ADAPTIVE NOISE CANCELLATION SYSTEM USING DEEP LEARNING

Group Members:

- Prathamesh Lakhotiya (BT22ECE102)
- Shreyansh Dwivedi (BT22ECE095)
- Devesh Jena (BT22ECE108)
- Suryansh Shukla (BT22MEC030)
- Aditi Verma (BT22MEC002)
- Jyoti Verma (BT21MEC082)

INTRODUCTION TO NOISE CANCELLATION

What is Noise Cancellation?

•Noise cancellation refers to the process of removing unwanted background noise from an audio signal, allowing the desired sound (e.g., speech, music) to be heard more clearly.

•It is widely used in applications such as telecommunication, music production, and hearing aids.

Types of Noise Cancellation:

1.Active Noise Cancellation (ANC): Uses microphones to detect external noise and generate opposing sound waves to cancel it.

2.Passive Noise Cancellation: Involves physical barriers to block external sounds (e.g., earplugs, insulated walls).

DEEP LEARNING FOR NOISE CANCELLATION

Why Use Deep Learning for Noise Cancellation?

•Traditional noise cancellation methods rely on filters and basic signal processing techniques, which may struggle with complex or dynamic noise.

•Deep learning leverages neural networks to learn complex patterns in noisy audio, making it more effective at handling a variety of noise environments (e.g., traffic, office, public spaces).

How Deep Learning Works:

•A neural network is trained on noisy and clean audio data.

•The network learns to predict the clean signal from the noisy input by learning patterns in the audio.

Applications:

•Speech enhancement in teleconferencing.

Improving audio clarity in consumer electronics like smartphones and headphones.

•Assisting hearing aids to better understand speech in noisy environments

SUMMARY

This project utilizes deep learning techniques to reduce noise in audio signals, using the Microsoft DNS (Deep Noise Suppression) dataset. A Conv1D-LSTM hybrid model was developed to enhance both spatial and temporal processing, effectively isolating clean audio from noisy inputs. The results showed a significant improvement in audio clarity, underscoring the model’s potential for applications in real-world noisy environments.

LITERATURE REVIEW

- **Traditional Approaches:** Spectral subtraction and Wiener filtering are effective but limited in dynamic noise conditions.

- **Deep Learning Advancements:** Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) handle complex audio data better than traditional methods, especially for non-stationary noise.

Relevant studies include:

- Dogra et al.: Utilized CNNs with features like MFCCs and Mel spectrograms, achieving high accuracy in noise reduction.
- Kejalakshmi et al.: A Convolutional Recurrent Network (CRN) model improved Signal-to-Noise Ratio (SNR) by 14.84 dB in active noise cancellation.
- Zhang & Wang: Their Deep ANC model, combining CNNs and RNNs, demonstrated superior performance in real-time noise suppression.

PROBLEM STATEMENT AND OBJECTIVES

This project addresses the problem of enhancing audio quality by reducing background noise in real-time. By using a deep learning approach, we aim to create a solution capable of filtering out noise, thus improving audio clarity across various applications.

●Develop a neural network model for noise cancellation using the Microsoft DNS dataset.
● Preprocess audio data and train a model to reconstruct cleaner signals.
● Evaluate model performance using metrics like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

METHODOLOGY OVERVIEW AND PROCEDURE

Tools and Technologies:

- **Libraries:** **TensorFlow, Keras, Librosa, Matplotlib, Scikit-Learn**
- **Platform**: **Python**

Procedure:

- **Load the Microsoft DNS dataset, containing clean and noisy audio files.**
- **Extract Mel Frequency Cepstral Coefficients (MFCCs) for feature representation.**
- **Design and train a Conv1D-LSTM model for noise reduction.**

MODEL ARCHITECTURE AND DATA COLLECTION & PRE-PROCESSING

- **The hybrid model combines Conv1D layers for spatial feature extraction and LSTM layers for temporal dependencies. Dense layers finalize the output, reconstructing a clean signal. This architecture is well-suited for audio data that requires both spatial and temporal analysis.**

- **Data Collection & Pre-processing**

- **Dataset: Microsoft DNS dataset, which includes clean and noisy audio samples.**
- **Feature Extraction: MFCCs were used to capture the essential audio characteristics.**
- **Normalization: Min-Max scaling ensured that features were consistently scaled.**
- **Synthetic Noise Addition: Created noisy samples to simulate real-world noise conditions.**

PROJECT OBJECTIVE

Objective of the Project:

•The main objective of this project is to develop a deep learning-based system for noise cancellation that effectively removes background noise from audio signals while preserving the clarity and quality of the desired sound.

