Adaptive Noise Cancellation System Using AI

Submitted By:

Aadityan Gupta
2110110002

Bhavya Puri
2110110166

Under Supervision of:

Mr. Aakash Kumar Sinha
Department of Electrical Engineering

SHIV NADAR
INSTITUTION OF EMINENCE DEEMED TO BE
UNIVERSITY
DELHI NCR
SCHOOL OF ENGINEERING

TABLE OF CONTENTS

ABSTRACT

The project titled ”Adaptive Noise Cancellation System Using AI” aims to design and implement an advanced noise cancellation system that dynamically reduces unwanted ambient sounds using artificial intelligence. The system will employ a microphone array to capture environmental noise, which will then be processed in real time using sophisticated signal processing algorithms. Leveraging adaptive machine learning techniques, the system will effectively filter out unwanted noise while preserving the integrity of the desired audio. This technology is intended to enhance audio quality in various settings, from personal audio devices to large-scale industrial applications. The adaptive nature of the AI-driven approach ensures that the system continuously learns and improves its noise cancellation capabilities, making it highly efficient across different environments and noise profiles.

INTRODUCTION & OVERVIEW

Introduction:

- AI-powered adaptive noise cancellation (ANC) systems combat noise disturbances in real-time.
- These systems eliminate background noises, enhancing the user's focus on preferred audio inputs like speech or music.

Overview:

- Adaptive ANC systems adjust dynamically to varying environments.
- They include three main components:
  - **Microphones:** Capture ambient sounds and user audio.
  - **Signal Processing Unit:** Analyzes incoming audio signals and applies noise reduction algorithms.
  - **Output Mechanism:** Delivers processed audio to headphones or speakers.

TECHNIQUES & APPLICATIONS

Techniques:

- **Active Noise Cancellation (ANC):** Uses microphones to detect and generate anti-noise signals, effectively cancelling out unwanted sounds.
- **Passive Noise Cancellation:** Relies on physical barriers to block external sounds.
- **AI-Based Noise Cancellation:** Enhances traditional ANC methods by analyzing and adapting to environmental sounds in real-time.
- **Hybrid Noise Cancellation:** Combines active and passive techniques for superior sound isolation and clarity.

Applications:

- Used in consumer electronics, telecommunications, automotive industries, and healthcare.
- Enhances listening experiences, voice clarity, and reduces cabin noise.
- Important in modern communication devices, hearing aids, and various noisy environments.

WORK DONE TILL MIDTERM

- **Objective:** Design and implement a system that dynamically reduces unwanted ambient noise while preserving desired audio using artificial intelligence.

- **Key Techniques:**
    - Leverage AI-based denoising using pre-trained models.
    - Implement real-time signal processing for audio clarity.

- **Dataset**:
    - Noise-reduced audio generated using tools like MyEditOnline and processed with Python libraries (e.g., librosa).
    - Performed Short-Time Fourier Transform (STFT) to analyze the frequency domain.

- **Midterm Focus:**
    - Experimented with existing pre-trained models to understand denoising behavior.
    - Processed noisy audio samples for denoising evaluation.

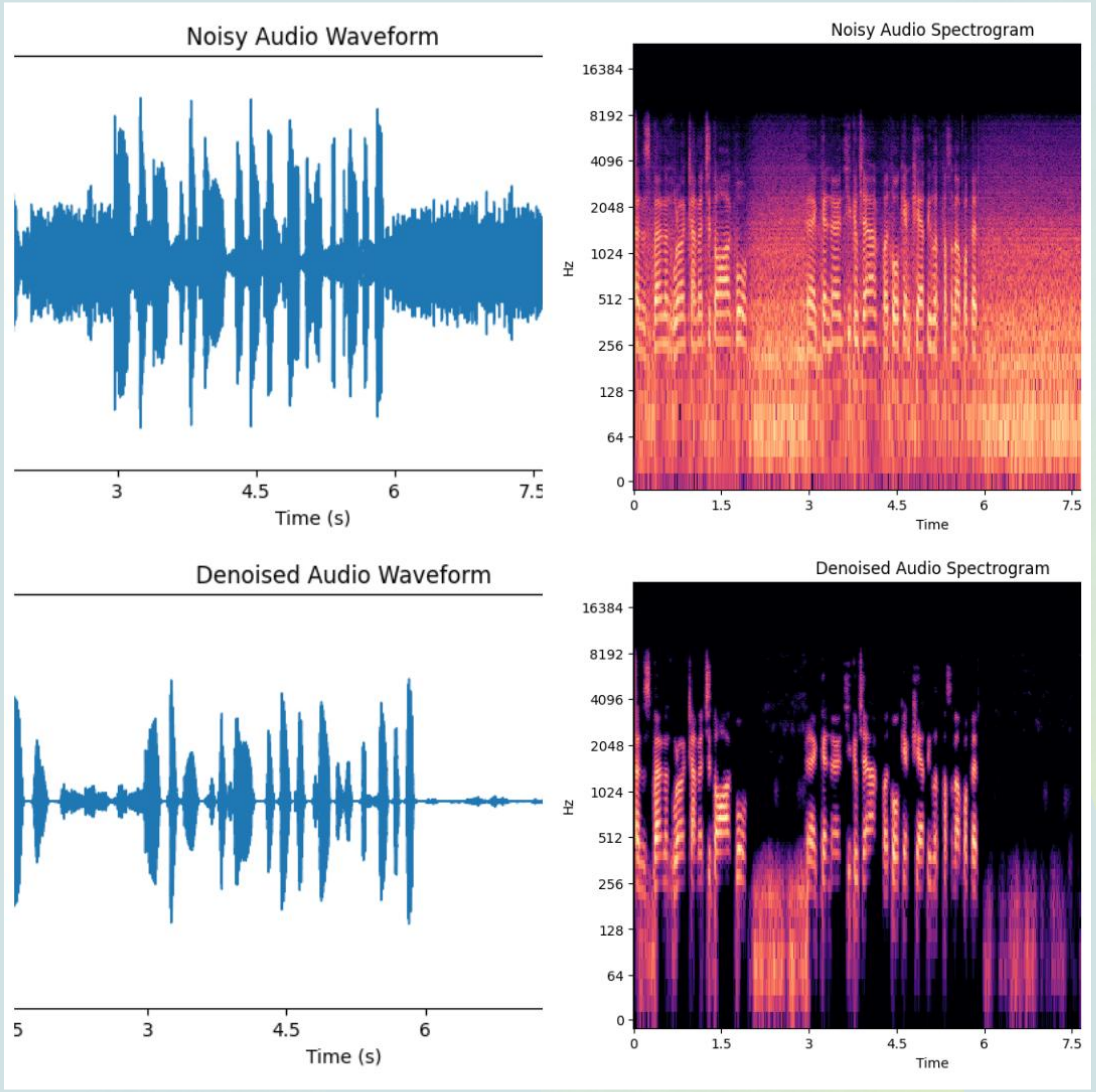
WORK DONE TILL MIDTERM

- **Implementation:**
  - Used the noisereduce library to reduce noise in an audio sample.
  - Visualized waveforms and spectrograms of noisy and denoised audio.

