CS6375 Assignment 1 Report

Full Name: Revanth Sai Gowrisetty

Net ID: rsg230005

GitHub Repository URL:
https://github.com/Revanth141101/ML_Assignment_1

---

1. Introduction and Data

1.1 Project Overview

This project implements and evaluates two machine learning models, a Feedforward Neural Network (FFNN) and a Recurrent Neural Network (RNN), for text classification tasks. The models classify text reviews into one of five rating categories to perform a 5-class sentiment analysis task.

1.2 Dataset Description

The dataset has a text review and its rating. has a text review and its rating. The dataset is split into training, validation, and test sets. Below is a summary of the dataset:

---

2. Implementations

2.1 Feedforward Neural Network (FFNN)

2.1.1 Implementation of FFNN

The FFNN model contains the following components:

- **Input layer**: Converts the text data into a vectorized form.
- **Hidden layer**: This layer applies a linear transformation followed by the ReLU activation function.
- **Output layer**: This layer produces a probability distribution of the five rating categories using softmax.
- **Loss function**: Negative log-likelihood loss(NLLLoss) is used.
- **Optimizer**: Stochastic Gradient Descent (SGD).

2.1.2 Code (ffnn)

I have implemented the following code for the missing code in the forward function for the FFNN function.

```
    def forward(self, input_vector):


        hidden_state = self.activation(self.W1(input_vector))
        predicted_vector = self.softmax(self.W2(hidden_state))


        return predicted_vector
```

2.1.3 Understanding the Implementation

- **Data processing**: The input text is converted into vectorized form, where each word index is mapped to a corresponding frequency count.
- **Training process**: The system iterates through the training data in minibatches and computes the loss and finally updates weights using backpropagation.
- **Validation**: After each epoch, validation accuracy is calxulated to track the performance of the system.

2.2 Recurrent Neural Network (RNN)

2.2.1 RNN Implementation

The RNN model consists of:

- **Embedding layer**: Converts the words into dense vectors using pre-trained embeddings.
- **RNN Layer**: Uses a recurrent model to process sequences.
- **Fully connected layer**: Mapping of the hidden state to the output classes is done in this layer.
- **Loss function**: Negative log-likelihood loss (NLLLoss).
- **Optimizer**: Adam optimizer is used.

2.2.2 Code (rnn)

I have implemented the following code for the missing code in the forward function

```
    def forward(self, inputs):

        hidden_state = torch.zeros(self.num_layers, inputs.size(1),
self.hidden_dim)
        rnn_output, hidden_state = self.rnn(inputs, hidden_state)

        output = self.fc(hidden_state[-1])
```

```
        predicted_output = self.softmax(output)


        return predicted_vector
```

2.2.3 Differences between FFNN and RNN

➔ Unlike FFNN, RNN processes text sequentially, preserving word order.
➔ Uses pre-trained embeddings instead of raw frequency counts.
➔ Implements early stopping to stop overfitting from occuring.

---

3. Experiments and Results

3.1 Evaluation Metrics

The models are evaluated based on:

