Group 6: Audio denoising using CleanUNet

Aditya Kumar Dwibedi
Tian Qiu
Thushar Thoreneur Govindaraju

Abstract

Background noise often makes it hard to hear speech clearly, whether it's during a phone call, a transcription, or even in a hearing aid. Our project tackles this using a deep learning model that removes noise while keeping the speech natural and clear. Unlike traditional methods that convert audio to spectrograms, which can mess up the sound quality, this works directly on raw audio waveforms. This helps it preserve the real sound of speech, including the phase. The model uses an encoder-decoder setup with gated convolutions and skip connections to clean the audio effectively. We trained it using both time-domain and frequency-domain loss functions to ensure it sounds good and accurate. We tested it on two datasets, DNS 2020 and UrbanSound8K, and evaluated the denoise result, comparing it with traditional filters. We found that it performs much better than older methods, especially in noisy, real-world conditions.

Section 1: Dataset

We use two datasets in this task: the DNS 2020 dataset and the UrbanSound8k dataset.

DNS 2020

This dataset comes from 'The INTERSPPECH 2020 Deep Noise Suppression (DNS) Challenge', which is held by Microsoft [1]. This challenge is intended to promote collaborative research in real-time single-channel Speech Enhancement aimed at maximizing the perceptual quality of the enhanced speech. And they built the DNS 2020 dataset for this challenge.

To train their models effectively, they collected some clean speech and noise data first. Then they combined these two to synthesize noisy data. Hence, they get noise-free speech pairs for training. The clean speech is derived from the public audiobooks dataset called Librivox [2]. It has recordings of volunteers reading over 10,000 public domain audiobooks in various languages, with the majority of which are in English. In total, there are 11,350 speakers and 441 hours of clean speech. The noise clips were selected from Audioset [3] and Freesound [4], which contained 150 audio classes and 60,000 clips. Then the noisy speech dataset is created by adding clean speech and noise at various Signal to Noise Ratio (SNR) levels from 0-40 dB. The sampling rate is 16 kHz. Each audio length is 30 seconds. The number of training samples is 60,000, the testing sample is 1,000.

UrbanSound8k

This dataset contains 8732 labeled sound excerpts (<=4s) of urban sounds from 10 classes: air conditioner, car horn, children playing, drilling, engine idling, gun shot, jackhammer, siren, and street music. The classes are drawn from the urban sound taxonomy [5]. This dataset only contains clean data, so we manually add 10% Gaussian noise (SNR 7dB) to it. The sampling rate is 44 kHz. The number of training samples is 8,000, and the testing sample is 732.

Section 2: Model Description

The primary model utilized in our project is an architecture tailored for audio denoising. The U-Net architecture was adapted due to its proven efficacy in tasks requiring detailed reconstruction from noisy input data, such as image segmentation and audio processing. And so our proposed architecture features an encoder-decoder design, which allows it to learn meaningful features from noisy audio inputs while reconstructing clean audio outputs.

Encoder Layer

The encoder layer is designed to extract high-level features from noisy input audio while reducing spatial resolution. Each encoder layer begins with a 1D convolution operation tailored to capture temporal features from the audio signal. The extracted features are then passed through a ReLU activation function, introducing non-linearity and enhancing the feature extraction process.

To further refine the samples, the encoder employs 1×1 convolutions with Gated Linear Unit (GLU) activation functions. These GLUs act as filters, selectively retaining essential features while discarding irrelevant ones, thereby ensuring dimensional consistency. Mathematically, the operation can be described as:

$$y=Conv1D(x, W)$$

, where x represents the input audio signal and W is the learned kernel. This approach ensures that only the most relevant features are passed to subsequent layers, thus enabling robust feature extraction.

Bottleneck Layer

The bottleneck layer serves as the intermediary between the encoder and decoder, bridging the two components by converting high-level features from the encoder into a compact latent representation. This layer is important in determining the model's capacity to learn meaningful information.

To capture long-term dependencies in audio data, the bottleneck layer uses dilated convolutions, which expand the receptive field without increasing the number of parameters. Batch normalization is applied to enhance training stability and to accelerate convergence, while dropout regularization uses stochasticity to prevent overfitting and improve generalization to unseen data.

