A Report on

CS671: Assignment 6

Recurrent Neural Network(RNN) and Long Short Term Memory(LSTM)

Team Members(Group 04):

Ananya Angra(S22025)
Shilpa Chandra(S22004)
Pushkar Kumar(S22038)

Course Instructor

Dr. Dileep A. D.

IIT Mandi Academic Year 2022-2023

Contents

Chapter 1

Sequential Learning

1.1 Difference between Sequential learning and Feed-forward networks

In feedforward and convolutional neural networks the size of the input is always fixed. Further, each input to the network was independent of the previous or future inputs.
However, in many applications the input is not of a fixed size.Further successive inputs may not be independent of each other.
Sequence learning refers to the process of building models that can make predictions or take actions based on input data that has a temporal component. In other words, sequence learning models are designed to work with data that comes in a sequence, such as time-series data or text data.
The goal of sequence learning is often to predict the next element in a sequence, based on the previous elements. For example, a sequence learning model might be trained to predict the next word in a sentence, given the previous words.
Sequence learning models can be built using a variety of techniques, including recurrent neural networks (RNNs) and transformers. These models are capable of capturing long-term dependencies between elements in a sequence, making them well-suited to a wide range of tasks, such as language modeling, machine translation, and speech recognition.

1.2 Reccurent Neural Networks

A Recurrent Neural Network (RNN) is a type of neural network designed to handle sequential data unlike feedforward neural networks, which take fixed-sized inputs and produce fixed-sized outputs. At the core of an RNN is a loop that allows information to be passed from one step of the sequence to the next.

At each step, the network takes in an input and produces an output, as well as a hidden state that captures information about the previous inputs. The hidden state is then used as input for the next step, allowing the network to incorporate information from previous steps into its predictions. RNNs are particularly well-suited for tasks that involve processing sequential data, such as speech recognition, natural language processing, and time series analysis. However, they can be difficult to train on long sequences, as the gradients can become very small or very large, leading to problems with vanishing or exploding gradients.

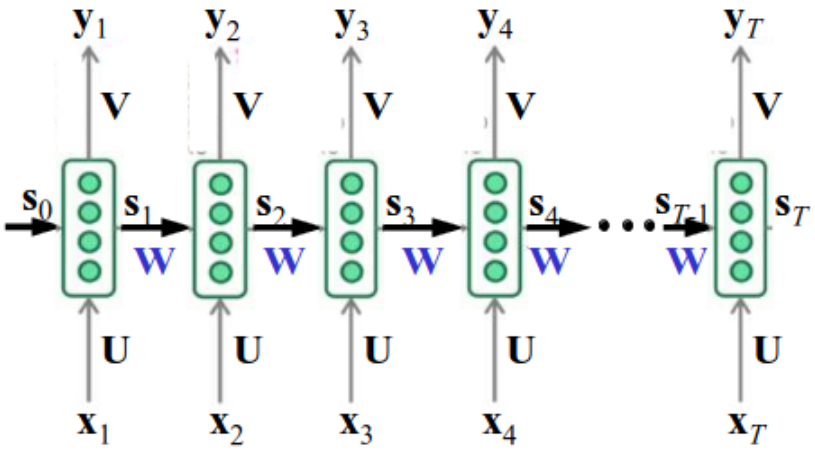


Figure 1.1: Neural Network Model for Sequence Learning Problem[1]

1.3 Training of RNN(Backpropagation through time)

Key steps in the training process for a Recurrent Neural Network (RNN):

- **Initialize the network parameters:**The weights and biases of the RNN are randomly initialized to small values.
- **Forward pass:**The input sequence is fed through the network, one time step at a time, and the network produces an output and a hidden state at each step.
- **Calculate the loss:**The output of the network is compared to the target output at each time step, and the loss is calculated using a loss function such as mean squared error or cross-entropy.
- **Backward pass:**The gradients of the loss with respect to the network parameters are calculated using backpropagation through time (BPTT), which involves propagating the gradients backwards through the network using the recurrent connections.
- **Update the parameters:** The gradients are used to update the network parameters using an optimization algorithm such as stochastic gradient descent (SGD). The learning rate and other hyperparameters may be adjusted during training to improve performance.
- **Repeat:**Steps 2-5 are repeated for a fixed number of epochs or until the training error converges to a minimum value.
- **Evaluate the model:** Once the training is complete, the performance of the model is evaluated on a separate validation or test set to assess its generalization performance.
- **Adjust the model:**Based on the evaluation results, the model may be adjusted or fine-tuned to improve its performance on the target task.