- **Developed scripts for:**
  - Loading and preprocessing audio.
  - Noise profile extraction and reduction.

- **Tools:** Google Colab, Python (librosa, matplotlib, torch, etc.).

- **Results:**
  - Successfully obtained denoised audio with improved clarity.
  - **Outputs:**
    1. Waveform and spectrogram plots comparing noisy and clean audio.
    2. Denoised Audio File.



Noisy Audio Waveform
3
4.5
6
7.5
Time (s)
Noisy Audio Spectrogram
16384
8192
4096
2048
1024
512
256
128
64
0
Hz
0
1.5
3
4.5
6
7.5
Time
Denoised Audio Waveform
5
3
4.5
6
Time (s)
Denoised Audio Spectrogram
16384
8192
4096
2048
1024
512
256
128
64
0
Hz
0
1.5
3
4.5
6
7.5
Time

WORK DONE POST MIDTERM

Data Collection

Added more audio samples from the Voicebank-DEMAND dataset.

Model Development

Built a custom neural network for noise cancellation

Training

Trained the model on noisy-clean audio pairs using MSE loss.

Evaluation

Assessed model performance with SNR and PESQ metrics.

Refinement

Tuned hyperparameters for better accuracy and generalization.

Integration

Integrated noise cancellation with visualization tools.

Testing

Tested the system with diverse noise profiles and gathered feedback.

Finalization

Finalized the system and prepared for presentation.

ACTIVE NOISE CANCELLATION (ANC) PROJECT OVERVIEW

End-Semester Objective

•Develop and train a custom ANC model without pre-existing libraries.

•Key Components:

- Audio Recording: Record or input real-time audio.
- Noise Removal: Use the trained model to filter out unwanted noise.
- Output Generation: Produce a clean, denoised audio file.

Dataset Used

1.VoiceBank-DEMAND:

1. Training Data: 11,572 paired clean-noisy speech files from 28 speakers.
2. Noise Types: Environmental sounds (e.g., street, park, kitchen).
3. Features: Ideal for supervised ANC tasks with standardized WAV format.

Comparison with Other Datasets

•DNS Challenge: Less realistic due to synthetic noise.

•CHiME: Focused on noisy ASR but lacks paired clean-noisy data.

•ESC-50/LibriSpeech: Requires preprocessing to focus on ANC tasks.

Dataset.py

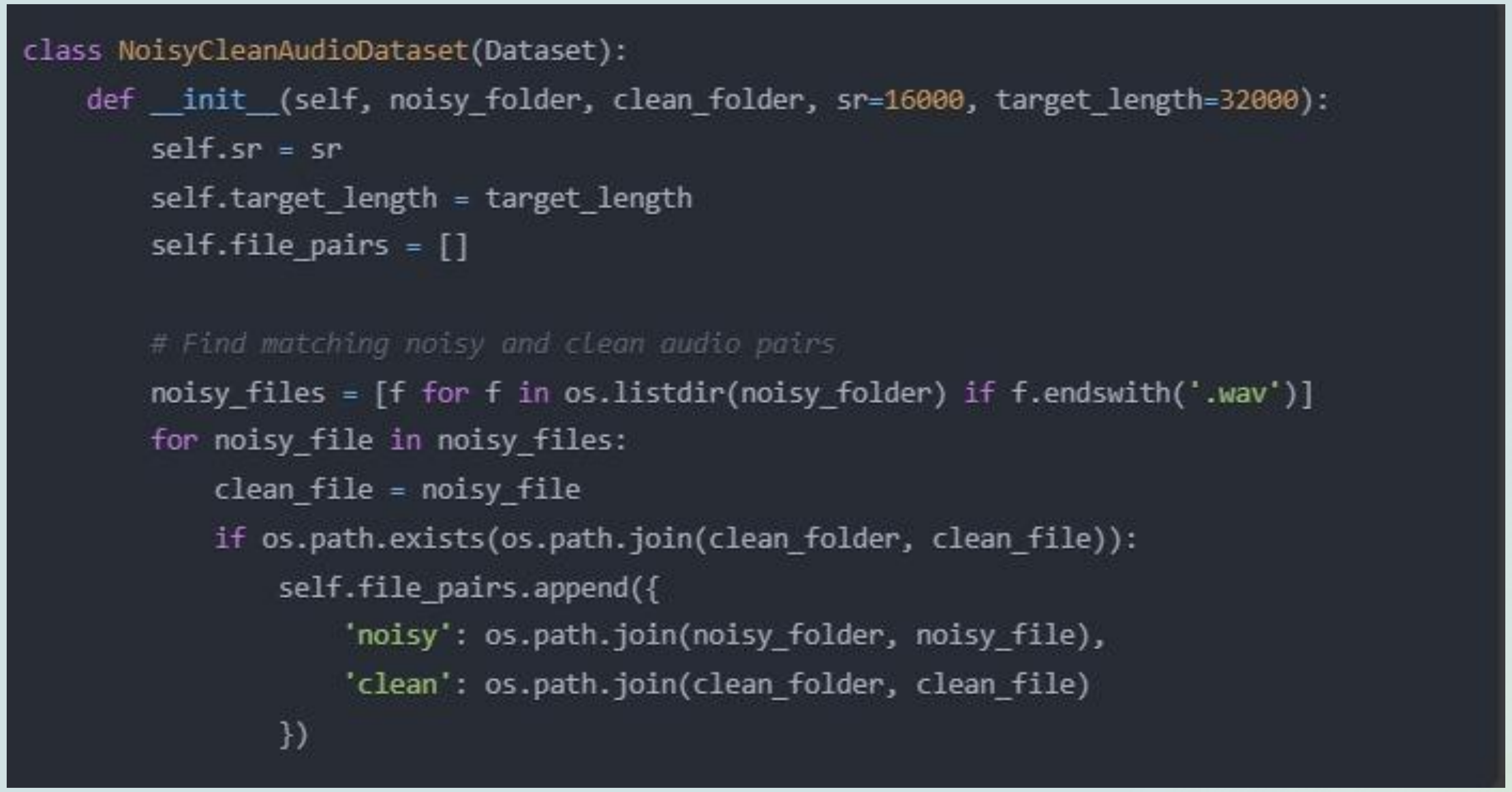
Data Implementation

Inherits from PyTorch's Dataset class:

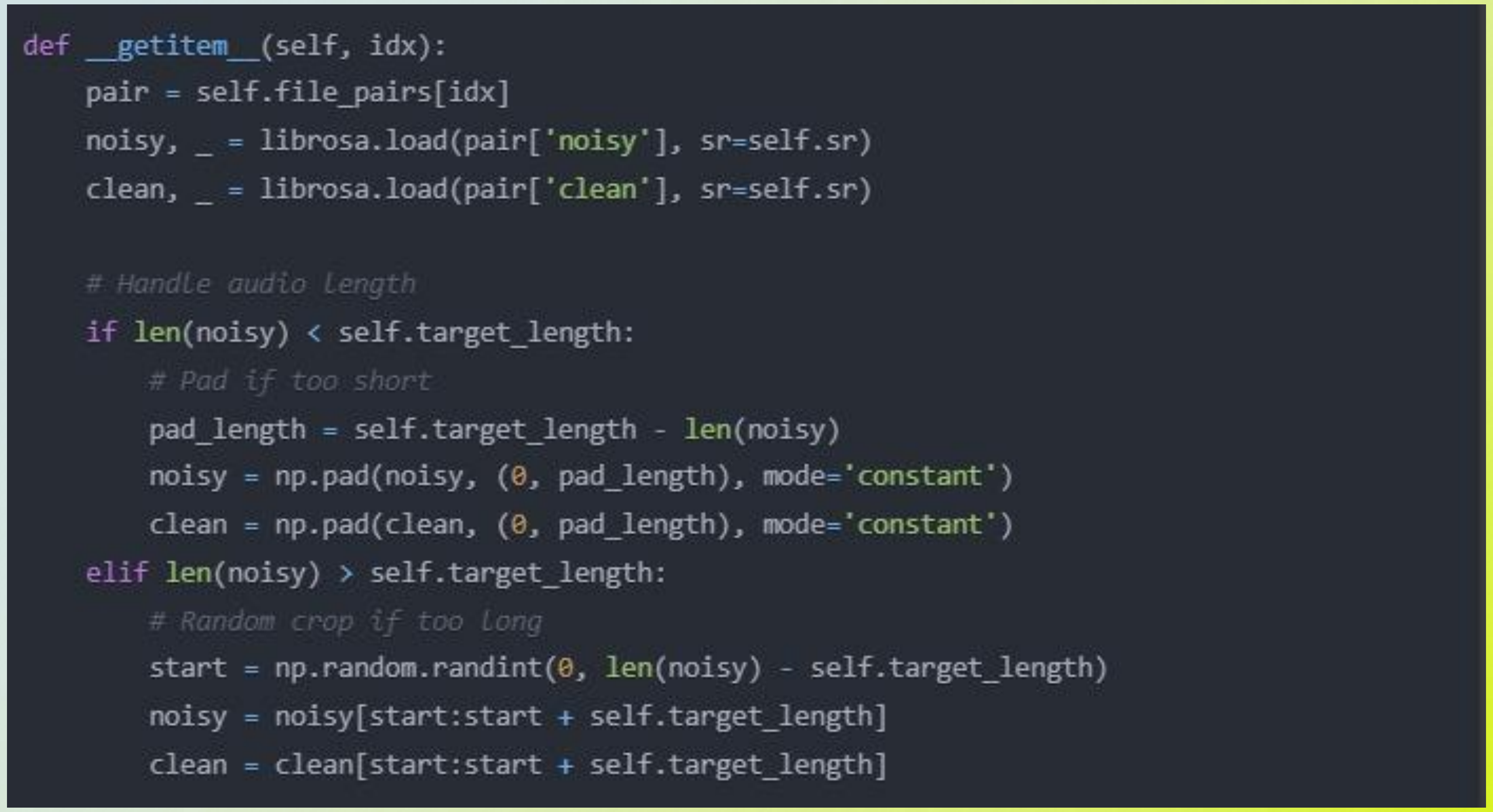
- Initializes with folders containing noisy and clean audio pairs
- Default sampling rate: 16kHz
- Target length: 32000 samples (2 seconds of audio)
- Creates pairs of matching noisy and clean files

Data Loading and Processing

- Loads audio pairs using librosa
- Handles variable length audio
- Pads shorter segments with zeros
- Randomly crops longer segments
- Returns tensors ready for model input



