Speech Recognition Using Recurrent Neural Networks (RNN)

Samudrala Hareesh-621243

Objective

The objective of this project is to implement a Recurrent Neural Network (RNN) for speech recognition. The model will process audio data to transcribe speech into text, utilizing the temporal dynamics inherent in the audio signal.

Theory

Recurrent Neural Networks (RNNs) are specialized neural networks designed to process sequential data, making them suitable for tasks involving time-series or natural language processing. The key components of RNNs include:

- **Recurrent Layers**: These layers maintain hidden states across time steps, allowing the model to remember past inputs and capture dependencies. RNNs can use their internal state (memory) to process sequences of inputs.
- **Long Short-Term Memory (LSTM) Cells**: A type of RNN designed to handle vanishing gradient problems, allowing better long-term memory retention.
- **Activation Functions**: Non-linear activation functions such as ReLU or tanh are used to introduce complexity in the learned patterns.
- **Loss Function**: The cross-entropy loss function measures the discrepancy between predicted and true sequences, ideal for speech transcription tasks.
- **Optimization**: The Adam optimizer is used to efficiently update the network weights based on calculated gradients.

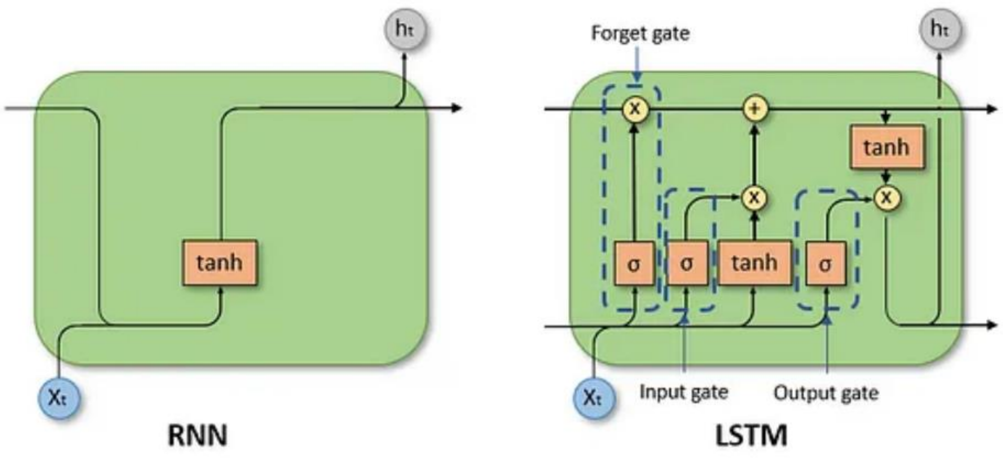
Method

The speech recognition task follows these steps:

1. **Dataset Preparation**
   - The database contains spoken digit files from 0-9 that have been recorded by different persons. All contain 1700 speech digits files. The main purpose of the database is to detect speech digits using deep learning. This kind of operation is mostly used in speech translation and transcription. Audio subtitling is also a good example in this context. The detection of speech digits falls under the category of

speech transcription system where the speech files are transcribed to text data. After the transcription of the digit data, the recognition can be done. So, the detection of speech digits will first need to pass through the speech transcription process and then the detection will be performed. The dataset includes labelled audio files of spoken phrases. Each audio file is pre-processed to convert it into a suitable format for model training. Preprocessing steps include sampling, normalization, and feature extraction using Mel Frequency Spectral Coefficients (MFCCs).

2. **Neural Network Design**
   - The architecture consists of multiple LSTM layers, followed by fully connected layers for classification.
     - Input: Audio features in the form of MFCCs.
     - LSTM Layers: Capture the temporal structure in the audio data.
     - Fully Connected Layer: Output layer with a softmax function to predict the probability distribution over the possible words.



3. **Training**
   - The model is trained for 10 epochs using the Adam optimizer with a batch size of 425. The loss function is cross-entropy loss, and the training involves backpropagation through time (BPTT) to update the weights.

