	0.6256	1.848	0.448
70	100	  Conv (70, 100)	11m 1s	0.8131	649	1.386	0.6177	1.835	0.446
30	80	  Conv (30, 80)	7m 37s	0.7583	649	1.392	0.7488	1.827	0.44
30	40	  Conv (30, 40)	7m 44s	0.6874	724	1.429	0.919	1.758	0.4365
10	100	  Conv (10, 100)	7m 20s	0.7381	717	1.425	0.8025	1.777	0.436

Figure 6: Grid search results sorted by test accuracy in descending order

Question 3

3. Using the optimal combination found in part (2), train the network by:

- Adding the momentum term with momentum $\gamma = 0.1$,
- Using RMSProp algorithm for learning,
- Using Adam optimizer for learning,
- Adding dropout (probability=0.5) to the two fully connected layers.

Plot the costs and accuracies against epochs (as in question 1(a)) for each case. Note that the sub-questions are independent. For instance, in (d), you do not need to modify the optimizer.

Source Code: Part_A_Object_Recognition -> obj_recognition_Q3n.ipynb

a. Momentum

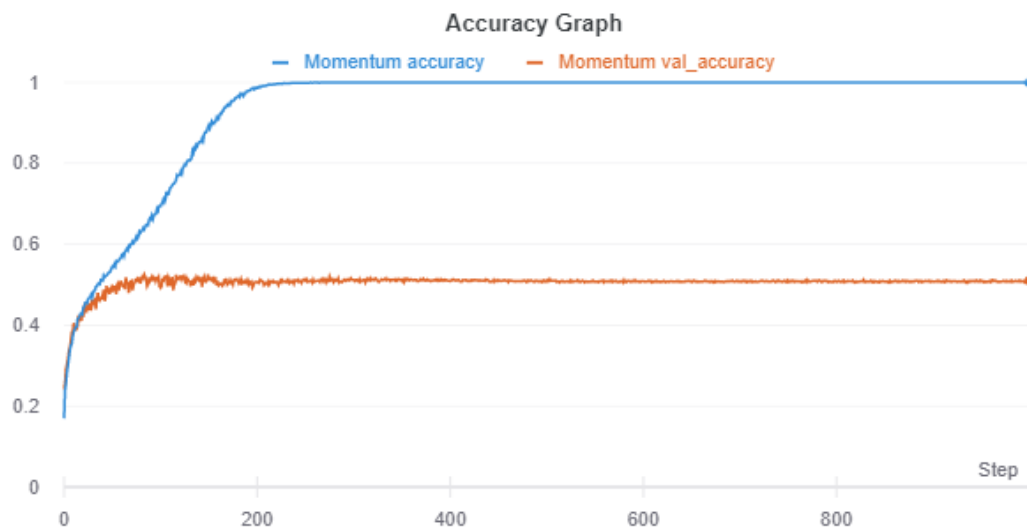


Figure 8: Accuracy against epochs with momentum added

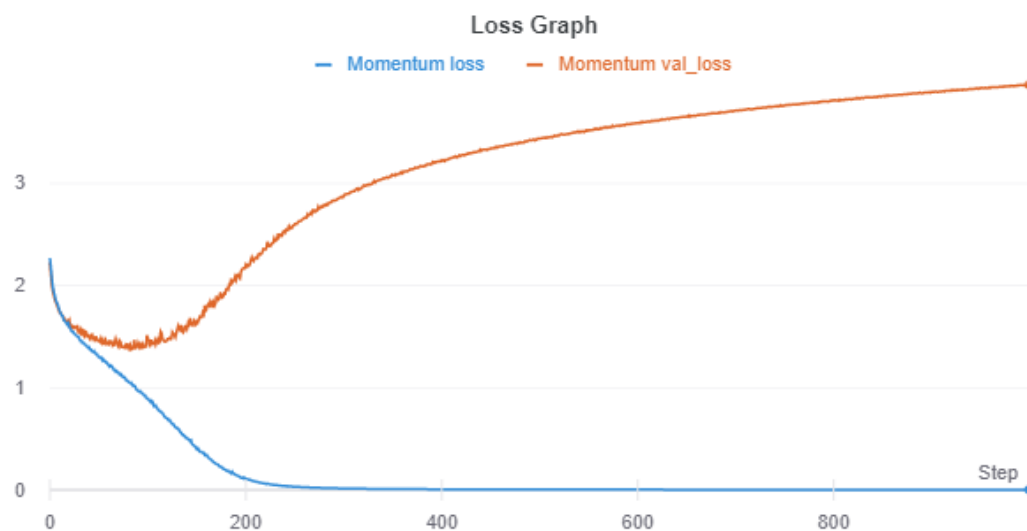


Figure 9: Loss against epochs with momentum added

b. RMSProp Optimizer

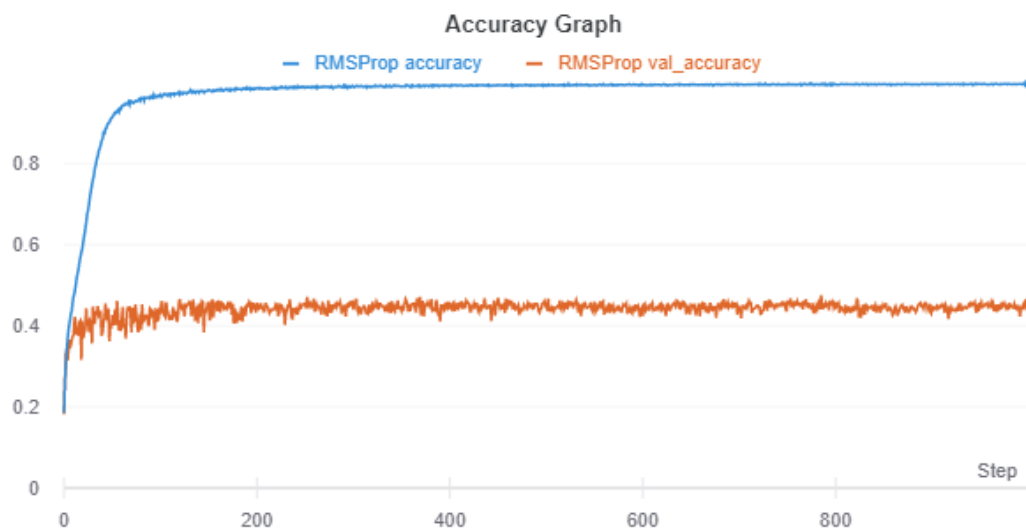


Figure 10: Accuracy against epochs with RMSProp optimizer used

