Comparative Analysis of Adaptive Filtering and Neural Network-Based Approaches for Audio Signal Denoising

Prosperi Marco

Signal, Image and Video - University of Trento

Contents

1 Introduction

This project investigates three approaches for noise reduction in audio signals: traditional methods such as Least Mean Squares (LMS) and Normalized Least Mean Squares (NLMS), along with a neural network-based implementation using the *PyTorch* library. The primary objective is to compare and evaluate the performance of these methods on both synthetic data and real-world audio signals [2][1]. This work is undertaken as part of the Signal, Image, and Video course in the Master's program in Artificial Intelligence Systems at the University of Trento, under the guidance of Prof. Rosani.

2 Background

Noise reduction in audio signals is a critical task in various fields, such as telecommunications, speech recognition, and hearing aids. The goal of noise reduction is to remove unwanted noise from audio signals while preserving the quality of the original sound. Traditional methods such as Least Mean Squares (LMS) and Normalized Least Mean Squares (NLMS) have been widely used for adaptive filtering in noise cancellation. These methods work by adjusting the filter coefficients in real-time to minimize the error between the desired signal and the noisy signal.

In recent years, deep learning techniques, particularly convolutional neural networks (CNNs) and other neural network architectures, have shown great potential for solving complex noise reduction problems. By learning patterns from large datasets, neural networks can provide high performance in handling non-stationary and time-varying noise, compared to traditional methods. This project aims to compare and evaluate the effectiveness of these traditional and neural network-based approaches on synthetic and real-world audio data.

3 Related Work

Noise reduction techniques have been extensively studied in the literature, with many approaches falling into two broad categories: traditional signal processing techniques and machine learning-based methods.

Traditional methods such as Least Mean Squares (LMS) and Normalized Least Mean Squares (NLMS) have been the foundation of adaptive filtering in noise reduction. These methods, as discussed by Widrow and Stearns (1985)[3], perform well in environments where the noise characteristics are stationary and predictable. However, in dynamic and non-stationary noise environments, these methods often fall short in maintaining high performance.

Recent works have explored the use of deep learning techniques for audio denoising, which offer the advantage of learning complex patterns from large datasets. For instance, Xu et al. (2015)[4] introduced a deep neural network architecture for speech enhancement, which outperformed traditional methods in terms of noise reduction and preservation of speech quality. Other studies,

such as those by Liu et al. (2019), have focused on the use of convolutional neural networks (CNNs) for real-time noise suppression. These studies demonstrate the potential of machine learning models to significantly improve performance in challenging noise conditions.

This project builds upon these existing works by comparing the performance of traditional adaptive filtering methods with a neural network-based approach, using both synthetic and real-world audio datasets.

4 Methodology

This section describes the methods used for noise reduction in the audio signals, including both traditional signal processing techniques and a neural network-based approach.

4.1 Traditional Methods: LMS and NLMS Filters

Two adaptive filters, the Least Mean Squares (LMS) filter and the Normalized Least Mean Squares (NLMS) filter, were implemented for noise reduction. These filters are adaptive because they iteratively adjust their filter weights to minimize the error between the predicted and the actual clean signal.

4.1.1 LMS Filter

The LMS filter works by iteratively adjusting its coefficients to minimize the error signal, which is the difference between the desired clean signal and the predicted signal. The filter operates on a sliding window of a fixed size over the noisy signal, updating its weights at each time step.

The update rule for the filter coefficients is:

$$w_{k+1} = w_k + \mu e_k x_k$$

where: - w_k are the filter coefficients, - μ is the step size, - e_k is the error signal ($e_k = y_k - \hat{y}_k$), - x_k is the input signal vector.

The LMS filter has the advantage of simplicity but can be sensitive to the choice of the step size μ, which must be chosen carefully to avoid instability or slow convergence.

The implementation in Python is as follows:

```
def lms_filter(noisy_signal, clean_signal, mu, filter_order):
    n_samples = len(noisy_signal)
    filtered_signal = np.zeros(n_samples)
    weights = np.zeros(filter_order)

    for i in range(filter_order, n_samples):
        x = noisy_signal[i-filter_order:i][::-1]
        y = np.dot(weights, x)
        error = clean_signal[i] - y
```

```
            w += 2 * mu * error * x  # Update filter
            filtered_signal[i] = y

    return filtered_signal, weights
```

4.1.2 NLMS Filter

The NLMS filter is a normalized version of the LMS filter. It normalizes the step size at each iteration, which helps to prevent issues when the input signal has varying power levels. The update rule is as follows:

$$w_{k+1} = w_k + \mu \frac{e_k x_k}{x_k^T x_k + \epsilon}$$

where ϵ is a small constant added to avoid division by zero.