A key innovation in the bottleneck layer is the integration of self-attention mechanisms. These mechanisms allow the model to dynamically focus on the most relevant parts of the input signal by weighting features based on their importance. By capturing global context alongside local patterns, the bottleneck layer makes sure that the latent representation is compact and informative.

Decoder Layer

The decoder layer reconstructs clean audio by upsampling the compact latent representation generated by the encoder and bottleneck layers. To achieve this, the decoder employs transposed 1D convolutions, which progressively restore spatial resolution and refine features.

To preserve high-resolution details that might have been lost during downsampling, the decoder incorporates skip connections from the corresponding encoder layers. These connections transfer high-resolution features directly to the decoder, ensuring that critical information is not lost and enhancing the overall reconstruction quality.

The transposed convolution operation in the decoder can be described as

$$x' = ConvTranspose1D(z, W^T)$$

, where z is the latent representation from the bottleneck layer and W^T is the transposed kernel learned during training. The symmetrical nature of the encoder-decoder structure and the use of skip connections allow the decoder to effectively reconstruct clean audio while maintaining contextual relationships between the noisy and denoised output.

Section 3: Loss Function

Our model optimizes a composite loss function that combines multiple components to ensure accuracy and natural audio denoising. Each component addresses specific aspects of the audio signal, balancing time-domain accuracy, frequency-domain fidelity, and high-frequency reconstruction.

Waveform L1 Loss

The first component of the loss function is the Waveform L1 Loss, which minimizes the absolute differences between the denoised and clean audio signals in the time domain. This loss is

particularly effective in preserving sudden spikes and transients in the audio signal. The mathematical formulation of the Waveform L1 Loss is:

$$Loss_{L1} = \frac{1}{n} \sum_{i=1}^{n} |\widehat{y} - y_i|$$

Where $\widehat{y}$ is the predicted waveform, and y_i is the clean waveform.

We chose this loss function due to its robustness to outliers, as unlike L2 loss, L1 loss does not penalize large deviations as heavily, preventing the loss function from "exploding" if the model makes a few bad predictions. Also, the function ensures that the denoised waveform closely follows the clean waveform's structure, capturing the rough temporal details effectively and therefore giving us accuracy in the time domain. A drawback of this function is that it does not address frequency-domain characteristics, such as tonal balance. So, to complement the L1 loss, we incorporate a Multi-Resolution STFT Loss.

Multi-Resolution Short-Time Fourier Transform (STFT) Loss

This component analyzes the denoised audio's frequency-domain characteristics by comparing the spectral magnitudes of the denoised and clean signals across multiple FFT (Fast Fourier Transform) window sizes. FFT, an algorithm for computing the Discrete Fourier Transform (DFT) and its inverse, efficiently converts a signal from the time domain to the frequency domain in O(nlogn) time instead of $O(n^2)$, where n is the number of data points. As human perception is highly sensitive to frequencies, STFT Loss ensures the denoised audio achieves a natural tonal balance. The equation of STFT loss is as follows:

$$Loss_{STFT} = \frac{1}{n} \sum_{i=1}^{n} \sum_{k} \left| \, |S_k(y_i)| \; - \; |S_k(\widehat{y}_i)| \, \right|$$

Where $S_k(y_i)$ represents the STFT magnitude computed using the kth FFT window size.

We chose STFT loss mainly because it makes sure that the denoised audio has the correct tonal balance, which leads to a more natural sound. Another advantage is that STFT loss gives us both coarse and fine frequency details by employing multiple FFT window sizes. However, one drawback is that STFT loss here doesn't consider the phase of the signal and might miss certain high-frequency details. Resolving this drawback is crucial as we need to make sure every frequency of human-audible sound is considered.

