School of Computer Science and Engineering

CE/CZ4042: Neural Network and Deep Learning

AY19/20 Semester 1

Project 2

Submitted by:

|  |  |  |
| --- | --- | --- |
| Student Name | Matriculation No. | Contribution |
| Fang Wen Jie | U1720196D | Part A: Object Recognition |
| Png Qun Jia | U1722491B | Part B: Text Classification |

[**1. Introduction**](#_po9p4zoe7qgk) **3**

[**2. Method**](#_qiz2ky7yyrqp) **3**

[2.1 Convolutional and Pooling Layer](#_tfu9402mt26b) 3

[2.2 SAME padding and VALID padding](#_v8ycqo5t1gee) 3

[2.3 Embedding Layer](#_zea18cn0lkzh) 3

[2.4 Dropouts](#_9ueapyxh47i6) 3

[2.5 Gradient Clipping](#_g14ttn9maw0h) 4

[**3. Experiments and Results**](#_uob9kz3gpucj) **4**

[3.1 Part A: Object Recognition](#_iqjx0847gb79) 4

[3.1.1 Designing a convolutional neural network](#_scapd8ptru85) 4

[3.1.2 Part A Question 1A](#_tyx1ttxmdlct) 6

[3.1.3 Part A Question 1B](#_42mqnxpef46r) 7

[3.1.4 Part A Question 2](#_k4tbzvn6ah3y) 9

[3.1.5 Part A Question 3A](#_2dydb6rhi0qq) 11

[3.1.6 Part A Question 3B](#_l2f66pjsqq0i) 11

[3.1.7 Part A Question 3C](#_9mry2ppps7qr) 12

[3.1.8 Part A Question 3D](#_ftfuhzbvc2ww) 12

[3.1.9 Part A Question 4](#_sudd8zsavzyc) 13

[3.2 Part B: Text Classification](#_isg1jsobsvcq) 15

[3.2.1 Part B Question 1 & 2](#_vht4pug6jp1u) 15

[3.2.2 Part B Question 3 & 4](#_sl54wpdravxs) 17

[3.2.3 Part B Question 5](#_24ymjojnd39o) 19

[3.2.4 Part B Question 6A](#_8us1b6l8envh) 21

[3.2.5 Part B Question 6B](#_7rxhhrdceckn) 22

[3.2.6 Part B Question 6C](#_d856sivafvsf) 23

# 

# 1. Introduction

The objective of this project is to allow us to understand the architecture of convolutional neural network (CNN) and recurrent neural network (RNN).

Part A works on object recognition using CNN, fine tuning the hyper parameters of the CNN and experimenting with dropouts.

Part B works on text classification using CNN and RNN. It also experiments with dropouts as well as noting down the time taken to train the different networks. The timing is then used to compare the effectiveness and performance of these networks.

# 2. Method

## 2.1 Convolutional and Pooling Layer

Features or weights of filters are learnt using back propagation in the convolution layer. The features or weights learnt are called filters or kernels. Activations are obtained by convolving the filters with the input activations with a predetermined strides and padding.

The output of the convolutional layer is then fed into the MAX pooling layer to reduce the dimension of the convolution layer.

## 2.2 SAME padding and VALID padding

SAME padding apply the filter to make the output size same as input size. VALID padding apply the filter wherever it completely overlaps with the input. In part A, VALID padding is applied in both convolutional and pooling layer. In part B (CNN network), VALID padding is applied in convolutional layer 1 and 2, and SAME padding is applied in pooling layer 1 and 2.

## 2.3 Embedding Layer

Word Embedding provides a dense representation of words and their relative meanings. This is an improvement over sparse representations used in simpler bag of word model representations.In Part B, embedding is applied to the dataset before it is fed to the input layer of RNN.

## 2.4 Dropouts

Dropout is a regularization technique to prevent overfitting. When dropout is used, some of the units in both visible and hidden layers are randomly “drop out” during training.

## 2.5 Gradient Clipping

Gradient Clipping is a technique used to prevent gradient exploding. It involves forcing the gradient values to a specific minimum or maximum value if the gradient exceeded an expected range. Gradient Clipping was used in Part B.