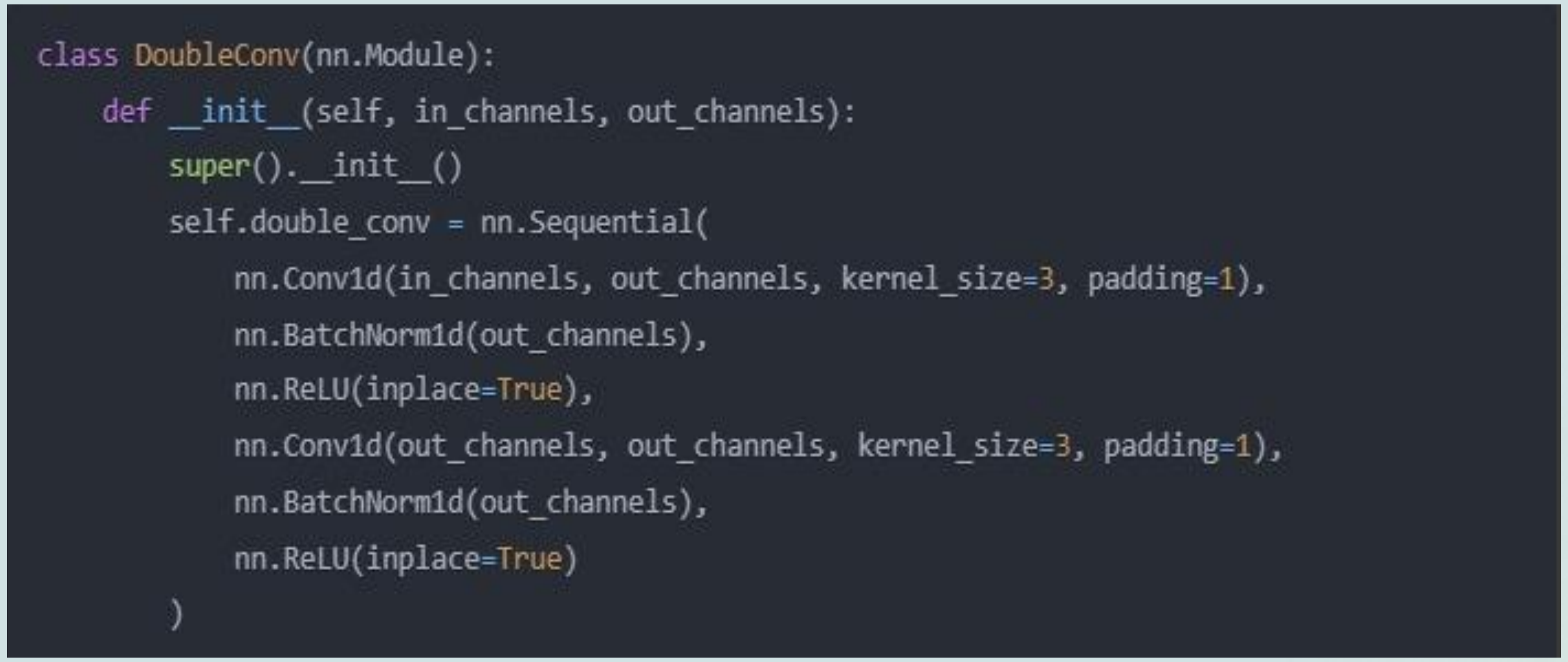
```python
class NoisyCleanAudioDataset(Dataset):
    def __init__(self, noisy_folder, clean_folder, sr=16000, target_length=32000):
        self.sr = sr
        self.target_length = target_length
        self.file_pairs = []

        # Find matching noisy and clean audio pairs
        noisy_files = [f for f in os.listdir(noisy_folder) if f.endswith('.wav')]
        for noisy_file in noisy_files:
            clean_file = noisy_file
            if os.path.exists(os.path.join(clean_folder, clean_file)):
                self.file_pairs.append({
                    'noisy': os.path.join(noisy_folder, noisy_file),
                    'clean': os.path.join(clean_folder, clean_file)
                })
```



```python
def __getitem__(self, idx):
    pair = self.file_pairs[idx]
    noisy, _ = librosa.load(pair['noisy'], sr=self.sr)
    clean, _ = librosa.load(pair['clean'], sr=self.sr)

    # Handle audio length
    if len(noisy) < self.target_length:
        # Pad if too short
        pad_length = self.target_length - len(noisy)
        noisy = np.pad(noisy, (0, pad_length), mode='constant')
        clean = np.pad(clean, (0, pad_length), mode='constant')
    elif len(noisy) > self.target_length:
        # Random crop if too long
        start = np.random.randint(0, len(noisy) - self.target_length)
        noisy = noisy[start:start + self.target_length]
        clean = clean[start:start + self.target_length]
```

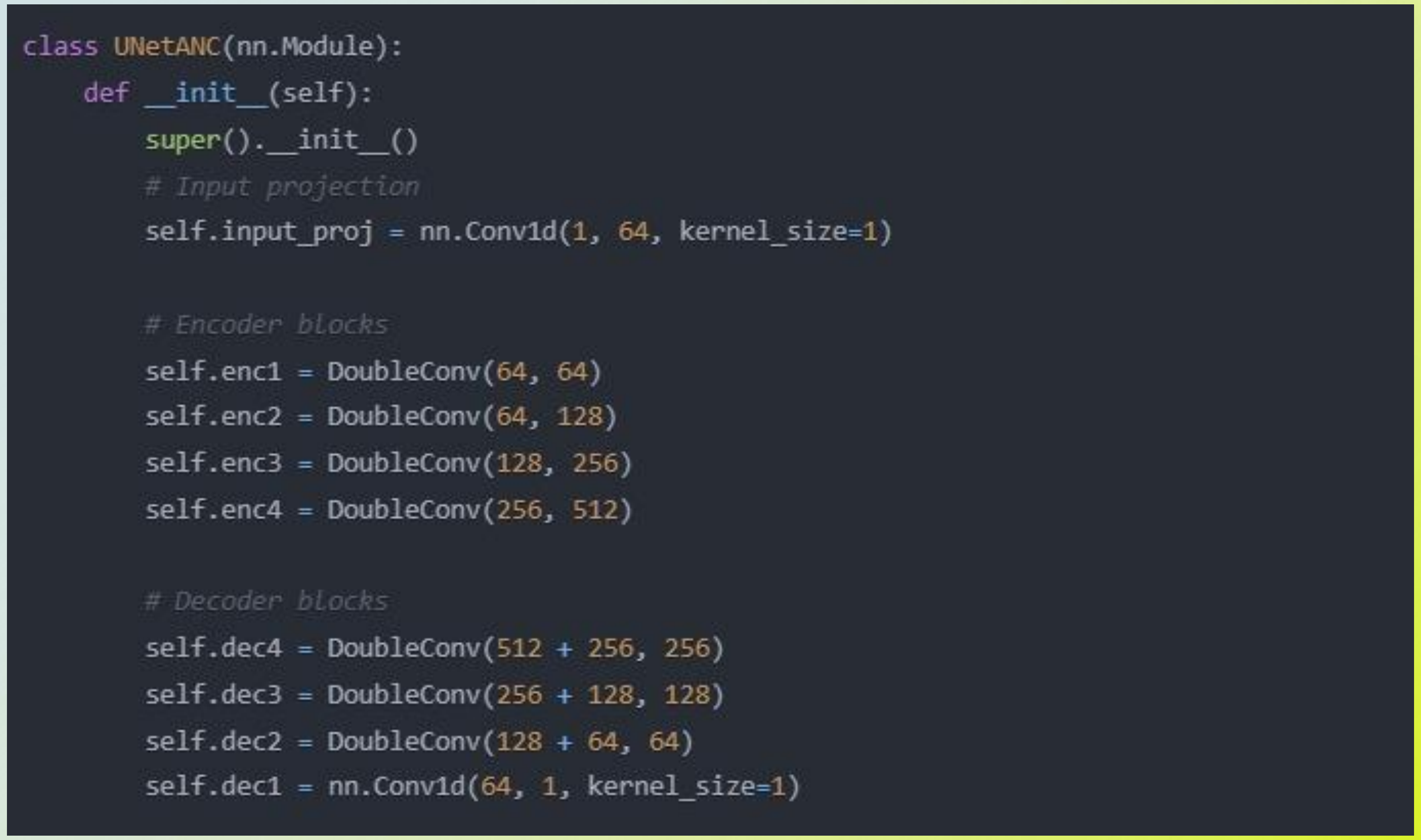
Model Architecture (model.py)