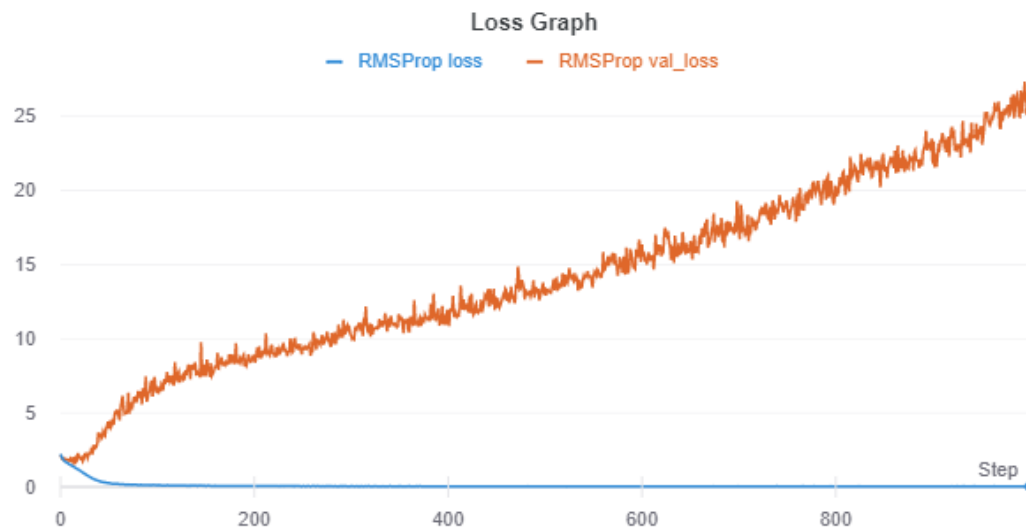


Figure 11: Loss against epochs with RMSProp optimizer used

c. Adam Optimizer

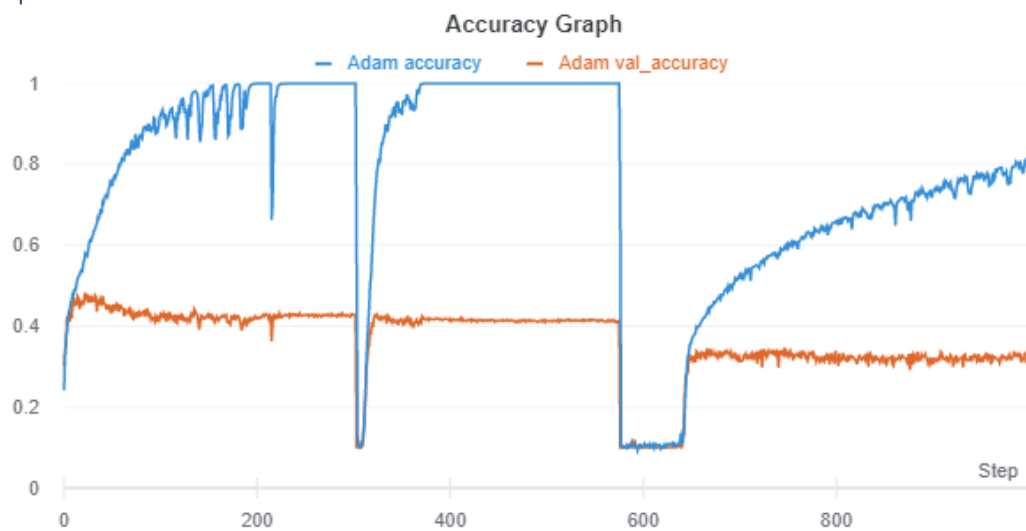


Figure 12: Accuracy against epochs with Adam optimizer used



Figure 13: Loss against epochs with Adam optimizer used

d. 2 Dropout Layers

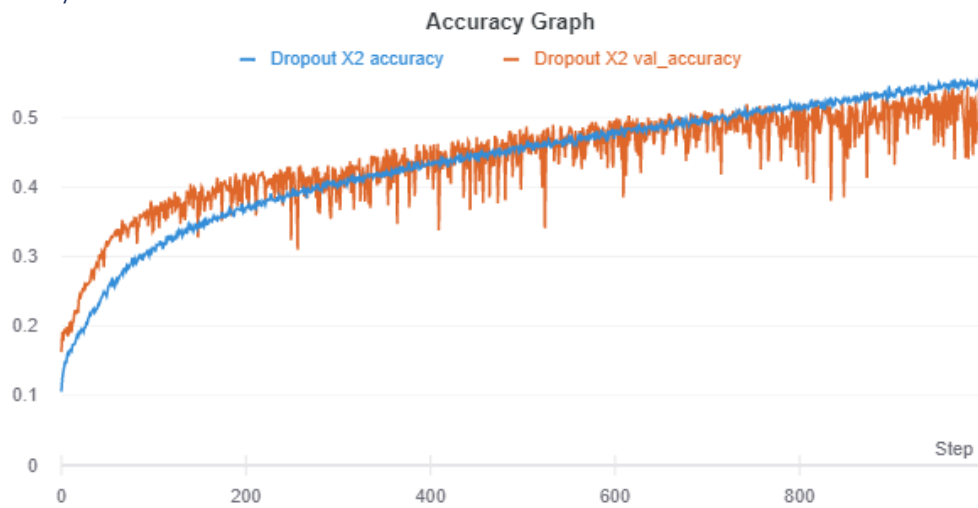


Figure 14: Accuracy against epochs with 2 dropout layers added (0.5)

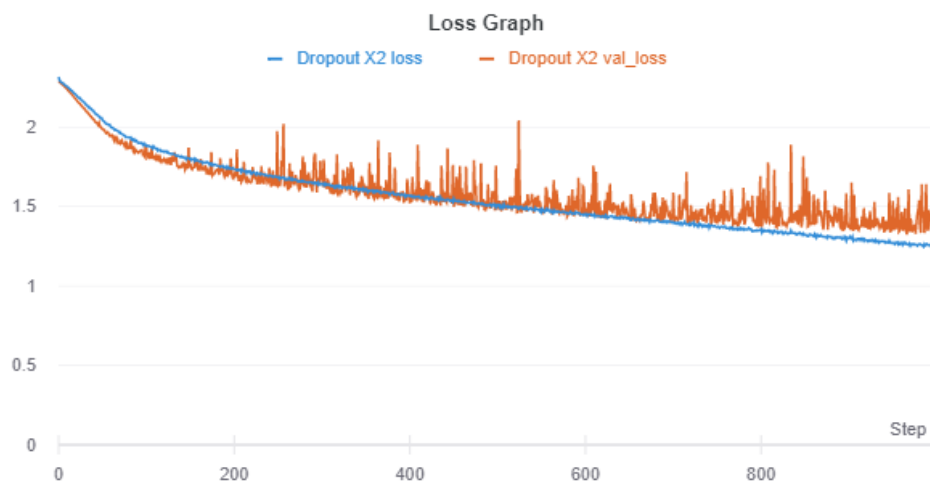


Figure 15: Loss against epochs with 2 dropout layers added (0.5)

e. 1 Dropout Layer (Additional)

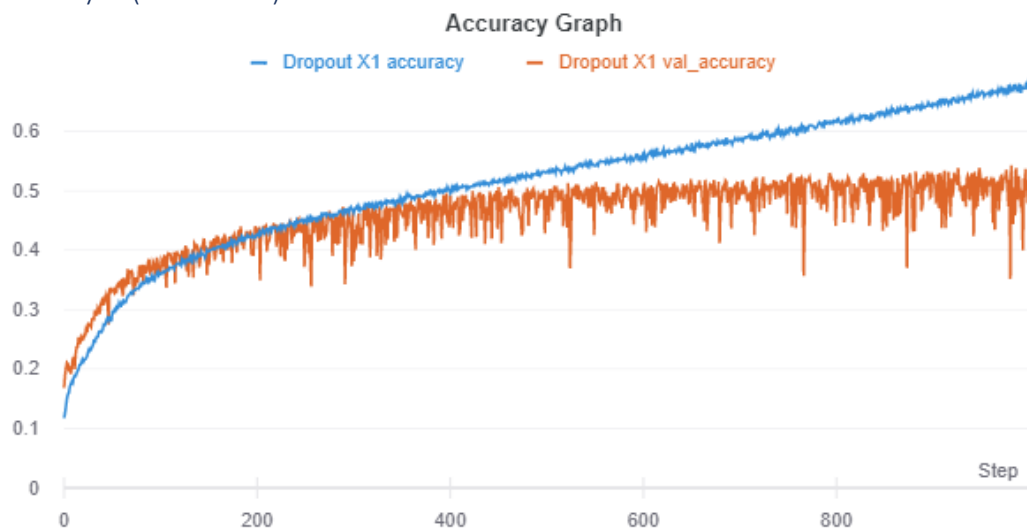


Figure 16: Accuracy against epochs with 1 dropout layer added (0.5)

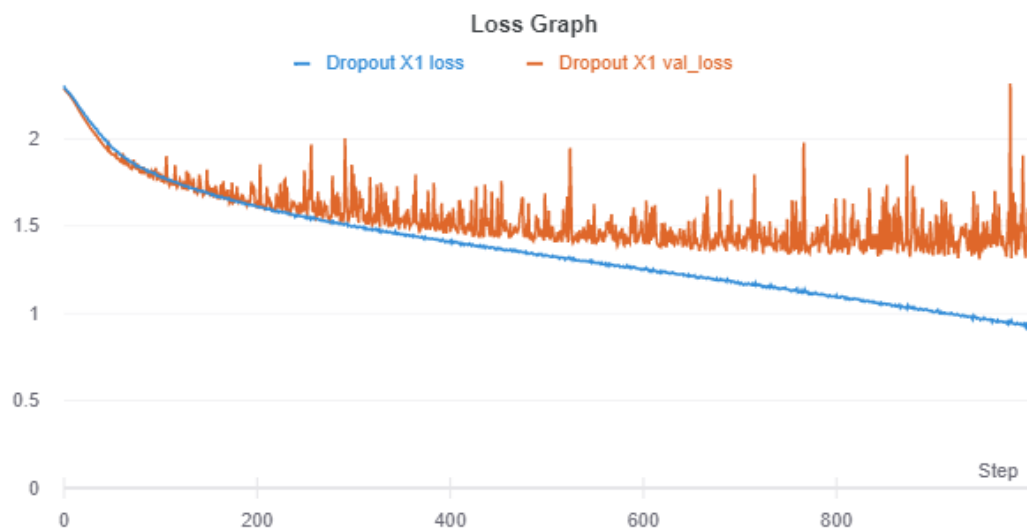


Figure 17: Loss against epochs with 1 dropout layer added (0.5)

Discussion of Results

Question 4

4. Compare the accuracies of all the models from parts (1) - (3) and discuss their performances.

Part 1

As we can observe from Figure 2 and 3, the accuracy of the model is not high. The test accuracy fluctuates a lot and starts to plateau after 500 epochs. There is also no significant drop in test loss as training loss decreases. This forms the baseline model.