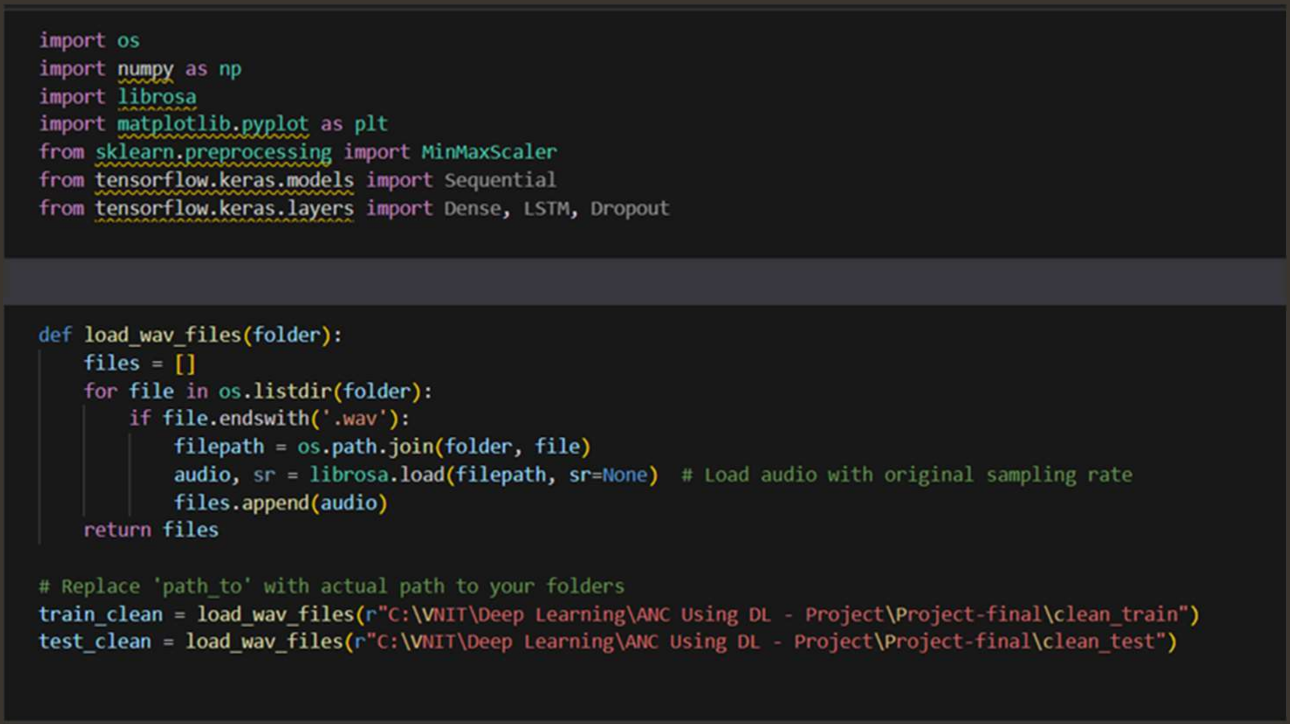
Key Goals:

•To train a deep learning model capable of distinguishing between noise and relevant audio in real-time.

•To reduce the noise levels in audio signals across different environments (e.g., traffic, wind, crowd noise).

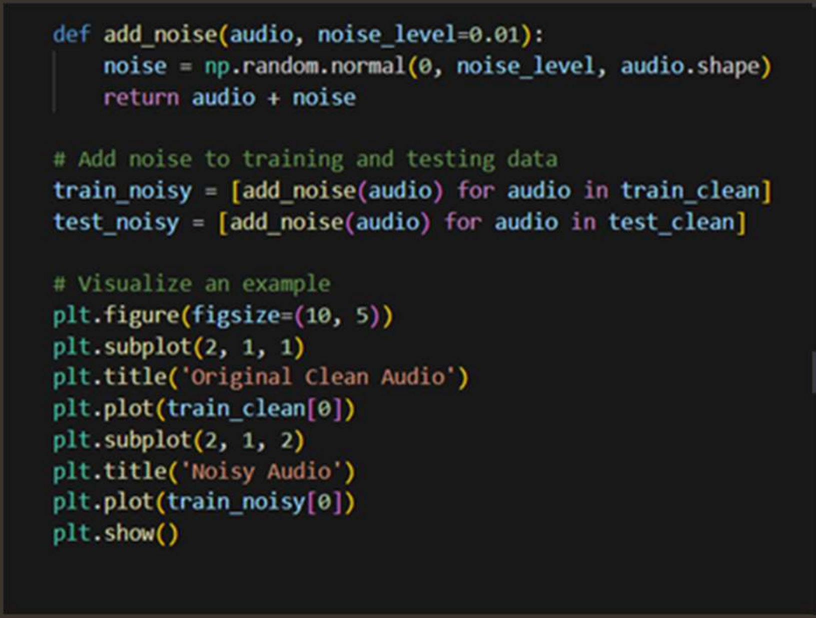
•To deploy a scalable and efficient solution that can be integrated into real-world applications like voice communication, video conferencing, and entertainment systems.

