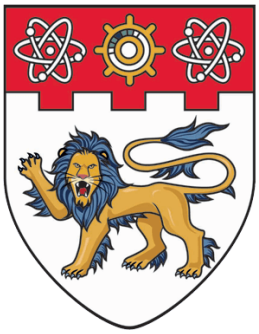


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

School of Computer Science and Engineering

CZ4042 - Neural Network & Deep Learning

Project 2

Name	Matriculation Number
Quek Chin Wei	U1620394D
Ching Jia Chin	U1620237E

Introduction

In this project, we are doing object recognition / classification and text classification.

For Part A, we need to construct a convolutional neural network and train the network to recognize image classes. The dataset used in the project is CIFAR-10 dataset which contains RGB colour images of size 32 x 32 and 10 classes. We then fine tune the CNN using different approaches such as momentum, RMSProp, Adam optimizer and dropout.

For Part B, we experience with building CNN , RNN, GRU, LSTM for text classification. We also try to use multiple hidden layer of RNN , dropout and gradient clipping in training. The dataset used in the project contains the first paragraphs collected from Wikipage and their labels about their category. We then compare the accuracies , running time and performance of different neural networks on text classification task.

Methods

In this section, we briefly discuss any methods or concepts used in the project, for example convolution and pooling, padding and regularization methods.

At convolution layer, filters/kernels are applied to the input image. The output activation produced by a particular filter is known as a feature map. Convolution learns the various aspects and underlying features of the input image. The output feature map is then fed into a pooling layer to further reduce the dimension of the feature map. The pooling operation used in this project is max pooling. In practice, one can fine tune the number of filters, activation function of filters, window size, strides, types of padding and pooling operation in convolution and pooling layer.

There are two types of padding namely SAME padding and VALID padding. For VALID padding, we don't pad the input and only apply filter whenever the filter completely overlap the input. For SAME padding, we pad the input such that the output produced has the same size as the input.

Gradient exploding is a common issue when we train on RNN as the network structure of RNN involves many time step sequences and has long term dependencies. Gradient clipping is used to overcome the issue of gradient exploding by forcing the gradient value to a specific value or normalize the gradient when the gradient exceeds certain threshold.

Dropout is a common regularization technique used to overcome overfitting on the training data. Dropout 'shut down' some neuron during training. Dropout rate of 0.2 is consistently applied throughout the project. Dropout rate is low in order for the network to train and learn properly because we apply dropout at convolution layer where the number of parameters to update is already low. We apply dropout after every layer etc convolution layer , pooling layer and fully connected layer. There is no specific rule on where we should apply dropout and in practice dropout can be applied at any layer as long as it increases the performance of the network.

Experiments and Results

Part A

1. Train the network by using mini-batch gradient descent learning. Set batch size =128, and learning rate α=0.001. Images should be scaled.

a. Plot the training cost and the test accuracy against learning epochs.

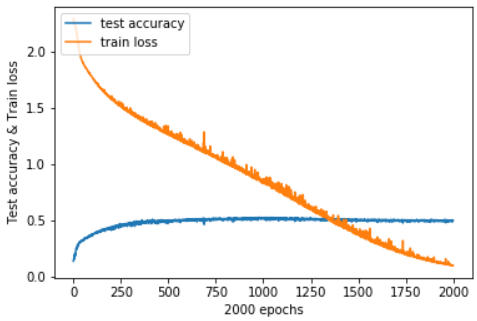
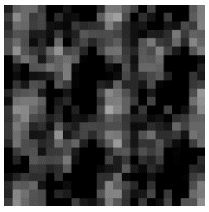
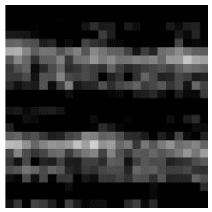
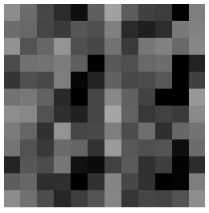
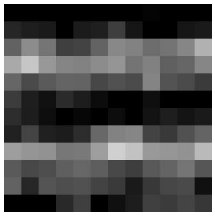


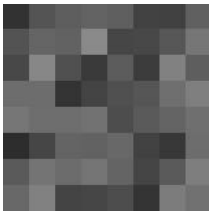
Figure 1a

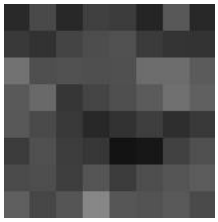
Test accuracy reached the maximum of around 0.5 at epoch 500. Afterwards, the loss is decreasing but accuracy stays the same, suggesting overfitting. Subsequent training will be done with 750 epochs.

b. For any two test patterns, plot the feature maps at both convolution layers (C_1 and C_2) and pooling layers (S_1 and S_2) along with the test patterns.

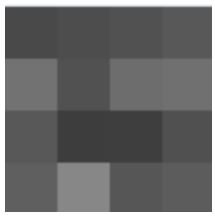
	Test Pattern 1	Test Pattern 2
Raw		
C1		
S1		

C2





S2



2. Using a grid search, find the optimal numbers of feature maps for part (1) at the convolution layers. Use the test accuracy to determine the optimal number of feature maps.