# 

# 3. Experiments and Results

## 3.1 Part A: Object Recognition

### 3.1.1 Designing a convolutional neural network

The CNN design for this question is based on the specifications below.

1. An input layer of 32x32
2. A convolution layer C1 of 50 filters of windows size 9x9, ‘VALID’ padding, and Relu neurons
3. A max pooling layer S1 with a pooling window of size 2x2, with stride 2, ‘VALID’ padding
4. A convolution layer C2 of 60 filters of window size 5x5, ‘VALID’ padding, and Relu neurons
5. A max pooling layer S2, with a pooling window of size 2x2, with stride 2, ‘VALID’ padding
6. A fully connected layer F3 of size 300
7. A softmax layer F4 of size 10

The first convolution layer C1 is implemented as such:

|  |
| --- |
|  |

The second convolution layer C2 is implemented as such:

|  |
| --- |
|  |

The fully connected layer and softmax output is implemented as such:

|  |
| --- |
|  |

## 

### 3.1.2 Part A Question 1A

|  |
| --- |
|  |

We trained the model using the full dataset with mini batch training of size 128 using gradient descent optimizer for 1000 epochs and achieved test accuracy of approximately 52%. From the graph above, it is observed that from about epoch 500-600, the training loss continue to decrease at a constant rate but the test accuracy appears to be saturated. This suggests that overfitting might have occurred in the network where the training is not yielding any improvements to the test data.

### 3.1.3 Part A Question 1B

Below are the test pattern and feature map at each convolution and pooling layers.

|  |  |
| --- | --- |
| First test pattern | |
|  | |
| Convolutional layer 1 (C1) | Pooling layer 1 (S1) |
|  |  |
| Convolutional layer 2 (C2) | Pooling layer 2 (S2) |
|  |  |

# 

# 

|  |  |
| --- | --- |
| Second test pattern | |
|  | |
| Convolutional layer 1 (C1) | Pooling layer 1 (S1) |
|  |  |
| Convolutional layer 2 (C2) | Pooling layer 2 (S2) |
|  |  |

# 

### 3.1.4 Part A Question 2

We used a grid search from [10, 10] to [100, 100] with an increment of 10 filters at each time instance to find the optimal number of feature maps for the convolutional layers.The grid search of [10, 10] to [100, 100] would yield us 100 different combinations. As such, we used a dictionary to make use of the key:value pairs to keep track of the accuracies for the different combinations with training epochs of 100.

The implementations of the dictionary and grid search is as such.

|  |
| --- |
|  |

As accuracies for 100 different instances is too much to show. We decided to only showed the top 5 accuracies. The graph below shows the top 5 accuracies for the different feature maps combinations.

|  |
| --- |
|  |

The table belows shows the accuracies of the feature maps starting from [10, 10] to [100, 100]. Note that highest accuracy rounded off to 3 d.p. is taken.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Convolution Layer C2 | | | | | | | | | | |
|  | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| 10 | 0.318 | 0.344 | 0.342 | 0.359 | 0.359 | 0.362 | 0.384 | 0.368 | 0.359 | 0.387 |
| 20 | 0.328 | 0.360 | 0.355 | 0.350 | 0.377 | 0.371 | 0.372 | 0.386 | 0.384 | 0.399 |
| 30 | 0.334 | 0.357 | 0.353 | 0.366 | 0.363 | 0.372 | 0.384 | 0.378 | 0.387 | 0.374 |
| 40 | 0.365 | 0.378 | 0.369 | 0.375 | 0.361 | 0.380 | 0.381 | 0.381 | 0.384 | 0.403 |
| 50 | 0.363 | 0.371 | 0.368 | 0.369 | 0.384 | 0.401 | 0.395 | 0.389 | 0.393 | 0.388 |
| 60 | 0.366 | 0.365 | 0.379 | 0.379 | 0.393 | 0.390 | 0.384 | 0.394 | 0.394 | 0.412 |
| 70 | 0.377 | 0.369 | 0.395 | 0.382 | 0.396 | 0.400 | 0.399 | 0.390 | 0.405 | 0.401 |
| 80 | 0.358 | 0.384 | 0.375 | 0.379 | 0.397 | 0.390 | 0.387 | 0.398 | 0.400 | 0.414 |
| 90 | 0.374 | 0.384 | 0.399 | 0.389 | 0.384 | 0.391 | 0.403 | 0.409 | 0.407 | 0.407 |
| 100 | 0.379 | 0.382 | 0.397 | 0.396 | 0.399 | 0.407 | 0.412 | 0.419 | 0.409 | 0.410 |

