CureMD™

GenAI

Bootcamp 2024

Prepared by
Rayyan Shabbir
Muntazir Ali
Hadi Khadim

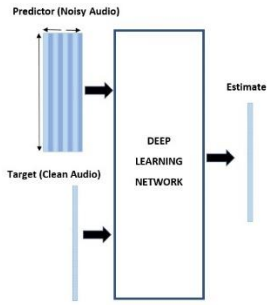
Introduction

Audio signal processing has experienced significant advancements through deep learning, enabling innovative applications such as noise removal, speech enhancement, and audio restoration. In this assignment, we aim to utilize deep learning techniques for two purposes: one is to transform **noisy podcast audio into clean**, high-quality audio, and the second is to transform the **male voice into the female voice.**

The process begins with inputting clean podcast audio, which we recorded specifically for this project. To simulate real-world noisy environments, we added noise to the recordings using a twofold approach: **Gaussian noise to represent static interference and environmental noise** (e.g., street noise and crowd chatter) to mimic typical background disturbances. These noisy audio clips served as inputs for training and testing the deep learning model.

Unlike many approaches that use spectrogram representations, we chose to process the audio directly in **its raw waveform vector format**, leveraging the Librosa library for audio handling and manipulation. By operating directly on raw vectors, we maintain the original waveform structure, enabling the model to learn features from the time-domain representation of the audio.

To remove noise, we designed a neural network based on the UNET architecture, known for its success in tasks requiring precise mappings between inputs and outputs. The model was trained to map noisy raw audio waveforms to their clean counterparts, learning the intricate relationships necessary to reconstruct clean audio and for the voice transformation we used UNET LSTM due to its ability to capture temporal dependencies.



Dataset Description

The dataset used in this project was custom-created to fulfill the assignment constraints, which required us to avoid using any pre-existing datasets. The dataset was generated by recording podcast-style audio in a controlled environment, ensuring clean and high-quality recordings. Relevant information about the dataset is as follows:

1. **Sampling Rate (SR):**

The audio was sampled at a frequency of 40000 Hz, providing an appropriate balance between audio fidelity and computational efficiency.

2. **Audio Format:**

The recordings were processed as mono audio, with a single channel used for simplicity and to reduce computational overhead.

3. **Data Chunking:**

Each audio clip was divided into fixed-length chunks of 1-second duration, ensuring uniform tensor sizes for processing. This chunking allows the neural network to handle input data efficiently and facilitates batch training.