CODE

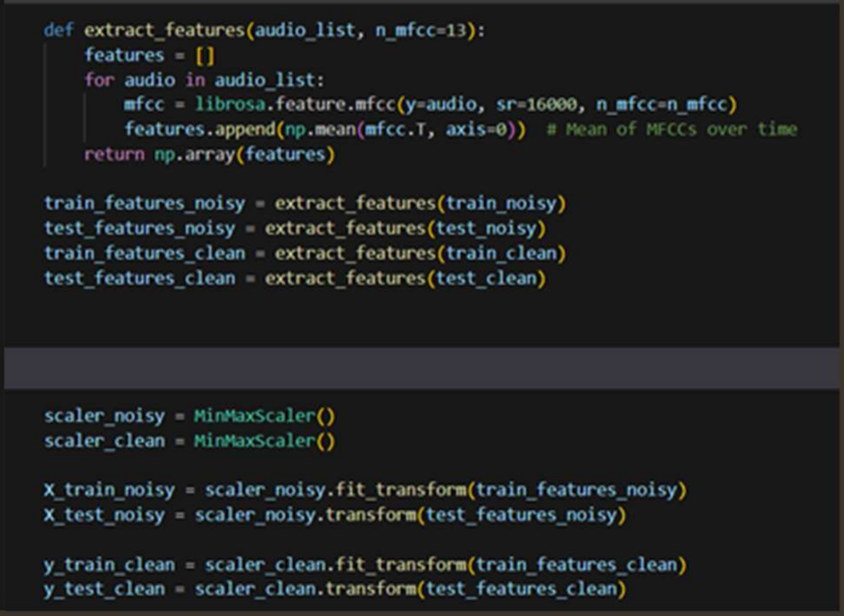


```
import os
import numpy as np
import librosa
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
```

```
def load_wav_files(folder):
    files = []
    for file in os.listdir(folder):
        if file.endswith('.wav'):
            filepath = os.path.join(folder, file)
            audio, sr = librosa.load(filepath, sr=None)  # Load audio with original sampling rate
            files.append(audio)
    return files

# Replace 'path_to' with actual path to your folders
train_clean = load_wav_files(r"C:\VNIT\Deep Learning\ANC Using DL - Project\Project-final\clean_train")
test_clean = load_wav_files(r"C:\VNIT\Deep Learning\ANC Using DL - Project\Project-final\clean_test")
```

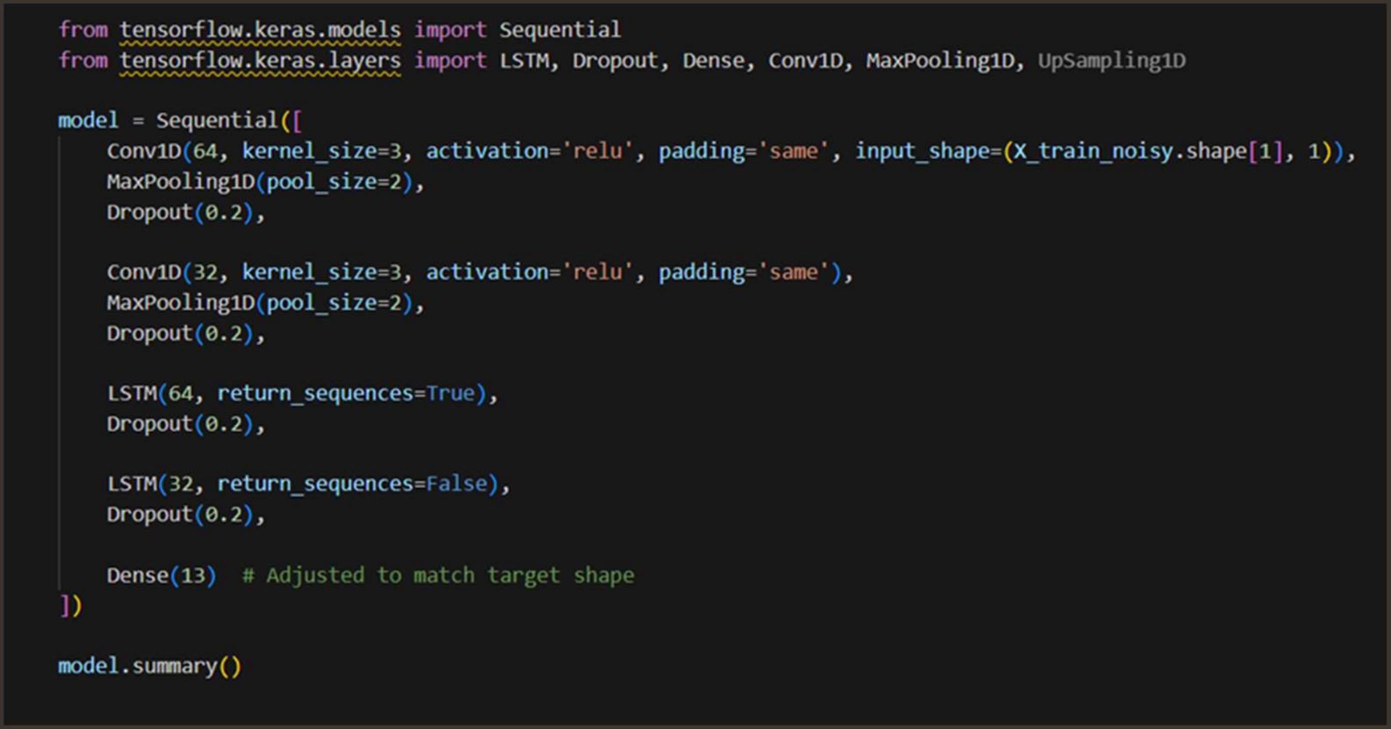


```python
def add_noise(audio, noise_level=0.01):
    noise = np.random.normal(0, noise_level, audio.shape)
    return audio + noise

# Add noise to training and testing data
train_noisy = [add_noise(audio) for audio in train_clean]
test_noisy = [add_noise(audio) for audio in test_clean]

# Visualize an example
plt.figure(figsize=(10, 5))
plt.subplot(2, 1, 1)
plt.title('Original Clean Audio')
plt.plot(train_clean[0])
plt.subplot(2, 1, 2)
plt.title('Noisy Audio')
plt.plot(train_noisy[0])
plt.show()
```