High-Band Emphasis Loss (optional)

To counter the STFT loss's drawback, we have an optional High-Band Emphasis Loss, which is included solely for high-frequency reconstruction (4 kHz–16 kHz). This includes sounds like "ch" or "z" and other such sounds that we need to consider in our formula for clear speech recognition. So once we calculate the STFT of both the clean and denoised outputs, we isolate

the high-frequency bits that fall in the 4 kHz–16 kHz range and then apply an L1 or L2 loss to the magnitude difference in this range.

$$Loss_{HighBand} = \frac{1}{n}\sum_{i=1}^{n} \left| |H\ (y_i)| - |H\ (\widehat{y}_i)| \right|$$

Where $H(y_i)$ represents the high-frequency extraction, such as through a bandpass filter.

Combination of Losses

By combining these components, we create a loss function that ensures both temporal accuracy and spectral fidelity. The total composite loss function is defined as:

$$Loss_{total} = \alpha Loss_{L1} + \beta Loss_{STFT} + \gamma Loss_{HighBand}$$

Where α, β, and γ are weights that balance the contributions of each loss component. These weights are then fine-tuned during experimentation to achieve the best performance.

This combined loss strategy allows us to balance time-domain precision with frequency-domain naturalness, which results in higher quality audio denoising performance.

Section 4: Optimization

We use Adam optimizer [5] for training. For the learning rate, we use a linear warmup and a cosine annealing to converge more smoothly, shown in Figure 1. Linear warmup means the learning rate starts from a small value (1e-5 in our training) and increases linearly over a specific number of epochs. This gradual increase helps the model stabilize during the initial training phase and avoids large updates based on poorly initialized weights. After reaching the maximum (2e-4 in our training), we use the annealing technique, which forces the learning rate to follow a cosine curve, gradually decreasing from the maximum to zero.

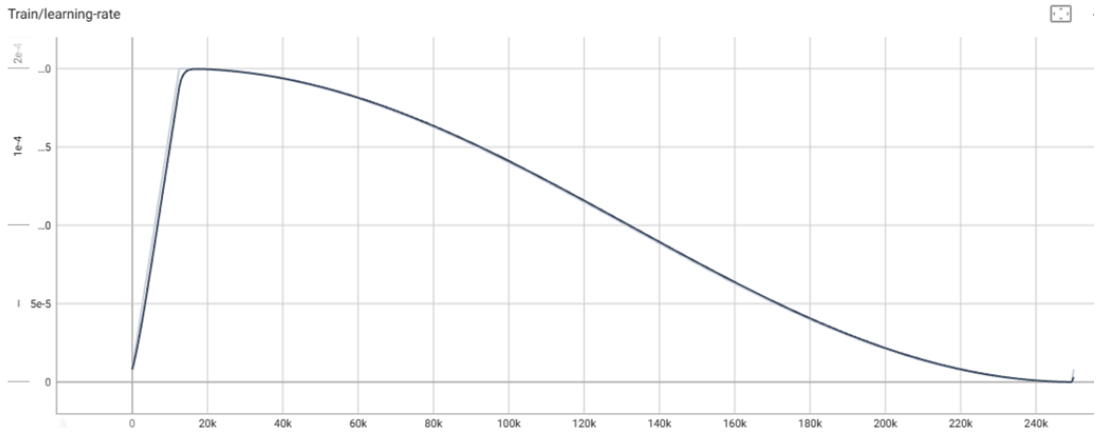

Figure 1 linear warmup with cosine annealing learning rate

Section 5: Metrics and Experimental Results

Metrics

1. PESQ value
   It's a recognized industry standard for audio quality that takes into consideration characteristics such as: audio sharpness, call volume, background noise, clipping, audio interference, etc. PESQ includes two parts: narrow band and wide band, both return a score between 0 and 4.5, with the higher scores indicating a better quality.
2. STOl value
   The Short-Time Objective Intelligibility (STOI) score is a metric used to objectively measure the intelligibility of speech signals, particularly in noisy or processed conditions. It predicts how well a listener can understand speech by comparing a degraded speech signal to a clean reference signal. The STOI score ranges from 0 to 1, with higher scores indicating a better quality.

Training step

For two datasets, we use almost the same training configuration except that the training epoch for the DNS dataset is 250,000, and for the UrbanSound8k dataset is 150,000. We use 8 NVIDIA RTX A6000 GPUs for training. The total training hours is about 12h. The loss decrease for the training set is shown in Figure 2.