Based on the table and graph, it is observed that the higher the number of feature maps for C1 and C2, the higher the test accuracy. Since feature maps of [100, 100] gave us the highest accuracy, we decided to choose [100, 100] for the feature maps for C1 and C2.

### 3.1.5 Part A Question 3A

|  |
| --- |
| Adding the momentum term with momentum 𝝲 = 0.1. |
|  |

### 

### 3.1.6 Part A Question 3B

|  |
| --- |
| Using RMSprop algorithm for learning |
|  |

### 

### 3.1.7 Part A Question 3C

|  |
| --- |
| Using Adam optimizer for learning |
|  |

### 3.1.8 Part A Question 3D

|  |  |
| --- | --- |
| GD with dropouts | Momentum with dropouts |
|  |  |
| RMSprop with dropouts | Adam optimizer with dropouts |
|  |  |

### 3.1.9 Part A Question 4

|  |  |
| --- | --- |
| Test Accuracies of all models without dropouts | Test Accuracies of all models with dropouts |
|  |  |
| Training cost of all models without dropouts | Training cost of all models with dropouts |
|  |  |

|  |  |
| --- | --- |
| Models (Based on 1000 epochs) | Accuracy |
| **Gradient Descent** with 50 and 60 Feature Maps | 52.0% |
| **Gradient Descent** with 100 and 100 Feature Maps | 52.7% |
| **Gradient Descent** with 100 and 100 Feature Maps + Dropouts | 53.0% |
| **Momentum Optimizer** with 100 and 100 Feature Maps | 52.1% |
| **Momentum Optimizer** with 100 and 100 Feature Maps + Dropouts | 53.6% |
| **RMSProp Optimizer** with 100 and 100 Feature Maps | 49.5% |
| **RMSProp Optimize**r with 100 and 100 Feature Maps + Dropouts | 49.6% |
| **Adam Optimizer** with 100 and 100 Feature Maps | 50.2% |
| **Adam Optimizer** with 100 and 100 Feature Maps + Dropouts | 48.7% |

Based on the graph and table shown above, it is observed that dropouts has little to no effects on the network where the accuracy varies by about 1 - 2%.

Instances where network with dropouts performs poorer is due to random neurons being ignored during training. This caused their contribution to the activation of downstream neurons to be temporarily removed on the forward pass and any weights update will not be back propagated. Dropouts affected the performance of the network as the network will be less sensitive to the specific weights of neurons.

It is also observed that RMSprop and Adam optimizer with/without dropout outperform the rest from epoch 0 to 400 with test accuracies converging at approximately 50 epochs. However, after 400 epochs, there are no significant improvements in their performance and Adam optimizer without dropout encounters a degradation problem. The test accuracy is then being caught up by momentum and gradient descent optimizer from epochs 400.

From the graph, Adam and RMSprop with dropouts takes slightly longer to converge as compared to no dropouts. Another noticeable difference is that there is no convergence with loss of 0 for Adam optimizer with dropouts. This is telling us that overfitting might occured in Adam optimizer without dropouts.

In terms of accuracy, the Momentum Optimiser with dropouts achieved the best accuracy at 53.6%. This is followed by the Gradient Descent Optimiser with [100, 100] feature maps. While these two optimizers have the best accuracy, it is important to note that they also took the most number of epochs to converge. Looking at their test accuracy graphs, it took them about 1000 epochs for the accuracy to converge whereas the other optimizer took only about 50 to 100 epochs.

