HW 3 Solutions

Author:
Fateme Noorzad

StudentID:
810198271

Deep Learning and Applications
Spring 2021

Contents

Chapter 1

Introduction

In this homework, we interact with LSTM in the first problem and BiLSTM in the second one. The network and preprocessing in each problem are altered in various ways to check their effect on the loss, accuracy, and other metric's.

Chapter 2

Question 1

In this problem, given the flickr8K dataset, we are to caption the given images based on the architecture proposed in a problem.

In the following sections, the questions regarding the stated goal and dataset are elaborated.

2.1 Part A: Preprocessing

2.1.1 A Review on Dataset

For this problem, the flickr8K dataset is utilized, consisting of 8091 images as well as their corresponding captions. In this dataset, there exists 5 unique captions for each image, hence the total number of captions in this dataset is 40455.

2.1.2 Preprocessing of Captions

In order to make the captions suitable for the neural network, it is mandatory to represent them as a vector of numbers. To do so, "Vocabulary" class is implemented to numericalize sentences, and an embedding layer, embeds these vectors.

To do the stated procedure, first of all, the captions are read and any punctuation is removed from the sentences. Afterwards, 90% of data is selected as training dataset and the remaining is set as the test dataset.

2.1.3 Preprocessing of Images

The images are resized to 300*300, and then random cropped into 256*256. Afterwards, They are represented as tensors and normalized. Same as captions, they are separated into training and test dataset.

2.1.4 FlickrDataset

In this class, the location of images, the location of .csv file containing the captions, the desired transformation on them, as well as an instance of "Vocabulary" are given as inputs. If the mode is set as train, by the given vocabulary instance, the vocabulary is built and learnt based on the given captions. In the case of test mode, the trained and built vocabulary is used.

In the "Vocabulary class", with the help of "Field" and "nltk", the given sentence is tokenized and numericalized. Based on what has been given, the first token of each sentence has to be "<sos>" [1], while the end of the sentence is marked by "<eos>" [2] token.

[1]Start of Sentence

[2]End of Sentence

On vital point in the creation of these tokenized and numericalized representations, is that the length of the sentences need to be the same. Thus, "<pad>" token is added so that all captions have the same length. Pads are added with "MyCollate" class, implemented to satisfy the mentioned need.

2.1.5 Dataloader

For the training dataset, the minimum frequency of the repetition of vocabulary, is set as 5, the batch size as 256, and the number of workers as 2. On the other hand, for the test dataset, the built vocabulary of training dataset is used, while batch size is set as 3, and the number of workers as 2.

2.2 Part B: Neural Network

The network consists of three parts which in the following subsections, they are elaborated.

2.2.1 Encoder

The encoder is a CNN [3] which its core a pre-trained "ResNet 18". In the first part of problem, besides the last layer, all the others are freezed. The last layer is also substituded by a linear layer. Afterwards, ReLU is employed as the activation function, and a drop out layer with 0.5 probability is also used for regularization.

2.2.2 Decoder

The decoder is a RNN [4]. In this network, an embedding layer is present, responsible for embedding the input. Its output then goes through a drop out layer with the aim of regulizing. Afterwards, features are added to the embedded captions to make a sequence. The result goes through a LSTM layer. Finally, the output is created after a linear layer.

2.2.3 Connector

The connector, is the only part of the network which needs training. In addition to this fact, this part is also responsible for connecting the encoder and decoder part of the whole network. Moreover, in this part, as has been explained in the given problem, a captioning function is implemented to caption the test data based on the trained model.

2.3 Part C: Training the Network

With the embedding size set as 300, the hidden size as 256, the number of epochs as 30, the learning rate as $3 * 10^{-4}$, the criterion as cross entropy, the optimizer as Adam, the network is trained.

[3]Convolutional Neural Network
[4]Recurrent Neural Network

2.3.1 Freezing Layers

As was mentioned, besides the last layer, all the other layers of the ResNet 18 is freezed, while the connector is trained with the stated parameters. Figure 2.1 depicts the loss on training dataset.