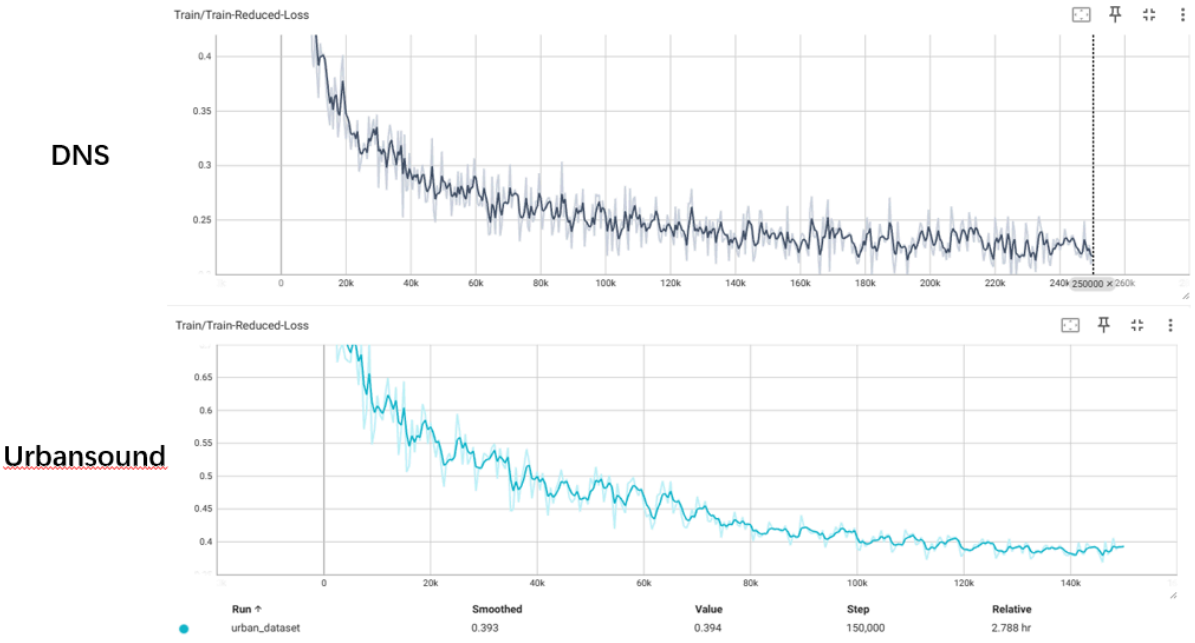

Figure 2 Loss decrease for two datasets

Comparison

To assess the effectiveness of the CleanUNet model, we compared it against traditional denoising techniques using three objective metrics: Signal-to-Noise Ratio (SNR) improvement, Short-Time Objective Intelligibility (STOI), and Perceptual Evaluation of Speech Quality (PESQ).

SNR Improvement

SNR improvement is used to quantify the reduction in background noise while retaining the speech signal. Based on the results visualized in the figures, traditional filters such as median, low pass, high pass, and Wiener filters showed only modest and inconsistent improvements. These approaches performed adequately for relatively stationary noise types, such as air conditioner, but often failed to handle impulsive or complex noises like jackhammer or gunshot. Sometimes, they introduced artifacts or distorted the speech, resulting in a net loss in perceived quality. In comparison, the CleanUNet model produced consistently higher SNR improvements across all tested sound classes. Its ability to learn noise patterns from data allowed it to adapt more effectively, achieving better noise suppression without sacrificing important speech features.

STOI: Speech Intelligibility

STOI scores provide an estimate of how understandable speech is after denoising. Traditional filtering methods generally led to negligible gains in STOI, especially in cases involving high-frequency noise. For classes such as street_music, dog_bark, STOI scores remained low even after filtering. CleanUNet demonstrated substantial improvements in STOI across all categories. It preserved speech intelligibility even under challenging noise conditions, with particularly strong gains in classes that initially had low baseline scores.