Part 2

In Part 2, we carry out a grid search to determine the optimal combination of number of channels at the convolution layers.

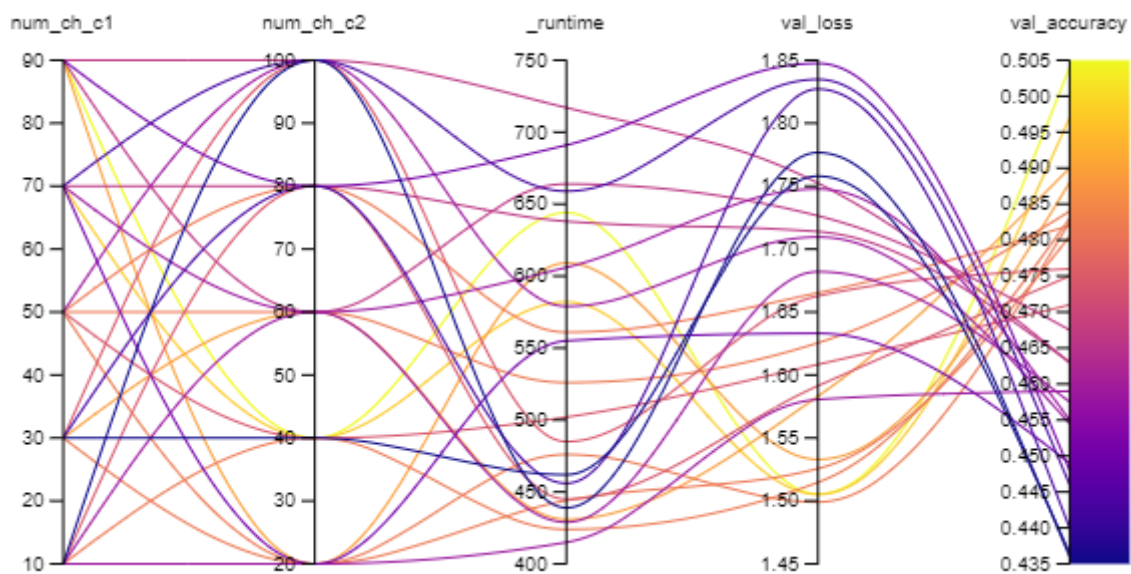


Figure 18: Parallel Coordinates graph showing hyperparameter sweep results

A hyperparameter sweep is carried out using Weights and Biases to determine the optimal combination. The best combination is a convolution channel size of 90 and 40 for the first and second layers as observed in Figure 6. However, this may not be the case as the range of values are extremely close to each other (0.436 to 0.504). As seen in Figure 1, the randomness inflicted may skew the results by up to 0.01. Additionally, a different random seed may result in a different optimal channel size. To correctly determine the best number of channels may involve running the sweep multiple times and getting the best number of channels using the average.

[Link to all 25 runs](#)

Part 3

Using the optimal hyperparameters derived from Part 2, we test the network by adding momentum, dropouts and changing the optimizer.

Momentum

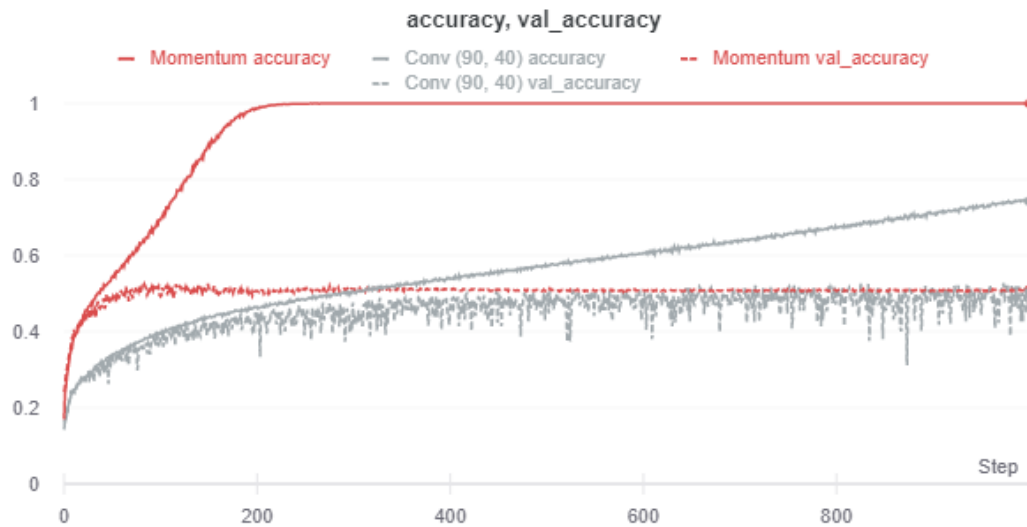


Figure 19: Comparing accuracies between adding momentum and without momentum

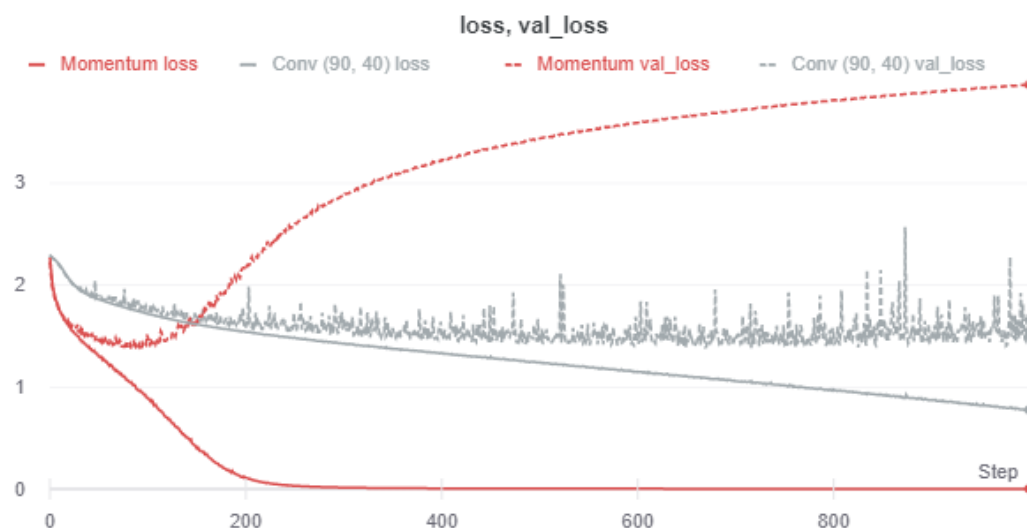


Figure 20: Comparing losses between momentum and SGD

The addition of momentum has 3 effects.

1. It causes the network to converge faster as seen in Figure 19 and 20
2. Reduced oscillation. The noise has decreased as a result of adding a momentum value of 0.9. Fluctuations in test loss and accuracy has been minimized.
3. This inadvertently causes the test loss to increase, overfitting the model.

Despite so, there is a minor improvement in the test accuracy from 0.504 to 0.51.