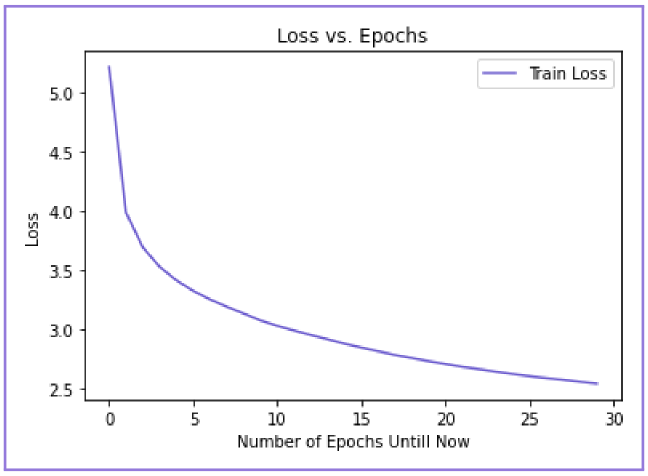
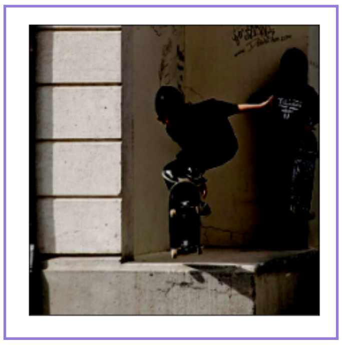
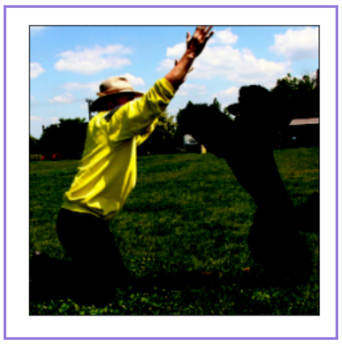


FIGURE 2.1: The loss on training data after 30 epochs while layers are freezed.

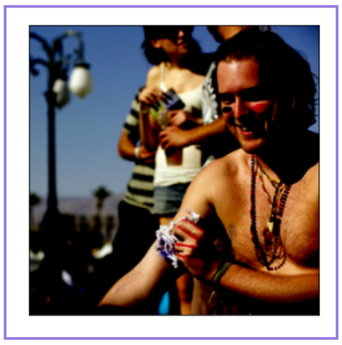
The below figures, represents the captions that the above network has predicted.



(A) <sos> a man in a red shirt is riding a bike on a skateboard <eos>



(B) <sos> a man in a black shirt and black shorts is standing in front of a large rock <eos>



(C) <sos> a woman in a black dress is holding a baby in a <unk> <eos>

FIGURE 2.2: The predicted captions on test images.

2.3.2 Unfreezing Layers

As was mentioned, in addition to the last layer, all the other layers of the ResNet 18 are unfreezed, while the connector is trained with the stated parameters. Figure 2.3 depicts the loss on training dataset.

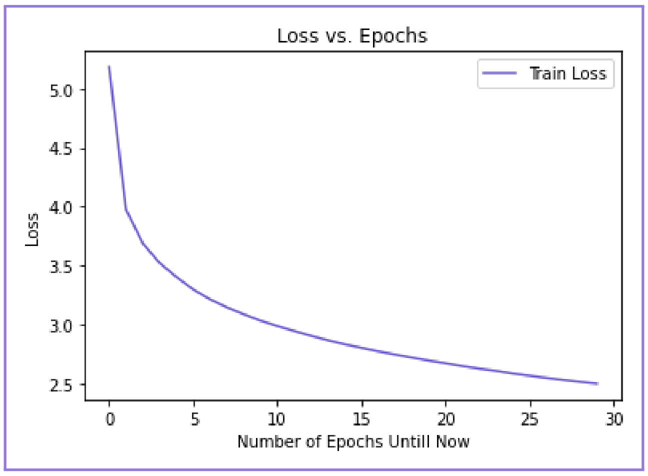
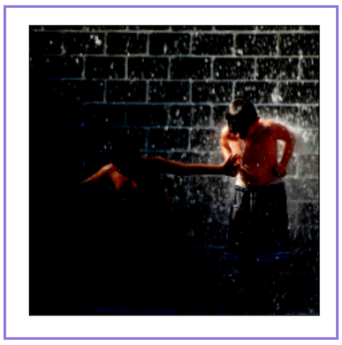
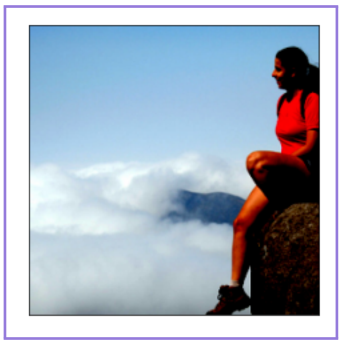


FIGURE 2.3: The loss on training data after 30 epochs while layers are unfreezed.