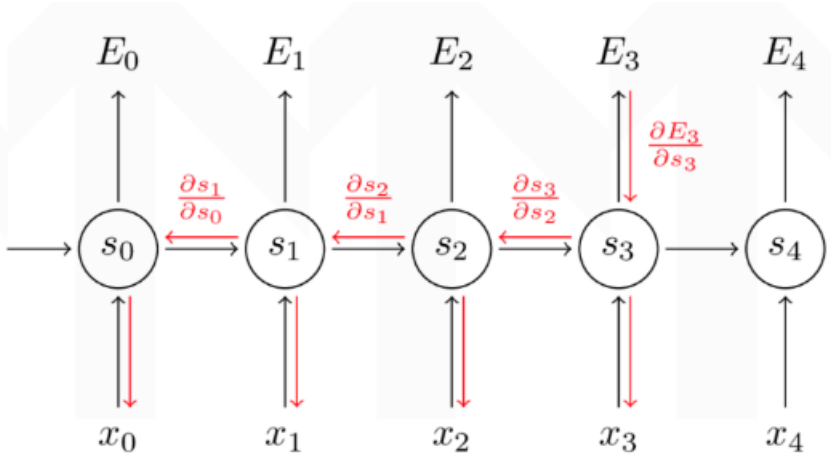


Figure 1.2: Backpropagation through time[2]

1.4 Long Short-Term Memory(LSTM)

- Long Short-Term Memory is a type of RNN architecture that is designed to better capture long-term dependencies in sequential data.
- LSTMs use a memory cell, which is a long-term memory unit that can store information over multiple time steps. The cell is controlled by gates, which are small neural networks that regulate the flow of information into and out of the cell.
- The gates use sigmoid activation functions to control the degree to which information is allowed to pass through them, and can selectively forget or remember information based on the current input and the contents of the memory cell.
- LSTMs are preferred over traditional RNNs because they are better able to deal with the vanishing and exploding gradient problems that can occur in traditional RNNs.
- The vanishing gradient problem occurs when the gradients become too small, making it difficult to update the network parameters effectively.
- The exploding gradient problem occurs when the gradients become too large, which can lead to unstable training and slow convergence.

- LSTMs are designed to address these problems by using a gating mechanism to selectively forget or remember information based on the current input and the contents of the memory cell.

1.5 Gated Recurrent Unit(GRU)

- Gated Recurrent Unit is a type of recurrent neural network that is similar to LSTM. Like LSTMs, GRUs are designed to address the vanishing gradient problem that can occur in traditional RNNs, which makes it difficult to learn long-term dependencies in sequential data.
- The main difference between LSTMs and GRUs is that GRUs have a simpler architecture, with fewer parameters to learn.
- GRUs use two gating mechanisms, a reset gate and an update gate, which are used to selectively forget or remember information from the previous time step.
- The reset gate in a GRU is used to decide how much of the previous hidden state should be used in computing the current hidden state. The update gate is used to decide how much of the current input should be used in computing the current hidden state.
- Because of their simpler architecture, GRUs are generally faster to train than LSTMs and require less memory to store the network parameters.
- However, LSTMs are still generally preferred over GRUs for tasks that require modeling long-term dependencies in sequential data, because LSTMs have been shown to be more effective at capturing long-term dependencies in the data.

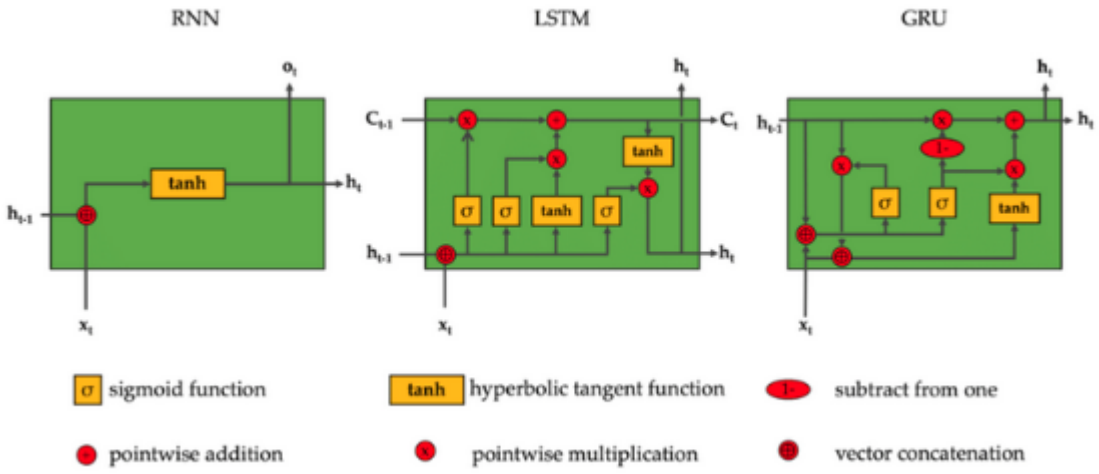


Figure 1.3: Computation-wise comparison of RNN, LSTM and GRU nodes[3]