Optimizer Comparison (Adam, SGD, RMSProp)

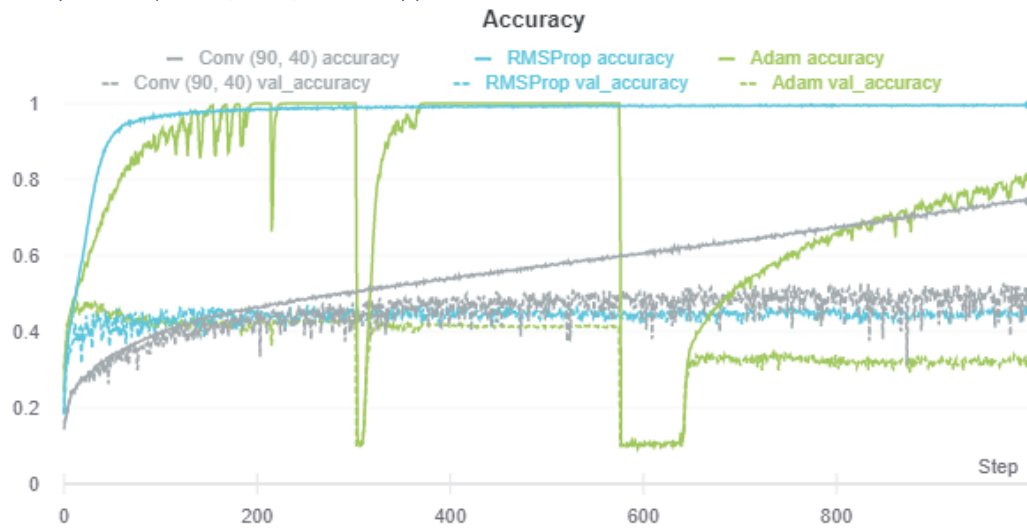


Figure 21: Comparing accuracies between the 3 optimizers

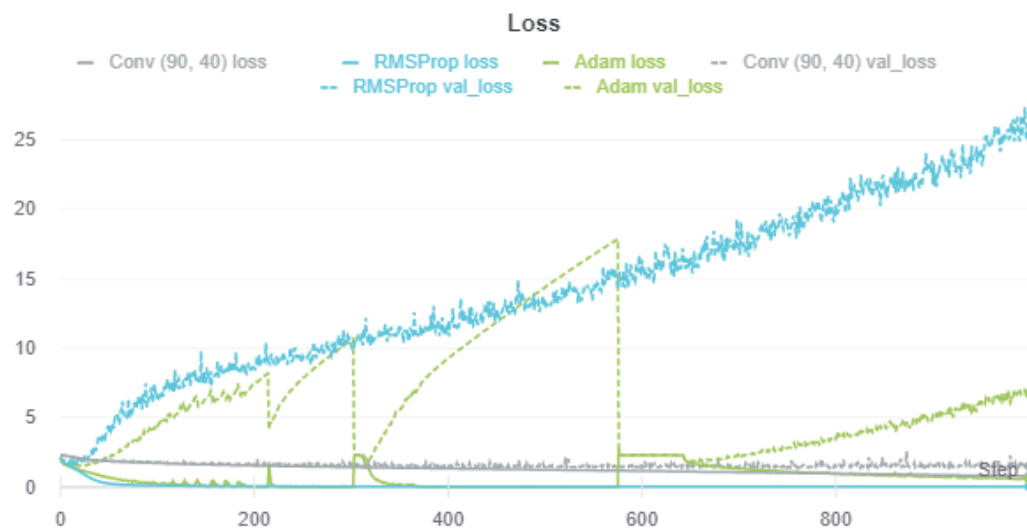


Figure 22: Comparing losses between the 3 optimizers

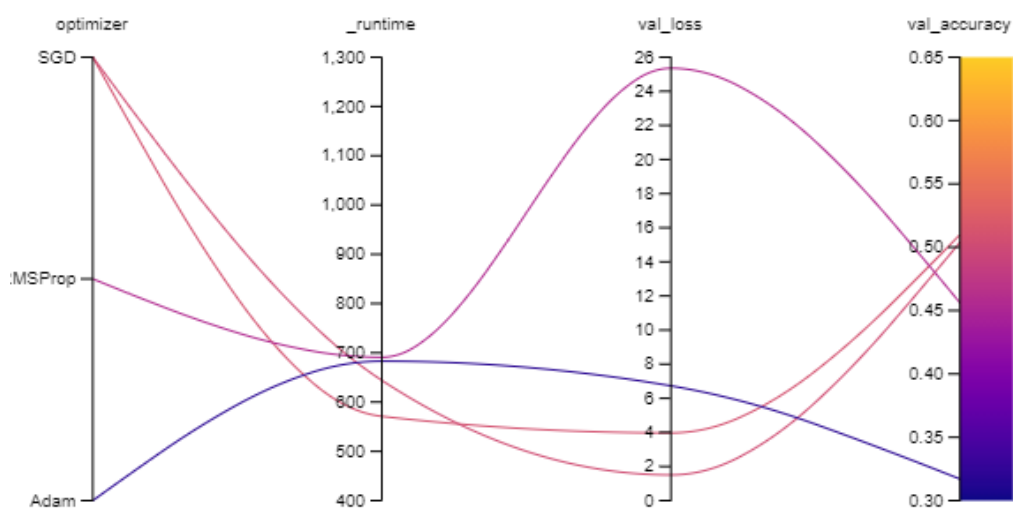


Figure 23: Parallel Coordinates Graph for test accuracy and loss comparison

Training a neural network can be slow. By choosing an appropriate optimizer, we are able to increase the speed of training a model

The RMSProp optimizer converges at a faster rate than the SGD optimizer but at the cost of a reduced performance in accuracy and significant overfitting.

The Adam optimizer is highly unstable as observed in Figure 12 and 13. Adam maintains a rolling geometric mean of recent gradients and squares of the gradients. The squares of the gradients are used to divide (another rolling mean of) the current gradient to decide the current step. However, when the gradient becomes and stays very close to zero, this will make the squares of the gradient become so low that they either have large rounding errors or are effectively zero, thus introducing instability.

The Momentum optimizer performs the best out of the 4 despite an increasing test loss.

Adding Dropout

The dropout rate is the probability of setting each input to the layer to zero. Referring to Figure 14, 15, 16 and 17, in this model, we add a dropout layer after the fully connected layers. With each progressive increase in no. of dropout layers, we can observe that overfitting tends to occur less, and the performance of the model improves.

Name (5 visualized)	accuracy	loss	val_loss	val_accuracy ▼	Runtime	Tags
● Dropout X2	0.5563	1.248	1.375	0.5205	11m 46s	Q3 Q4 additional
● Momentum	1	0.0009605	3.97	0.51	14m 20s	Q3 Q4 additional
● Conv (90, 40)	0.7429	0.7773	1.506	0.504	10m 45s	Q3
● RMSProp	0.9961	0.02006	25.35	0.456	12m 38s	Q3
● Adam	0.8075	0.553	6.728	0.317	12m 4s	Q3

Figure 24: Table of Results for Part 3

Improving Performance (Additional)

Taking the two best performing models (Momentum added, and 2 dropout layers added), we can build on the model to improve its performance. Beyond adding momentum and dropout to regularize the model, one way of improving the model performance is to add more training data. By adding more training data, the model is able to learn more characteristics related to the 10 classes. Some other ways of improving the performance may also include changing the model architecture or lowering the learning rate. However, such modifications are beyond the scope of this assignment. One interesting observation is that the oscillations were dramatically reduced once more training data was added. (Figure 25). The accuracy of the model was also greatly improved from 0.5205 to 0.636.



Figure 25: Test accuracies with full dataset with momentum or SGD optimizer

Name (5 visualized)	accuracy	loss	val_loss	val_accuracy	Runtime	Tags
● SGD Full Data	0.694	0.8637	1.041	0.636	53m 53s	additional
● MM Full Data	0.8256	0.493	1.353	0.6129	22m 33s	additional
● MM + Dropout X2	0.967	0.09937	2.35	0.542	13m 30s	additional
● Dropout X2	0.5563	1.248	1.375	0.5205	11m 46s	Q4 additional
● Momentum	1	0.0009605	3.97	0.51	14m 20s	Q4 additional

Figure 26: Table of Results for additional model improving

Source Code: Part_A_Object_Recognition -> Additional

Conclusion

In part A of the assignment, we first designed a simple CNN model. The feature maps were then plotted using the Keract library. A grid search was then carried out to determine the best number of channels that will ensure the best performance. Lastly, several experiments were carried out to reduce overfitting (by adding dropouts) and to determine the best optimizer. One challenge aspect of this part was ensuring that the results were not affected by the randomness of the results.