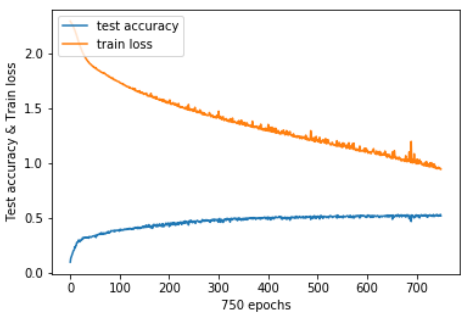
Searching these grids:
C1 = {40,50,60}
C2 = {50,60,70}

Test Accuracy	C1:40	C1:50	C1:60
C2:50	0.517	0.5165	0.5185
C2:60	0.515	0.516	0.516
C2:70	0.5155	0.5075	0.5265

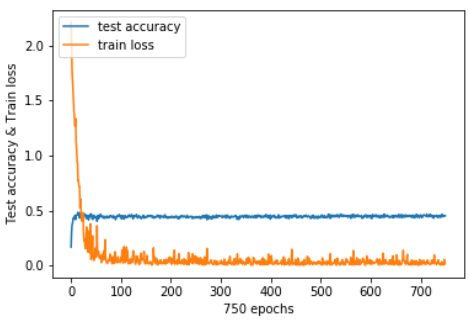
From the table above, optimal number of feature maps {C1, C2} is {60, 70}.

3. Using the optimal number of filters found in part (2), train the network by:
a. Adding the momentum term with momentum γ=0.1.
b. Using RMSProp algorithm for learning
c. Using Adam optimizer for learning
d. Adding dropout to the layers

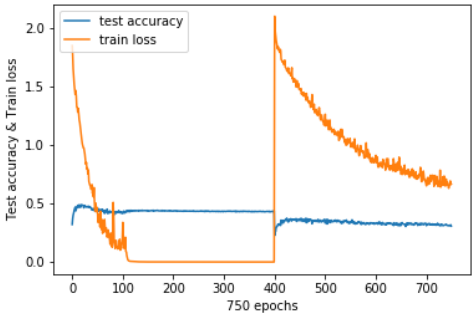
Plot the training costs and test accuracies against epochs for each case.



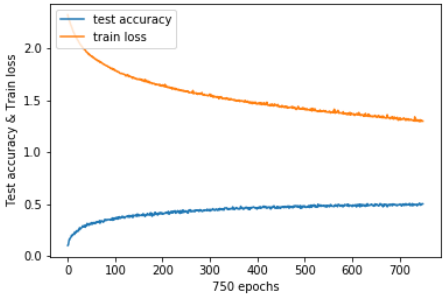
Momentum



RMSProp



Adam Optimizer



Dropouts

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

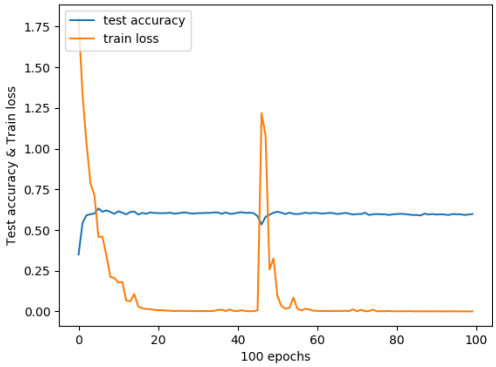
GradientDescent – GradientDescent has the slowest training rate, reach 0.5 accuracy after 500 epochs. There are also many small spikes during training.
Momentum – In this case, Momentum has a similar performance to Gradient Descent. This might be because the gradient during training is too gentle for Momentum to pick up any speed. Alternatively, the constant spikes has cancelled out any built up momentum.
RMSProp – RMSProp has the fastest training rate, finishing training before 100 epochs. When slowing down, there are huge spikes while its stabilising. However, there is a slight loss in accuracy compared to GradientDescent.
Adam – Adam has the second fastest training rate, finishing training at around 120 epochs. When slowing down, it has similar spikes to RMSProp. After flatlining for a while, there is a sudden spike in loss at 400 epochs. Afterwards, training loss decreases and seems to converge at 0.7. There is an apparent loss of 0.1 accuracy as well. It could be that epoch 400 has a particularly unlucky mini-batch, causing the loss to spike.
Dropouts- After introducing Dropouts to Gradient Descent, there are less spikes during training but training takes a longer time.

For all the models, the test accuracy does not increase past 0.5 despite the decrease in loss. If we compare models where loss = 0, accuracy stays around 0.4-0.5. This suggests that the model has a max performance of 50% accuracy when used outside of training. To fix this, we might have to introduce more robust datasets or change the model.

