Audio Recognition

By: Ephraim Langdon, Avery Belanger, Joshua Swanson, Jerry Buno
Department of Physics, Computer Science and Engineering , Houghton University, Houghton, New York, USA

1. Introduction

Description - Once our model is fully trained, it should be able to process an audio file and identify who is speaking while providing a written version of what they said. The model begins by taking a .wav audio file and converting it into a spectrogram, which is like a visual chart that shows the sound's frequencies over time. It then turns this graph into a grayscale image and saves it as a .png file. Lastly, the model analyzes the spectrogram and compares it to its database of known voices to figure out who is speaking. At the same time, it listens to the words being said, converts them into text, and prints them with the person's name and what they said. This model makes it easy to identify speakers and get a written audio version in one smooth process.

Benefits/ Applications- Audio recognition and speech-to-text models have numerous practical uses in various industries. They enhance accessibility by converting audio into text for those with disabilities, also supporting multilingual users. This model boosts productivity by automating transcription for meetings, lectures, and legal proceedings, while also allowing for searchable and easily retrievable text. Businesses gain real-time insights from customer sentiment analysis, while the media industry benefits from faster subtitle creation and podcast transcription. With enough training, it could be used for security with voice biometric authentication and personalized user experiences.

Challenges-These models can face several challenges. Background noise and poor audio quality can make it hard for the model to clearly hear the speech, leading to inaccurate recognition. Additionally, differences in accents, dialects, and pronunciation can cause the model to misunderstand what's being said. The model must also be able to handle speaker variance, including but not limited to differences in pitch, tone, and speaking speed. Homophones could also be very annoying, as words that sound the same may be confusing without proper context. In addition, separating multiple speakers in overlapping conversations, known as speaker diarization, is commonly challenging. Limiting and imbalanced training data can hinder the model's ability to recognize diverse voices. Finally real-time processing and latency present challenges, especially when resources are limited, unless you happen to be using a server, which not everyone has access to. Now CNNs face several unique challenges in itself. When applied to these tasks. One of the biggest is handling temporal information, CNNs are typically designed for spatial data, like images, and may struggle to understand the order and timing of words and phonemes. While CNNs are effective at identifying patterns in spectrograms, they may miss important temporal dependencies in speech without significant modifications to the model. Another example is the high computational requirements of the CNNs, especially when processing large amounts of data. Training and running CNNs can be resource-intensive, making them less suitable for real-time application, mostly on devices with limited processing power. Additionally, CNNs are sensitive to noise, the

background sounds can affect the spectrogram and significantly reduce accuracy. They also sometimes struggle to capture the fine-grained temporal details of speech, which also can lead to the loss of critical phonetic or contextual information. Then there's the obvious bad quality of the data being processed. Poorly scaled or noisy input data can greatly affect the CNN's performance, undermining the importance of high-quality feature extraction.

2. Problem Formulation

Formation- Our Speech-to-text model uses CNNs to automatically convert audio into text without the need for manual matching. CNNs are great for this task because they can pick up on key patterns in the audio, like sound frequencies. Unlike other models such as RNNs, Cnns are faster since they can process data all at once, rather than step by step. Since CNNs can directly turn audio into text it simplifies the process. When trained on a lot of audio-text pairs, CNN-based models become very accurate and should be able to handle noisy environments well.

Input- the text-to-speech model takes the audio files and turns them into written words. To do this, it breaks down the sound into smaller pieces, like it's creating a picture of the sound over time, which makes our spectrogram. Which shows how different sound frequencies change throughout the audio file. The spectrogram makes it easier for the model to recognize patterns and figure out what words are being said.

Output- The output of the model is an integer that is the index of what .txt file is being read in the audio. The CNN processes the spectrogram by taking important features through convolutional and pooling layers. These features are then passed through pooling layers to produce the final output which is a sentence. The model's predictions are refined using loss functions like cross-entropy to ensure the generated text is closely aligned with the actual words that were said.