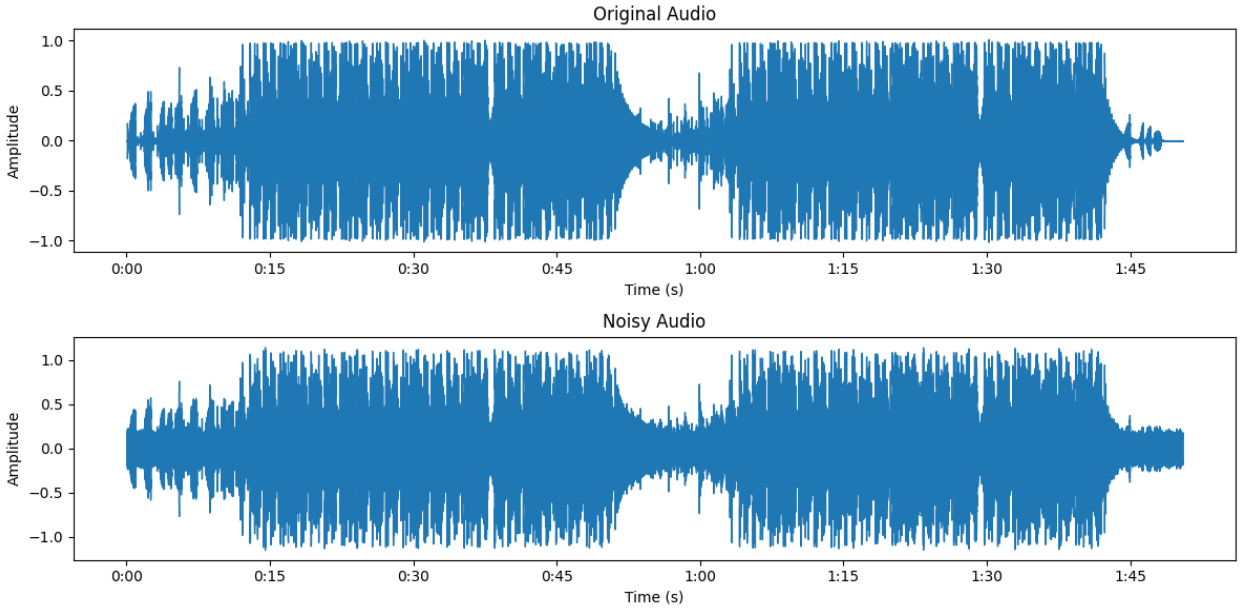
4. **Dataset Size:**

The dataset contains a total of 10 minutes of audio, which was divided into Y chunks, each representing a 1-second tensor of raw audio data.

5. **Noise Addition:**

Noise was artificially added to the clean audio using:

- Gaussian Noise to simulate static interference.



These noisy samples formed the input to the neural network, while the corresponding clean audio served as the ground truth for supervised training.

6. **Dataset Splitting:**

The dataset was divided into training and testing sets with a ratio of 80:20. This split ensured that the model was trained on a significant portion of the data while retaining a separate subset for evaluating its performance on unseen audio.

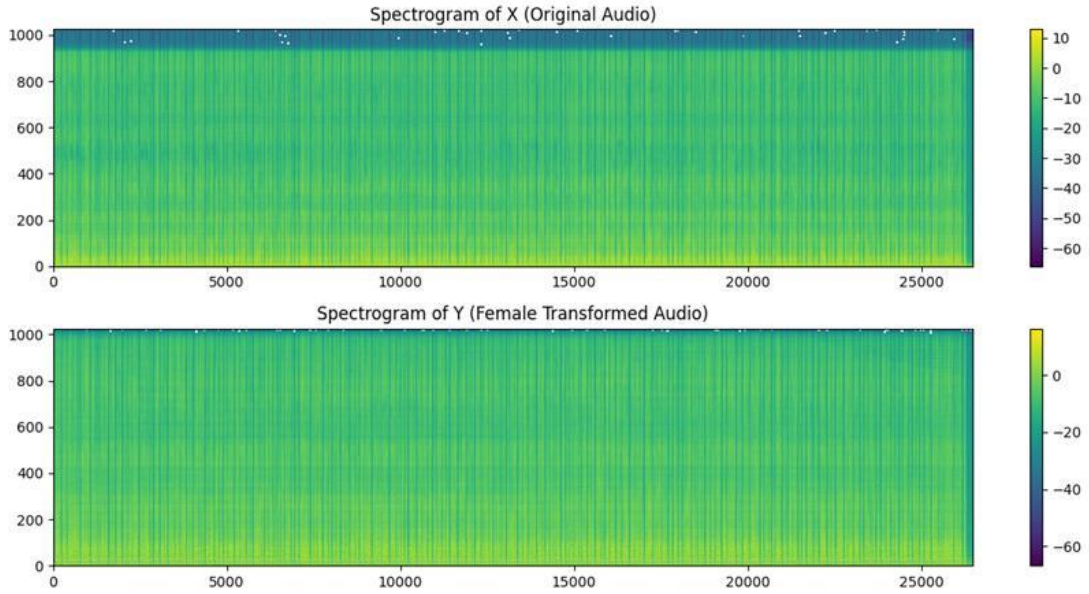
This dataset design and preparation were tailored to meet the project requirements and provide a realistic simulation of noise removal in podcast audio using raw waveform processing. The division into fixed-length chunks and the inclusion of diverse noise types ensure that the model can generalize well across various noise scenarios.

Preprocessing

The preprocessing pipeline was carefully designed to prepare the audio data for training and testing the neural network while adhering to the constraints of the project. The steps taken in preprocessing are as follows:

1. **Audio Loading and Handling:**

The Librosa library was used to handle audio loading and manipulation. All audio files were loaded at a sampling rate of 40000 Hz and converted to mono format to ensure consistency across the dataset. But before this approach, we converted the audio into the spectrogram; the problem with this approach was that due to the feedback of our output to the input, the size of the spectrogram was mismatched.



2. **Noise Addition:**

To create noisy versions of the clean audio, we used the random normal distribution to generate Gaussian noise. This noise was then added to the clean audio to simulate real-world interference. Additionally, the noisy audio was saved to disk as separate files, allowing us to inspect and verify the noise addition process visually and audibly.