Basic building block of the U-Net
Features:
- Two 1D convolution layers
- Batch normalization for stability
- ReLU activation for non-linearity
- Maintains temporal dimension with padding



```python
class DoubleConv(nn.Module):
    def __init__(self, in_channels, out_channels):
        super().__init__()
        self.double_conv = nn.Sequential(
            nn.Conv1d(in_channels, out_channels, kernel_size=3, padding=1),
            nn.BatchNorm1d(out_channels),
            nn.ReLU(inplace=True),
            nn.Conv1d(out_channels, out_channels, kernel_size=3, padding=1),
            nn.BatchNorm1d(out_channels),
            nn.ReLU(inplace=True)
        )
```

Explanation: -
- U-Net architecture adapted for 1D audio
- Encoder: progressively reduces temporal dimension
- Decoder: reconstructs audio with skip connections
- Skip connections preserve fine detail information

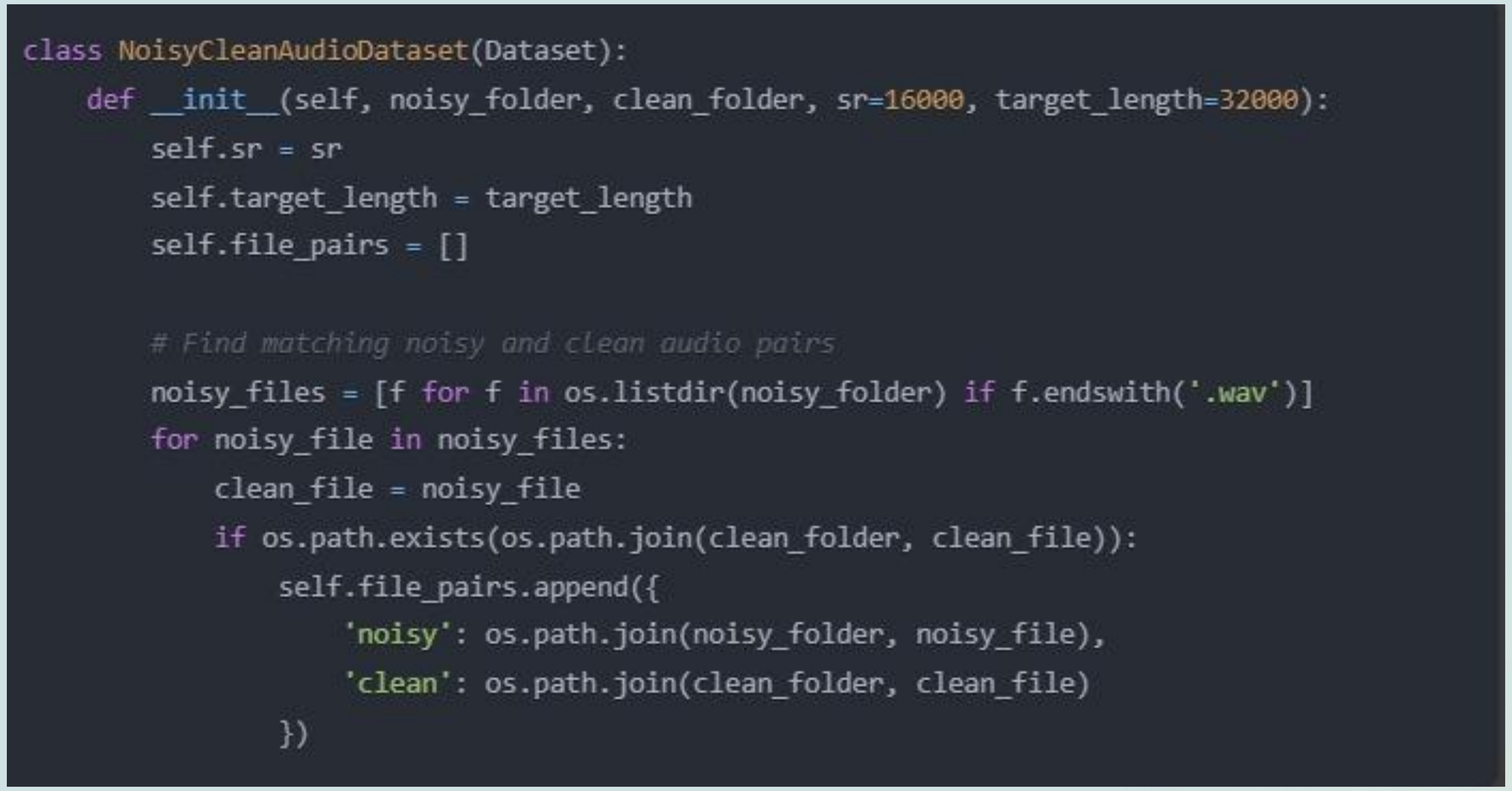


```python
class UNetANC(nn.Module):
    def __init__(self):
        super().__init__()
        # Input projection
        self.input_proj = nn.Conv1d(1, 64, kernel_size=1)

        # Encoder blocks
        self.enc1 = DoubleConv(64, 64)
        self.enc2 = DoubleConv(64, 128)
        self.enc3 = DoubleConv(128, 256)
        self.enc4 = DoubleConv(256, 512)

        # Decoder blocks
        self.dec4 = DoubleConv(512 + 256, 256)
        self.dec3 = DoubleConv(256 + 128, 128)
        self.dec2 = DoubleConv(128 + 64, 64)
        self.dec1 = nn.Conv1d(64, 1, kernel_size=1)
```