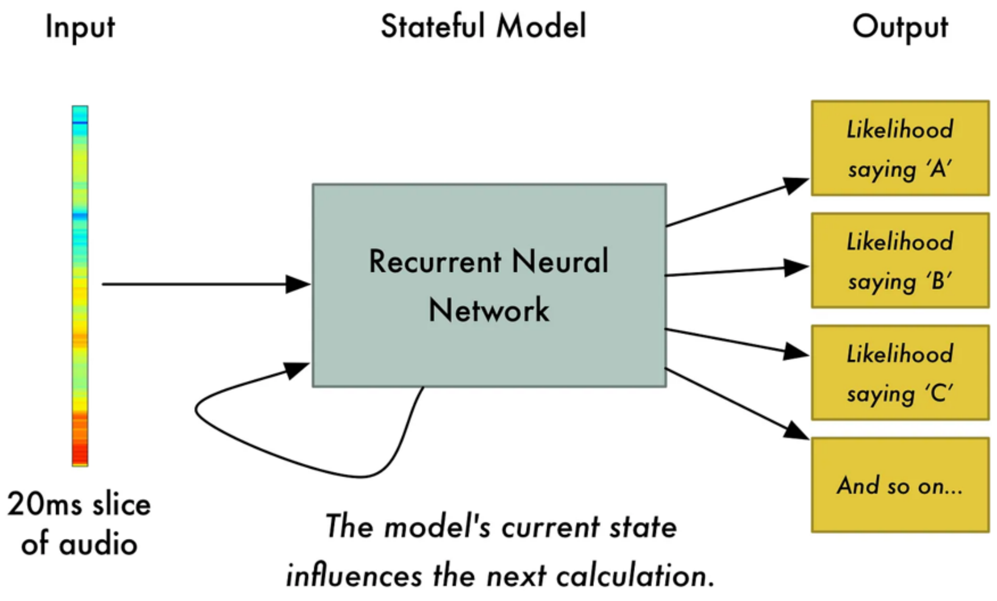
4. **Evaluation**
   - The model is evaluated on a test set of speech recordings. The primary metrics used are accuracy and word error rate (WER).

Dataset Preparation

- **Audio Preprocessing**: Each audio file is converted into MFCC features, which represent the short-term power spectrum of sound.
- **Normalization**: The data is normalized to ensure that input features are on a consistent scale.
- **Dataset Splitting**: The dataset is divided into training, validation, and test sets to ensure robust performance evaluation.

Neural Network Design

- **Input Layer**: MFCC features are fed into the RNN.
- **LSTM Layers**: Two stacked LSTM layers capture the sequential dependencies in the speech data.
- **Fully Connected Layers**: The final layer predicts the transcribed text.



Recognizing character from short sound

Training

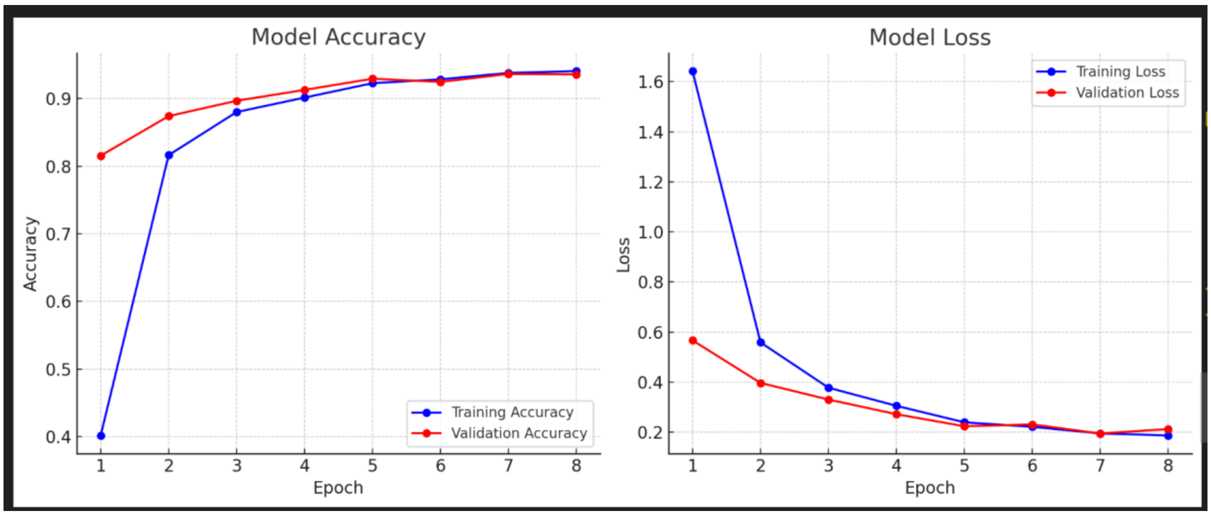
The model was trained for 10 epochs using the Adam optimizer. During each epoch, the following metrics were tracked:

- **Training Loss**: Measures how well the model is learning from the training data.
- **Validation Accuracy**: Indicates how well the model generalizes to unseen data.

Evaluation

The final model was tested on a separate dataset, achieving the following results:

- **Accuracy**: 0.9442
- **Word Error Rate (WER)**: 0.1725



Results

The model achieved a test accuracy of 94.42% with a word error rate of 17.25%. Further improvements can be made by experimenting with deeper network architectures and incorporating advanced techniques like attention mechanisms.

Summary

In this project, a simple RNN architecture was successfully applied to the task of speech recognition, achieving good performance. Future work can explore more complex models such as transformers, data augmentation, and transfer learning to further improve accuracy and generalization.