3. **Audio Chunking:**

Each audio file was divided into 1-second chunks to standardize the input size for the neural network. This chunking ensured that each input tensor represented a fixed duration of audio, simplifying the training process.

4. **Tensor Conversion:**

The clean and noisy audio chunks were converted into tensors using PyTorch. The noisy tensors formed the input (X), while the corresponding clean tensors served as the target output (Y) for the supervised learning process.

5. **Dataset Splitting:**

The dataset was split into training and testing sets using a random split method, with 80% of the data allocated for training and the remaining 20% for testing. This split ensures the model is trained on a diverse set of examples while retaining a separate subset for evaluation.

6. **Batch Processing:**

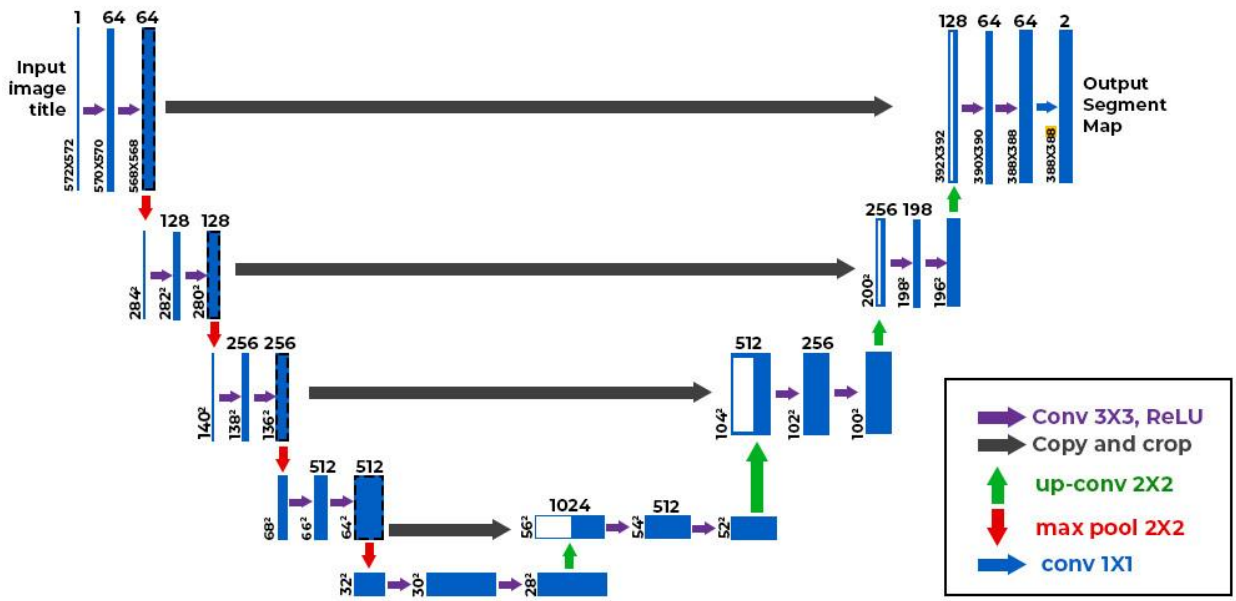
To facilitate efficient training, the data was further divided into batches of size 16. Mini-batching reduces memory usage and speeds up training by processing multiple samples simultaneously in each forward and backward pass of the model.

By following this preprocessing pipeline, the audio data was prepared in a format suitable for deep learning. The use of random noise addition, chunking, and batching ensures that the model is exposed to varied input scenarios while maintaining a consistent structure for learning.

Model Architecture

The core of this project revolves around using a deep learning model based on the UNET architecture to denoise audio. Originally designed for image segmentation tasks, UNET's encoder-decoder structure with skip connections makes it highly suitable for tasks requiring precise input-output mappings, such as audio denoising. I have used UNet for noise removal because **it efficiently captures and restores fine-grained details in audio**, separating noise from the clean signal.

1. **Simple UNET:**



UNET Architecture Details:

1. **Input and Output:**

- The model takes 1-second chunks of noisy audio in the form of raw waveform tensors as input.
- The output is the corresponding denoised waveform tensor, attempting to reconstruct the original clean audio.