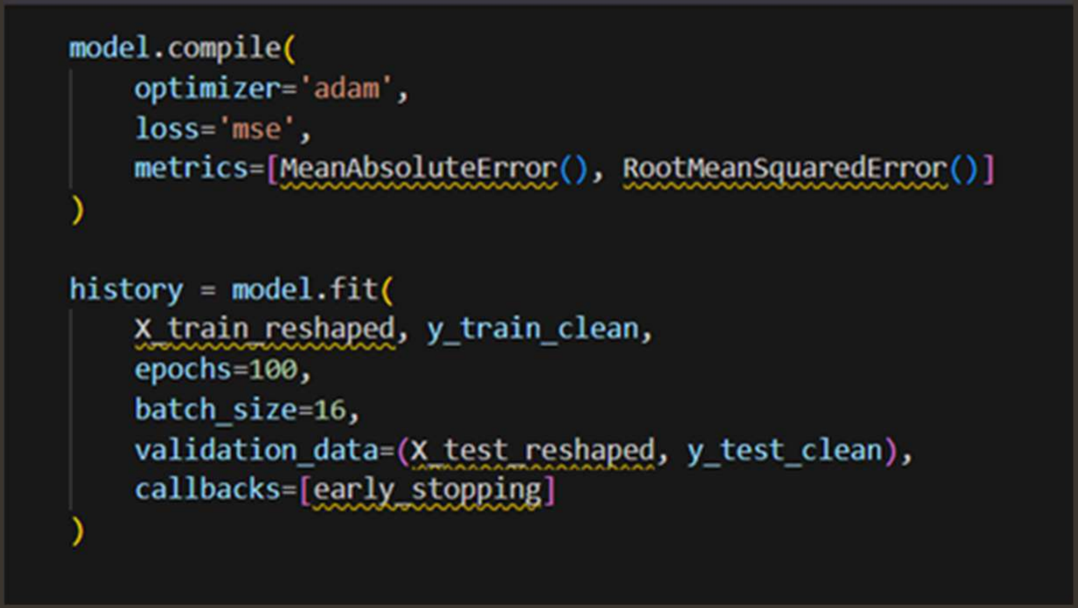


```python
def extract_features(audio_list, n_mfcc=13):
    features = []
    for audio in audio_list:
        mfcc = librosa.feature.mfcc(y=audio, sr=16000, n_mfcc=n_mfcc)
        features.append(np.mean(mfcc.T, axis=0))  # Mean of MFCCs over time
    return np.array(features)

train_features_noisy = extract_features(train_noisy)
test_features_noisy = extract_features(test_noisy)
train_features_clean = extract_features(train_clean)
test_features_clean = extract_features(test_clean)
```

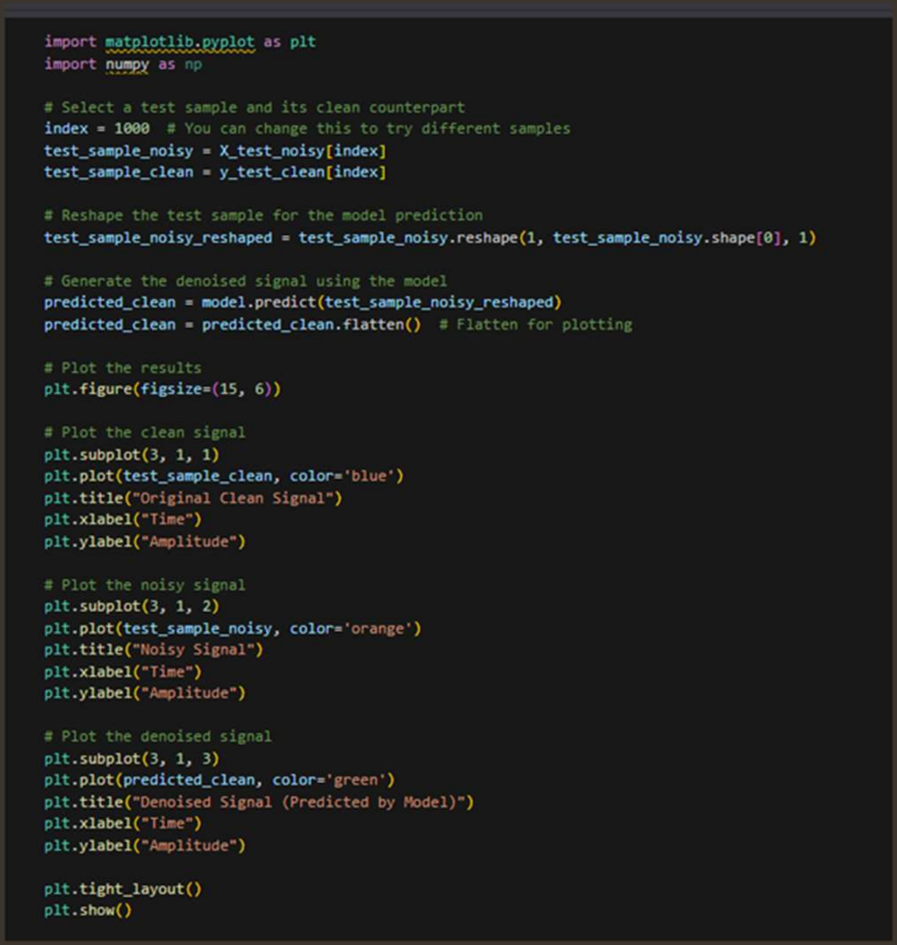
```python
scaler_noisy = MinMaxScaler()
scaler_clean = MinMaxScaler()

X_train_noisy = scaler_noisy.fit_transform(train_features_noisy)
X_test_noisy = scaler_noisy.transform(test_features_noisy)

y_train_clean = scaler_clean.fit_transform(train_features_clean)
y_test_clean = scaler_clean.transform(test_features_clean)
```



```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dropout, Dense, Conv1D, MaxPooling1D, UpSampling1D

model = Sequential([
    Conv1D(64, kernel_size=3, activation='relu', padding='same', input_shape=(X_train_noisy.shape[1], 1)),
    MaxPooling1D(pool_size=2),
    Dropout(0.2),

    Conv1D(32, kernel_size=3, activation='relu', padding='same'),
    MaxPooling1D(pool_size=2),
    Dropout(0.2),

    LSTM(64, return_sequences=True),
    Dropout(0.2),

    LSTM(32, return_sequences=False),
    Dropout(0.2),

    Dense(13)  # Adjusted to match target shape
])

model.summary()
```