In conclusion, we can’t conclude that the Momentum Optimiser with dropout is the best as it could be too early in the training phase and dropouts might have little effect on this dataset.

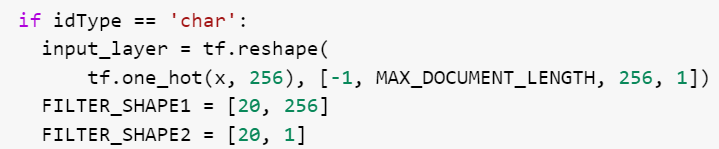
Looking at Adam optimiser, it only train for 300 epochs before it faced degradation problem. However, when dropouts are added to it, it managed to achieve an accuracy of 46.4%. Training error of RMSProp and Adam Optimiser also consistently achieved lower values than the Momentum Optimiser.

Such discrepancies could be due to overfitting of the model to the training data. Using cross-validation techniques might allow us to achieve better results for the RMSProp and Adam Optimiser.

## 3.2 Part B: Text Classification

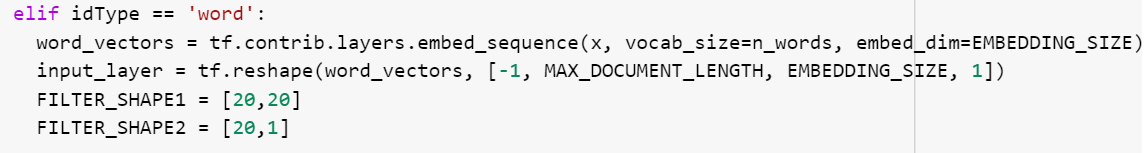
### 3.2.1 Part B Question 1 & 2

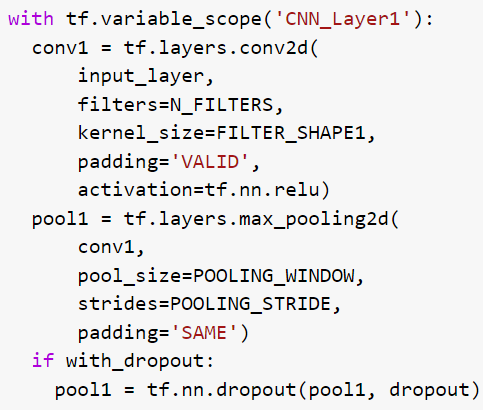
**Input layer for Character ids**

****

**Input layer for Word ids**

* Embedding layer of size 20

****

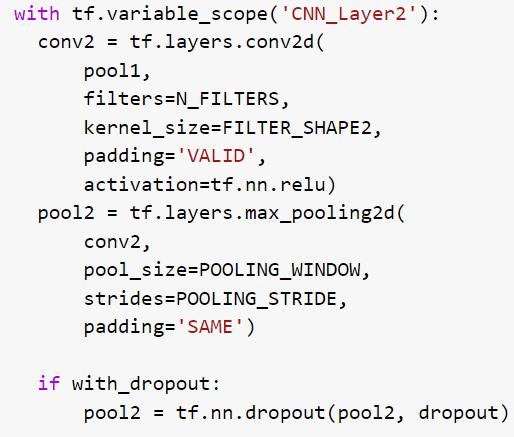


**CNN layers Implementation**

A convolution layer 𝐶1

* 10 filters
* filter window size 20x256 for character ids
* filter window size 20x20 for word ids
* VALID padding
* ReLU neurons.

A max pooling layer 𝑆1

* pooling window of size 4x4
* stride = 2
* padding = 'SAME'.
* If with\_dropout is “True”, apply dropout to pooling layer 𝑆1 (probability = 0.9)

A convolution layer 𝐶2

* 10 filters of window size 20x1
* VALID padding
* ReLU neurons.