1.6 Selective Write, Selective Read and Selective Forget

Selective Write refers to the process of selectively updating the contents of the memory cell based on the current input and the previous hidden state. In other words, the network determines which information is important to store in the memory cell and updates it accordingly.
Selective Read refers to the process of selectively retrieving information from the memory cell based on the current input and the previous hidden state. The network determines which information is relevant for the current prediction and retrieves it from the memory cell.
Selective Forget refers to the process of selectively removing information from the memory cell based on the current input and the previous hidden state. This is particularly important in situations where some information is no longer relevant and can even be harmful to the accuracy of predictions. The network determines which information should be forgotten and removes it from the memory cell

Chapter 2

Handwritten character dataset

2.1 About the Dataset

Handwritten character dataset:

- The dataset consists of subset of handwritten characters from Kannada/Telugu script. Each character is seen as sequence of 2-dimensional points (x and y coordinates) of one stroke of character (pen down to pen up)
- In order to ensure that all the characters are in same scale, x and y coordinates in each file should be normalized to the range of 0 to 1.
- Each data file is considered as one sample. Each file includes an array of elements.
- First element indicates the number of 2-d sequential points in that file.
- Second element onwards correspond to the 2-d sequential data points. They need to be considered in pairs as follows: first 2 numbers (i.e., 2nd and 3rd elements) are corresponding to first sequential point and so on.

Character (5 Classes):

5 images (after normalization) from each of the handwritten character classes

Figure 2.1: 5 images (after normalization) from each of the handwritten character classes

2.2 Architectures used for the dataset:

- **We have experimented with 5 architectures.**
- **Some hyper-parameters that were consistent throughout these 5 architectures were:**
  - **Adam optimizer with lr = 0.001 was used.**
  - **The difference between average error of successive epochs falling below a threshold of 10$\hat{}$(-4) is used as the stopping criteria.**

The following 5 architectures were used:

Architecture Number	Architecture Description
1	RNN/LSTM layer 1: 64 units Dropout: 0.2 RNN/LSTM layer 2: 32 units Dropout: 0.2 RNN/LSTM layer 3: 32 units Dropout: 0.2
2	RNN/LSTM layer 1: 128 units Dropout: 0.2 RNN/LSTM layer 2: 32 units Dropout: 0.2 RNN/LSTM layer 3: 32 units Dropout: 0.2
3	RNN/LSTM layer 1: 64 units Dropout: 0.2 RNN/LSTM layer 2: 32 units Dropout: 0.2
4	RNN/LSTM layer 1: 64 units Dropout: 0.2 RNN/LSTM layer 2: 64 units Dropout: 0.2
5	RNN/LSTM layer 1: 32 units Dropout: 0.2 RNN/LSTM layer 2: 32 units Dropout: 0.2 RNN/LSTM layer 3: 32 units Dropout: 0.2

Figure 2.2: Architectures for Handwritten character dataset

- **The input to each of the architectures was (178, 2) with mask values 2 as padding was done on the dataset with value = 2. The max sequence length was 178 as a result all the other sequence length were padded upto that length with 2. The values of the dataset after padding were normlaized within 0 to 1(other than values (2,2).**
- **The output of the architectures was Dense layer of size 5 along with softmax activation value.**