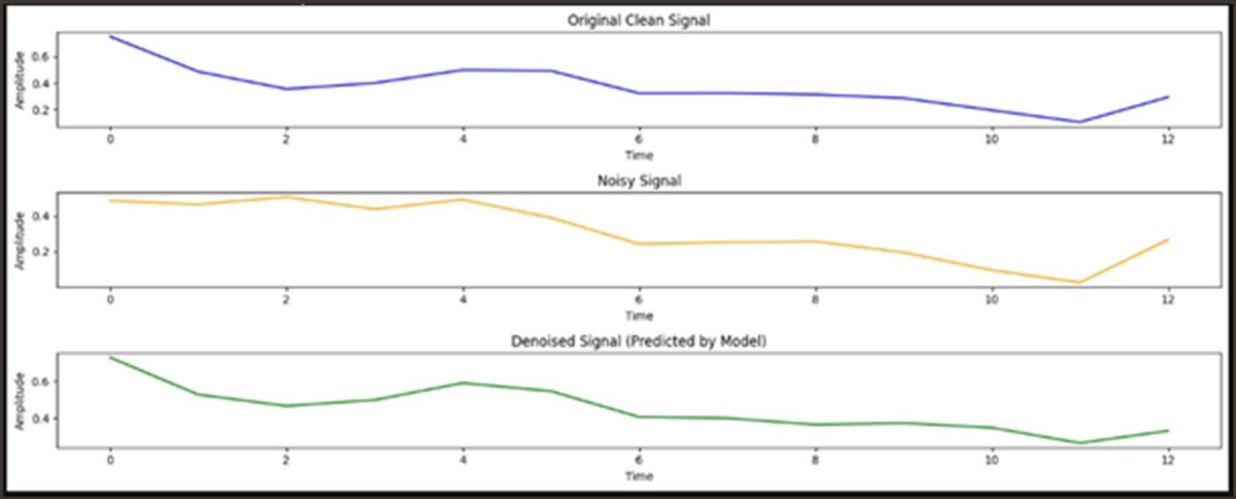
```python
model.compile(
    optimizer='adam',
    loss='mse',
    metrics=[MeanAbsoluteError(), RootMeanSquaredError()]
)

history = model.fit(
    X_train_reshaped, y_train_clean,
    epochs=100,
    batch_size=16,
    validation_data=(X_test_reshaped, y_test_clean),
    callbacks=[early_stopping]
)
```

