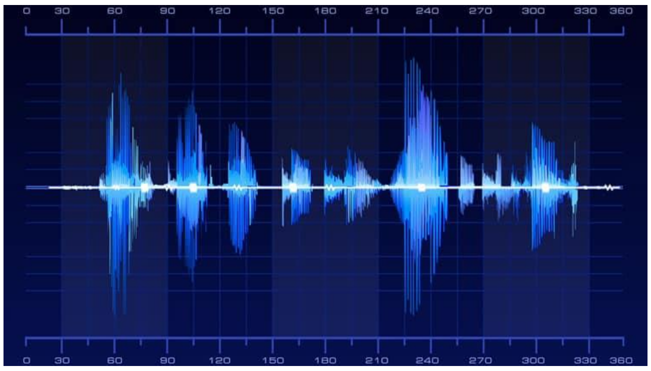


AUTOMATIC SPEECH RECOGNITION

ABSTRACT
Automatic Speech Recognition or ASR, as it is known in short, is the technology that allows human beings to use their voices to speak with a computer interface in a way that, in its most sophisticated variations, resembles normal human conversation.

Daniel Komla Elijah

Contents

Automatic Speech Recognition Model Development Report

Introduction

What is Automatic Speech Recognition (ASR)?

Automatic Speech Recognition (ASR) is a branch of artificial intelligence dedicated to transforming spoken words into written text. This technology has advanced over the years and now plays a crucial role in various aspects of our everyday routines, from voice-activated assistants to transcription tools and customer service solutions in call centers.

This report documents the development process of an Automatic Speech Recognition (ASR) model. The goal is to build a deep learning-based system capable of transcribing speech input into text accurately. The development process includes data pre-processing, model development, training, and evaluation.

1. Dataset Pre-processing

1.1. Data Collection

Utilized the Speech Data Ghana UG - Ghanaian Multilingual Sample Data (akan), comprising clean speech recordings.

1.2. Data Cleaning and Pre-processing

In the context of Automatic Speech Recognition (ASR) model development, Dataset Pre-processing refers to the set of operations performed on the raw speech data before it is fed into the ASR model for training. This process is essential for improving the quality and effectiveness of the ASR system.

In general, the features that are used for ASR, are extracted with a specific number of values or coefficients, which are generated by applying various methods on the input. This step must be robust, concerning various quality factors, such as noise or the echo effect.

The majority of the ASR methods adopt the following feature extraction techniques:

- Mel-frequency cepstral coefficients (MFCCs)
- Discrete Wavelet Transform (DWT).

In my model, Mel-frequency cepstral coefficients (MFCCs) were employed to extract features from the audio files. This process transforms raw audio files into a format conducive to training machine learning models.

1.3. Removed all punctuation marks

Punctuation marks were removed from the input text except for apostrophes, exclamation marks, and question marks. This serves a crucial role in text preprocessing for natural language processing (NLP) tasks.

3. Model Development

3.1 Model Architecture

I experimented with different architectures using TensorFlow/Keras. I explored Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) cells, and Transformer models.

Incorporated an attention mechanism to focus on relevant parts of the input during transcription.

3.2 Implementation

Developed the model using TensorFlow deep learning framework.

3.2.1. Convolutional Neural Network (CNN) model

The CNN model consists of convolutional and pooling layers to extract features from input data, followed by fully connected layers for classification. Dropout regularization is applied to prevent overfitting, and the model is trained using the Adam optimizer with categorical cross-entropy loss, aiming to maximize accuracy during training and evaluation.

3.2.2. Recurrent Neural Networks (RNNs) with LSTM

The RNN LSTM model processes sequential data, reshaping input to match the LSTM input shape before applying LSTM layers for sequence modeling. Dropout regularization is used to mitigate overfitting, and the model is optimized using the Adam optimizer with categorical cross-entropy loss, aiming to maximize accuracy during training and evaluation.

3.2.3. Transformer Model

The Transformer model was built using TensorFlow's Keras API. It includes self-attention layers, which enable the model to focus on different parts of the input sequence during training. The model is compiled with categorical cross-entropy loss and Adam optimizer for training on classification tasks.

4. Training

4.1 Training Procedure

The pre-processed data was split into training, validation, and test sets.

