

**CZ4042 Neural Networks and Deep Learning**

*Project 2*

Yong Hao

U1722282A

Contents

[1. Introduction 3](#_Toc56192776)

[1.1 Part A 3](#_Toc56192777)

[1.2 Part B 3](#_Toc56192778)

[2. Method 3](#_Toc56192779)

[2.1. Part A 3](#_Toc56192780)

[2.1.1. Model Development 3](#_Toc56192781)

[2.1.2. Hyperparameter Optimization 4](#_Toc56192782)

[2.2. Part B 4](#_Toc56192783)

[2.2.1. Model Development 4](#_Toc56192784)

[2.1.2. Hyperparameter Optimization 5](#_Toc56192785)

[3. Experiments and Results 5](#_Toc56192786)

[3.1. Part A 5](#_Toc56192787)

[3.1.1. Question 1 Training the network 5](#_Toc56192788)

[3.1.2. Question 2 Use a grid search to find the optimal combination of the numbers of channels at the convolution layers. 11](#_Toc56192789)

[3.1.3. Question 3 Experiments with variations 13](#_Toc56192790)

[3.1.4. Question 4 Compare the accuracies of all the models from parts (1) - (3) and discuss their performances. 17](#_Toc56192791)

[3.2. Part B 18](#_Toc56192792)

[3.2.1. Question 1 Design a Character CNN Classifier that receives character ids and classifies the input 18](#_Toc56192793)

[3.2.2. Question 2 Design a Word CNN Classifier that receives word ids and classifies the input 20](#_Toc56192794)

[3.2.3. Question 3 Design a Character RNN Classifier that receives character ids and classify the input. 22](#_Toc56192795)

[3.2.4. Question 4 Design a word RNN classifier that receives word ids and classify the input. 24](#_Toc56192796)

[3.2.5. Question 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. 26](#_Toc56192797)

[3.2.6. Question 6 For RNN networks implemented in (3) and (4), perform the following experiments with 30](#_Toc56192798)

[the aim of improving performances, compare the accuracies and report your findings 30](#_Toc56192799)

[4. Conclusions 38](#_Toc56192800)

[4.2. Part A Conclusions 38](#_Toc56192801)

[4.2. Part B Conclusions 38](#_Toc56192802)

# 1. Introduction

The objective of this project is to experiment with Convolutional Neural Network and Recurrent Neural Network architectures and different sets of hyperparameters to determine the optimal structure of models to target specific problems.

## 1.1 Part A

In this project we are required to build convolutional neural networks with different structure and hyperparameters to classify images from CIFAR-10 dataset. The dataset contains RGB images of size 32 \* 32 and are categorized with 10 labels from 0 to 9. The training data contains 10,000 training samples and testing data contains 2,000 testing samples.

The default structure of the network consists an input layer, two pairs of convolution layer and max pooling layer which we will implement grid search to determine their optimal number of channels, one fully connected layer with no activation and one fully connected layer with Softmax activation. We will also explore the effects of different optimizers on the performance of the models.

## 1.2 Part B

In this project we are required to build convolutional neural networks and recurrent neural networks which takes characters and words as inputs respectively to classify texts from the dataset. The output layer of the model is a Softmax layer.

The dataset contains the first paragraphs collected from Wikipage entries and are categorized with different labels.

We will experiment the performances of the model on the word-level input and character-level input. We will also explore the effects of a dropout layer before the output layer and the effects of different network structure and optimizers on the performance of the networks. We have selected the accuracy, loss, and training time of a model as metrics to evaluate its performance.

# 2. Method

## 2.1. Part A

### 2.1.1. Model Development

We start with a default convolutional neural network model with the following architecture:

• An Input layer of 32x32x3 dimensions.

• A convolution layer 𝐶1 with 50 channels, window size 9x9, VALID padding, and ReLU activation.

• A max pooling layer 𝑆1 with a pooling window of size 2x2, stride = 2, and VALID padding.