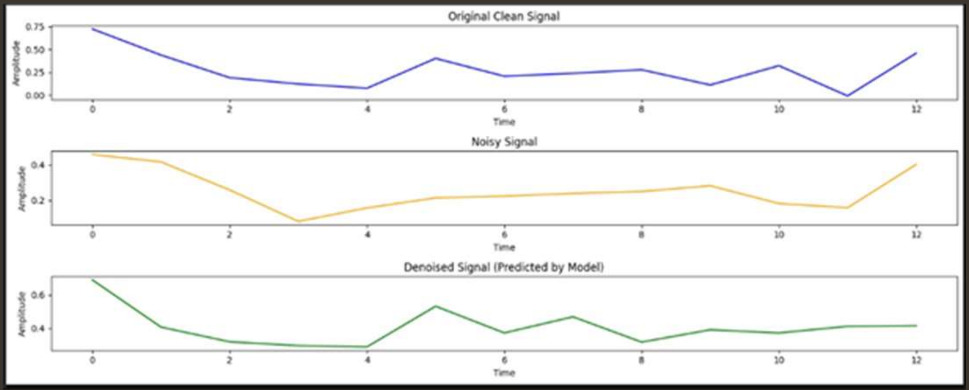


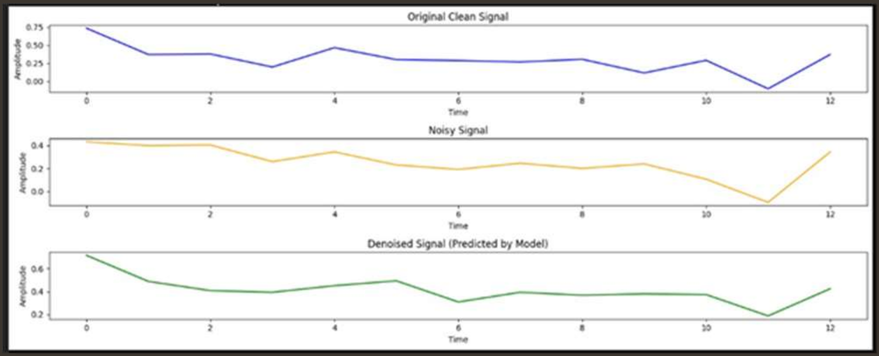
```python
import matplotlib.pyplot as plt
import numpy as np

# Select a test sample and its clean counterpart
index = 1000  # You can change this to try different samples
test_sample_noisy = X_test_noisy[index]
test_sample_clean = y_test_clean[index]

# Reshape the test sample for the model prediction
test_sample_noisy_reshaped = test_sample_noisy.reshape(1, test_sample_noisy.shape[0], 1)

# Generate the denoised signal using the model
predicted_clean = model.predict(test_sample_noisy_reshaped)
predicted_clean = predicted_clean.flatten()  # Flatten for plotting

# Plot the results
plt.figure(figsize=(15, 6))

# Plot the clean signal
plt.subplot(3, 1, 1)
plt.plot(test_sample_clean, color='blue')
plt.title("Original Clean Signal")
plt.xlabel("Time")
plt.ylabel("Amplitude")

# Plot the noisy signal
plt.subplot(3, 1, 2)
plt.plot(test_sample_noisy, color='orange')
plt.title("Noisy Signal")
plt.xlabel("Time")
plt.ylabel("Amplitude")

# Plot the denoised signal
plt.subplot(3, 1, 3)
plt.plot(predicted_clean, color='green')
plt.title("Denoised Signal (Predicted by Model)")
plt.xlabel("Time")
plt.ylabel("Amplitude")

plt.tight_layout()
plt.show()
```

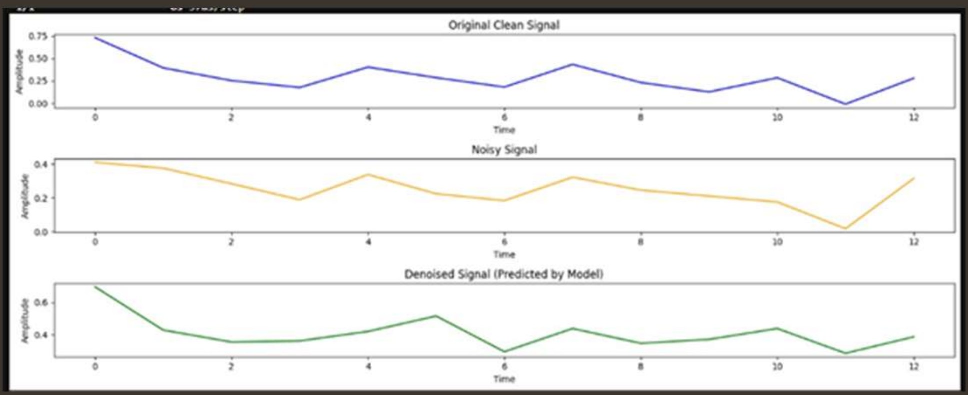
RESULTS

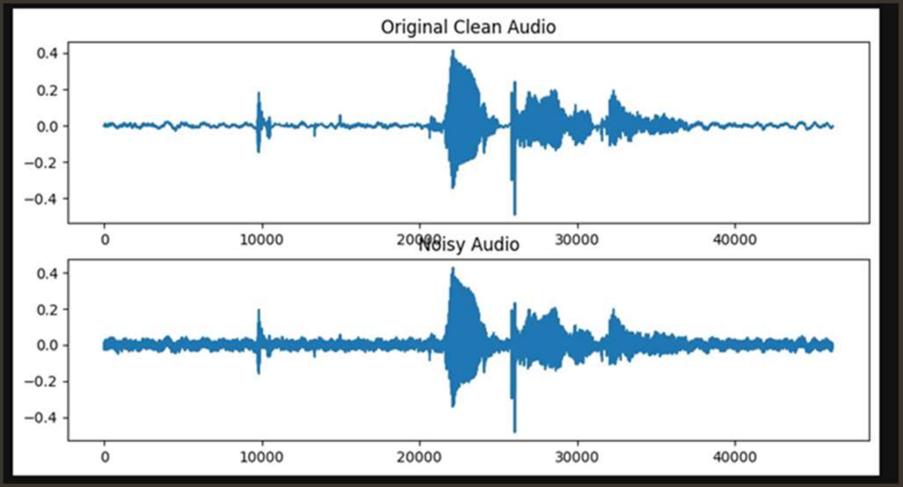
The Conv1D-LSTM model achieved low MAE and RMSE scores, indicating strong performance in reconstructing clean audio. Visual results showed significant noise reduction, with cleaner audio outputs in test samples.