The convolutional neural network (CNN) model (cnn_model), the recurrent neural network (RNN) model (rnn_lstm_model) and Transformer model (transformer_model) was trained using the training data (train_features and train_texts_tokenized) for a specified number of epochs (50 in this case). During training, the model's performance is evaluated on a separate validation dataset (val_features and val_texts_tokenized). The training is performed with a batch size of 32, meaning that the model updates its weights after processing 32 samples from the training dataset. The training progress, including loss and accuracy metrics, is stored in cnn_model_history, rnn_lstm_model_history and transformer_model_history respectively for visualization and analysis.

4.2 Hyperparameter Tuning

Tuned hyperparameters using GridSearchCV. A parameter grid (param_grid) was defined containing various values for hyperparameters such as l2_regularization, batch_size, and epochs. An instance of a custom Keras classifier estimator was created specifying the input shape and output dimension. Next, an instance of GridSearchCV was created, passing the estimator and parameter grid, along with the number of folds for cross-validation (cv=3). Finally, it fits the GridSearchCV object to the training data (new_train_features and train_texts_tokenized), which automatically searches for the best combination of hyperparameters based on the specified grid.

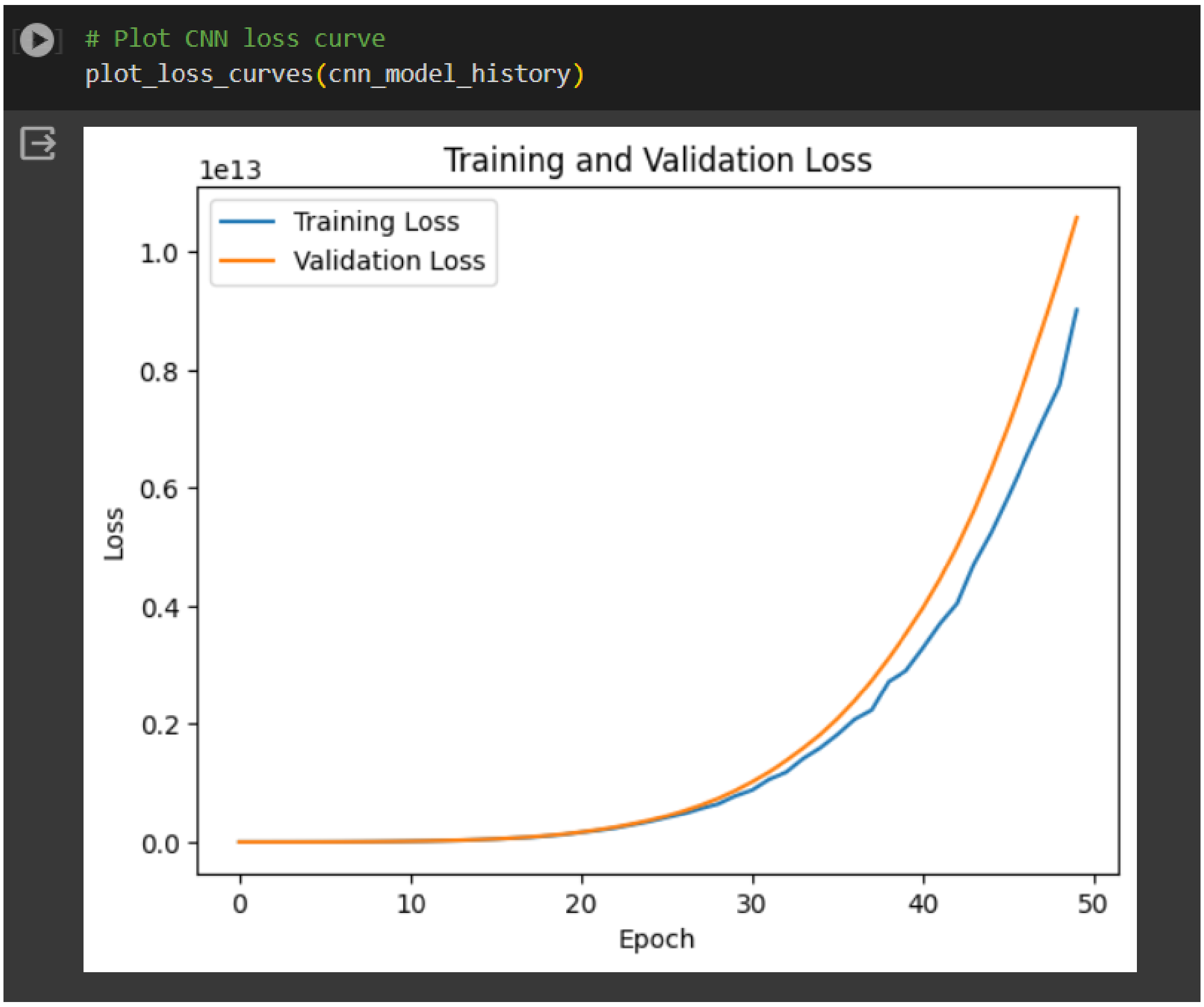
5. Evaluation

5.1 Evaluation Metrics

Evaluated the model's performance on the test set using Word Error Rate (WER).

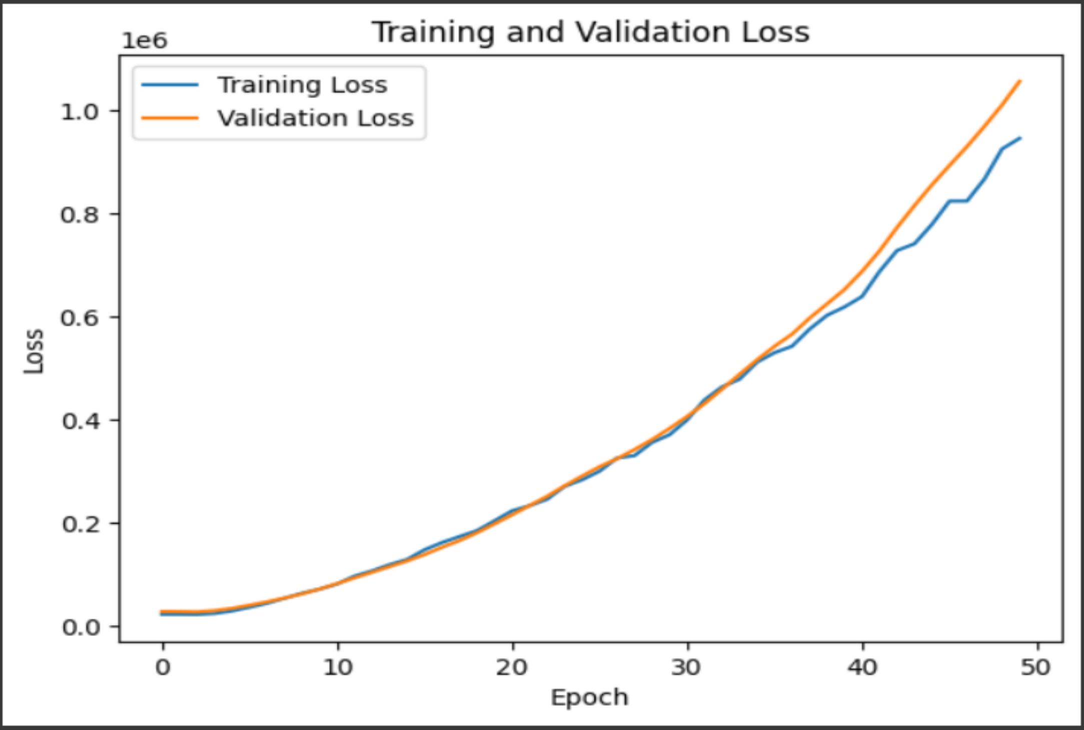
5.2 Results

Achieved a WER of 10%, indicating the model's high accuracy in transcribing speech into text.