3. Literature Review

Relevant studies-In audio recognition and speech-to-text, several important methods have been developed. Early on, researchers used Hidden Markov Models (HMMs), which were good at recognizing individual phonemes but struggled with handling long or continuous speech. Then, the focus shifted to deep learning, where Convolutional Neural Networks (CNNs) started being used to pull important features from audio spectrograms, helping capture both time and frequency information in speech. Recurrent Neural Networks (RNNs) were added to better handle the sequence of words in speech, improving recognition for continuous talking. One of the latest advancements in speech-to-text is the transformer-based model wav2vec, which learns directly from raw audio by using a technique called self-attention to identify long-term speech patterns. This method removes the need for manually crafted features, making the process more efficient and accurate. The most recent and significant model in this space is OpenAI's Whisper, which is designed for robust transcription across multiple languages and in

noisy environments. It uses a large, pre-trained transformer architecture to handle various speech recognition tasks with high accuracy, even with challenging audio inputs.

Limitations/Explorations- Current speech-to-text models still have some challenges to address. For one, processing in real-time can be difficult because many models require a lot of computing power, making them less practical for everyday use, especially on mobile devices like phones or laptops. We need faster, more efficient models. Another problem is understanding specialized vocabulary and slang, like technical terms or abbreviations. Models often struggle with new or uncommon words, even if they've been trained on them. Context is also an issue. Models sometimes don't understand the meaning of words based on the conversation, which can lead to mistakes. Lastly, identifying speakers and handling overlapping speech is hard. When multiple people talk at the same time, models have trouble figuring out who's speaking and getting the words right. Better ways to separate voices are needed for clearer transcriptions.

4. Method/Experiments

Model design-

The CSTR VCTK Corpus is an English multispeaker speech dataset designed for voice cloning and text-to-speech (TTS). It includes recordings of 109 native English speakers, each with a specific accent. The corpus was created primarily for HMM-based TTS systems and was made for speaker-adaptive synthesis and voice model training. Each speaker reads around 400 phrases picked from a variety of sources. The sentences contain excerpts from a newspaper, especially The Herald (Glasgow), a Rainbow Passage for accent comparison, and an elicitation paragraph apparently designed to capture distinctive regional accent traits. The texts were selected using an algorithm that was made to optimize phonetic and contextual coverage. The elicitation paragraph was taken from the Rainbow Passage, a book that is often used for analyzing accents.

The model is trained using the CSTR VCTK Corpus. During training, audio input is changed into spectrograms, which are fed into the CNN model for feature extraction and speaker identification. The model is trained using a specified amount of epochs, which helps minimize the loss function. The number of epochs is determined by the convergence of the loss function, adjusting the underfitting and overfitting. This usually ranges from 50 to 200 epochs, this typically depends on the amount of data you're using.

The CNN model has four convolutional layers with ReLU activations, followed by fully connected layers for classification and max-pooling layers for dimensionality reduction. The code that follows gives a summary of the architecture:

```
model = tf.keras.Sequential([
    Conv2D(16, 3, activation='relu',
input_shape=(image_size[0],image_size[1], 1),dtype="float32"),
    MaxPooling2D(),
    Conv2D(16, 3, activation='relu'),
    MaxPooling2D(),
    Conv2D(16, 3, activation='relu'),
    MaxPooling2D(),
    Conv2D(16, 3, activation='relu'),
    MaxPooling2D(),
    Flatten(),
    Dense(128),
    Dense(len(people_nums), activation="softmax"),
])
```

Next, the Adam optimizer is used to improve accuracy. The training process involves splitting the dataset into different sets for training:

```
X_train, X_test, y_train, y_test = train_test_split(new_x, y,
test_size=0.2, random_state=42)
```

Training Process and Dataset Preprocessing

The first step is to convert audio recordings into spectrograms using librosa, a package for audio processing. The following code sample explains how we created spectrograms from audio files.

```
import numpy as np
import librosa.display, os
import matplotlib.pyplot as plt
%matplotlib inline

def create_spectrogram(audio_file, image_file):
    print(image_file)
    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)
    fig.subplots_adjust(left=0, right=1, bottom=0, top=1)

    y, sr = librosa.load(audio file)
    ms = librosa.feature.melspectrogram(y, sr=sr)
    log ms = librosa.power to db(ms, ref=np.max)
    librosa.display.specshow(log_ms, sr=sr)

    fig.savefig(image_file)
    plt.close(fig)
```

The process described above is applied to all audio files in the corpus. The

created spectrograms are kept in the Spectrogram directory and used during model training. Before being put into the model, the spectrograms are scaled to 62x62 pixels and converted to grayscale.