2.3 Results with Recurrent Neural Networks

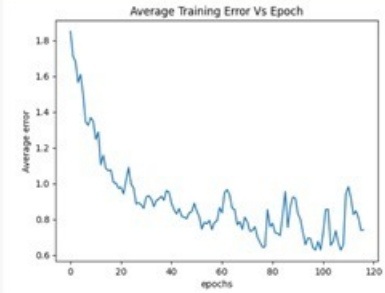
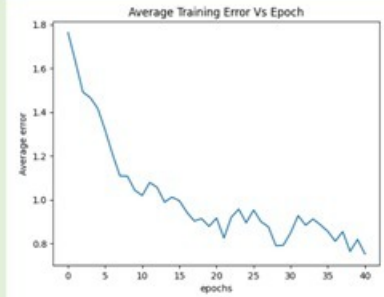
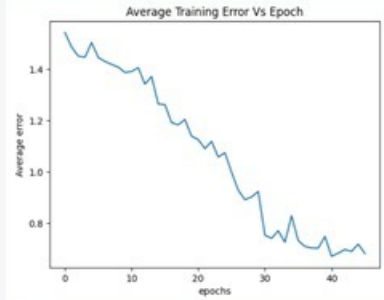
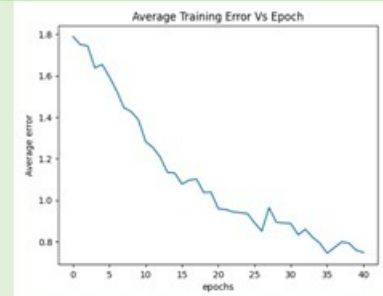
Architecture	Training error (y-axis) vs. epochs (x-axis)
1	
2	
3	
4	
5	

Figure 2.3: training error (y-axis) vs. epochs (x-axis) for all 5 architectures

Architecture	Training Accuracy	Test Accuracy
1	0.7536	0.7900
2	0.6638	0.7900
3	0.7072	0.6899
4	0.7304	0.7500
5	0.6985	0.7501

Figure 2.4: Training and testing accuracies with each of the 5 architectures

Architecture 1 gave the best testing and training accuracy out of the other 4. And is our best architecture. The confusion matrix is as follows:

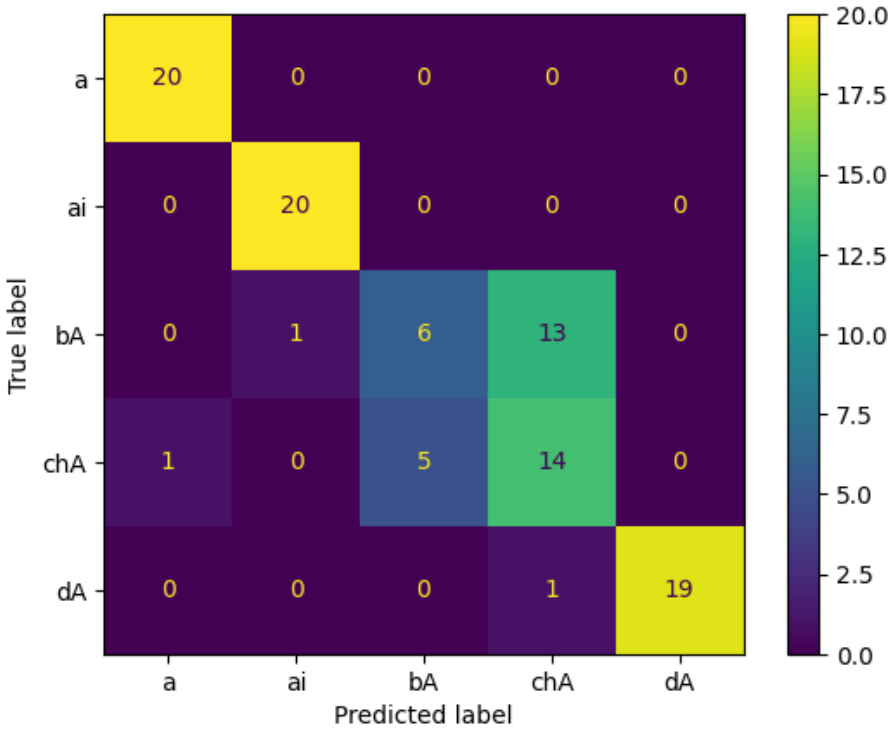


Figure 2.5: Confusion Matrix of Architecture 1

2.4 Results with long short-term memory networks(LSTM)

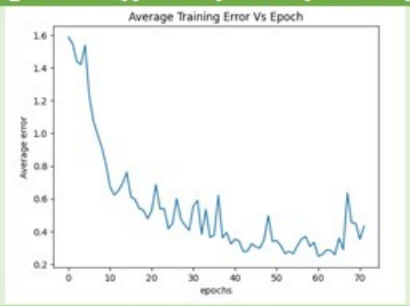
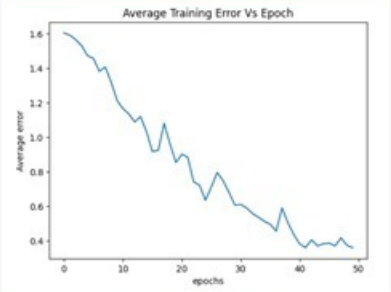
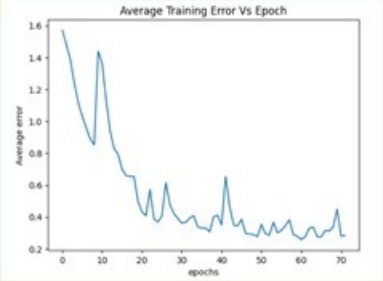
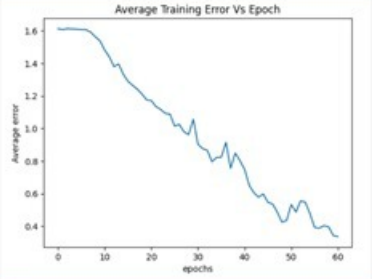
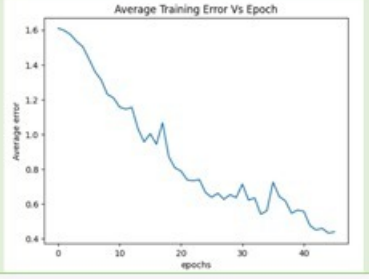
Architecture	Training error (y-axis) vs. epochs (x-axis)
1	
2	
3	
4	
5	

Figure 2.6: training error (y-axis) vs. epochs (x-axis) for all 5 architectures

Architecture	Training Accuracy	Test Accuracy
1	0.9217	0.8799
2	0.8869	0.8600
3	0.8956	0.8399
4	0.8115	0.8899
5	0.7971	0.8399

Figure 2.7: Training and testing accuracies with each of the 5 architectures

Architecture 4 gave the best testing and training accuracy out of the other 4. And is our best architecture. The confusion matrix is as follows:

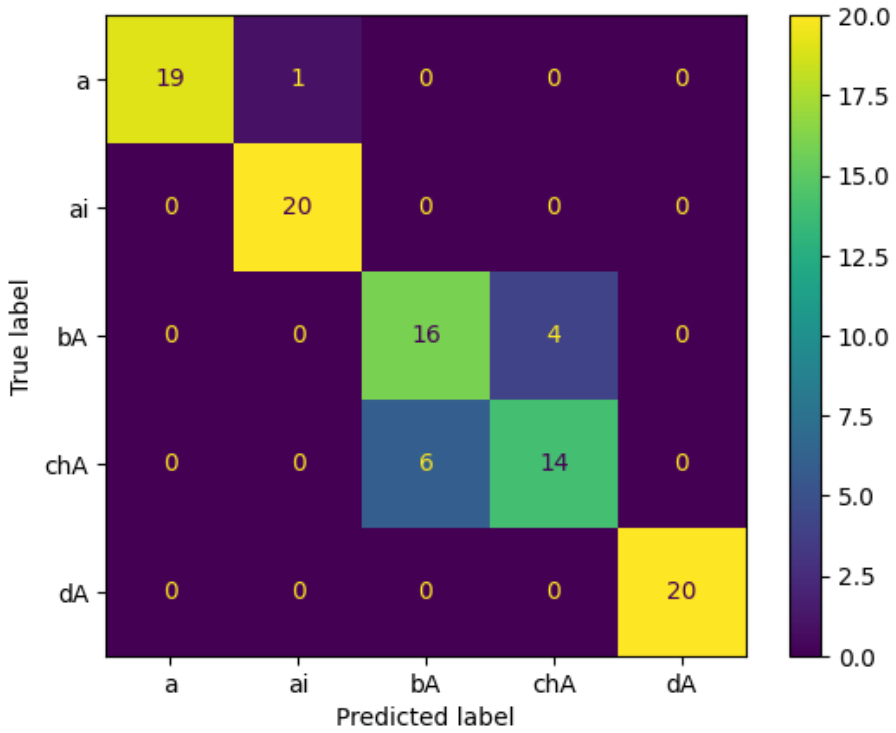


Figure 2.8: Confusion Matrix of Architecture 4

- **From the above it can be seen that LSTM performs better as compared to RNN.**
- **Also LSTM is more efficient in comparing in between 3rd and 4th class. The characters of 3rd and 4th class are almost similar**

in shape so the models tends to confuse between the 2 (as seen in Fig 2.1).