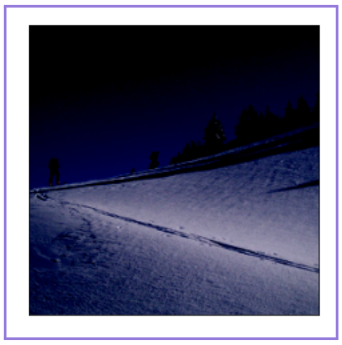
The below figures, represents the captions that the above network has predicted.



(A) <sos> two children are playing in the water <eos>



(B) <sos> a woman in a blue shirt and a backpack is standing on a rock <eos>



(C) <sos> two people are riding a bicycle on a dirt track <eos>

FIGURE 2.4: The predicted captions on test images.

Chapter 3

Question 2

In this problem, given the SemEval-2010 Task 8 dataset, we are to predict the relations between two specific entities based on the architecture proposed in a problem.

In the following sections, the questions regarding the stated goal and dataset are elaborated.

3.1 Part A: A Review on Dataset and Initial Preprocessing

For this problem, the SemEval-2010 Task 8 dataset is utilized, consisting of 8000 training data which only the first 7110 of them are used. In addition, there are 2717 data is utilized as test dataset. About 20% of training data is separated as used as validation data as well.

First of all, the sentences are read and separated by "\n" so that each line is separated from the other. Afterwards, the lines which start with "Comment" are removed, since they have no rule in training and testing the network. As the next step, sentences and relations are separated. In fact sentences are data and relations are the corresponding labels helping the network in the training process.

3.2 Part B: A)Weights are Learnt

3.2.1 Preprocessing of Sentneces

In order to make the sentences suitable for the neural network, it is mandatory to represent them as a vector of numbers. To do so, "Vocabulary" class is implemented to numericalize sentences, and an embedding layer, embeds these vectors.

To do the stated procedure, the desired length of them is selected. Afterwards, "\t", numbers, and entity indicators are removed from them. Now the sentences are ready for furthur usages.

3.2.2 Preprocessing of Relations

In order to make the relations suitable for the neural network, it is mandatory to represent them as a vector of numbers, meaning each class needs to be represented in numerical form. To do so, first of all, the desired length of them is selected. Then, the additional punctuations are removed as well. As the last step, each class is represented in numerical manner. Now the relations are ready for furthur usages.

3.2.3 SemEval-2010 Task 8

In this class, the location .csv file containing the sentences and relations preprocessed in the last sections, as well as an instance of "Vocabulary" class are given as inputs. If the mode is set as train, by the given vocabulary instance, the vocabulary is built and learnt based on the given sentences. In the case of test mode, the trained and built vocabulary is used.

In the "Vocabulary" class, with the help of "Field" and "nltk", the given sentence is tokenized and numericalized. Based on what has been given, the first token of each sentence has to be "<sos>" [1], while the end of the sentence is marked by "<eos>" [2] token.

On vital point in the creation of these tokenized and numericalized representations, is that the length of the sentences need to be the same. Thus, "<pad>" token is added so that all captions have the same length. Pads are added with "MyCollate" class, implemented to satisfy the mentioned need.

3.2.4 Dataloader

For the training dataset, the minimum frequency of the repetition of vocabulary, is set as 5, the batch size as 8, and the number of workers as 2. The validation dataloader follows the same parameters as well. On the other hand, for the test dataset, the built vocabulary of training dataset is used, while batch size is set as 1, and the number of workers as 2.

3.3 Part B: Neural Network

The network consists of following layers:

- **Embedding**: With the embedding dimension set as 100.
- **LSTM**: With its bi-directional flag set as true, hidden size set as 150, number of layers set as 2, and drop out value as 0.5.
- **Linear**: With the in feature size set as 2 * hidden size.
- **Dropout**: With its probability set s 0.7.
- **LogSoftmax**: Although in the problem it is stated to employ "nn.Softmax" layer, due to the usage of negative log-likelihood, instead of "nn.Softmax", "nn.LogSoftmax" is used.[3]

In the training procedure the last two indexes of the hidden output of LSTM is concatinated and fed into the linear layer.