• A convolution layer 𝐶2 with 60 channels, window size 5x5, VALID padding, and ReLU activation.

• A max pooling layer 𝑆2 with a pooling window of size 2x2, stride = 2, and VALID padding.

• A fully connected layer 𝐹3 of size 300 with no activation.

• A fully connected layer 𝐹4 of size 10 with Softmax activation.

The hyperparameters we will explore with the networks are:

1. The number of channels in convolution layer C1.
2. The number of channels in convolution layer C2.
3. Whether to apply dropout layers with drop rate of 50% to the fully connected layers.
4. Different optimizers like SGD, SGD with momentum (0.1), RMSProp, and Adam.

### 2.1.2. Hyperparameter Optimization

We will adopt the accuracy on the testing data set as the metric for determining the performance of the model.

We adopted grid search and limited the range of possible number of channels of the first convolution layer to [10, 30, 50, 70, 90] and the possible number of channels of the second convolution layer to [20, 40, 60, 80, 100], this will give us a combination of 25 models.

The performances of the models will be compared and analyzed in the conclusion section in the Experiments and Results section of Part A.

## 2.2. Part B

### 2.2.1. Model Development

In this project we develop four baseline models, which are Character CNN, Word CNN, Character RNN, Word RNN.

For character-level networks, we implement one-hot encoding for the input data for a total vocabulary size of 256. For word-level networks, we implement an embedding layer of size 20 before feeding the data to the network.

Architecture for Character CNN:

• A convolution layer 𝐶1 of 10 filters of window size 20x256, VALID padding, and ReLU neurons. A max pooling layer 𝑆1 with a pooling window of size 4x4, with stride = 2, and padding = 'SAME'.

• A convolution layer 𝐶2 of 10 filters of window size 20x1, VALID padding, and ReLU neurons. A max pooling layer 𝑆2 with a pooling window of size 4x4, with stride = 2 and padding = 'SAME'.

Architecture for Word CNN:

• A convolution layer 𝐶1 of 10 filters of window size 20x20, VALID padding, and ReLU neurons. A max pooling layer 𝑆1 with a pooling window of size 4x4, with stride = 2, and padding = 'SAME'.

• A convolution layer 𝐶2 of 10 filters of window size 20x1, VALID padding, and ReLU neurons. A max pooling layer 𝑆2 with a pooling window of size 4x4, with stride = 2 and padding = 'SAME'.

Architecture for Character RNN:

• A recurrent layer R1 with GRU cell and a hidden layer size of 20.

Architecture for Word RNN:

• A embedding layer of size 20 for data to pass before following layers.

• A recurrent layer R1 with GRU cell and a hidden layer size of 20.

### 2.1.2. Hyperparameter Optimization

We will adopt the accuracy on the testing data set and total training time as the metric for determining the performance of the model.

We will experiment with adding dropout layers with drop rate of 0.5 before the output layer to analyze its effect on the performance on the model.

We will also experiment with the types of RNN layers, the number of layers and hyperparameters like gradient clipping threshold for optimizers.

The performances of the models will be compared and analyzed in the conclusion section in the Experiments and Results section of Part B.

# 3. Experiments and Results

## 3.1. Part A

### 3.1.1. Question 1 Training the network

We developed the model using the following code:



The function takes in three parameters, **num\_ch\_c1**, **num\_ch\_c2**, and **use\_dropouts**. **num\_ch\_c1** and **num\_ch\_c2** indicates the number of channels in the convolution layers This will serve as a baseline for the following sections.