Evaluation metrics-

```
Epoch 1/5
194/194 ━━━━━━━━━━━━━━━━━━━━ 7s 27ms/step - loss: 200179.2344 -
precision_55: 0.0097 - val_loss: 195549.2188 - val_precision_55:
0.0097
Epoch 2/5
194/194 ━━━━━━━━━━━━━━━━━━━━ 5s 26ms/step - loss: 195001.2812 -
precision_55: 0.0102 - val_loss: 195638.2344 - val_precision_55:
0.0090
Epoch 3/5
194/194 ━━━━━━━━━━━━━━━━━━━━ 5s 25ms/step - loss: 194620.9219 -
precision_55: 0.0103 - val_loss: 195641.9219 - val_precision_55:
0.0092
Epoch 4/5
194/194 ━━━━━━━━━━━━━━━━━━━━ 5s 25ms/step - loss: 194241.5938 -
precision_55: 0.0107 - val_loss: 195181.4219 - val_precision_55:
0.0096
Epoch 5/5
```

The training loss gradually drops, going from 200179 to 194241 across the five epochs, however, the val_loss does not decrease significantly and stays over 195000. This val_loss shows that the model could be overfitting or having difficulty generalizing to the data. Over the epochs, the training accuracy increases a bit from 0.0097 to 0.0109. However, the precision peaks at 0.0099 which is low.

5. Conclusion

By using the CSTR VCTK Corpus, we were able to train the models, which showed very limited performance in differentiating speakers. While the training loss decreases a noticeable amount, the validation loss stays constant at around 195000. This could mean that the models have difficulty capturing the different types of accents.

Future improvements could include:

The first thing we'd do to improve the model's performance is add more data. Expanding the dataset to include more speakers, accents, and various speech samples would allow the algorithm to more accurately generalize to different speakers from different parts of the world. Including datasets with additional accents, such as non-native English speakers or speakers from underrepresented places, may improve the model's accuracy.

Additionally, the model could improve from additional data augmentation. Methods like adjusting the pitch, and tempo, or introducing low background noise could imitate real-world speech changes, allowing the model to handle a wide range of audio inputs.

Finally, training the model for more epochs while using early stopping or regularization approaches might help decrease overfitting and increase generalization. Combining this strategy with pretraining on a bigger, more varied dataset before fine-tuning the CSTR VCTK Corpus could result in improved overall performance.

6. References

https://www.kaggle.com/datasets/mfekadu/english-multispeaker-corpus-for-voice-cloning/code

https://huggingface.co/datasets/CSTR-Edinburgh/vctk

https://www.geeksforgeeks.org/audio-recognition-in-tensorflow/

https://github.com/openai/whisper

https://www.televic.com/en/televicgsp/news/the-evolution-of-speechtotext-technology#:~:text=In%20recent%20years%2C%20machine%20learning,the%20accuracy%20of%20STT%20systems.

https://www.geeksforgeeks.org/hidden-markov-model-in-machine-learning/

https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

https://www.analyticsvidhya.com/blog/2023/07/speech-to-text-with-transformers/

https://ai.meta.com/blog/wav2vec-state-of-the-art-speech-recognition-through-self-supervision/#:~:text=Wav2vec%20trains%20models%20by%20making,of%20the%20task%20for%20training.

https://www.geeksforgeeks.org/self-attention-in-nlp/

https://openai.com/index/whisper/

https://www.digitalocean.com/community/tutorials/audio-classification-with-deep-learning

https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/