A max pooling layer 𝑆2

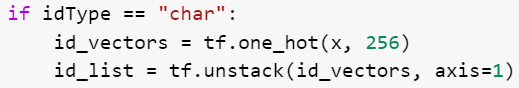
* with a pooling window of size 4x4
* stride = 2
* padding = 'SAME'.
* If with\_dropout is “True”, apply dropout to pooling layer 𝑆2 (probability = 0.9)

**Results**

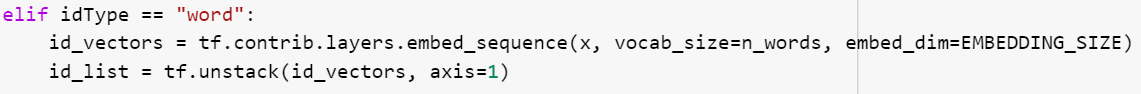
|  |  |
| --- | --- |
| CNN Character (Question 1) | CNN Character with dropouts |
|  |  |
| CNN Word (Question 2) | CNN Word with dropouts |
|  |  |

### 3.2.2 Part B Question 3 & 4

**Get Character List**

****

**Get Word List**

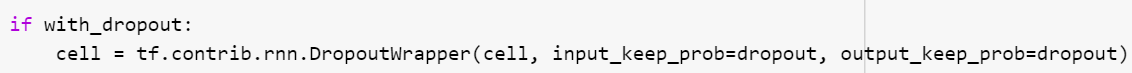
****

**GRU layer with hidden-layer size of 20**

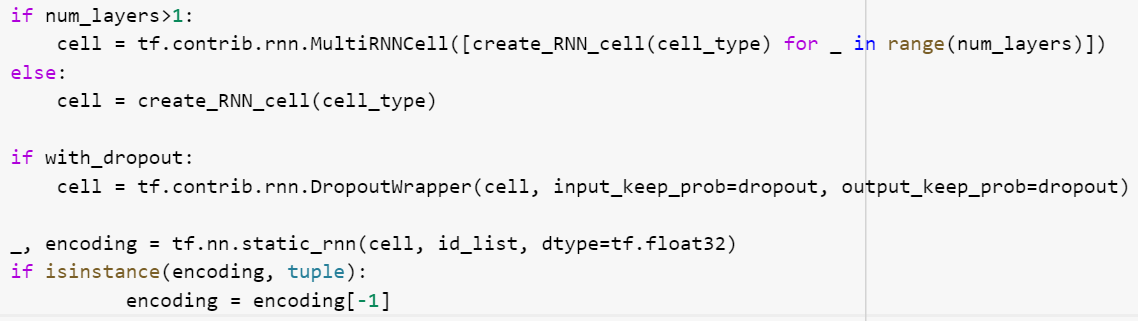
****

**Dropout**

If with\_dropout is “True”, apply dropout to cell (probability = 0.9)

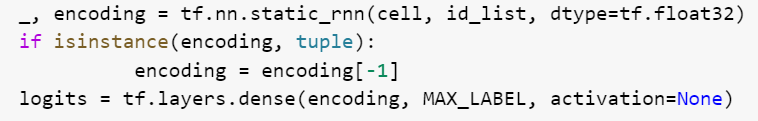
****

**Create a recurrent neural network specified by RNNCell**



**Create logits**

If encoding is a tuple, return the final state in encoding

****

**Results**

|  |  |
| --- | --- |
| RNN Character (Question 3) | RNN Character with dropouts |
|  |  |
| RNN Word (Question 4) | RNN Word with dropouts |
|  |  |

### 3.2.3 Part B Question 5

The time taken and the best test accuracy for each network is as follows

|  |  |  |
| --- | --- | --- |
| Network Type | Time Taken | Best Test Accuracy |
| CNN Character | 5:13 mins | 0.432 |
| CNN Character with Dropouts | 5:18 mins | 0.456 |
| RNN Character | 17:43 | 0.507 |
| RNN Character with Dropouts | 20.41 | 0.496 |
| CNN Word | 1:50 | 0.924 |
| CNN Word with Dropouts | 1:54 | 0.927 |
| RNN Word | 15:34 | 0.875 |
| RNN Word with Dropouts | 18:00 | 0.870 |