3.4 Part C: Training the Network

With the embedding size set as 100, the hidden size as 150, the number of epochs as 70, the learning rate as $3 * 10^{-4}$, the criterion as negative log-likelihood, the optimizer

[1]Start of Sentence

[2]End of Sentence

[3]"nn.Softmax" + "nn.CrossEntroopyLoss" has almost the same result as "nn.LogSoftmax" + "nn.NLLLoss". Therefore, both of these two combinations were tested in this problem and the second combo had slightly better results. So the second one is used.

as Adam, the network is trained.

The result of the above conditions can be seen as below:

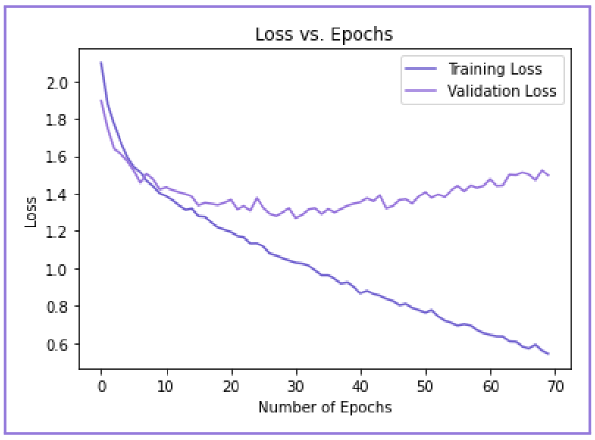


FIGURE 3.1: The loss on training and validation data after 70 epochs.

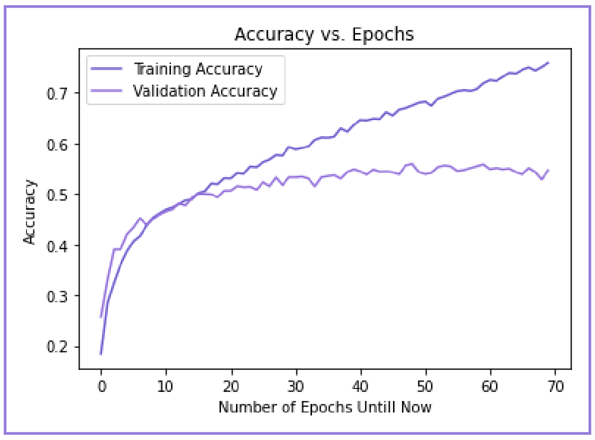


FIGURE 3.2: The accuracy on training and validation data after 70 epochs.

As the above figures suggest that after 20-30 epochs, the network is overfitted on the training data, hence the loss on the validation dataset is increasing and its accuracy is constant while the same two parameters on training data are acting in the opposite direction.

The results of recall, precision, and f1-score on test dataset is represented in table. 3.1

	Precision	Recall	F1-Score
Class 0	0.72	0.80	0.76
Class 1	0.00	0.00	0.00
Class 2	0.48	0.32	0.38
Class 3	0.71	0.84	0.77
Class 4	0.64	0.57	0.61
Class 5	0.71	0.90	0.80
Class 6	0.45	0.50	0.47
Class 7	0.40	0.73	0.52
Class 8	0.46	0.50	0.48

	Precision	Recall	F1-Score
Class 9	0.32	0.23	0.27

TABLE 3.1: Precision, recall, and f1-score of test data

3.5 Part B: B)Using GloVe

3.5.1 Preprocessing of Sentneces

The preprocessing of the sentences is the same as the last part with one slight difference. The difference lies in the creation of vocabulary in the "Vocabulary" class in which the vectors are set to glove.42B.300d.

3.5.2 Preprocessing of Relations

The preprocessing of the relations is the same as the last part.

3.5.3 SemEval-2010 Task 8

This sections is also the same as what has been carried out in the corresponding section in part B.

3.5.4 Dataloader

The exact same parameters are employed.

