50.039 Theory and Practice of Deep Learning

Coding Homework #8: Recurrent Neural Networks

Keshigeyan 1002327

Task 1

The data was split into 80% for training set, 10% for validation set and 10% for test set. The accuracies for validation and test set for all the experiments are reported below. The hyper-parameters are reported below as well.

LSTM Architecture

To make the model better, I used an embedding layer (character) between the inputs and the hidden layer. Though this made the model better, it was computationally a bit expensive to run on a CPU.

	# Hidden Layers = 200	# Hidden Layers = 250	# Hidden Layers = 300
LSTM layers = 1	Experiment 1 Validation acc. =0.821 Test acc. =0.814	Experiment 2 Validation acc. = 0.802 Test acc. = 0.805	Experiment 3 Validation acc. = 0.795 Test acc. = 0.785
LSTM layers = 2	Experiment 4 Validation acc. = 0.796 Test acc. = 0.797	Experiment 5 Validation acc. = 0.802 Test acc. = 0.813	Experiment 6 Validation acc. = 0.794 Test acc. = 0.804

Batch-size	Learning rate	Weight decay	Epochs
32	0.01	0.01	10

Task 2

Using a batch size of 1 took a very long time to run. Hence, I ran the model for batch sizes of 10, 20, 30 and 40 to observe any general trends. But except for improvement in training accuracy, I was not able to observe any trends in validation or test accuracy. I suspect that since validation and test set were not split class-wise, it is difficult to draw concrete conclusions about the effect of batch size on model performance.

Experiment 1

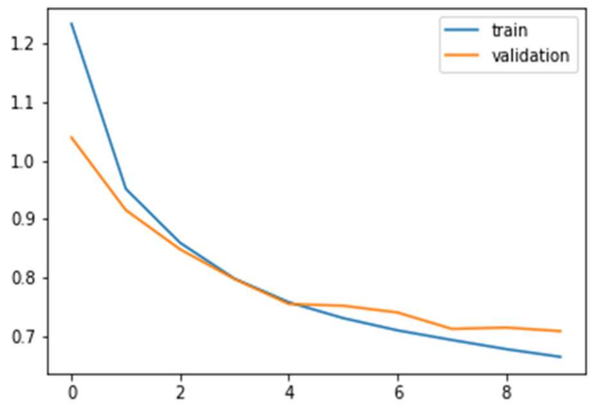
Batch-size	Learning rate	Weight decay	Epochs
10	0.01	0.01	10

50.039 Theory and Practice of Deep Learning

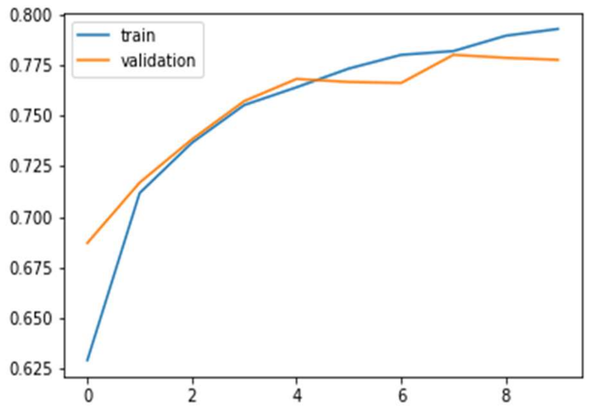
Coding Homework #8: Recurrent Neural Networks

Keshigeyan 1002327

Best Training Loss	Best Training Accuracy	Best Validation Loss	Best Validation Accuracy	Test Loss	Test Accuracy
0.6667	0.793	0.7078	0.778	0.6713	0.784



Training vs Validation Loss

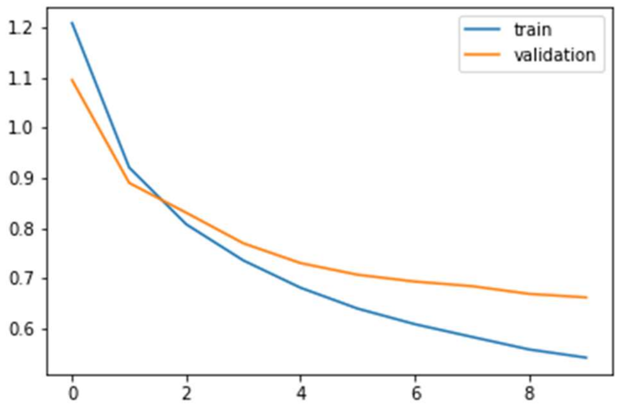


Training vs Validation Accuracy

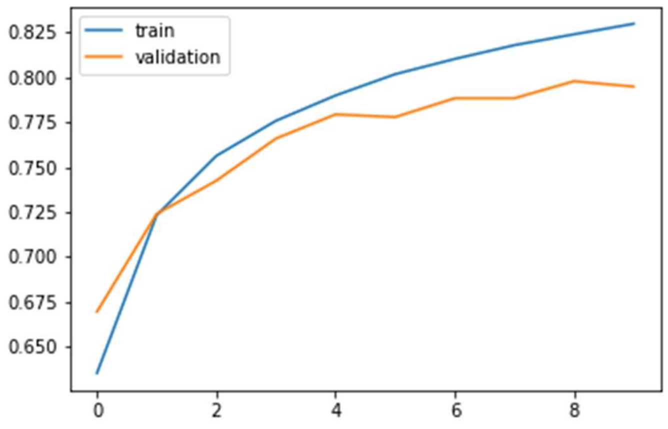
Experiment 2

Batch-size	Learning rate	Weight decay	Epochs
20	0.01	0.01	10

Best Training Loss	Best Training Accuracy	Best Validation Loss	Best Validation Accuracy	Test Loss	Test Accuracy
0.5423	0.830	0.6623	0.795	0.6139	0.807