- **The data for question 1 is very less so taking large number of parameters makes the model ovefit to the data , hence sometimes the model gives very high accuracy on the train data and low on test data.**
- **Sometimes the model has better test accuracy as compared to the train accuracy. This maybe due to regularization done to avoid overfitting on training data. These methods can sometimes lead to lower training accuracy, but can improve generalization on unseen data.**

Chapter 3

Consonant Vowel (CV) segment dataset

3.1 About the Dataset

Consonant Vowel (CV) segment dataset:

- **The dataset consists of subset of CV segments from a conversational speech data spoken in Hindi language**
- **The 39-dimensional Mel frequency cepstral coefficient (MFCC) features extracted frame-wise from utterances of a particular CV segment uttered by multiple people are provided.**
- **Each data file is considered as one sample. Each row in a data file indicates one 39-dimensional MFCC feature vector. The number of such feature vectors (rows) depend on the duration of speech segment.**
- **We are given with 5 classes: hI, ne, ni, nii, pa**

The spectogram of one sample of each class (using 39-dimensional Mel frequency cepstral coefficient (MFCC) features) are plotted below:

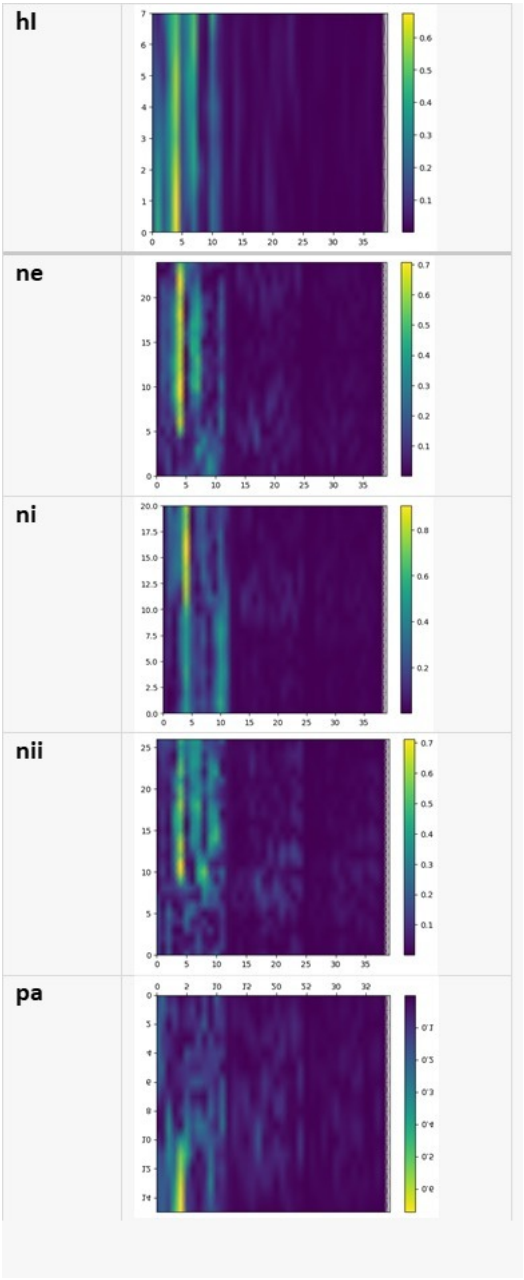


Figure 3.1: spectogram of one sample of each class(using 39-dimensional Mel frequency cepstral coefficient (MFCC) features)

3.2 Architectures used for the dataset:

- **We have experimented with 5 architectures.**
- **Some hyper-parameters that were consistent throughout these 5 architectures were:**
  - **Adam optimizer with lr = 0.001 was used.**
  - **The difference between average error of successive epochs falling below a threshold of 0.0001 is used as the stopping criteria.**

The following 5 architectures were used:

Architecture Number	Architecture Description
1	RNN/LSTM layer 1: 256 units Dropout: 0.2
2	RNN/LSTM layer 1: 256 units Dropout: 0.2 RNN/LSTM layer 2: 256 units Dropout: 0.2
3	RNN/LSTM layer 1: 128 units Dropout: 0.3
4	RNN/LSTM layer 1: 64 units RNN/LSTM layer 2: 64 units
5	RNN/LSTM layer 1: 32 units Dropout: 0.2 RNN/LSTM layer 2: 32 units Dropout: 0.2

Figure 3.2: Architectures for Consonant Vowel (CV) segment dataset

- **The input to each of the architectures was (64, 39) with mask values -1 as padding was done on the dataset with value = -1. The max sequence length was 64 as a result all the other sequence length were padded upto that length with -1.**
- **The output of the architectures was Dense layer of size 5 along with softmax activation value.**