3.6 Part B: Neural Network

The network consists of following layers:

- **Embedding**: With the embedding dimension set as 300. The embedding vectors are copied as its weights and its weights are not trained.
- **LSTM**: With its bi-directional flag set as true, hidden size set as 150, number of layers set as 2, and drop out value as 0.5.
- **Linear**: With the in feature size set as 2 * hidden size.
- **Dropout**: With its probability set s 0.7.
- **LogSoftmax**: Although in the problem it is stated to employ "nn.Softmax" layer, due to the usage of negative log-likelihood, instead of "nn.Softmax", "nn.LogSoftmax" is used.[4]

3.7 Part C: Training the Network

The training procedure is same as the last section.
The result of the above conditions can be seen as below:

[4] "nn.Softmax" + "nn.CrossEntroopyLoss" has almost the same result as "nn.LogSoftmax" + "nn.NLLLoss". Therefore, both of these two combinations were tested in this problem and the second combo had slightly better results. So the second one is used.

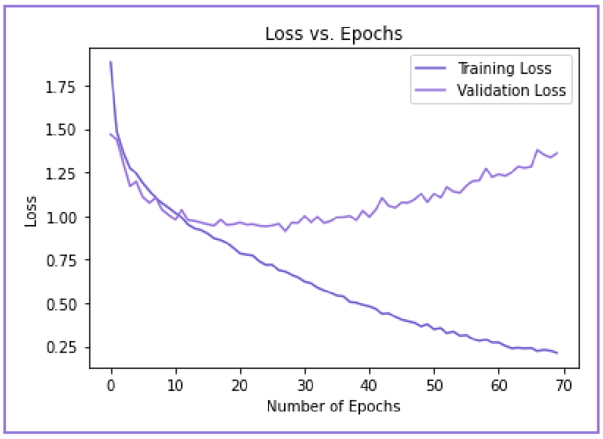


FIGURE 3.3: The loss on training and validation data after 70 epochs.

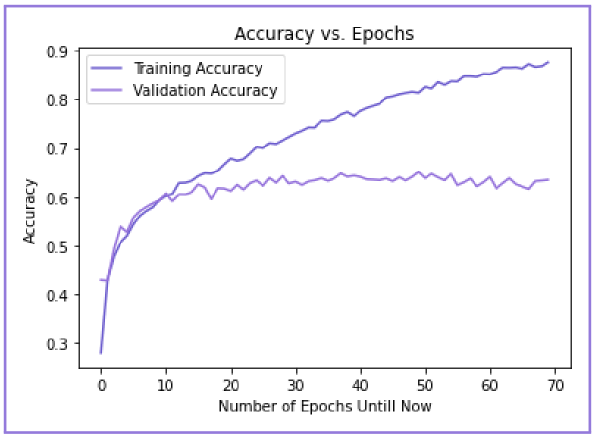


FIGURE 3.4: The accuracy on training and validation data after 70 epochs.

As the above figures suggest that after 20-30 epochs, the network is overfitted on the training data, hence the loss on the validation dataset is increasing and its accuracy is constant while the same two parameters on training data are acting in the opposite direction.

The results of recall, precision, and f1-score on test dataset is represented in table 3.2.

	Precision	**Recall**	**F1-Score**
Class 0	0.75	0.87	0.80
Class 1	0.00	0.00	0.00
Class 2	0.45	0.56	0.50
Class 3	0.71	0.85	0.78
Class 4	0.72	0.66	0.69
Class 5	0.76	0.85	0.80
Class 6	0.61	0.67	0.64
Class 7	0.58	0.68	0.62
Class 8	0.65	0.70	0.68
Class 9	0.32	0.25	0.28

TABLE 3.2: Precision, recall, and f1-score of test data

3.8 Part D: C)Using GloVe and Index of Entities

3.8.1 Preprocessing of Sentneces

In order to make the sentences suitable for the neural network, it is mandatory to represent them as a vector of numbers. To do so, "Vocabulary" class is implemented to numericalize sentences, and an embedding layer, embeds these vectors.

To do the stated procedure, the desired length of them is selected. Afterwards, "\t" and numbers are removed. The next step, is the difference of this section with the last two parts. In this section, before eliminating the entity indicators, the location of the are also stored in a dictionary. Then, the entity indicators are removed from sentences. Now the sentences are ready for furthur usages.

3.8.2 Preprocessing of Relations

This part, is the same as the last two parts.

3.8.3 SemEval-2010 Task 8

This part, is the same as the last two parts.