PESQ: Perceptual Audio Quality

The PESQ metric assesses audio quality in a way that aligns more closely with human perception. It evaluates factors such as sharpness, volume consistency, and the presence of distortions or artifacts. Traditional filters typically achieved PESQ scores in the range of 1.5 to 2.5, reflecting moderate quality improvements but with noticeable degradation due to speech distortion or incomplete noise removal. CleanUNet consistently achieved PESQ scores above 3.0, with several sound classes reaching values close to 3.5 – 4.0. This indicates that the model not only removed noise effectively but also preserved the natural characteristics of the speech signal, resulting in a cleaner and more pleasant listening experience.

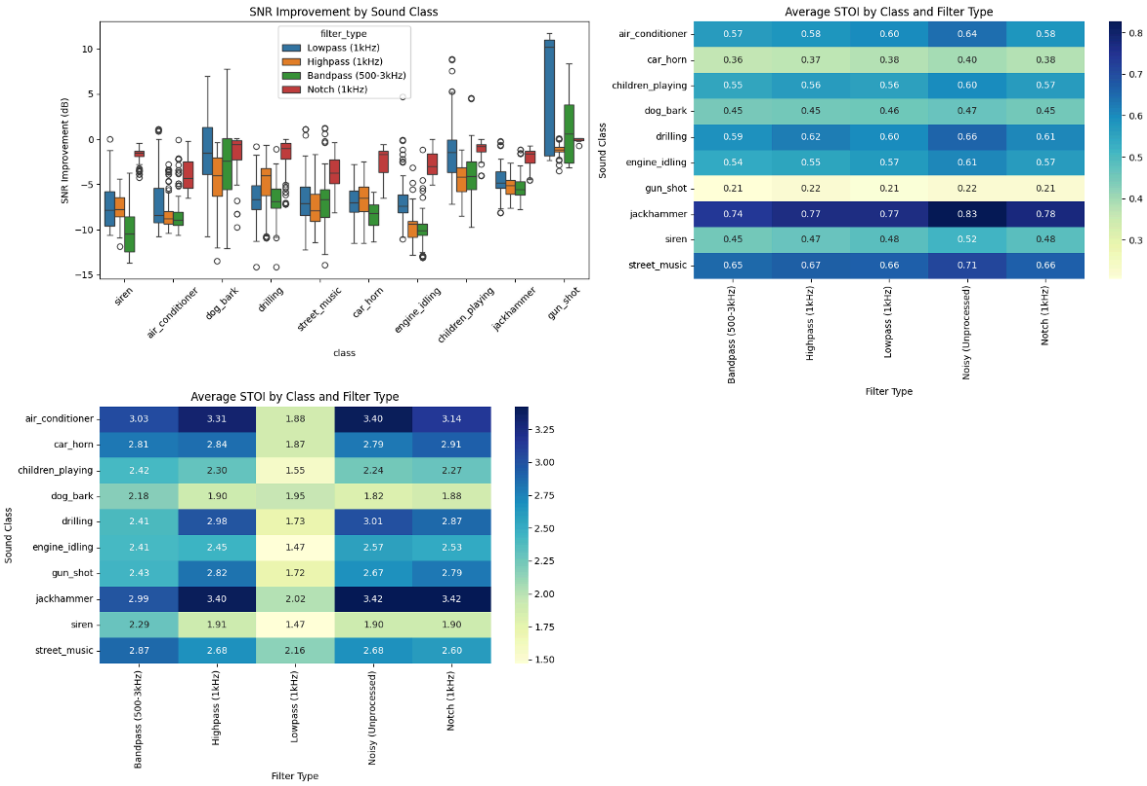

Figure 3: Metrics of Traditional model

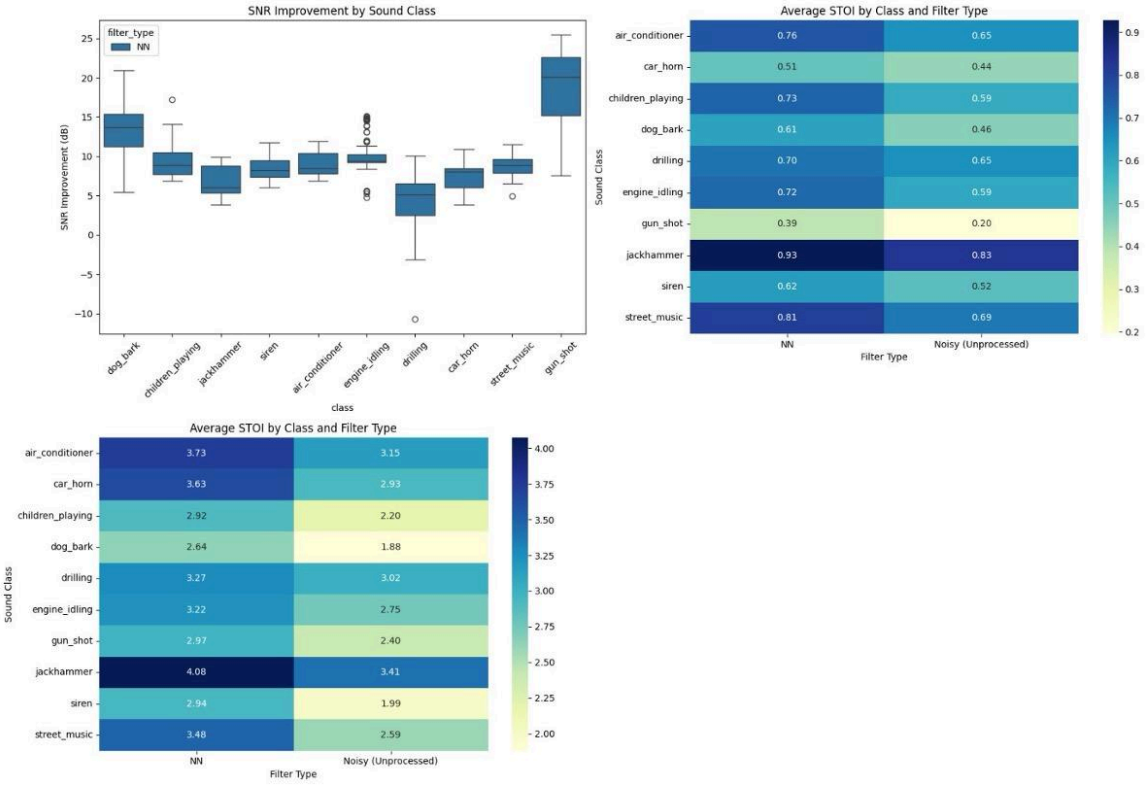

Figure 4: Metrics of CleanUNet

References

1. Reddy, Chandan KA, et al. "The interspeech 2020 deep noise suppression challenge: Datasets, subjective testing framework, and challenge results." *arXiv preprint arXiv:2005.13981* (2020).
2. Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error log-spectral amplitude estimator," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 33, no. 2, pp. 443-445, April 1985, doi: 10.1109/TASSP.1985.1164550.
3. Reddy, Chandan Karadagur Ananda, et al. "An individualized super-Gaussian single microphone speech enhancement for hearing aid users with smartphone as an assistive device." *IEEE signal processing letters* 24.11 (2017): 1601-1605.
4. Wolfe, Patrick J., and Simon J. Godsill. "Simple alternatives to the Ephraim and Malah suppression rule for speech enhancement." *Proceedings of the 11th IEEE Signal Processing Workshop on Statistical Signal Processing (Cat. No. 01TH8563)*. IEEE, 2001.
5. Salamon, Justin, Christopher Jacoby, and Juan Pablo Bello. "A dataset and taxonomy for urban sound research." *Proceedings of the 22nd ACM international conference on Multimedia*. 2014.
6. Kingma, Diederik P. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

Contributions and GitHub

Contribution percentage

Name	UBIT	Contribution (%)
Aditya Kumar Dwibedi	adwibedi	33%
Tian Qiu	tqiu5	33%
Thushar Thorenur Govindaraju	thushart	33%

GitHub link - https://github.com/HandleGod101/CSE676_DL_GroupRepo