Part B: Text Classification

Part B: Introduction

The dataset used in this project contains the first paragraphs collected from Wikipage entries and the corresponding labels about their category.

The training and test datasets are read from 'train_medium.csv' and 'test_medium.csv' files. The training dataset contains 5600 entries and test dataset contains 700 entries. The label of an entry is one of the 15 categories such as people, company, schools, etc.

Objective

The purpose of this project is to perform text classification using CNN and RNN to predict the category of the line of text.

Experiments and Results

Question 1

1. Design a Character CNN Classifier that receives character ids and classifies the input. The CNN has two convolution and pooling layers:

- A convolution layer $C1$ of 10 filters of window size 20×256 , VALID padding, and ReLU neurons. A max pooling layer $S1$ with a pooling window of size 4×4 , with stride = 2, and padding = 'SAME'.
- A convolution layer $C2$ of 10 filters of window size 20×1 , VALID padding, and ReLU neurons. A max pooling layer $S2$ with a pooling window of size 4×4 , with stride = 2 and padding = 'SAME'.

Plot the entropy cost on the training data and the accuracy on the testing data against training epochs.

Source Code: Part_B_Text_Classification -> Q1a_CharCNN.ipynb

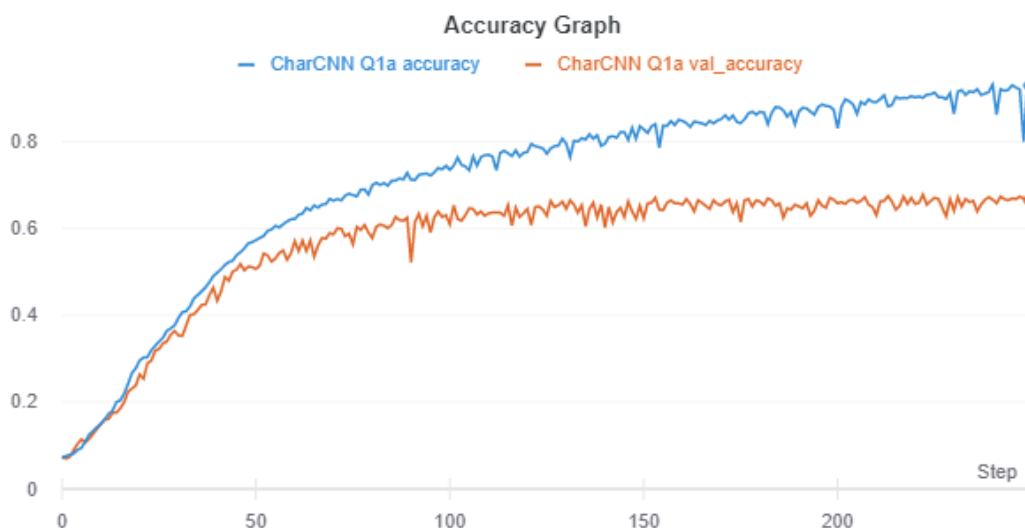


Figure 27: Accuracy against epochs for character CNN

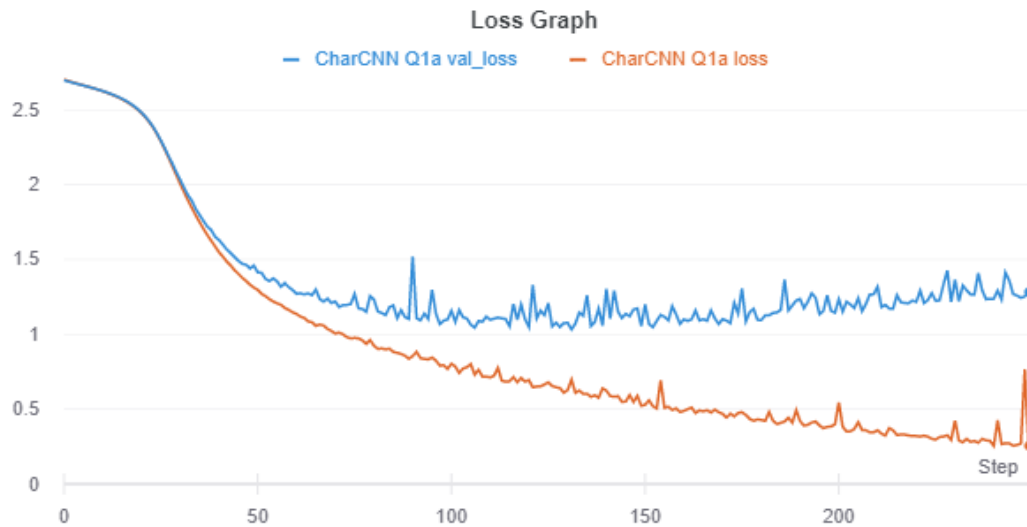


Figure 28: Loss against epochs for character CNN

Question 2

2. Design a Word CNN Classifier that receives word ids and classifies the input. Pass the inputs through an embedding layer of size 20 before feeding to the CNN. The CNN has two convolution and pooling layers with the following characteristics:

- A convolution layer $C1$ of 10 filters of window size 20×20 , VALID padding, and ReLU neurons. A max pooling layer $S1$ with a pooling window of size 4×4 , with stride = 2 and padding = 'SAME'.
- A convolution layer $C2$ of 10 filters of window size 20×1 , VALID padding, and ReLU neurons. A max pooling layer $S2$ with a pooling window of size 4×4 , with stride = 2 and padding = 'SAME'.

Plot the entropy cost on training data and the accuracy on testing data against training epochs.

Source Code: Part_B_Text_Classification -> Q2a_WordCNN.ipynb

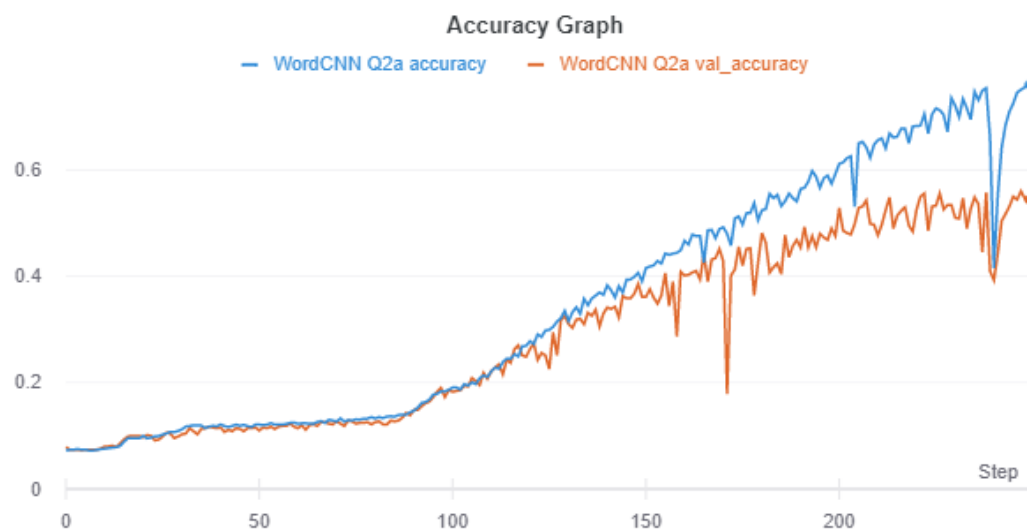


Figure 29: Accuracy against epochs for word CNN

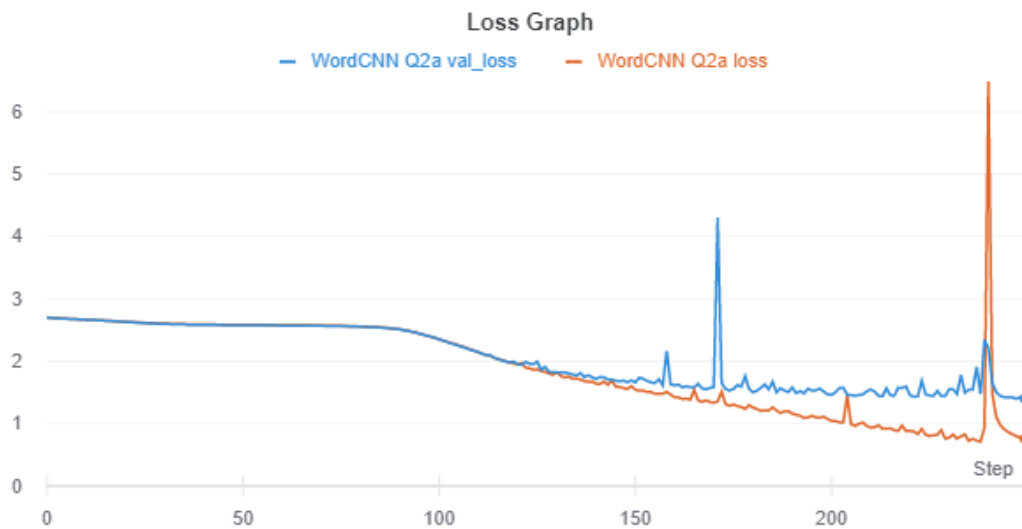


Figure 30: Loss against epochs for word CNN

Question 3

3. Design a Character RNN Classifier that receives character ids and classify the input. The RNN is GRU layer and has a hidden-layer size of 20. Plot the entropy cost on training data and the accuracy on testing data against training epochs.