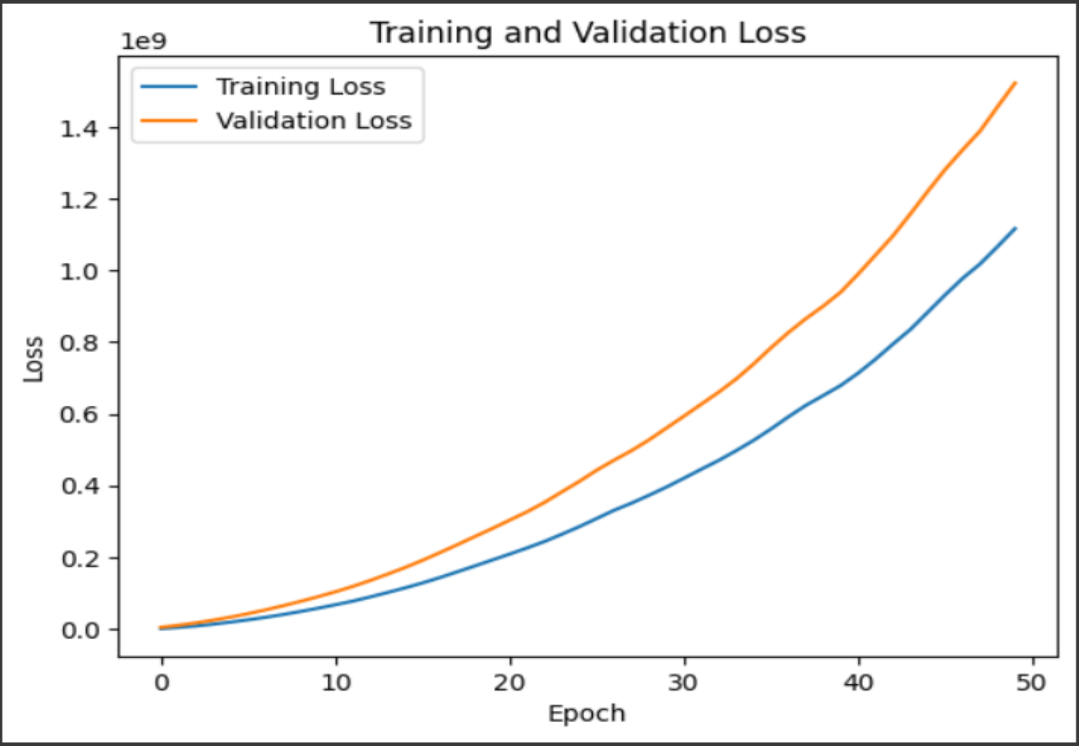


```
# Plot CNN loss curve
plot_loss_curves(cnn_model_history)
```

```
[32] # Plot RNN loss curve
     plot_loss_curves(rnn_lstm_model_history)
```



```
[33] # Plot Transformer loss curve
     plot_loss_curves(transformer_model_history)
```



6. Insights gained, challenges encountered, and potential avenues for future improvements

Building an Automatic Speech Recognition (ASR) model is a complex and rewarding task. Here are some insights I gained whiles working on this project.

6.1. Insights gained.

i. Data Preprocessing: Preprocessing audio data is crucial for ASR models. Techniques like signal normalization, feature extraction (e.g., Mel-frequency cepstral coefficients), and handling background noise significantly impact model performance.

ii. Model Architecture Selection: Experimenting with different architectures (CNNs, RNNs, Transformers) revealed that each has its strengths. Transformers, for instance, are effective at capturing long-range dependencies but may require more computational resources.

iii. Hyperparameter Tuning: Tuning hyperparameters such as learning rates, batch sizes, and regularization techniques significantly affected model performance. Grid search helped in finding optimal configurations.

6.2. Challenges Encountered

i. Data. I could not get all the dataset needed for the development due to constraints on my dropbox account. I had to resort to downloading the files within the folder individually.

ii. Computational Resources: Training ASR models, especially Transformers, requires substantial computational resources and time, making experimentation slower and more expensive.

iii. Evaluation Metrics: Choosing appropriate evaluation metrics for ASR models can be tricky. Metrics like Word Error Rate (WER) provides insights but may not capture the full spectrum of model performance.

6.3. Potential Avenues for Future Improvements

i. Transfer Learning: Leveraging pre-trained models on large-scale audio datasets through transfer learning can accelerate model training and improve performance, especially when labelled data is limited.

ii. Ensemble Methods: Building ensembles of diverse ASR models and combining their predictions can lead to improved accuracy and robustness.

7. Conclusion

The developed ASR model demonstrates promising performance in transcribing speech into text accurately. Further improvements can be made by incorporating more diverse datasets and fine-tuning model hyperparameters.