3.8.4 Dataloader

For the training dataset, the minimum frequency of the repetition of vocabulary, is set as 5, the batch size as 16, and the number of workers as 2. The validation dataloader follows the same parameters as well. On the other hand, for the test dataset, the built vocabulary of training dataset is used, while batch size is set as 1, and the number of workers as 2.

3.9 Part B: Neural Network

The network consists of following layers:

- **Embedding**: With the embedding dimension set as 300. The embedding vectors are copied as its weights and its weights are not trained.
- **LSTM**: With its bi-directional flag set as true, hidden size set as 150, number of layers set as 2, and drop out value as 0.5.
- **Max Pooling**: With the kernel size set as hidden size+1, it fixes the dimension of LSTM output.
- **Linear**: With the in feature size set as 2 * hidden size.
- **Dropout**: With its probability set s 0.7.
- **LogSoftmax**: Although in the problem it is stated to employ "nn.Softmax" layer, due to the usage of negative log-likelihood, instead of "nn.Softmax", "nn.LogSoftmax" is used.[5]

[5]"nn.Softmax" + "nn.CrossEntroopyLoss" has almost the same result as "nn.LogSoftmax" + "nn.NLLLoss". Therefore, both of these two combinations were tested in this problem and the second combo had slightly better results. So the second one is used.

In the training procedure, the index of the entities save in the preprocessing of sentences is used. Meaning that the values corresponding to these indexes are averages and the concatinated. Afterwards, they are fed into the linear and then softmax layer.

3.10 Part C: Training the Network

With the embedding size set as 300, the hidden size as 150, the number of epochs as 30, the learning rate as $3 * 10^{-4}$, the criterion as negative log-likelihood, the optimizer as Adam, the network is trained.

The result of the above conditions can be seen as below:

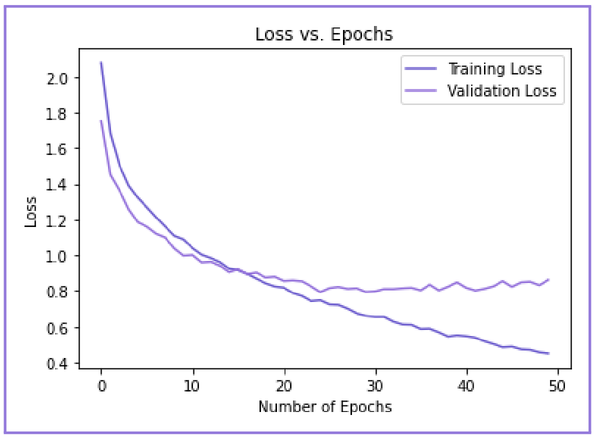


FIGURE 3.5: The loss on training and validation data after 70 epochs.

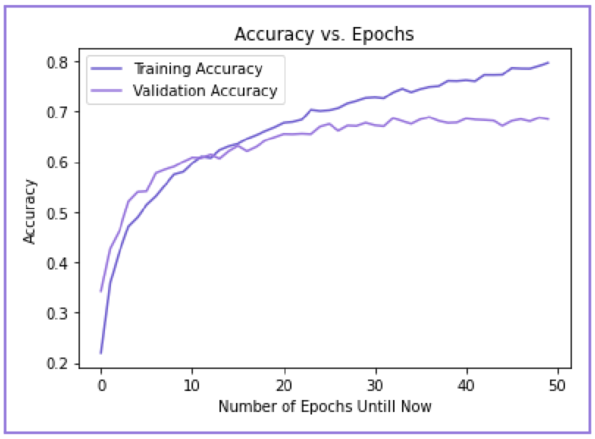


FIGURE 3.6: The accuracy on training and validation data after 50 epochs.

As the above figures suggest that after 40 epochs, the network is almost overfitted on the training data, hence the loss on the validation dataset is starting to increase and its accuracy is almost constant while the same two parameters on training data are acting in the opposite direction.