Source Code: Part_B_Text_Classification -> Q3a_CharRNN.ipynb

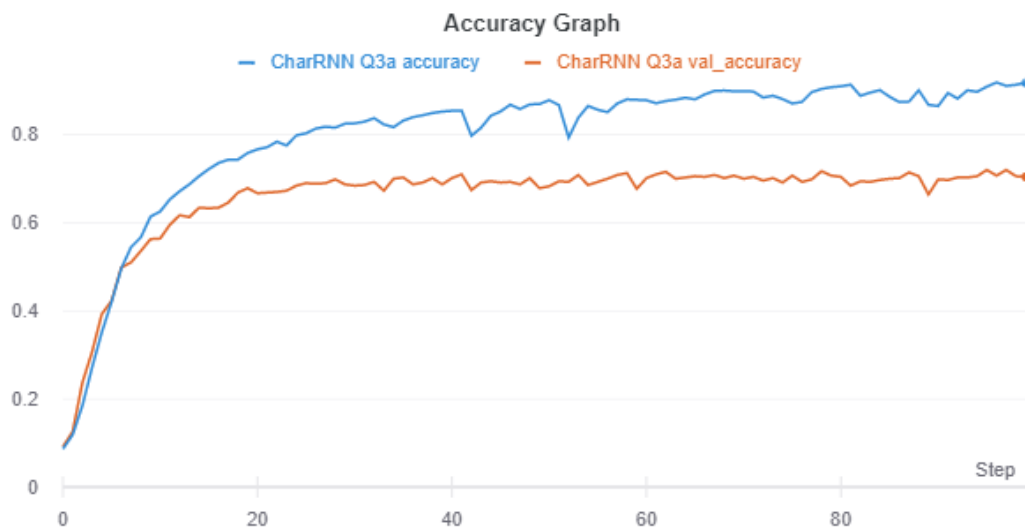


Figure 31: Accuracy against epochs for character RNN

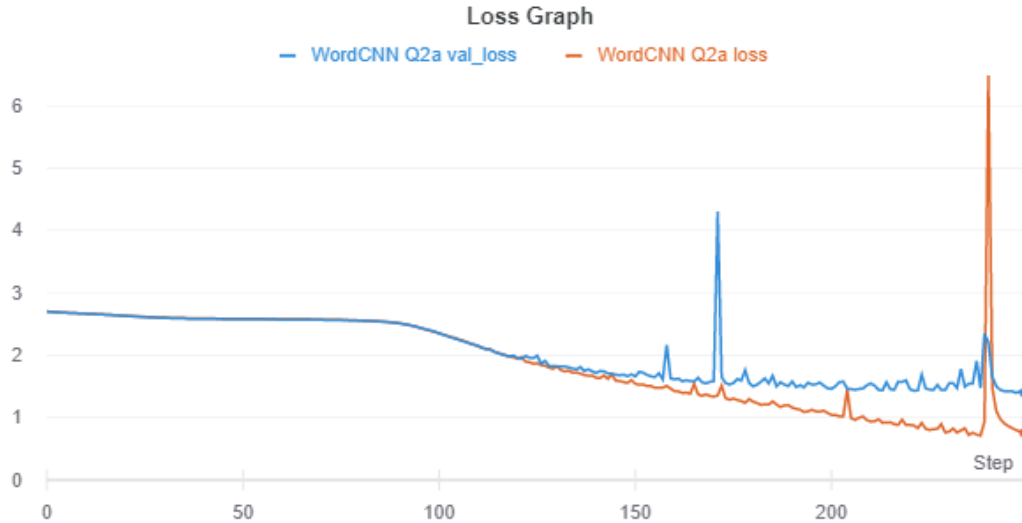


Figure 32: Loss against epochs for character RNN

Question 4

4. Design a word RNN classifier that receives word ids and classify the input. The RNN is GRU layer and has a hidden-layer size of 20. Pass the inputs through an embedding layer of size 20 before feeding to the RNN. Plot the entropy on training data and the accuracy on testing data versus training epochs.