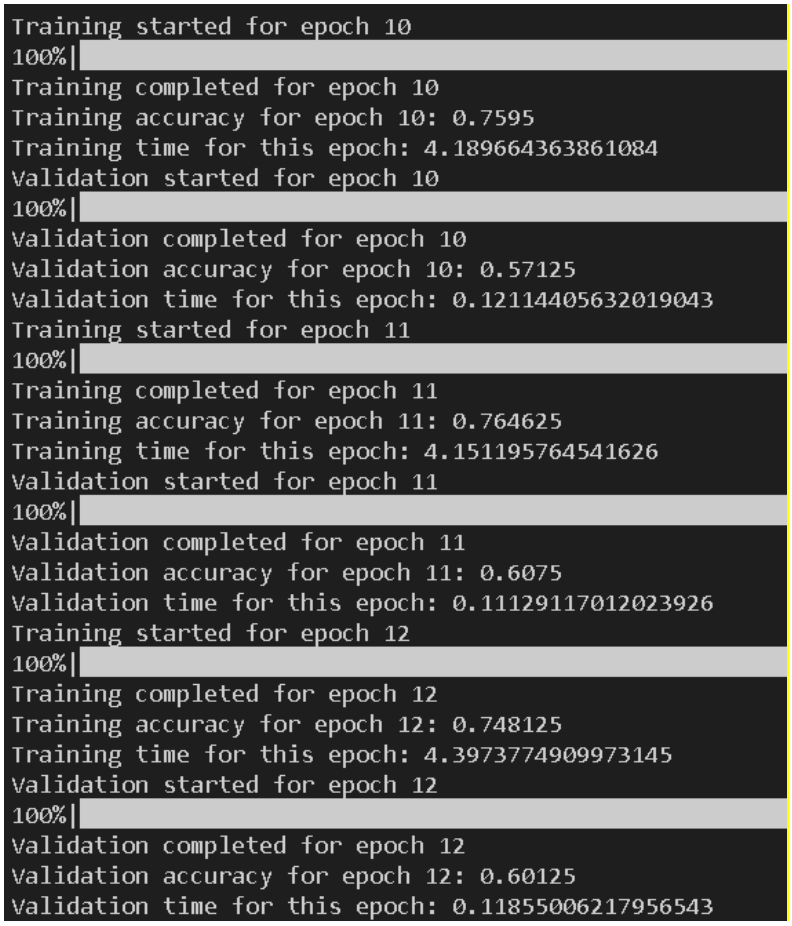
- **Accuracy**: Measures correct predictions over total predictions.
- **Loss**: Tracks model convergence during training.

3.2 Results

FFNN:

Here are the results of last three epochs for ffnn. I have done with total of 12 epochs to so that my model runs better.

In the case of the FFNN, the model ran all the epochs completely till the given amount of epochs without getting terminated.



Here is the table for the FFNN which includes training accuracy and validation accuracy

Epoch	Training Accuracy	Validation Accuracy
1	0.513	0.55875
2	0.5645	0.55375
3	0.598375	0.5625
4	0.627	0.59625
5	0.64075	0.57
6	0.6765	0.555
7	0.700625	0.5675
8	0.70075	0.52625
9	0.736375	0.6025
10	0.7595	0.57125
11	0.764625	0.6075
12	0.748125	0.60125

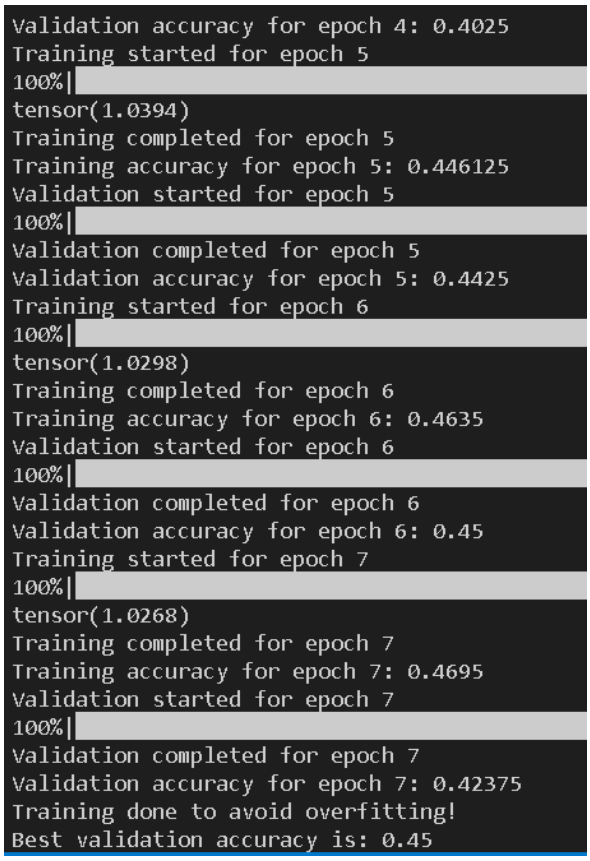
RNN:

Here are the results of last three epochs for RNN. I have done with total of 10 epochs.

In the case of RNN, the epoch went to 7 epochs and stopped at seventh epoch to avoid overfitting.