2. **Architecture Structure:**

- **Encoder**: The encoder extracts hierarchical features from the noisy audio by applying multiple convolutional layers with increasing filter sizes and downsampling through max pooling.
- **Bottleneck**: The bottleneck layer captures the most compressed and abstract representation of the audio signal.
- **Decoder**: The decoder reconstructs the audio waveform by upsampling the compressed features and integrating contextual information from the encoder using skip connections.
- **Skip Connections**: These connections between the encoder and decoder allow the model to retain fine-grained details during the reconstruction process, essential for recovering subtle audio features.
- **Feedback**: After the UNET model processes the noisy input audio, it generates a cleaned audio output. This generated audio is saved and then manually fed back as the input for the next round of predictions. This process is done to see how well the model can refine its predictions when it starts from its own output rather than using it for further training.

3. **Activation Functions:**

- ReLU (Rectified Linear Unit) activation was used in the hidden layers for non-linearity.

- No activation function is used in the final layer

4. **Loss Function:**

I have used Mean Squared Error (MSE) and Signal Noise Ratio (SNR) as the loss function, As MSE which measures the difference between the predicted and ground truth audio waveforms. MSE was chosen because it effectively penalizes deviations in continuous signal reconstruction tasks. SNR< ensures the output audio has minimal noise and high clarity.

5. **Optimizers:**

To optimize the model's performance, two different optimizers were used with varying Signal-to-Noise Ratios (SNR):

Nadam Optimizer:

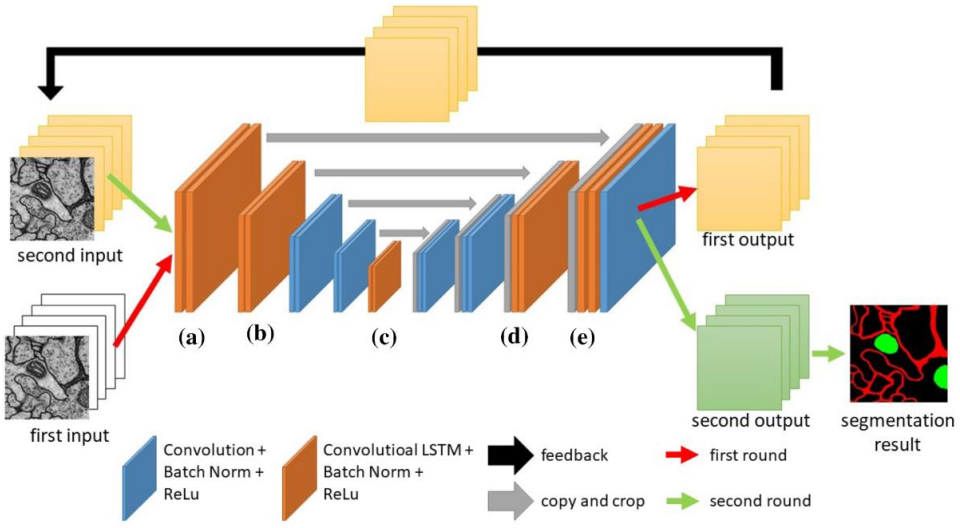
- This optimizer, an adaptive gradient descent variant of Adam, was used with audio chunks having SNR = -12.
- Nadam helps with faster convergence and better generalization in scenarios with low signal clarity.

Adam Optimizer:

- This standard adaptive optimizer was applied to audio chunks with SNR = 0.25, where the noise interference was less severe.
- Adam effectively balances the learning rate for each parameter, enabling stable convergence.

By leveraging the UNET architecture with these optimizers, the model was able to adapt to varying noise levels in the input audio and perform denoising effectively. The combination of architectural features and tailored optimization strategies played a crucial role in achieving high-quality audio restoration.

2. **UNET LSTM:**



UNet-LSTM for Voice Transformation

UNet-LSTM for male-to-female **voice transformation to combine spatial (frequency) and temporal (time-dependent) features,** ensuring smooth and natural voice modulation.

1. **Input and Output:**

Input: The model takes an audio signal, leveraging the Librosa library for audio handling and manipulation. By operating directly on raw vectors, we maintain the original waveform structure, enabling the model to learn features from the time-domain representation of the audio.

Output: The output is a transformed spectrogram that modifies key features of the male voice to resemble a female voice, such as pitch, formants, and timbre.

