Ashay Sawant (ahsawant)

Sentence Level Text Classification using CNN and RNN

Dataset: IMDB Movie reviews sentiment classification[1]

IMDb is an online database of information related to world films, television programs, home videos and video games and internet streams. IMDb are very famous for authenticate movie ratings and reviews. Dataset contains 25000 movie reviews, labeled by sentiment (positive/negative). Movie reviews are pre-processed, and each review is encoded as a sequence of word indexes based on overall frequency in the dataset.

Task 1: Sentence Level Classification using CNN

- **Architecture and Implementation**
  I am using Convolutional Neural Network (CNN) to learn patterns in a sentence with the help of Convolution1D from Keras.
  My approach is to tokenize the sentence and limit its length to a pre-defined value (400). Using Conv1D with window size 3, learn a filter and perform max pooling operation. Then use Feed Forward Neural Network with Dense layer and ReLU as activation function. Finally, I used single unit output layer with sigmoid function to classify review as either positive or negative.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 400, 50)           250000
_________________________________________________________________
dropout_1 (Dropout)          (None, 400, 50)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 398, 250)          37750
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 250)               0
_________________________________________________________________
dense_1 (Dense)              (None, 250)               62750
_________________________________________________________________
dropout_2 (Dropout)          (None, 250)               0
_________________________________________________________________
activation_1 (Activation)    (None, 250)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 251
_________________________________________________________________
activation_2 (Activation)    (None, 1)                 0
=================================================================
Total params: 350,751
Trainable params: 350,751
Non-trainable params: 0
_________________________________________________________________
```

- **Training Procedure**
  The model is trained over 5 epochs.

- **Evaluation methods and Results**

  I have used "**binary_crossentropy**" as a loss function and "**adam**" optimizer is used to minimize the loss function.

```
Build model...
Train on 25000 samples, validate on 25000 samples
Epoch 1/5
25000/25000 [==============================] - 370s 15ms/step - loss: 0.4726 - acc: 0.7559 - val_loss: 0.3067 - val_acc: 0.8689
Epoch 2/5
25000/25000 [==============================] - 368s 15ms/step - loss: 0.2508 - acc: 0.8970 - val_loss: 0.2630 - val_acc: 0.8892
Epoch 3/5
25000/25000 [==============================] - 339s 14ms/step - loss: 0.1811 - acc: 0.9303 - val_loss: 0.2733 - val_acc: 0.8886
Epoch 4/5
25000/25000 [==============================] - 201s 8ms/step - loss: 0.1300 - acc: 0.9531 - val_loss: 0.2835 - val_acc: 0.8905
Epoch 5/5
25000/25000 [==============================] - 198s 8ms/step - loss: 0.0877 - acc: 0.9708 - val_loss: 0.3126 - val_acc: 0.8888
```

After 5 epochs, Training accuracy went up to 97.08% and Validation accuracy to 88.88% Training loss is 0.0877 and validation loss is 0.3126.

Task 2: Sentence Level Classification using RNN

- **Architecture and Implementation**

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 100, 50)	750000
lstm_1 (LSTM)	(None, 100, 100)	60400
lstm_2 (LSTM)	(None, 100, 50)	30200
lstm_3 (LSTM)	(None, 100, 20)	5680
lstm_4 (LSTM)	(None, 100, 10)	1240
global_max_pooling1d_1 (Glob	(None, 10)	0
dropout_1 (Dropout)	(None, 10)	0
dense_1 (Dense)	(None, 1)	11

Total params: 847,531
Trainable params: 847,531
Non-trainable params: 0

- **Training Procedure**

The model is trained over 5 epochs.

- **Evaluation methods and Results**

  I have used "**binary_crossentropy**" as a loss function and "**RMSProp**" optimizer is used to minimize the loss function.