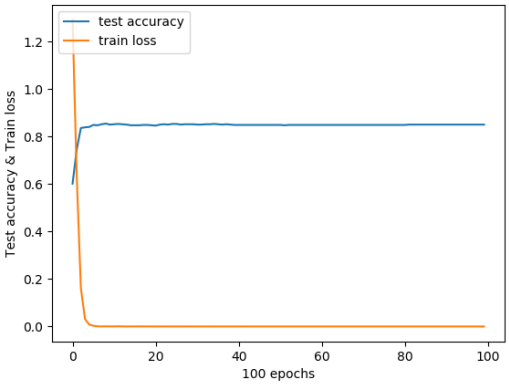
In this case, RMSProp has the best performance out of all the models. Despite not showing better accuracy, it finished its training in 100 epochs, which is 5 times faster than some models.

Part B: Text classification

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

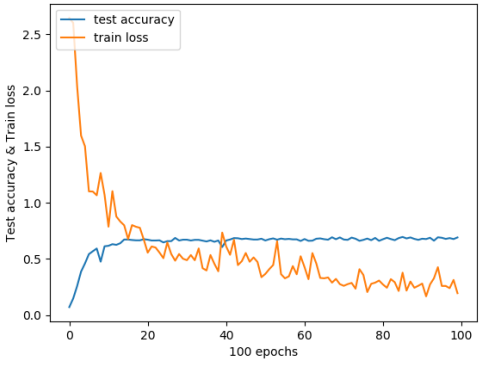


2. Plot the entropy cost on training data and the accuracy on testing data against training epochs.



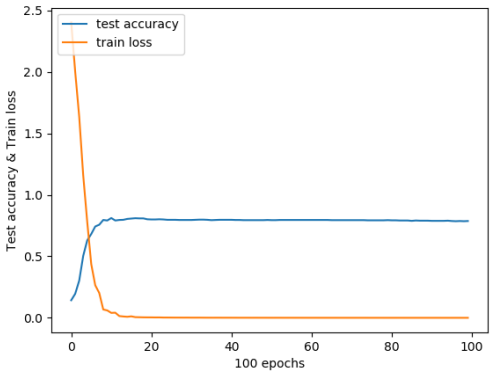
3. Design a Character RNN Classifier that receives character ids and classify the input. The RNN is GRU layer and has a hidden-layer size of 20.

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



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.



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

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.

Model	Test Accuracy	Running Times (seconds)
Q1 without dropouts	0.5985714	1106
Q2 without dropouts	0.85	183
Q3 without dropouts	0.72	439
Q4 without dropouts	0.7871429	141
Q1 with dropouts	0.6414286	1030
Q2 with dropouts	0.9028571	186
Q3 with dropouts	0.71	426
Q4 with dropouts	0.89	138

Run times:

1. Comparing models with & without dropouts, both runtimes are similar.
2. Character processing models takes substantially longer time to train than word processing models.
3. RNN models are substantially faster to train compared to CNN models.

Accuracies:

1. For the most part, models with dropouts have better accuracies than models without dropouts.
2. Word processing models are more accurate than character processing models
3. CNN & RNN models have similar performance for accuracies.

In this experiment, both CNN & RNN models have similar performance. However, RNN models have a slight advantage over CNN in running times.

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:
a. 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.

Model	Test Accuracy
Char RNN with Vanilla RNN layer	0.108571425
Word RNN with Vanilla RNN layer	0.07
Char RNN with LSTM layer	0.69142854
Word RNN with LSTM layer	0.73285717
Char RNN with 1 RNN layer (q3)	0.72
Word RNN with 1 RNN layer (q4)	0.7871429
Char RNN with 2 RNN layers	0.7214286
Word RNN with 2 RNN layers	0.8157143
Char RNN with gradient clipping	0.69285715
Word RNN with gradient clipping	0.87142855

6a) The result with vanilla RNN is bad and gives low test accuracy. This may be caused by exploding gradient in the network. This happens when the network updates the gradient during backpropagation, going through deep network with too many multiplications in chain rules.

With LSTM, we can get a better result than vanilla RNN. The test accuracy for Char and Word classifier with LSTM layer are significantly higher. LSTM special structure incorporates memory units that allow the network to learn better in long term dependencies.

6b) With 2 layers of GRU, we can obtain slightly better test accuracy. Char RNN and word RNN with one layer has test accuracy of 0.72 and 0.7871429 respectively, while char RNN and word RNN with two layers has test accuracy of 0.7214286 and 0.8157143.

6c) Gradient clipping set the gradient to a certain value when the gradient exceeds some threshold. We set clipping threshold of 2 for this question. Gradient clipping seems to perform better especially in word RNN.

Conclusion

From Part A, we learnt that CNN is very computationally intensive and requires a GPU to run practically. This makes sense as CNN requires 20x100 neurons of calculations done for one layer simultaneously, taking up a lot of memory.

There are also many feature maps that have no features. This might suggest that there may be too many filters than necessary.

From Part B, we learnt that CNN is surprisingly useful for text recognition. Its performance is on par with most RNN networks while using a more human understandable model.

RNN GRU and LSTM has a good performance for text recognition compared to vanilla RNN. We should be careful of exploding gradients for RNN.

Finally, we should use CNN for spatial datasets such as images while using RNN for temporal datasets such as sentences or computational states. Perhaps, it might be possible for RNN to parse images if we convert each image into a sequence of pixels.