Source Code: Part_B_Text_Classification -> Q4a_WordRNN.ipynb

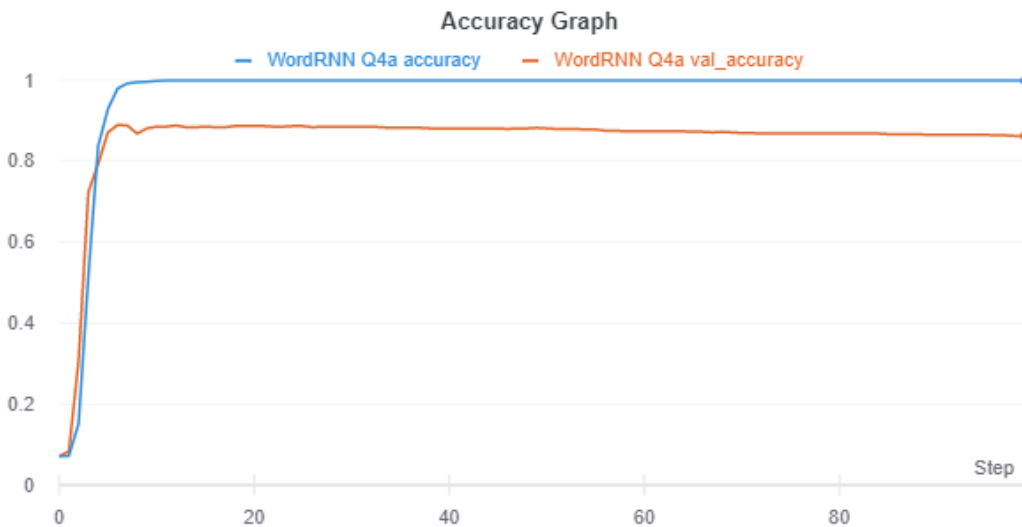


Figure 33: Accuracy against epochs for word RNN

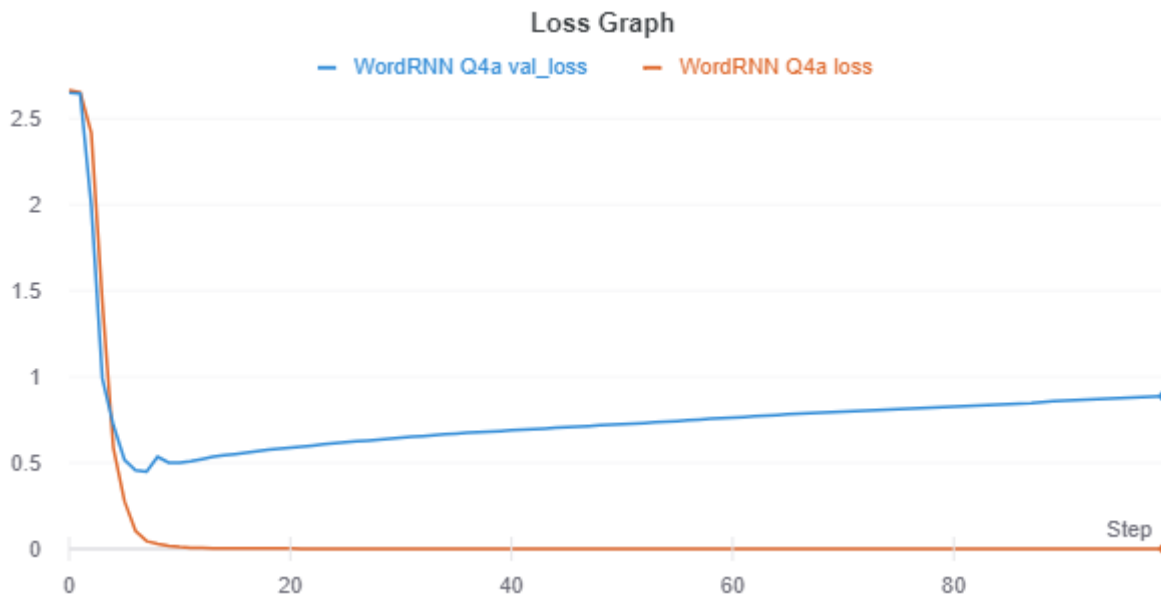


Figure 34: Loss against epochs for word RNN

Question 5

5. Compare the test accuracies and the running times of the networks implemented in parts (1) – (4).

Source Code: Part_B_Text_Classification -> Q1/2/3/4a_<model_name>.ipynb

Name (4 visualized)	Runtime	epochs	accuracy	loss	val_loss	val_accuracy ▾	Tags
WordRNN Q4a	6m 27s	100	1	0.0000552	0.8874	0.8629	WordRNN base_model
CharRNN Q3a	3m 47s	100	0.9182	0.2443	1.349	0.7043	CharRNN base_model
CharCNN Q1a	2m 19s	250	0.9305	0.255	1.284	0.6657	base_model
WordCNN Q2a	1m 49s	250	0.7609	0.7564	1.4	0.5443	base_model

Figure 35: Comparison of test accuracies and runtimes for all 4 models

From Figure 35, we can infer that the best performing model is the word RNN model which shows a test accuracy of 0.8629. It also has the longest runtime out of all the models. Although the number of epochs differ for the RNN (100 epochs) and the CNN (250 epochs) model, the RNN models still has a longer runtime than the CNN models.

5. Experiment with adding dropout to the layers of networks in parts (1) – (4) and report the test accuracies. Compare and comment on the accuracies of the networks with/without dropout.

Source Code: Part_B_Text_Classification -> Q1/2/3/4b_<model_name>.ipynb

For comparison of similar models, models without dropouts are named with part (a) while model with dropouts are named with part (b). An example of how they are named are written in the table below.

Model Name	Question Number	No Dropout (a) / Dropout (b)	Combined Name
CharRNN	Q1	a	CharRNN Q1a
CharRNN	Q1	b	CharRNN Q1b

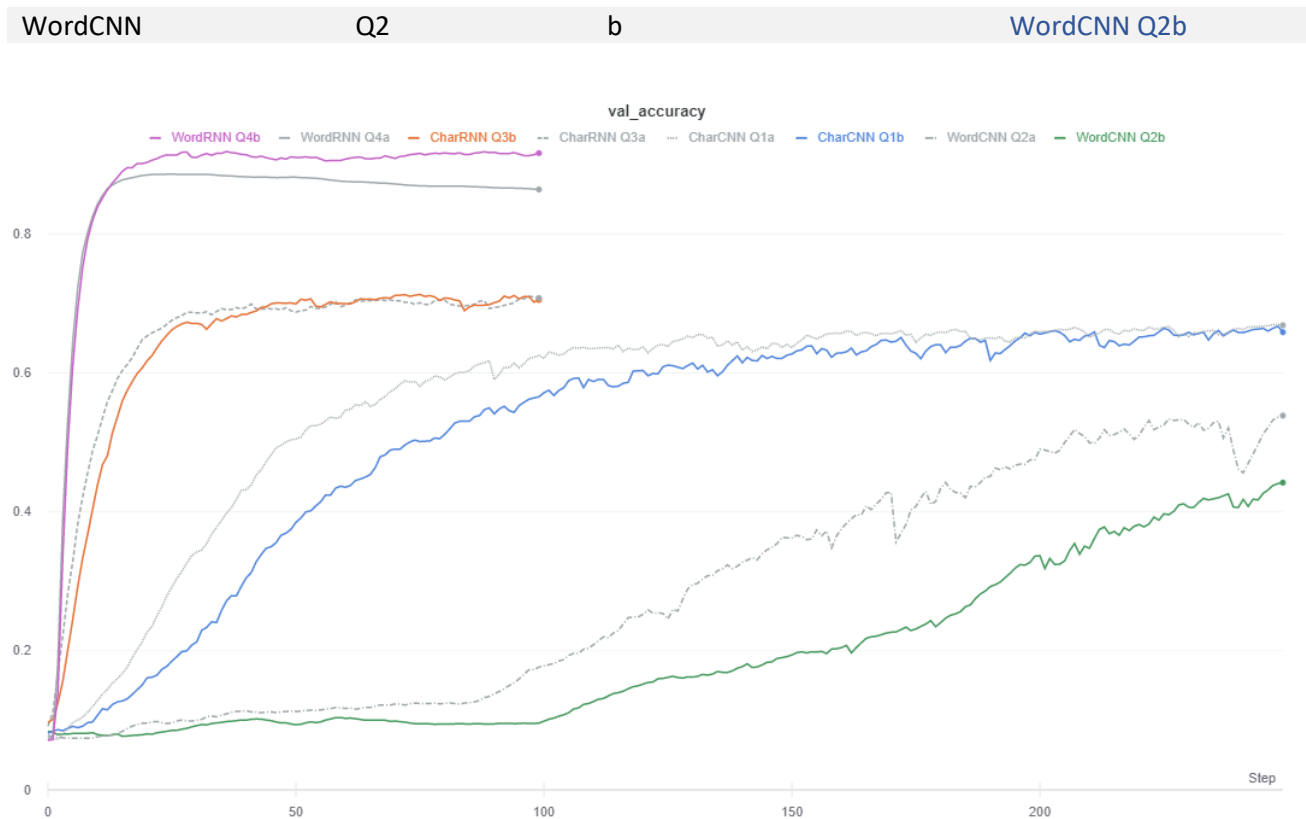


Figure 36: Comparison of test accuracies for all 4 models with (a) no dropout and (b) dropout

As seen in Figure 36, the grey line represents the model without dropout (Q1a, Q2a, Q3a, Q4a) and the coloured line represents the model with dropout added (Q1b, Q2b, Q3b, Q4b). Both RNN models have an improved performance when a dropout layer is added (value = 0.5) while the CNN models have a decreased performance when a dropout layer is added (value = 0.5). The actual figures can be seen in Figure 37 below.

Name (8 visualized)	Runtime	epochs	accuracy	loss	val_loss	val_accuracy ▼
WordRNN Q4b	5m 25s	100	0.9989	0.004229	0.4137	0.9229
WordRNN Q4a	6m 27s	100	1	0.0000552	0.8874	0.8629
CharRNN Q3b	3m 47s	100	0.7604	0.7227	1.036	0.7129
CharRNN Q3a	3m 47s	100	0.9182	0.2443	1.349	0.7043
CharCNN Q1a	2m 19s	250	0.9305	0.255	1.284	0.6657
CharCNN Q1b	1m 58s	250	0.8316	0.5009	1.329	0.6371
WordCNN Q2a	1m 49s	250	0.7609	0.7564	1.4	0.5443
WordCNN Q2b	1m 41s	250	0.4339	1.513	1.602	0.4457

Figure 37: Test Accuracies of the 4 models with (a) no dropout and (b) dropout

Question 6

6. For RNN networks implemented in (3) and (4), perform the following experiments with the aim of improving performances, compare the accuracies and report your findings:

- Replace the GRU layer with (i) a vanilla RNN layer and (ii) a LSTM layer

- b. Increase the number of RNN layers to 2 layers
- c. Add gradient clipping to RNN training with clipping threshold = 2

Source Code: Part_B_Text_Classification -> <model_name>_Q6a/b/c.ipynb

Character RNN

Name (6 visualized)	Runtime	epochs	accuracy	loss	val_loss	val_accuracy ▼	Tags	Notes
● CharRNN Q6c	7m 25s	100	0.8882	0.3527	1.071	0.7557	CharRNN	Add notes...
● CharRNN Q6b	7m 42s	100	0.8684	0.4049	1.159	0.7414	CharRNN	Add notes...
● CharRNN Q3b	3m 47s	100	0.7604	0.7227	1.036	0.7129	CharRNN Dropout Q5	Add notes...
● CharRNN Q6aii	4m 4s	100	0.6123	0.9949	1.241	0.5914	CharRNN	Add notes...
● CharRNN Q6ai	1m 23s	100	0.09536	2.624	2.606	0.1	CharRNN	Add notes...
● CharRNN Q6ai-2	2m 28s	100	0.06036	15136824	8651415	0.05857	CharRNN	linear activation

Figure 38: Table of Results for Question 6 Character RNN

The results for the character RNN are quite distinct from each other. As seen in Figure 38, after experimenting with switching the cell within the RNN layer, the best model implements the use of the GRUCell. Subsequently, by increasing the RNN layers and adding gradient clipping, we are able to further improve the performance of the model and reduce overfitting.