3.3 Results with Recurrent Neural Networks

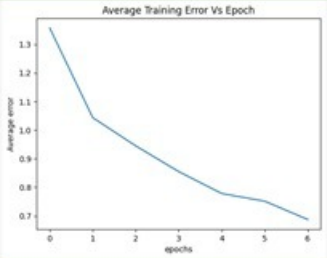
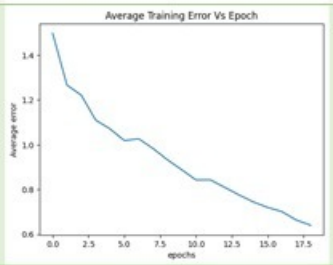
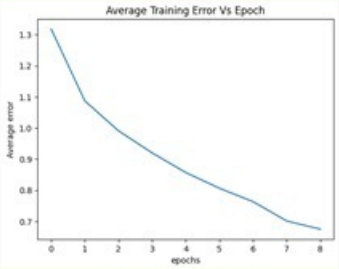
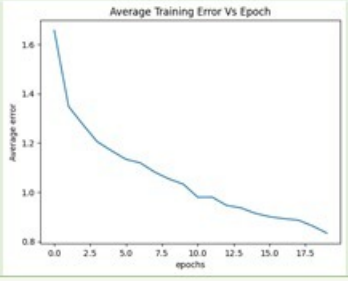
Architecture	Training error (y-axis) vs. epochs (x-axis)
1	
2	
3	
4	
5	

Figure 3.3: training error (y-axis) vs. epochs (x-axis) for all 5 architectures

Architecture	Training Accuracy	Test Accuracy
1	0.8346	0.6682
2	0.8007	0.6795
3	0.8108	0.6907
4	0.7233	0.6930
5	0.7182	0.6749

Figure 3.4: Training and testing accuracies with each of the 5 architectures

Architecture 4 gave the best testing and training accuracy out of the other 4. And is our best architecture. The confusion matrix is as follows:

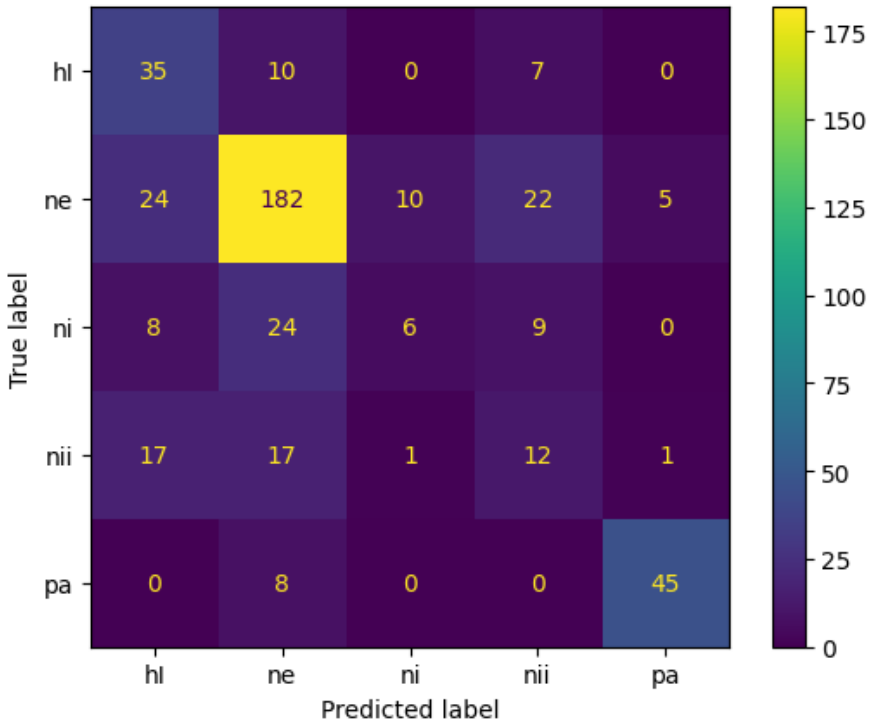


Figure 3.5: Confusion Matrix of Architecture 4

- **the segment 'pa' is classfied the best as it hardly overlaps with other 5 segments.**
- **The segment 'ni' is missclassified as 'ne' as it has similarities with it.**

- **Architecture 4 gives best results as two layer RNN with 64 units each is optimum considering the size of our dataset. Also dropouts are used for regularization**