Train_model.py

Early Stopping

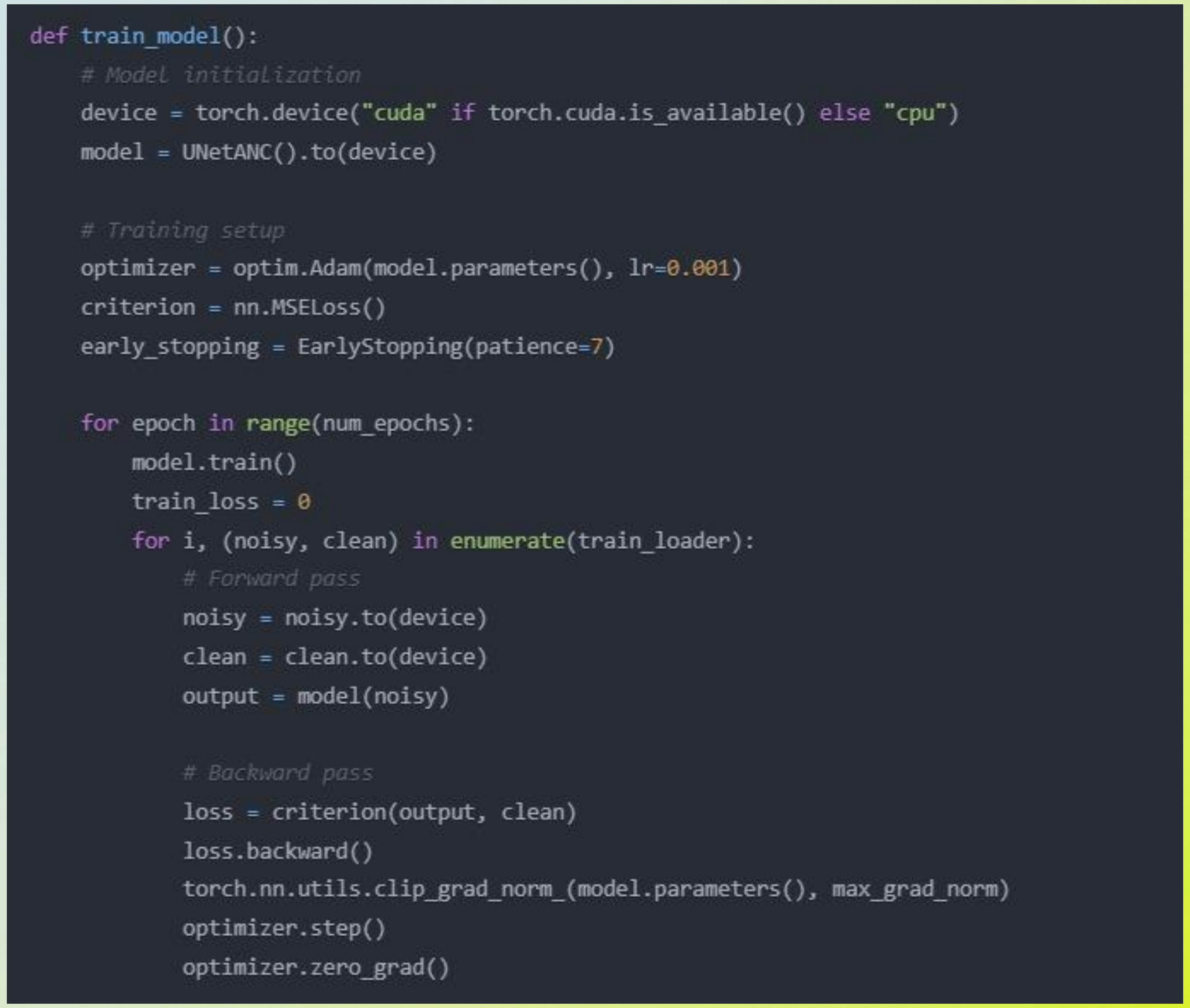
- Monitors validation loss
- Stops training if no improvement for 'patience' epochs
- Helps prevent overfitting
- Saves training time



```python
class NoisyCleanAudioDataset(Dataset):
    def __init__(self, noisy_folder, clean_folder, sr=16000, target_length=32000):
        self.sr = sr
        self.target_length = target_length
        self.file_pairs = []

        # Find matching noisy and clean audio pairs
        noisy_files = [f for f in os.listdir(noisy_folder) if f.endswith('.wav')]
        for noisy_file in noisy_files:
            clean_file = noisy_file
            if os.path.exists(os.path.join(clean_folder, clean_file)):
                self.file_pairs.append({
                    'noisy': os.path.join(noisy_folder, noisy_file),
                    'clean': os.path.join(clean_folder, clean_file)
                })
```

Training Loop

- Complete training pipeline-
- Features:
- Automatic device selection (CPU/GPU)
- Adam optimizer with learning rate 0.001
- MSE loss for audio quality
- Gradient clipping for stability



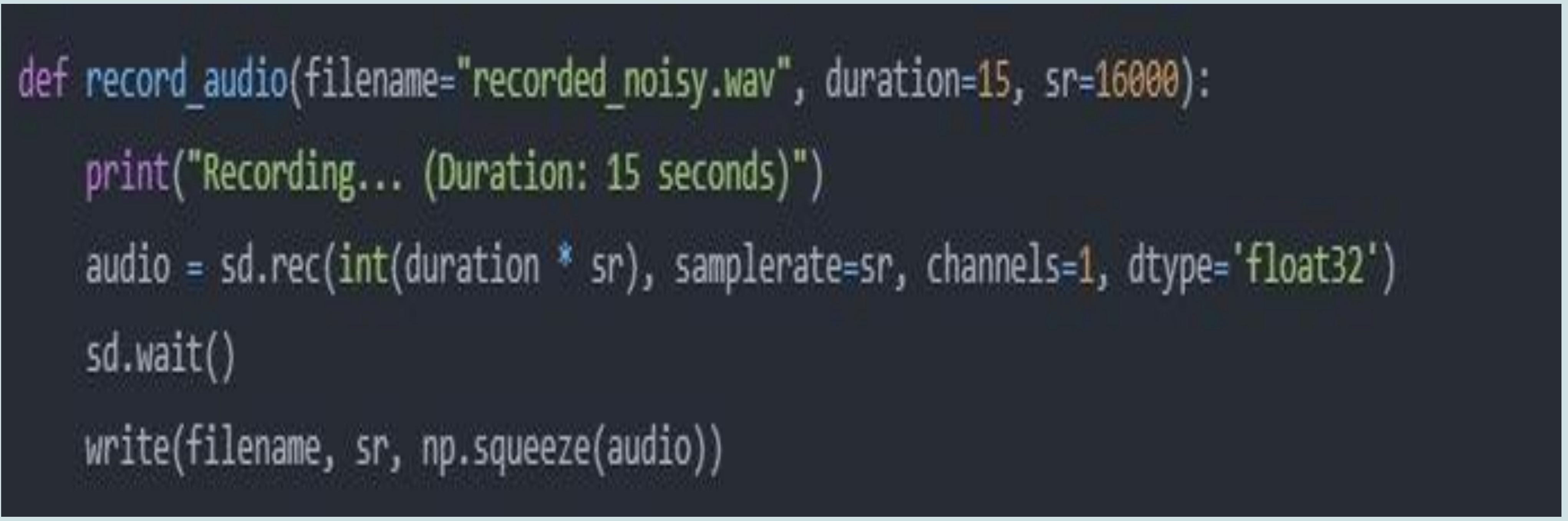
```python
def train_model():
    # Model initialization
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model = UNetANC().to(device)

    # Training setup
    optimizer = optim.Adam(model.parameters(), lr=0.001)
    criterion = nn.MSELoss()
    early_stopping = EarlyStopping(patience=7)

    for epoch in range(num_epochs):
        model.train()
        train_loss = 0
        for i, (noisy, clean) in enumerate(train_loader):
            # Forward pass
            noisy = noisy.to(device)
            clean = clean.to(device)
            output = model(noisy)

            # Backward pass
            loss = criterion(output, clean)
            loss.backward()
            torch.nn.utils.clip_grad_norm_(model.parameters(), max_grad_norm)
            optimizer.step()
            optimizer.zero_grad()
```