The implementation in Python is as follows:

```
    def nlms_filter(noisy_signal, desired_signal, mu, filter_order, epsilon=1
        n_samples = len(noisy_signal)
        weights = np.zeros(filter_order)
        filtered_signal = np.zeros(n_samples)

        for i in range(filter_order, n_samples):
            x = noisy_signal[i-filter_order:i][::-1]
            y = np.dot(weights, x)
            error = desired_signal[i] - y
            normalization_factor = np.dot(x, x) + epsilon
            weights += (2 * mu * error * x) / normalization_factor
            filtered_signal[i] = y

        return filtered_signal, weights
```

4.2 Neural Network-based Approach

A neural network model was used as a more advanced approach to noise reduction. The network is a simple feedforward architecture that processes noisy signals and outputs a denoised signal. The model consists of three fully connected layers with ReLU activations, designed to capture the underlying relationship between the noisy input and the clean output.

The network is defined as follows:

```
class AdvancedNN(nn.Module):
    def __init__(self, input_size):
        super(AdvancedNN, self).__init__()
        self.fc1 = nn.Linear(input_size, 64)
        self.fc2 = nn.Linear(64, 32)
```

```
        self.fc3 = nn.Linear(32, 1)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

The model was trained using the Mean Squared Error (MSE) loss function, which measures the difference between the predicted output and the actual clean signal. The Adam optimizer was used to minimize the loss function during training, with a learning rate of 10^{-3}.

The implementation of the neural network model setup is as follows:

```
# Initialize the model
model = AdvancedNN(window_size)
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=learning_rate)
```

The model was trained for 15 epochs, with a batch size of 64.

4.3 Data Preprocessing

The data preprocessing involved in dividing the noisy and clean audio signals into overlapping windows of a fixed size defined by the *window_size* parameter. For each window extracted from the noisy signal, the corresponding next value from the clean signal was used as the target output. This approach constructs a supervised dataset to train a model capable of predicting the clean signal value given a segment of the noisy signal.

4.4 Experimental Setup

The experiments were conducted using synthetic noisy data as well as real-world noisy audio signals. The synthetic data included a clean signal mixed with white noise. The real-world data came from the NOIZEUS Speech Corpus [2], which contains speech signals with various noise types. I decided to use the dataset with train noise added to clean signals. The performance of each method was evaluated using the Signal-to-Noise Ratio (SNR), Mean Squared Error (MSE), Mean Absolute Error (MAE), Peak-Signal-to-Noise-Ratio (PSNR) and the coefficient of determination between the denoised and clean signals.

4.5 Evaluation Metrics

The following evaluation metrics were used to assess the effectiveness of the noise reduction methods:

- **Signal-to-Noise Ratio (SNR)**: Measures the ratio of the signal power to the noise power.

- **Mean Squared Error (MSE)**: Measures the average squared difference between the processed signal and the clean signal.
- **Mean Absolute Error (MAE)**: Measures the average absolute difference between the processed signal and the clean signal.
- **Peak Signal-to-Noise Ratio (PSNR)**: Measures the ratio between the maximum possible signal value and the error between the processed signal and the clean signal.
- **R-squared (R2)**: Measures the proportion of the variance in the clean signal that is predictable from the processed signal.

5 Experimental Setup

The experiments were conducted using a MacBook Pro M3 Pro with 18GB of unified memory. The neural network was implemented using PyTorch, while the LMS and NLMS filters were implemented in Python with the NumPy library. The audio signals were processed using the SciPy library for reading and writing .WAV files.

6 Results and Discussion

In this experiment, we compared the performance of a Neural Network (NN), Least Mean Squares (LMS), and Normalized Least Mean Squares (NLMS) filters for audio noise reduction. The performance of each method was evaluated using several metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), Peak Signal-to-Noise Ratio (PSNR), Signal-to-Noise Ratio (SNR), and the coefficient of determination (R^2). The metrics were computed across a set of test signals, and the average values with standard deviations are presented below.

6.1 Neural Network (NN) Results

- **MSE:** 0.0023 ± 0.0008
- **MAE:** 0.0294 ± 0.0053
- **PSNR:** 26.5947 ± 1.6139
- **SNR:** 7.4483 ± 0.7732
- **R^2:** 0.8172 ± 0.0319

The NN-based filter exhibited exceptional performance, achieving the lowest MSE of 0.0023 among the evaluated methods, indicating a minimal residual error between the noisy and clean signals. The PSNR of 26.59 reflects a very high-quality denoised signal, and the SNR of 7.45 highlights its superior ability

to suppress noise. Additionally, the R^2 value of 0.8172 suggests that the NN captured over 81% of the variance in the clean signals, demonstrating its ability to accurately approximate the clean signal.

6.2 LMS Filter Results

- **MSE:** 0.0038 ± 0.0011
- **MAE:** 0.0353 ± 0.0061
- **PSNR:** 24.2412 ± 1.3910
- **SNR:** 5.0947 ± 1.0480
- **R^2:** 0.6815 ± 0.0773

The LMS filter demonstrated good performance, with an MSE of 0.0038, though it was higher than the NN's MSE, indicating a slightly larger residual error. The PSNR of 24.24 and SNR of 5.09 indicate effective noise suppression, though not as robust as the NN's performance. The R^2 value of 0.6815 shows that the LMS filter explained approximately 68% of the variance in the clean signal, which is lower than the NN but still significant.

6.3 NLMS Filter Results

- **MSE:** 0.0040 ± 0.0014
- **MAE:** 0.0322 ± 0.0069
- **PSNR:** 24.1591 ± 1.7036
- **SNR:** 5.0126 ± 0.9056
- **R^2:** 0.6779 ± 0.0651

The NLMS filter performed slightly worse than the LMS filter in terms of MSE (0.0040) and PSNR (24.16). While its MAE of 0.0322 was marginally lower, the SNR of 5.01 and R^2 of 0.6779 suggest that it was slightly less effective in approximating the clean signal and suppressing noise compared to the LMS and NN methods.

6.4 Discussion

From these results, it is evident that the NN approach outperformed both the LMS and NLMS filters across all metrics. Its lower MSE, higher PSNR, and superior SNR demonstrate its effectiveness in denoising the audio signals. The LMS filter also performed well, showing competitive results but falling short of the NN in terms of MSE, PSNR, and SNR. The NLMS filter, while similar in approach to the LMS, demonstrated slightly worse performance, particularly in terms of variance explained (R^2).

The strong performance of the neural network can be attributed to its ability to model complex signal patterns, whereas the LMS and NLMS methods rely on simpler adaptive algorithms. However, the LMS filter remains an effective choice for scenarios where computational simplicity and real-time adaptability are required.

Metric	**NN**	**LMS**	**NLMS**
MSE	0.0023 ± 0.0008	0.0038 ± 0.0011	0.0040 ± 0.0014
MAE	0.0294 ± 0.0053	0.0353 ± 0.0061	0.0322 ± 0.0069
PSNR	26.5947 ± 1.6139	24.2412 ± 1.3910	24.1591 ± 1.7036
SNR	7.4483 ± 0.7732	5.0947 ± 1.0480	5.0126 ± 0.9056
R^2	0.8172 ± 0.0319	0.6815 ± 0.0773	0.6779 ± 0.0651

Table 1: Performance metrics for NN, LMS, and NLMS filters

7 Conclusion

In this study, we compared three methods for audio noise reduction: Neural Network (NN), Least Mean Squares (LMS), and Normalized Least Mean Squares (NLMS) filters. The results revealed that the neural network achieved the best performance across all metrics, making it the most effective choice for denoising audio signals. The LMS filter showed competitive performance, while the NLMS filter was slightly less effective, particularly in terms of MSE and R^2.

Overall, the NN approach demonstrates significant potential for high-quality audio noise reduction, particularly in scenarios where complex signal patterns are present. The LMS filter, with its simplicity and adaptability, remains a viable option for real-time applications. Future work could explore the integration of machine learning and adaptive filtering techniques for even better performance in noise reduction tasks.

References

[1] Yi Hu and Philipos C. Loizou. "Subjective evaluation and comparison of speech enhancement algorithms". In: *Speech Communication* 49.7-8 (2007), pp. 588–601.

[2] Philipos C. Loizou. *NOIZEUS Speech Corpus*. Accessed: 2024-12-23. 2024. URL: `https://ecs.utdallas.edu/loizou/speech/noizeus/`.

[3] B. Widrow and S.D. Stearns. *Adaptive Signal Processing*. EDITED BY ALAN V. OPPENHEIM. Prentice-Hall, 1985. ISBN: 9780130040299. URL: `https://books.google.it/books?id=X74QAQAAMAAJ`.

[4] Zhiqiang Zhang, Shuang Li, and Wei Zuo. "A Comparative Study of Adaptive Filtering Algorithms for Noise Reduction in Speech Signals". In: *Signal Processing* 119 (2015), pp. 1–11.