2. **Architecture Structure:**

Encoder:

- Extracts hierarchical features from the input audio.
- Convolutional layers process the spatial (frequency-related) details, while downsampling layers (e.g., max pooling) reduce the raw vector, focusing on prominent patterns like harmonics and formant structures.

Bottleneck (with LSTM):

- LSTM layers integrate temporal (time-dependent) information across the sequence to capture how audio features evolve over time. This ensures smooth transitions and continuity in the transformed audio.
- The bottleneck combines frequency-domain features (from convolutional layers) and time-domain dependencies (from LSTM) to effectively model voice characteristics.

Decoder:

- Reconstructs the modified spectrogram by upsampling and combining the hierarchical features captured by the encoder.
- Skip connections link corresponding encoder and decoder layers, ensuring preservation of important frequency and time-domain details.

Skip Connections:

- These help maintain fine-grained details about the original audio, ensuring that the output voice retains intelligibility while transforming characteristics like pitch and tone.

Feedback:

- In the LSTM model, the feedback happens automatically. The audio output generated at the last layer of the model is directly passed back to the first layer as input.
- This allows the model to refine its predictions continuously within the same prediction loop.

3. **Activation Functions:**

ReLU:

- Used in intermediate layers to enable non-linear feature transformations, allowing the model to learn complex patterns specific to male and female voice differences.

Sigmoid:

- Applied in the output layer to constrain the spectrogram values to a realistic range, ensuring the transformed spectrogram is usable for audio synthesis.

Result:

To evaluate the performance of our UNET-based denoising model, we conducted experiments with three different configurations of noise levels, optimizers, and learning rates. The results were assessed using the Signal-to-Noise Ratio (SNR) as the primary metric, which measures the quality of the reconstructed audio relative to the noise present. Below are the findings from the three versions:

1. Our Dataset:

This is the dataset in which we recorded audio having a length of 10 minutes. Approximately then, we added some noise to it and then trained through **UNET** to denoise the audio.

Hyperparameter	Noise Level	Optimizer	Learning Rate	SNR Loss
Version(V1)	0.02	Adam	1e-3	0.25
Version(V2)	0.05	Adam	1e-3	0.25
Version(V3)	0.05	NAdam	0.01	-12

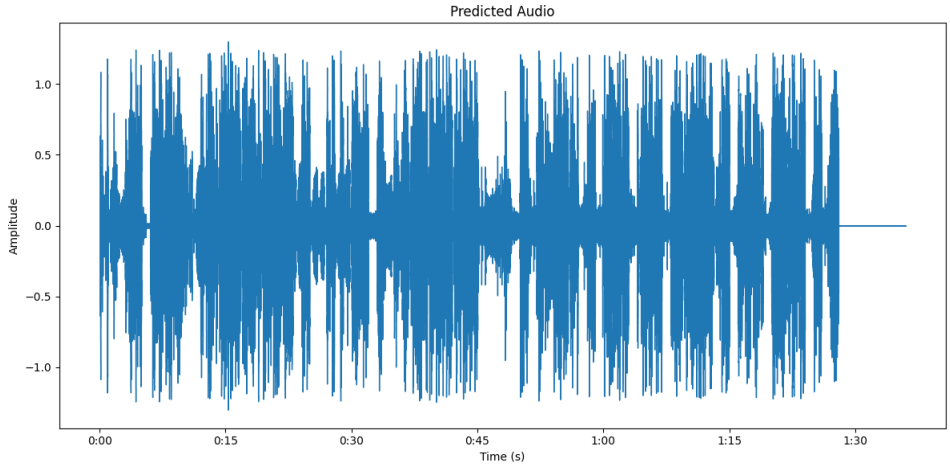
The model's performance was best at lower noise levels (V1). As the noise level increased, the denoising capability decreased slightly, with more residual noise present in the output.

2. Music Dataset:

This is the dataset which we have got the music from the internet and we added the noise in it and feed into the deep learning model of **simple UNET architecture** to denoise the audio.

Hyperparameters	Noise Level	Optimizer	Learning Rate	Batch Size	SNR Train	SNR Val
Version (V1)	0.02	Adam	1e-3	16	-19.71	-20.25
Version (V2)	0.02	NAdam	1e-3	16	-17.10	-15.03
Version (V3)	0.02	Adam	1e-3	16	0.0011 (MSE)	0.0015 (MSE)
Version (V4)	0.05	NAdam	0.01	16	-12.82	-13.66

For our predicted audio this is the waveform from the deep learning model.