#### 3.1.1.(a)Plot the accuracies and loss of training and testing against training epoch

The accuracies and loss of the model against each training epoch is given below:

|  |  |
| --- | --- |
| Model Accuracy | Chart  Description automatically generated |
| Model Loss | Chart, line chart, histogram  Description automatically generated |

#### 3.1.1.(b) For the first two test images, plot the feature maps at both convolution layers

#### and pooling layers along with the test images.

We used the following code to display the images:



The after training the model, we used the following section of code to display the intermediate results:



The result of the first two images and their intermediate results after the convolution layers and polling layers are given below:

|  |  |
| --- | --- |
| Image 0 | Chart  Description automatically generated |
| Feature Map at C1 | Graphical user interface, chart  Description automatically generated |
| Feature Map at S1 | Graphical user interface, chart  Description automatically generated |
| Feature Map at C2 | A picture containing graphical user interface  Description automatically generated |
| Feature Map at S2 | A picture containing chart  Description automatically generated |

|  |  |
| --- | --- |
| Image1 | Chart, treemap chart  Description automatically generated |
| Feature Map at C1 | A picture containing clock  Description automatically generated |
| Feature Map at S1 | Graphical user interface, chart  Description automatically generated |
| Feature Map at C2 | A picture containing graphical user interface  Description automatically generated |
| Feature Map at S2 | A picture containing chart  Description automatically generated |

### 3.1.2. Question 2 Use a grid search to find the optimal combination of the numbers of channels at the convolution layers.

We applied the grid search to train the 25 models aforementioned in the method section and recorded their accuracy and loss as following:

|  |  |  |  |
| --- | --- | --- | --- |
| C1 | C2 | Accuracy | Loss |
| 10 | 20 | 0.4335 | 1.600266547203064 |
| 10 | 40 | 0.474 | 1.6105144748687743 |
| 10 | 60 | 0.447 | 1.722028130531311 |
| 10 | 80 | 0.395 | 2.060461317062378 |
| 10 | 100 | 0.4495 | 1.728692039489746 |
| 30 | 20 | 0.4625 | 1.571728699684143 |
| 30 | 40 | 0.404 | 1.9098822059631348 |
| 30 | 60 | 0.425 | 1.8802436389923096 |
| 30 | 80 | 0.3745 | 2.206295244216919 |
| 30 | 100 | 0.4145 | 2.1070687370300294 |
| 50 | 20 | 0.461 | 1.6028640966415406 |
| 50 | 40 | 0.4675 | 1.7147203922271728 |
| 50 | 60 | 0.4525 | 1.9518405380249024 |
| 50 | 80 | 0.4105 | 1.7616415233612062 |
| 50 | 100 | 0.448 | 1.96665834236145 |
| 70 | 20 | 0.474 | 1.5943922576904297 |
| 70 | 40 | 0.455 | 1.8243213157653808 |
| 70 | 60 | 0.3975 | 1.9165659065246583 |
| 70 | 80 | 0.455 | 1.9572435817718505 |
| 70 | 100 | 0.4455 | 1.9626891288757324 |
| 90 | 20 | 0.442 | 1.7560296392440795 |
| 90 | 40 | 0.4465 | 1.887927869796753 |
| 90 | 60 | 0.4475 | 2.029411937713623 |
| 90 | 80 | 0.443 | 2.12908487701416 |
| 90 | 100 | 0.3925 | 1.8592973976135254 |

By observing from the evaluation data, we can conclude that the model with 70 channels in the first convolution layer and 20 channels in the second convolution layer has the highest accuracy and lowest validation loss. We will use this setting as the baseline for the following questions.

The accuracies and loss of training and testing against epochs for the (70, 20) model are given below:

|  |  |
| --- | --- |
| Model Accuracy | Chart  Description automatically generated |
| Model Loss | Chart, histogram  Description automatically generated |

### 3.1.3. Question 3 Experiments with variations

We adopted the (70, 20) pair as the optimal model for this section.

The following variations are experimented:

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 (probability=0.5) to the two fully connected layers.