Training vs Validation Loss



Training vs Validation Accuracy

50.039 Theory and Practice of Deep Learning

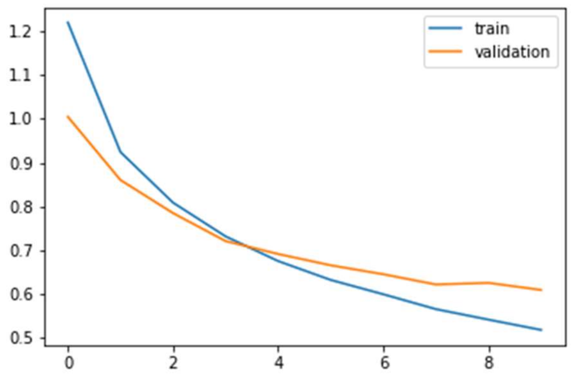
Coding Homework #8: Recurrent Neural Networks

Keshigeyan 1002327

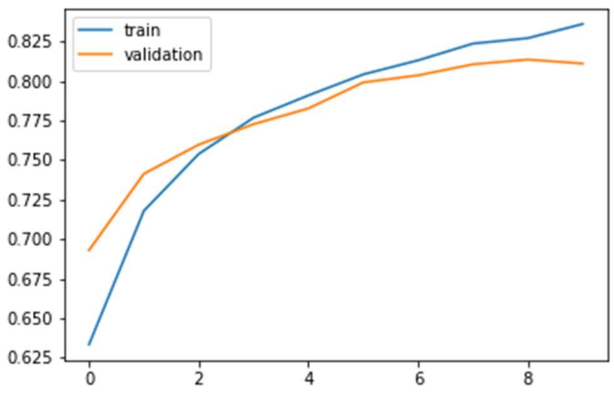
Experiment 3

Batch-size	Learning rate	Weight decay	Epochs
30	0.01	0.01	10

Best Training Loss	Best Training Accuracy	Best Validation Loss	Best Validation Accuracy	Test Loss	Test Accuracy
0.5186	0.836	0.6098	0.811	0.5936	0.820



Training vs Validation Loss

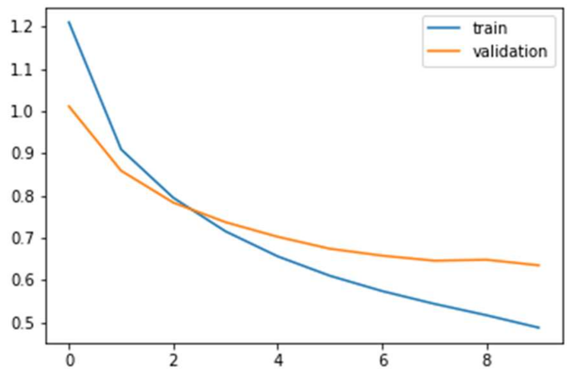


Training vs Validation Accuracy

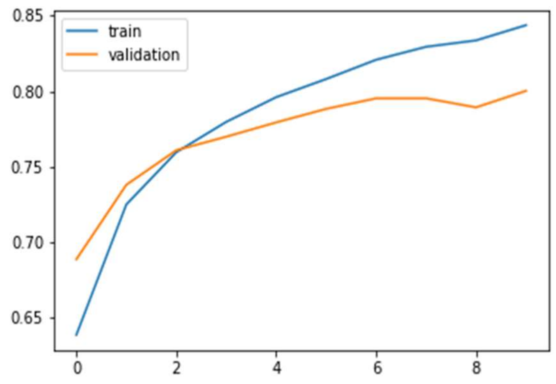
Experiment 4

Batch-size	Learning rate	Weight decay	Epochs
40	0.01	0.01	10

Best Training Loss	Best Training Accuracy	Best Validation Loss	Best Validation Accuracy	Test Loss	Test Accuracy
0.4871	0.844	0.6349	0.800	0.5729	0.810



Training vs Validation Loss



Training vs Validation Accuracy

50.039 Theory and Practice of Deep Learning

Coding Homework #8: Recurrent Neural Networks

Keshigeyan 1002327

How to reproduce the code?

1. Create the data folder containing the names and labels
2. Under main.py, edit the arguments for the train function in **line 148** to reproduce the code. An example run is shown below. All the arguments have default values as well.

```
if __name__ == '__main__':
  train(
        seed=100,
        data_dir= "./data/names/",
        hidden_dim = 200,
        num_layers = 1,
        learning_rate=0.01,
         weight_decay=0.01,
        batch_size=32,
        num_epochs = 10
  )
```