Inference System (inference.py)

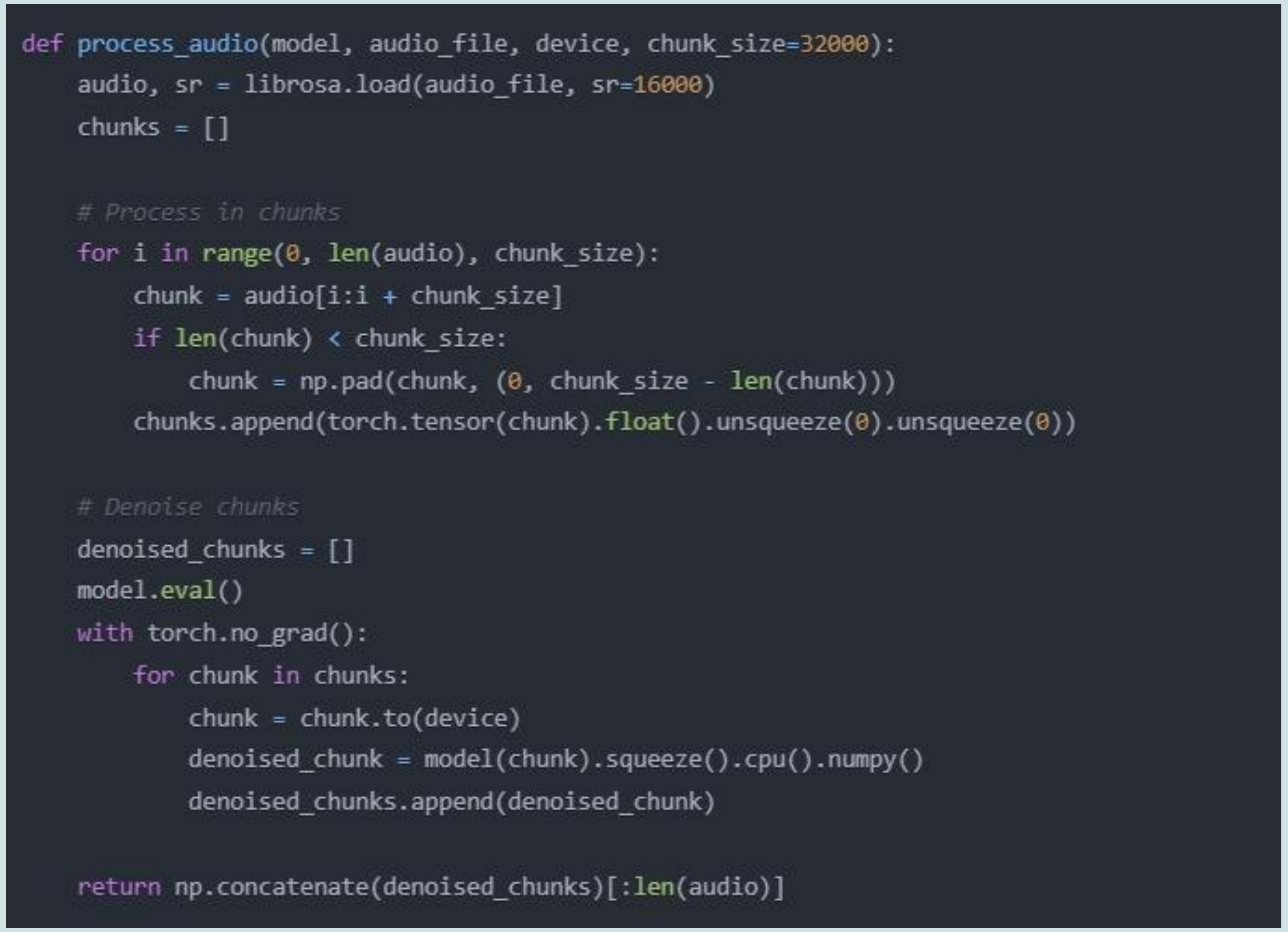
- Processes audio in fixed-size chunks
- Memory-efficient processing
- Handles arbitrary length audio - Concatenates results seamlessly

Records audio using sounddevice library

- 15-second duration
- 16kHz sampling rate
- Single channel (mono) recording



```
def record_audio(filename="recorded_noisy.wav", duration=15, sr=16000):
    print("Recording... (Duration: 15 seconds)")
    audio = sd.rec(int(duration * sr), samplerate=sr, channels=1, dtype='float32')
    sd.wait()
    write(filename, sr, np.squeeze(audio))
```



```
def process_audio(model, audio_file, device, chunk_size=32000):
    audio, sr = librosa.load(audio_file, sr=16000)
    chunks = []

    # Process in chunks
    for i in range(0, len(audio), chunk_size):
        chunk = audio[i:i + chunk_size]
        if len(chunk) < chunk_size:
            chunk = np.pad(chunk, (0, chunk_size - len(chunk)))
        chunks.append(torch.tensor(chunk).float().unsqueeze(0).unsqueeze(0))

    # Denoise chunks
    denoised_chunks = []
    model.eval()
    with torch.no_grad():
        for chunk in chunks:
            chunk = chunk.to(device)
            denoised_chunk = model(chunk).squeeze().cpu().numpy()
            denoised_chunks.append(denoised_chunk)

    return np.concatenate(denoised_chunks)[:len(audio)]
```

Output Obtained

Noisy Audio Waveform

•Description: Displays amplitude (y-axis) over time (x-axis).
•Observations:

- Highly distorted with irregular and spiked fluctuations.
- Noise obscures the structure of the clean audio signal.