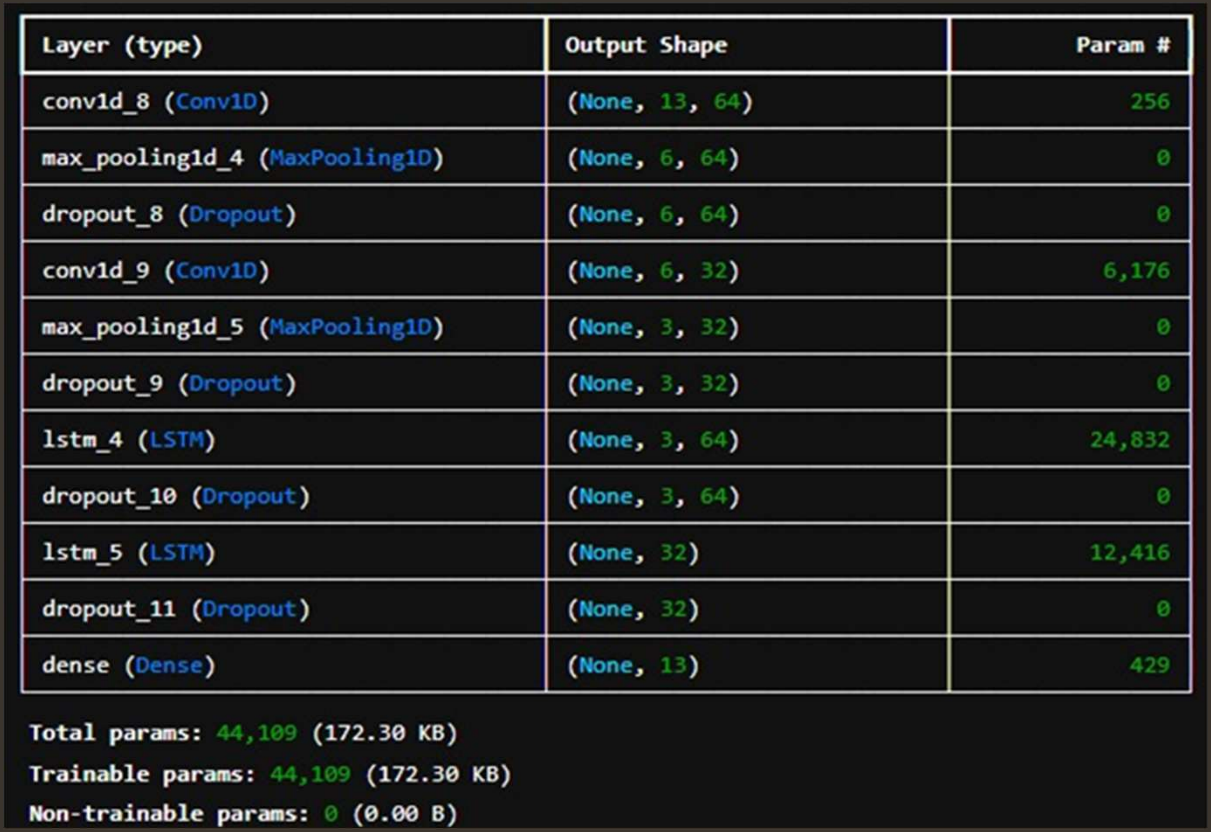



Original Clean Signal
Noisy Signal
Denoised Signal (Predicted by Model)
Amplitude
Time

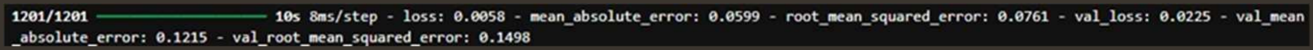

Original Clean Signal
Noisy Signal
Denoised Signal (Predicted by Model)
Amplitude
Time


Original Clean Signal
Noisy Signal
Denoised Signal (Predicted by Model)
Amplitude
Time