```
Build model...
Train on 25000 samples, validate on 25000 samples
Epoch 1/5
25000/25000 [==============================] - 153s 6ms/step - loss: 0.5723 - acc: 0.7102 - val_loss: 0.4387 - val_acc: 0.8110
Epoch 2/5
25000/25000 [==============================] - 240s 10ms/step - loss: 0.4213 - acc: 0.8372 - val_loss: 0.3951 - val_acc: 0.8273
Epoch 3/5
25000/25000 [==============================] - 291s 12ms/step - loss: 0.3603 - acc: 0.8682 - val_loss: 0.4311 - val_acc: 0.8224
Epoch 4/5
25000/25000 [==============================] - 285s 11ms/step - loss: 0.3269 - acc: 0.8865 - val_loss: 0.4048 - val_acc: 0.8266
Epoch 5/5
25000/25000 [==============================] - 279s 11ms/step - loss: 0.3022 - acc: 0.8970 - val_loss: 0.4164 - val_acc: 0.8239
```

After 5 epochs, Training accuracy went up to 89.7% and Validation accuracy to 82.39%

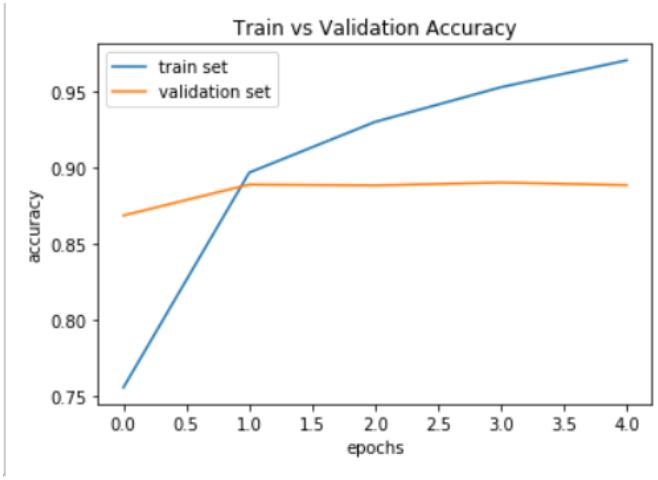
Training loss is 0.3022 and validation loss is 0.4164.

Training Complexity:

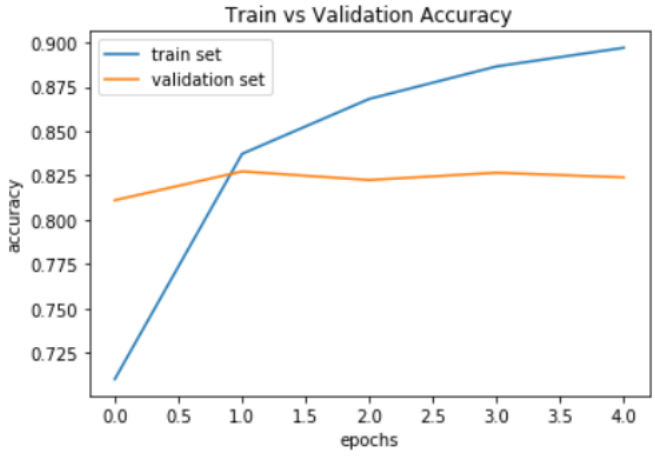
CNN has 350,751 number of trainable params, whereas RNN has 847,531 trainable params. Hence, RNN has more complex architecture.

Performance Results:

1. Accuracy:



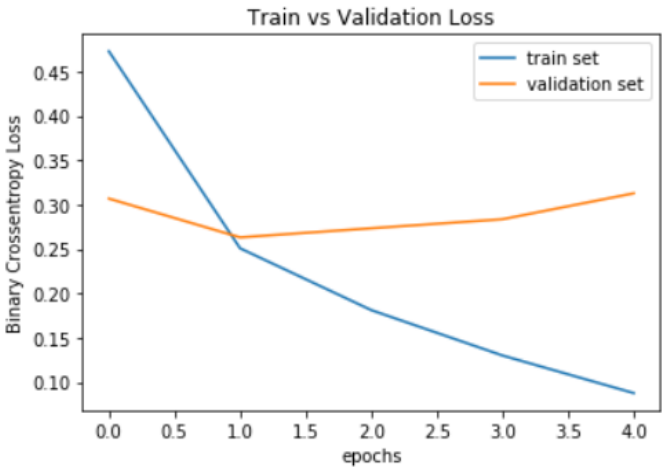
a. Accuracy for CNN model

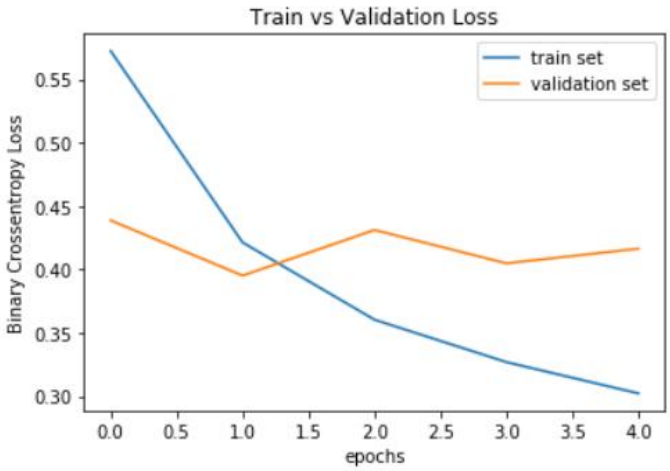


b. Accuracy for RNN model

As we can see in Accuracy plots for both CNN and RNN, CNN performs much better than RNN. Accuracy on validation set for CNN is 88.88%

2. Loss





As we can see in Loss plots for both CNN and RNN, CNN performs much better compared to RNN. Loss on validation set for CNN is 0.3126.

Reference:

[1] Datasets: Keras Documentation https://keras.io/datasets/