Word RNN

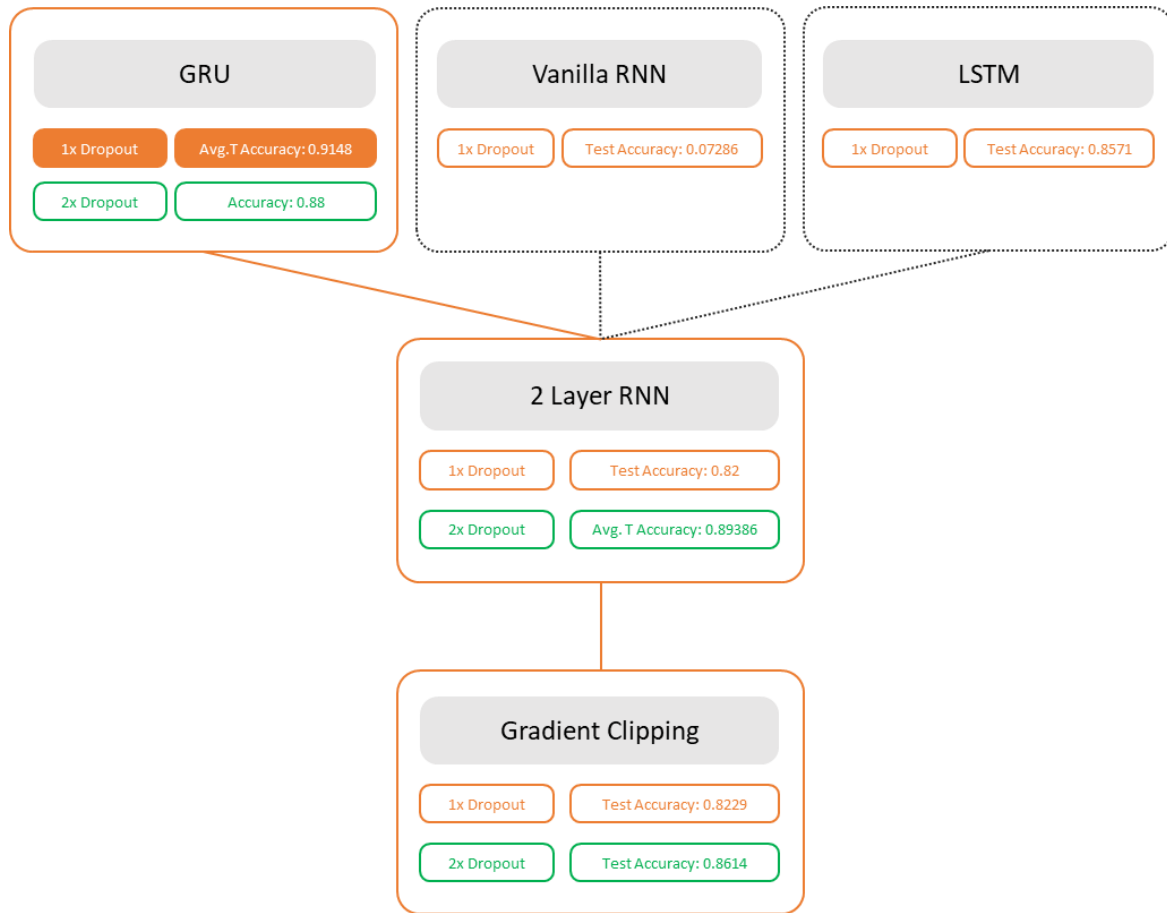


Figure 39: Overview of results for word RNN

The results of the tuning process for word RNN is displayed above. We first test the use of 2 dropout layers instead of 1 when GRU is used. Next, we test the use of a vanilla RNN cell and LSTM cell. From the results, we can derive that GRU has the best performance. Hence, the GRU Cell is used for the 2-layer RNN and gradient clipping in Q6b and Q6c.

Name (0 visualized)	epochs	accuracy	loss	val_loss	val_accuracy	Tags
WordRNN Q6b-2 SC2	100	0.9814	0.06006	0.9281	0.8843	6b
WordRNN Q6b-2 SC	100	0.9827	0.06525	0.7194	0.9086	6b SC
WordRNN Q6b-2	100	0.9805	0.06478	0.9007	0.8886	6b SC WordRNN

Figure 40: Second best performing model (results over 3 runs)

Name (1 visualized)	epochs	accuracy	loss	val_loss	val_accuracy	Tags
WordRNN Q4b SC2	100	0.9993	0.003264	0.4668	0.9029	4b SC
WordRNN Q4b SC	100	0.9984	0.004867	0.4108	0.9186	4b SC
WordRNN Q4b	100	0.9989	0.004229	0.4137	0.9229	4b Dropout Q5 WordRNN

Figure 41: Best performing model (results over 3 runs)

Naming of Models

For comparison of similar models, models without dropouts are named with part (a) while model with dropouts are named with part (b). Additionally, if the model has 2 dropout layers, “-2” is added after “b”. If only 1 dropout layer is added, nothing is added. SC stands for Sanity Check where an experiment is repeated again for verification. A few examples of how they are named are written in the table below.

Model Name	Question Number	Dropout (b)	2x dropout (-2)	Sanity Check (SC)	Combined Name
WordRNN	Q6	b	-2	SC	WordRNN Q6b-2 SC2
WordRNN	Q4	b	NIL (no dropout)		WordRNN Q4b
WordRNN	Q4	b	NIL	SC	WordRNN Q4b SC

Verification of Results

From the initial results, we know that the 2-layer RNN performs significantly better when more dropouts are added. Therefore, the 2-layer RNN with 2 dropout layers is repeated to take an average of the 3 test accuracies.

Average Test Accuracy of WordRNN Q6b-2 = $(0.8843 + 0.9086 + 0.8886) / 3 = 0.89386$

Similarly, the original 1-layer RNN with GRU cell has the best performance, to verify that it beats the performance of the 2-layer RNN with 2 dropout layers, we repeat the experiment and take the average test accuracy out of 3 runs.

Average Test Accuracy of WordRNN Q4b = $(0.9029 + 0.9186 + 0.9229) / 3 = 0.9148$

Best Model

Thus, the best performing model is the original 1-layer RNN with 1 dropout layer and an average test accuracy of 0.9148.

Conclusion

In Part B of the assignment, we designed 4 different models (Character CNN, Character RNN, Word CNN and Word RNN). RNN performs much better as compared to CNN. Using the results obtained, we compared the 4 models and eliminate the CNN model architecture in favour of the RNN model. 1 or 2 dropout layers were also added to reduce overfitting of the model. Lastly, experiments were carried out to tweak the model architecture and reduce overfitting. There could be more work done to reduce the overfitting of the model, but it is not within the scope of the questions in the assignment.

Appendix

Weights and Biases



Weights and Biases is a developer tool used to track, compare and visualize ML experiments. Below are the links to the experiment results. They can be found in the Experiment Results folder as well.

Part A (Q2 Sweep Only)

<https://wandb.ai/todayisagreatday/NNA2%20Test%20Runs/sweeps/ccro3cg5?workspace=user-todayisagreatday>

Part A

<https://wandb.ai/todayisagreatday/NNA2%20Test%20Runs/?workspace=user-todayisagreatday>

Part B

<https://wandb.ai/todayisagreatday/NNA2%20Part%20B%20Redo?workspace=user-todayisagreatday>

Part B Check Results

<https://wandb.ai/todayisagreatday/NNA2%20Part%20B%20Checks/?workspace=user-todayisagreatday>

Source Codes

Source codes are found in the Part_A_Object_Recognition and Part_B_Text_Classification folders. Alternatively, [they can be accessed here](#).