Original Clean Audio
Noisy Audio



```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃        Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━┩
│ conv1d_8 (Conv1D)                    │ (None, 13, 64)              │            256 │
├──────────────────────────────────────┼─────────────────────────────┼────────────────┤
│ max_pooling1d_4 (MaxPooling1D)       │ (None, 6, 64)               │              0 │
├──────────────────────────────────────┼─────────────────────────────┼────────────────┤
│ dropout_8 (Dropout)                  │ (None, 6, 64)               │              0 │
├──────────────────────────────────────┼─────────────────────────────┼────────────────┤
│ conv1d_9 (Conv1D)                    │ (None, 6, 32)               │          6,176 │
├──────────────────────────────────────┼─────────────────────────────┼────────────────┤
│ max_pooling1d_5 (MaxPooling1D)       │ (None, 3, 32)               │              0 │
├──────────────────────────────────────┼─────────────────────────────┼────────────────┤
│ dropout_9 (Dropout)                  │ (None, 3, 32)               │              0 │
├──────────────────────────────────────┼─────────────────────────────┼────────────────┤
│ lstm_4 (LSTM)                        │ (None, 3, 64)               │         24,832 │
├──────────────────────────────────────┼─────────────────────────────┼────────────────┤
│ dropout_10 (Dropout)                 │ (None, 3, 64)               │              0 │
├──────────────────────────────────────┼─────────────────────────────┼────────────────┤
│ lstm_5 (LSTM)                        │ (None, 32)                  │         12,416 │
├──────────────────────────────────────┼─────────────────────────────┼────────────────┤
│ dropout_11 (Dropout)                 │ (None, 32)                  │              0 │
├──────────────────────────────────────┼─────────────────────────────┼────────────────┤
│ dense (Dense)                        │ (None, 13)                  │            429 │
└──────────────────────────────────────┴─────────────────────────────┴────────────────┘

 Total params: 44,109 (172.30 KB)
 Trainable params: 44,109 (172.30 KB)
 Non-trainable params: 0 (0.00 B)
```



```
1201/1201 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - loss: 0.0058 - mean_absolute_error: 0.0599 - root_mean_squared_error: 0.0761 - val_loss: 0.0225 - val_mean
_absolute_error: 0.1215 - val_root_mean_squared_error: 0.1498
```

DISCUSSION AND FUTURE WORK & RECOMMENDATIONS

The model demonstrated robust noise cancellation abilities, adapting to various noise types. This deep learning approach provides flexibility and accuracy superior to traditional filtering methods, especially in dynamic noise environment.

Future Work & Recommendations

·Experiment with transformer-based models to explore potentially greater noise suppression.
·Extend the model for real-time audio processing in streaming applications.

SUMMARY

Our project aims to develop a deep learning-based noise cancellation system that efficiently eliminates unwanted background noise, ensuring the clarity of desired audio like speech and music. This system is designed for real-world applications such as voice communication, video conferencing, and entertainment systems.

Key Objectives:

- Train a deep learning model to differentiate between noise and relevant audio in real-time.
- Minimize noise in diverse environments, including traffic and crowd noise.
- Deploy a scalable, efficient solution for real-world use cases.

In summary, this project leverages deep learning and the DNS Challenge dataset to create an efficient noise cancellation system for real-world audio applications.

REFERENCES

1. Dogra, M., Borwankar, S., & Domala, J. Noise Removal from Audio Using CNN and Denoiser.
2. Cherukuru, P., & Mustafa, M. B. CNN-based Noise Reduction for Multi-Channel Speech Enhancement System with Discrete Wavelet Transform (DWT) Preprocessing.
3. Park, S. R., & Lee, J. W. A Fully Convolutional Neural Network for Speech Enhancement.
4. Silva, T. S. How to Build a Deep Audio De-Noiser Using TensorFlow 2.0. Better Programming, December 1, 2019.
5. Shajeesh, K. U., & Soman, K. P. Noise Cancellation Method for Robust Speech Recognition.
6. Kejalakshmi, V., Kamatchi, A., & Anusuya, M. (Year). Active Noise Cancellation using Deep Learning.
7. Zhang, H., & Wang, D. (Year). Deep ANC: A Deep Learning Approach to Active Noise Control.
8. Cha, Y-J., Mostafavi, A., & Benipa, S. S. (Year). DNoiseNet: Deep Learning-Based Feedback Active Noise Control.
9. Lin, C-M., TsaoChu, H-C., Lan, S-W., & Fang, S-H. (Year). Adaptive Noise Cancellation Using DCMAC.
10. Beniwa, P., & Jain, D. (Year). Noise Cancellation Using Adaptive Filters.
11. DNS Challenge Dataset. (n.d.). Retrieved from https://github.com/microsoft/DNS-Challenge/tree/master
12. Noise Reduction. (n.d.). Retrieved from https://github.com/topics/noise-reduction
13. Dhriti03/Noise-Reduction. (n.d.). Retrieved from https://github.com/Dhriti03/Noise-Reduction
14. Rdadlaney/Audio-Denoiser-CNN. (n.d.). Retrieved from https://github.com/rdadlaney/Audio-Denoiser-CNN
15. Ahmetcanaydemir/sekte. (n.d.). Retrieved from https://github.com/ahmetcanaydemir/sekte
16. TensorFlow Models: YamNet. (n.d.). Retrieved from https://github.com/tensorflow/models/tree/master/research/audioset/yamnet
17. NVIDIA Blog. (n.d.). NVIDIA Real-Time Noise Suppression with Deep Learning. Retrieved from https://developer.nvidia.com/blog/nvidia-real-time-noise-suppression-deep-learning/
18. EncoraDigital/SAB-cnn-audio-denoiser. (n.d.). Retrieved from https://github.com/EncoraDigital/SAB-cnn-audio-denoiser
19. Microsoft DNS Challenge Dataset (2020). [Link to Dataset]
20. Y. Ephraim and D. Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator," in IEEE Transactions on Acoustics, Speech, and Signal Processing, 1984.
21. P. C. Loizou, "Speech Enhancement: Theory and Practice," CRC Press, 2007.

Thank You