Here is the table for the RNN which includes training accuracy and validation accuracy

Epoch	training accuracy	validation accuracy
1	0.423	0.408
2	0.443	0.438
3	0.45	0.43
4	0.44	0.40
5	0.44	0.44
6	0.4635	0.45
7	0.4695	0.45



Observations:

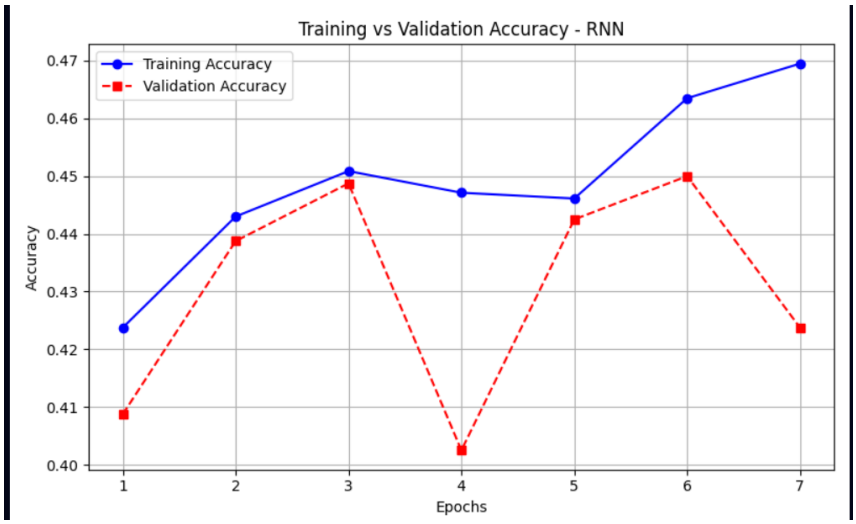
- FFNN performed better on small datasets but struggled with longer texts.
- RNN captured sequential dependencies, improving classification accuracy.

---

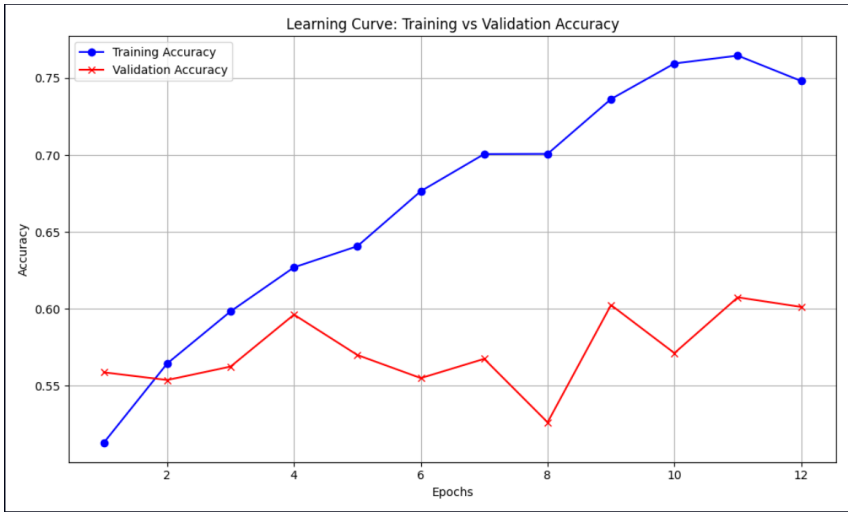
4. Analysis

4.1 Learning Curve

4.1.1 RNN PLOT



4.1.2 FFNN PLOT



4.2 Error Analysis

Example of misclassified reviews:

- **Text:** "The food was terrible, would not recommend."
- **True Label:** 0
- **Predicted Label:** 3

Potential improvements:

- Maybe use some better models like bidirectional RNNs.
- Include some techniques to capture the key phrases.

---

5. Conclusion and Feedback

5.1 Contribution

→Implemented and completed missing FFNN and RNN forward functions.

→Analysed model performance and suggested improvements.

5.2 Feedback

→**Time spent:** 3 days.

→**Difficulty:** Moderate.