**Compare and comment on the accuracy of the networks**

|  |  |
| --- | --- |
| **Observation** | **Remarks** |
| Word ids perform better than Character ids | Word ids should perform better than character ids for text classification. This is because a character does not contain any sentiment by itself. For example, we cannot determine if the character “a” belongs to “people”, “company”, or “schools”. |
| CNN is faster than RNN | CNN usually outperforms RNN in terms of speed |
| CNN perform better than RNN for network that receives word ids | CNN is better for task that requires feature detection. (e.g. searching for angry terms like “sadness”, “abuses”)  RNN is better for task that requires semantic dependency (e.g. “dog” is an animal, but “hot dog” is a food. The semantic of “dog” is dependant on “hot”)  This shows that feature detection is more important for this dataset. |
| RNN perform better than CNN for network that receives char ids | Even though a single character does not give any sentiment (as mentioned above), a sequence of characters may have some meaning. |
| Some networks without dropout have a lot of spikes in plot of training loss | This is a sign of overfitting. In overfitting, a statistical model describes random error or noise instead of the underlying relationship |
| Adding dropouts to all network (except CNN) removes spikes from the plot of training loss | Dropout is a type of regularization; it reduces overfitting and improving the generalization for neural network |
| Adding dropouts to all network (except CNN) allows training loss to decrease consistently, but has no impact on testing accuracy | The model improves on the training data, but when it tries to predict new data from the test set, the accuracy did not improve. This shows that the model needs a larger training set. |
| Adding dropouts to CNN network that receives word ids will cause spikes in the plot of training loss | Unlike RNN, CNN does not have a lot of parameters in this setup. Adding a regulation (i.e. dropouts) to CNN will lower CNN’s small model capacity even further, thus lowering performance. |

### 3.2.4 Part B Question 6A

**Replace the GRU layer with vanilla RNN layer**

|  |  |
| --- | --- |
| Vanilla RNN layer Receiving Character IDs | Vanilla RNN layer Receiving Word IDs |
|  |  |

Both training loss and test accuracy did not improve after 500 epoch when the vanilla RNN layer was used. This is because of vanishing gradient.

**Replace the GRU layer with LSTM layer**

|  |  |
| --- | --- |
| LSTM layer Receiving Character IDs | LSTM layer Receiving WordIDs |
|  |  |

When receiving word ids, LSTM have fixed the vanishing gradient problem by introducing new gates, such as input and forget gates. Therefore, the graph shows better result. However, the training time took significantly longer than vanilla RNN.

When receiving character ids, the training loss and test accuracy starts to improve after epoch 100. However, even though the training loss gradually decreases with random spikes, the test accuracy did not improve. This is a sign of overfitting.

### 3.2.5 Part B Question 6B

**Increase the number of RNN layers to 2 layers**

|  |  |
| --- | --- |
| 2 GRU Layers Receiving Character IDs | 2 GRU Layers Receiving WordIDs |
|  |  |

When receiving word ids, GRU with 2 layers achieved the same test accuracy as GRU with 1 layer. This is because the training data is small; the network has over fitted, and is unable to generalise to new unseen data. Therefore, increasing the number of layers will not help to improve accuracy. It will, however, increase the training time.

When receiving character ids, exploding gradient can be seen at approximately epoch 380. To resolve exploding gradient, gradient clipping can be used (discussed in the next section)

# 

### 3.2.6 Part B Question 6C

**Add Gradient Clipping to RNN training (clipping threshold = 2)**

|  |  |
| --- | --- |
| GRU Layers Receiving Character IDs  (**without** Gradient Clipping) | GRU Layers Receiving Character IDs  (**with** Gradient Clipping) |
|  |  |

Gradient clipping is added to 1 GRU layer receiving character ids (done in question 3). The result is quite similar to each other, because there was no signs of vanishing or exploding gradient. Therefore, gradient clipping has no effect in this scenario.

|  |  |
| --- | --- |
| 2 GRU Layers Receiving Character IDs  (**without** Gradient Clipping) | 2 GRU Layers Receiving Character IDs  (**with** Gradient Clipping) |
|  |  |

Gradient clipping is added to GRU 2 layer receiving character ids (done in question 6B). This network had an exploding gradient problem, but is resolved with gradient clipping.