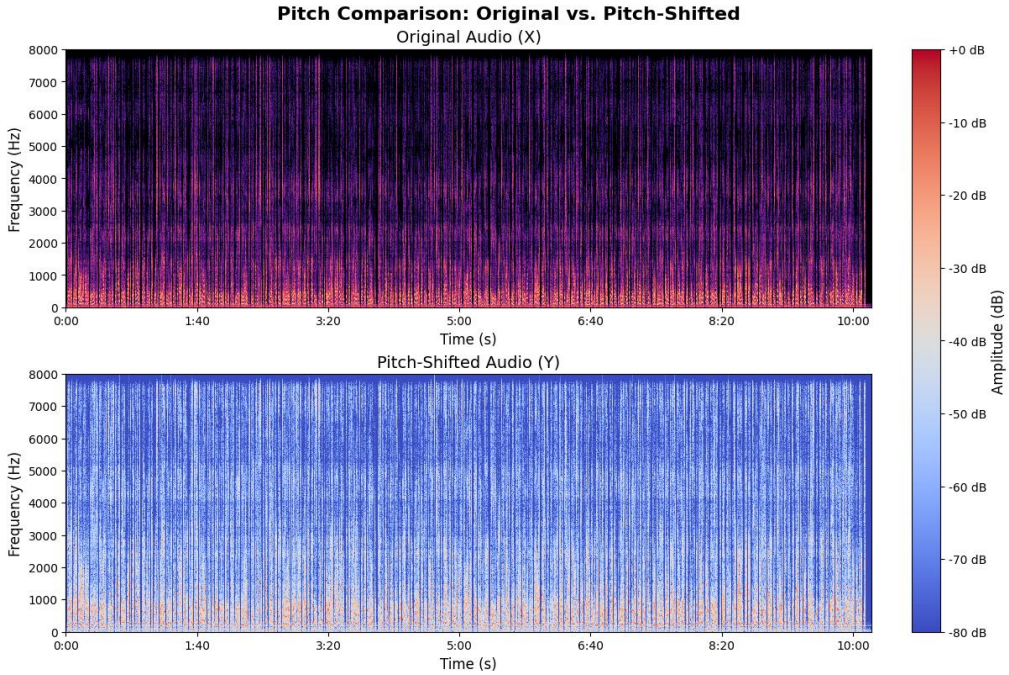


3. BookChapter:

This dataset has an audio of a book reading of a male person and feeds into our **UNET LSTM** model so that it can change the male voice into the female voice by changing the pitch of the voice.

Hyperparameter	Pitch Level	Optimizer	Learning Rate	Batch Size	SNR Train	SNR Val
Version (V1)	12 steps	Adam	1e-3	16	0.0093	0.0044
Version (V2)	12 Steps	NAdam	0.01	16	0.013	0.056
Version (V3)	12 Steps	Adam	0.01	16	0.007	0.005

This graph illustrates a comparison between the original audio (male voice) and the pitch-shifted audio (converted female voice).



The pitch-shifted spectrogram demonstrates an overall upward shift in frequency components, consistent with the characteristics of a typical female voice. This visualization highlights the transformation of vocal characteristics through pitch modification techniques.

Challenges and Observations

During the project, one significant challenge was maintaining the quality of audio outputs while effectively removing noise. For instance, higher noise levels in the dataset led to residual artifacts in the cleaned audio, making it harder for the model to generalize. Another observation was the difference in performance between the UNET and UNET-LSTM models. While UNET excelled in noise removal for simpler datasets, the LSTM component in UNET-LSTM proved more effective for voice transformation tasks due to its ability to capture temporal dependencies. Additionally, implementing the feedback loop required careful handling to avoid compounding errors over successive predictions.

Conclusion

This project successfully demonstrated the application of deep learning models for audio denoising and voice transformation tasks. By leveraging custom datasets and designing UNET and UNET-LSTM architectures, we achieved significant improvements in audio quality and realistic voice modifications. The feedback mechanisms enhanced predictions by reusing generated outputs effectively. While the models showed promising results, further refinement with larger datasets and hybrid architectures could improve performance and generalization. Overall, this project highlighted the potential of deep learning in advancing audio signal processing applications.