The results of recall, precision, and f1-score on test dataset is represented in table. 3.3

	Precision	Recall	F1-Score
Class 0	0.82	0.89	0.86
Class 1	0.00	0.00	0.00
Class 2	0.56	0.76	0.64
Class 3	0.70	0.86	0.77
Class 4	0.74	0.80	0.77
Class 5	0.83	0.89	0.86
Class 6	0.66	0.72	0.69
Class 7	0.71	0.85	0.77
Class 8	0.74	0.78	0.76
Class 9	0.47	0.33	0.39

TABLE 3.3: Precision, recall, and f1-score of test data

The confusion matrix is depicted as below:

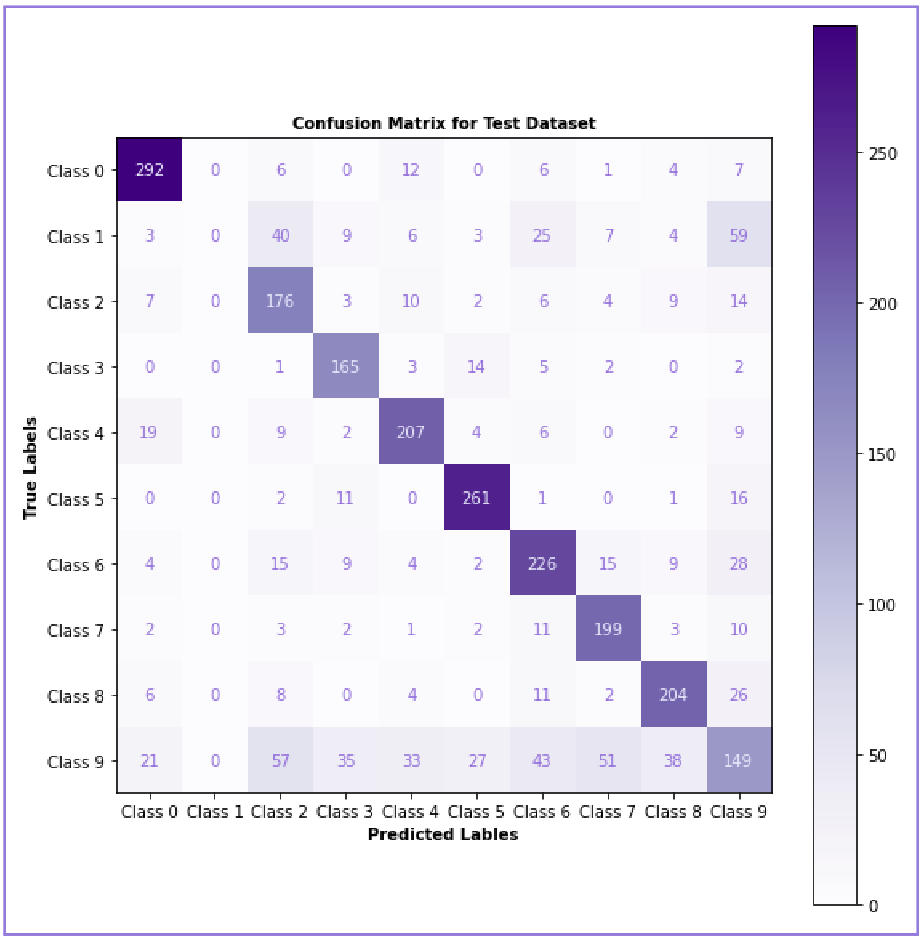


FIGURE 3.7: The confusion matrix on test data.

As can be seen, the results in the last part is slightly better than the first and second one since vectors have higher dimension as well as the corresponding index is given, making it having a better prediction tool in hand.

Chapter 4

Appendix: How to Run the Codes

4.1 Problem 1

All the codes can be found in the codes folder, Problem1. Note that first of all, the directory of files need to be set. Afterwards, running each notebook will show the results. Besides, my results can also be found in the codes.

4.2 Problem 2

All the codes can be found in the codes folder. Note that first of all, the directory of files need to be set. Afterwards,running each notebook will show the results. Besides, my results can also be found in the codes.

Chapter 5

References

- How to use TorchText for neural machine translation, plus hack to make it 5x faster
- Use torchtext to Load NLP Datasets — Part I
- Use torchtext to Load NLP Datasets — Part II
- Image captioning and build vocabulary
- Image captioning model
- Loader custom text
- SemEval 2010 Task 8 preprocessing
- Pytorch bi-directional LSTM example
- Understanding Bidirectional RNN in PyTorch
- Bi-directional LSTM output
- Simple relation extraction with a bi-LSTM model
- Getting entities