3.1.3.(a) adding the momentum term with momentum 𝛾 = 0.1

|  |  |
| --- | --- |
| Accuracy | Loss |
| 0.4355 | 1.7893178691864013 |

The plots for accuracies and loss of training and testing against training epochs are given below:

|  |  |
| --- | --- |
| Model Accuracy | Chart, histogram  Description automatically generated |
| Model Loss | Chart, line chart, histogram  Description automatically generated |

3.1.3.(b) using RMSProp algorithm for learning

|  |  |
| --- | --- |
| Accuracy | Loss |
| 0.417 | 17.560549850463868 |

|  |  |
| --- | --- |
| Model Accuracy | Chart  Description automatically generated |
| Model Loss | Chart, line chart  Description automatically generated |

3.1.3.(c) using Adam optimizer for learning

|  |  |
| --- | --- |
| Accuracy | Loss |
| 0.305 | 13.802277282714844 |

|  |  |
| --- | --- |
| Model Accuracy | Chart  Description automatically generated |
| Model Loss | Chart, line chart  Description automatically generated |

3.1.3.(d) adding dropout (probability=0.5) to the two fully connected layers

|  |  |
| --- | --- |
| Accuracy | Loss |
| 0.482 | 1.4474995489120484 |

|  |  |
| --- | --- |
| Model Accuracy | Chart, line chart  Description automatically generated |
| Model Loss | Chart, histogram  Description automatically generated |

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

We aggregated the performances from models developed from part (2) – part (3) to better observe the effects of hyperparameter optimization and different optimizers. The accuracies for models from part (1) are recorded in the section itself.

|  |  |
| --- | --- |
| Model Architecture | Accuracy |
| Gradient Descent (c1=70, c2=20, same for below) | 0.474 |
| Gradient Descent with momentum = 0.1 | 0.4355 |
| RMSProp Algorithm | 0.417 |
| Adam Algorithm | 0.305 |
| Adding Dropout layer with drop rate of 0.5 | 0.482 |

We can analyse the observations from the following perspectives:

* From the experiments we did in section (1) we can conclude that adding more channels to both convolution layers does not necessarily increase the accuracy of the models. The optimal settings of both hyperparameters still rely on experiments like grid search.
* From the accuracies we recorded above we can conclude that GD with the dropout layer achieved the highest accuracy of 48.2% followed by the vanilla GD model of 47.4%, while other variations including adding momentum, using RMSProp Algorithm, and using Adam Algorithm did not seem to improve the accuracy of the model significantly.
* Following the conclusion above, we can observe the models with RMSProp, and Adam algorithm converges significantly earlier than GD algorithm variations. Both algorithms converge in the first 200 epochs, while the continuation of training actually led to a decrease in testing accuracy. This is because overfitting has happened and the model, along with more training epochs, biases towards the training data which lead to a decrease in accuracy for testing data. To tackle this problem, early stopping can be applied to the training process and we could expect to witness a higher accuracy for these two algorithms.
* In this CNN architecture, momentum does not seem to pose an observable impact on the testing accuracy of the model.
* Adding dropouts to the CNN model generally result a higher accuracy.

## 3.2. Part B

### 3.2.1. Question 1 Design a Character CNN Classifier that receives character ids and classifies the input

We used the following code to build the required model:



The plots the entropy cost on the training data and the accuracy on the testing data against training epochs are given below:

|  |  |
| --- | --- |
| Training Cost | Chart, histogram  Description automatically generated |
| Testing Accuracy | Chart, scatter chart  Description automatically generated |

### 3.2.2. Question 2 Design a Word CNN Classifier that receives word ids and classifies the input

We used the following code to build the required model:



The plots the entropy cost on the training data and the accuracy on the testing data against training epochs are given below:

|  |  |
| --- | --- |
| Training Cost | Chart, histogram  Description automatically generated |
| Model Loss | Chart, line chart  Description automatically generated |

### 3.2.3. Question 3 Design a Character RNN Classifier that receives character ids and classify the input.

We used the following code to build the required model:



The plots the entropy cost on the training data and the accuracy on the testing data against training epochs are given below:

|  |  |
| --- | --- |
| Training Cost | Chart, histogram  Description automatically generated |
| Testing Accuracy | Chart  Description automatically generated |

### 3.2.4. Question 4 Design a word RNN classifier that receives word ids and classify the input.

We used the following code to build the required model:



The plots the entropy cost on the training data and the accuracy on the testing data against training epochs are given below:

|  |  |
| --- | --- |
| Training Cost | Chart  Description automatically generated |
| Testing Accuracy | Chart  Description automatically generated |

### 3.2.5. Question 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.

The plots for training loss and testing accuracies of each setting of model after adding the dropout layer is listed as following:

|  |  |
| --- | --- |
| Char CNN | |
| Model Accuracy | Chart  Description automatically generated |
| Model Loss | Chart, histogram  Description automatically generated |

|  |  |
| --- | --- |
| Word CNN | |
| Model Accuracy | Chart  Description automatically generated |
| Model Loss | Chart, histogram  Description automatically generated |

|  |  |
| --- | --- |
| Char RNN | |
| Model Accuracy | Chart  Description automatically generated |
| Model Loss | Chart, histogram  Description automatically generated |

|  |  |
| --- | --- |
| Word RNN | |
| Model Accuracy | Chart, line chart  Description automatically generated |
| Model Loss | Chart, histogram  Description automatically generated |

The performances of four neural network architectures are summarised as following:

|  |  |  |  |
| --- | --- | --- | --- |
| NN Architecture | Use Dropout | Testing Accuracy | Training Time in seconds |
| Char CNN | False | 0.668571412563324 | 330.4538435935974 |
| Char CNN | True | 0.631428599357605 | 304.46557354927063 |
| Word CNN | False | 0.6928571462631226 | 859.0286507606506 |
| Word CNN | True | 0.7628571391105652 | 932.8975763320923 |
| Char RNN | False | 0.7071428298950195 | 2606.2206382751465 |
| Char RNN | True | 0.6842857003211975 | 2793.671418428421 |
| Word RNN | False | 0.8557142615318298 | 3058.287944793701 |
| Word RNN | True | 0.8157142996788025 | 3085.323560476303 |

From the results above we can make the following observation:

* On the perspective of time, training a character-level Neural Network generally takes less time than training a word-level Neural Network for both CNN and RNN. Training RNNs generally takes significantly longer time than training CNNs. In the case of this project, the training time for RNN could be 4-10 times to the time taken for training a CNN. Additionally, adding a dropout layer does not seem to pose an observable impact on training time.
* On the perspective of accuracy, adding a dropout layer does not seem to pose an observable impact on enhancing the testing accuracy with only one exception, the Word CNN model. One explanation to this reduction is that the network size is relatively small compared to the dataset, thus regularization would lower the capacity of the model which results in a lower performance.

### 3.2.6. Question 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

In this section we explored the following variations with the char RNN and word RNN models built in Question (3) and Question (4):

1. (i) Vanilla RNN Layer/ (ii) LSTM Layer
2. Increase the number of RNN layers to 2. For the additional layer, we used the default GRU layer.
3. Gradient clipping with threshold = 2

The plots of training cost and testing accuracies against each epoch are given below:

|  |  |
| --- | --- |
| Char RNN with vanilla RNN layer | |
| Model Accuracy | Chart, line chart  Description automatically generated |
| Model Cost | Chart, histogram  Description automatically generated |

|  |  |
| --- | --- |
| Char RNN with vanilla LSTM layer | |
| Model Accuracy | Chart  Description automatically generated |
| Model Cost | Chart, histogram  Description automatically generated |

|  |  |
| --- | --- |
| Char RNN with 2 RNN layer | |
| Model Accuracy | Chart  Description automatically generated |
| Model Cost | Chart, line chart  Description automatically generated |

|  |  |
| --- | --- |
| Char RNN with clipping threshold = 2 | |
| Model Accuracy | Chart  Description automatically generated |
| Model Cost | Graphical user interface, chart, histogram  Description automatically generated |

|  |  |
| --- | --- |
| Word RNN with vanilla RNN layer | |
| Model Accuracy | Chart, line chart  Description automatically generated |
| Model Cost | Chart, histogram  Description automatically generated |