Denoised Audio Waveform

•Description: Output waveform after denoising, with the same axes.
•Observations:

- Cleaner and more structured waveform.
- Noise is suppressed, preserving the original audio's integrity.

Noisy Audio Spectrogram

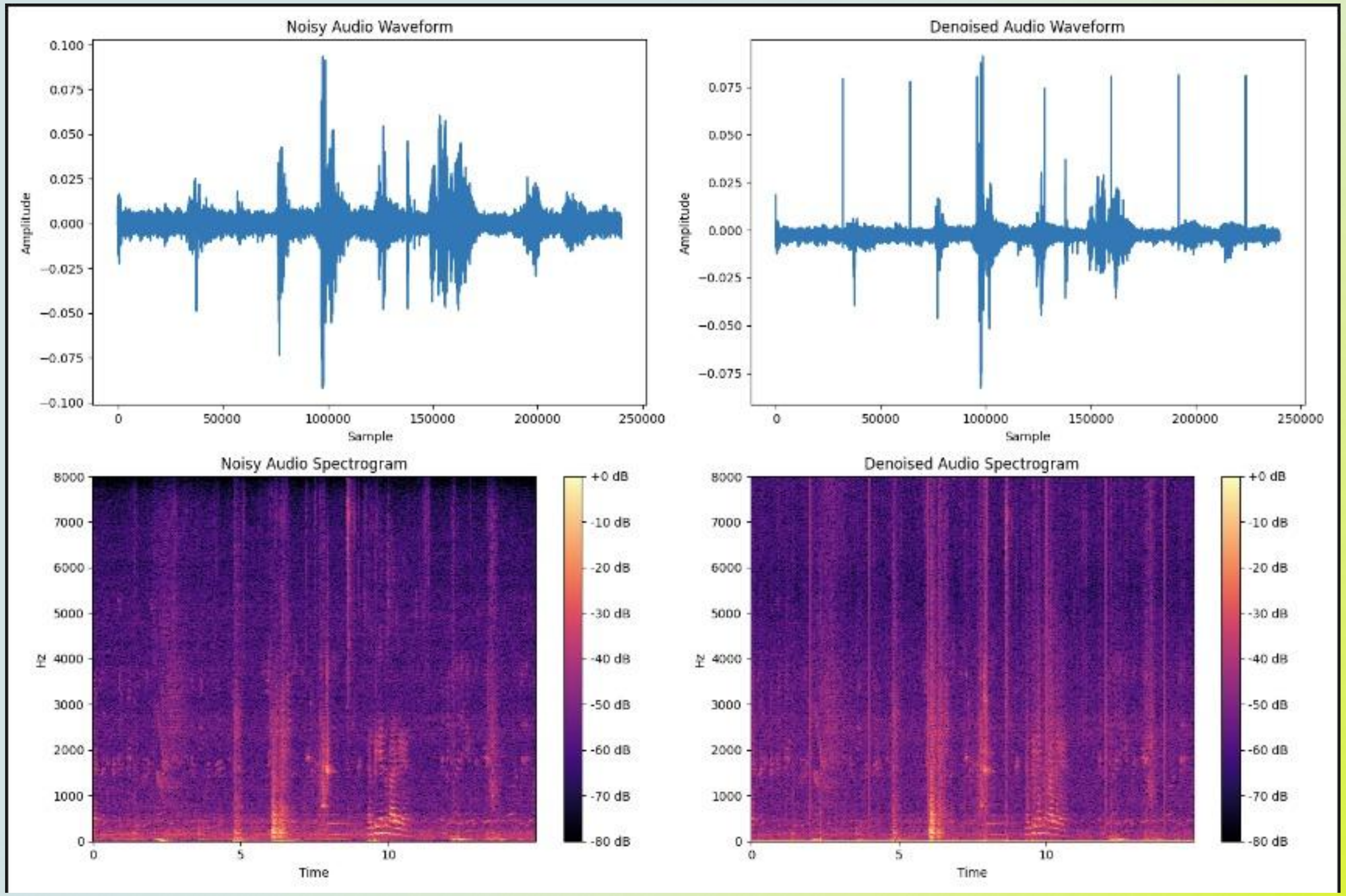
•Description: Visualizes frequency (y-axis) over time (x-axis) with intensity as a color scale.
•Observations:

- Scattered high-frequency components dominate, masking meaningful patterns.

Denoised Audio Spectrogram

•Description: Spectrogram after noise removal.
•Observations:

- Cleaner with reduced noise artifacts.
- Prominent frequency bands are well-defined and consistent.



Future Works

Enhanced Model Architecture

- Transformers and attention-based mechanisms
- Hybrid models combining traditional filters with AI approaches

Real-Time Implementation

- Low latency optimization for immediate processing
- Model quantization and pruning for embedded systems

Dataset Expansion

- Diverse noise profiles (industrial, urban, wildlife)
- Multi-language speech datasets
- Simulated and real-world data mixtures

Future Works

Custom Hardware Integration

- Development of ASICs and edge devices
- Power-efficient solutions for consumer electronics

Advanced Use Cases

- AR/VR applications
- Telecommunication systems
- Healthcare integration (hearing aids)

Improved Evaluation Metrics

- Psychoacoustic studies
- Human tester evaluations
- Real-world performance validation

Bibliography

1.VoiceBank-DEMAND Dataset

1. Valentini-Botinhao, C., et al. *"Speech Enhancement Models Based on Paired Clean and Noisy Speech Datasets."*
2. Accessed from: https://datashare.ed.ac.uk

2.UrbanSound8K Dataset

1. Salamon, J., Jacoby, C., & Bello, J. P. *"A Dataset and Taxonomy for Urban Sound Research."* Proceedings of the ACM International Conference on Multimedia, 2014.
2. Accessed from: https://urbansounddataset.weebly.com

3.Denoising Techniques

1. Vincent, P., Larochelle, H., Lajoie, I., et al. *"Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion."* Journal of Machine Learning Research, 2010.

4.Spectrogram Analysis

1. Griffin, D., & Lim, J. *"Signal Estimation from Modified Short-Time Fourier Transform."* IEEE Transactions on Acoustics, Speech, and Signal Processing, 1984.

1.Active Noise Cancellation (ANC)

1. Elliott, S. J., & Nelson, P. A. *"Active Noise Control."* IEEE Signal Processing Magazine, 1993.

2.Machine Learning Frameworks

1. Abadi, M., Agarwal, A., Barham, P., et al. *"TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems."* 2015.

\

Q&A

Avery Davis, Marketing Director
Ingoude Company