3.4 Results with long short-term memory networks(LSTM)

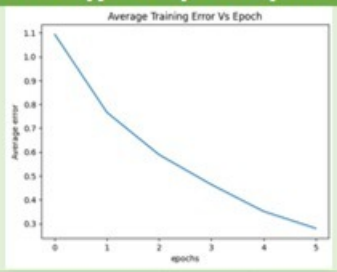
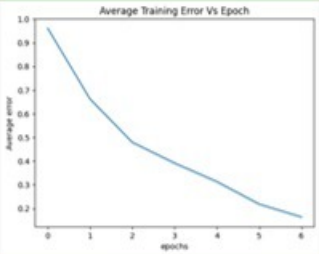
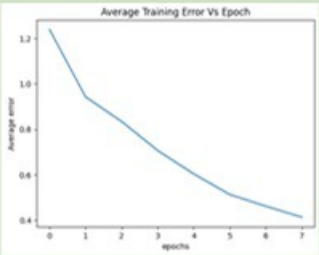
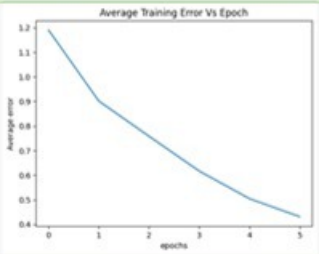
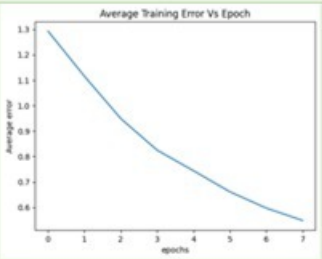
Architecture	Training error (y-axis) vs. epochs (x-axis)
1	
2	
3	
4	
5	

Figure 3.6: training error (y-axis) vs. epochs (x-axis) for all 5 architectures

Architecture	Training Accuracy	Test Accuracy
1	0.8922	0.7878
2	0.9610	0.7991
3	0.9125	0.7743
4	0.8684	0.7878
5	0.8357	0.7336

Figure 3.7: Training and testing accuracies with each of the 5 architectures

Architecture 2 gave the best testing and training accuracy out of the other 4. And is our best architecture. The confusion matrix is as follows:

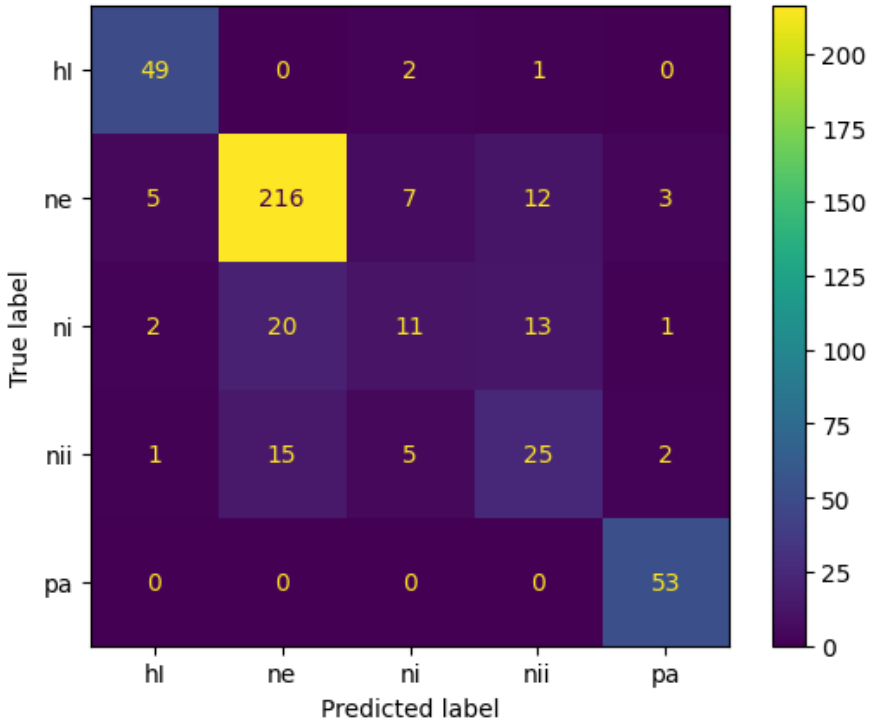


Figure 3.8: Confusion Matrix of Architecture 2

- **From the above it can be seen that LSTM performs better as compared to RNN.**
- **the segment 'hI' is classfied the better than that of RNN.**

- **Here segment 'ni' is less missclassified as 'ne' than it was in RNN.**
- **Architecture 2 gives best results with two layer RNN with 256 units each. This enhances the perfomance of LSTM even further.**

References

[1] **Lecture notes of CS671 Feb 2023-24 by Dr.Dileep A D**

[2] **https://zhangruochi.com/BackPropagation-through-time/2019/10/12/**

[3] **https://www.researchgate.net/figure/Computation-wise-comparison-of-RNN-LSTM-and-GRU-nodes_fig4_343462095**