|  |  |
| --- | --- |
| Word RNN with vanilla LSTM layer | |
| Model Accuracy | Chart, line chart  Description automatically generated |
| Model Cost | A picture containing chart  Description automatically generated |

|  |  |
| --- | --- |
| Word RNN with 2 RNN layer | |
| Model Accuracy | A picture containing chart  Description automatically generated |
| Model Cost | Chart, line chart  Description automatically generated |

|  |  |
| --- | --- |
| Word RNN with clipping threshold = 2 | |
| Model Accuracy | Chart, line chart  Description automatically generated |
| Model Cost | Chart, line chart  Description automatically generated |

We aggregated the accuracies and training times of these variations together with the original RNN models:

|  |  |  |
| --- | --- | --- |
| Model | Testing Accuracy | Training Time in seconds |
| Original Char RNN | 0.7071428298950195 | 2606.2206382751465 |
| Char RNN with vanilla RNN layer | 0.10285714268684387 | 817.8206958770752 |
| Char RNN with LSTM layer | 0.6557142734527588 | 1693.6792585849762 |
| Char RNN with 2 RNN layers | 0.74 |  |
| Char RNN with clipping threshold = 2 | 0.70115656444550195 | 2991.314649581909 |
| Original Word RNN | 0.8557142615318298 | 3058.287944793701 |
| Word RNN with vanilla RNN layer | 0.07285714149475098 | 1212.7567870616913 |
| Word RNN with LSTM layer | 0.6757143139839172 | 2167.067813873291 |
| Word RNN with 2 RNN layers | 0.8899999856948853 | 6856.745208024979 |
| Word RNN with clipping threshold = 2 | 0.8285714387893677 | 2202.778781414032 |

From the recorded plots above and the accuracy and time of each model, we can make the following key observations:

1. For both Char RNN and Word RNN, applying the vanilla RNN layer SimpleRNN result in the most unstable performance (from the turbulation observed in the plots) and lowest accuracy (both around 10%). Whereas the training time was reduced significantly to around 1/3 of the original model’s training time.
2. Experiments with LSTM layers also produced worse performance compared to the original RNNs. We could still observe a reduction in training time when using LSTM. This observation together with observation 1 can deduce a general conclusion of the relationship between the accuracy and training time of a model – one can be improved at the cost of sacrificing the other.
3. Implementing 2-layers with an additional GRU layer significantly improved performance for both Char RNN and Word RNN models.
4. Implementing clipping thresholds with value 2 does not seem to pose an observable improvement to both models.

# 4. Conclusions

## 4.2. Part A Conclusions

We can conclude the following key observations from the experiments:

1. The number of channels of each convolution layer in a deep convolutional neural network has important impact on the accuracy of the model. However, the enhancement of accuracy is not linearly proportional to the number of channels, which means we cannot simply increase the number of channels. Empirical Experiments are needed to determine the optimal number of channels, as well as other hyperparameters.
2. The type of optimizer applied also has important impact on the accuracy of the model. Empirically experimenting different optimization algorithm can lead to enhancement of the performance of the model.
3. Adding dropout layer to the network generally can enhance the performance of the model by preventing the model from overfitting and produce more generalized result.
4. Other hyperparameters, for instance, learning rate and number of layers can also be empirically studies to enhance the performance of the model.

## 4.2. Part B Conclusions

We can conclude the following key observations from the experiments:

1. It can be observed that compared to CNN models, RNN models have better performance regarding sequential textual data that are related in the context, from the result that any one of the RNN models has a higher accuracy than one from CNN models.
2. It can be observed that compared to CNN models, RNN models generally takes significantly longer time for training.
3. RNN models with more layers generally has better performance than RNN models with fewer layers. This is due to more RNN layers can capture the precedent intermediate results more accurately.
4. The type of RNN layer implemented in the RNN model has important impact on the performance of the model. Among all RNN layers, GRU layer has the best performance, followed by LSTM layer, while SimpleRNN layer has significantly low accuracy compared to other types of layers.