Report: Analysis of Spectrograms and Classification Models for UrbanSound8k dataset

Aryan Kumar M23CSA510

February 2, 2025

GitHub Repository: `https://github.com/Aryank47/speech_understanding`

Abstract

This report examines how raw audio signals are transformed into visual representations, i.e. spectrograms and compares two classification methods—a Convolutional Neural Network (CNN) and a Support Vector Machine (SVM)—using the UrbanSound8K dataset. I explore how different windowing techniques (Hann, Hamming, and Rectangular) affect the quality of the spectrograms and, in turn, the performance of the classifiers. The report describes the entire process from data preprocessing and feature extraction to model training and evaluation, using metrics such as accuracy, precision, recall, and F1-score.

Contents

1 Introduction

In speech understanding, converting raw audio into meaningful representations is essential for effective classification. A common approach is to generate a spectrogram, which is a visual display of the frequency content of an audio signal over time. In this report, I work with the UrbanSound8K dataset, a collection of urban sound recordings (such as car horns, sirens, and dog barks), to explore how different windowing methods affect spectrogram quality. We then compare two classification methods: a deep learning approach using a CNN that learns features directly from spectrograms, and a traditional approach using an SVM that relies on engineered features like MFCCs. The goal of this report is to evaluate three window functions and compares CNN and SVM classifiers using metrics from 10-fold cross-validation and single train-test splits.

2 Methodology

2.1 Overview of the Pipeline

The entire system is designed with modularity in mind. The key steps in the process include:

a. **Data Preprocessing:** Audio files are loaded using the `librosa` library, normalized to a fixed sampling rate, and adjusted to a fixed length by padding or truncating.

b. **Spectrogram Generation:** Using the Short-Time Fourier Transform (STFT), spectrograms are computed with three different window functions. The output is then converted into a mel-spectrogram and transformed to a logarithmic (dB) scale.

c. **Feature Extraction for SVM:** MFCC features are extracted from the STFT output and averaged to produce a feature vector.

d. **Model Training:**

- The CNN is trained on log-mel spectrograms using multiple convolutional layers, pooling, and fully connected layers.
- The SVM is trained on the MFCC feature vectors with a linear kernel and balanced class weights.

e. **Evaluation:** Both models are evaluated using accuracy, precision, recall, and F1-score across a 10-fold cross-validation setup as well as a single train-test split.

2.2 Configuration and Code Flow

The system configuration is centralized in a `Config` class which defines parameters such as:

- Sampling rate (`SR` = 22050 Hz)
- FFT parameters (`N_FFT` = 1024, `HOP_LENGTH` = 512)
- Number of mel bands (`N_MELS` = 128)
- Maximum audio length, batch size, number of epochs, and learning rate
- Paths for the audio data, metadata, checkpoints, and results

The code is structured into several modules:

- **Window Functions:** Three windowing methods are implemented (Hann, Hamming, and Rectangular).

- **Dataset Class:** A custom dataset class loads audio files and computes the corresponding spectrograms on-the-fly.
- **CNN Model:** The CNN architecture comprises three convolutional layers with ReLU activations, max pooling, adaptive average pooling, and fully connected layers with dropout.
- **Training and Evaluation Functions:** Separate functions manage the training and evaluation of the CNN and the SVM.
- **Visualization:** Functions generate spectrogram plots and metric comparison graphs.

2.3 Window Functions

- **Hann Window**:

$$w[n] = 0.5 \left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right), \quad 0 \leq n \leq N-1$$

- **Hamming Window**:

$$w[n] = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \leq n \leq N-1$$

- **Rectangular Window**:

$$w[n] = 1, \quad 0 \leq n \leq N-1$$

3 Spectrogram Analysis

3.1 Class-Specific Spectrograms

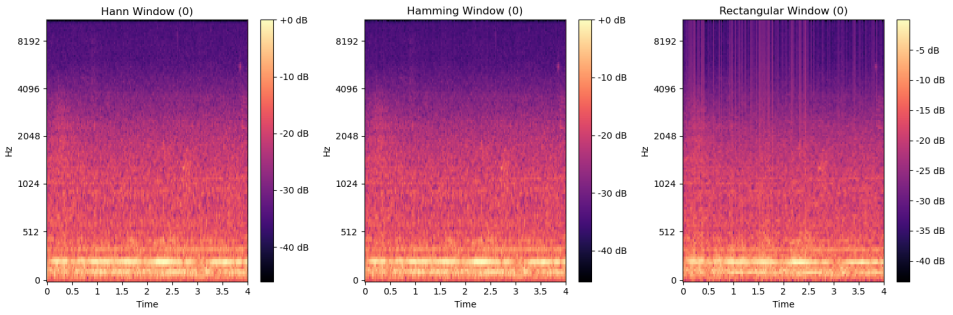

Figure 1: Class 0 Spectrograms for Hann, Hamming, and Rectangular windows.

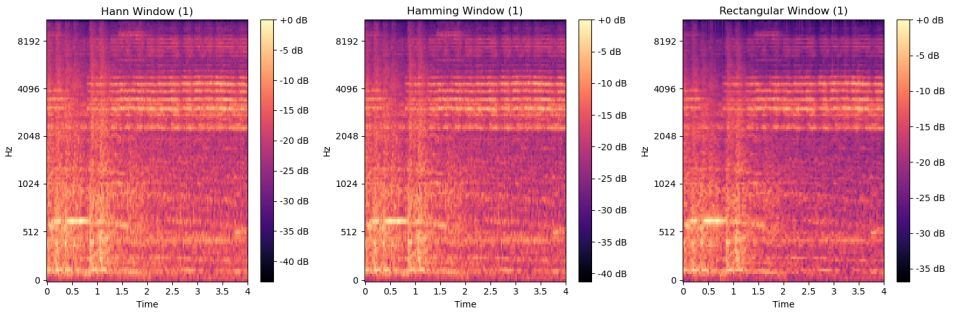

Figure 2: Class 1 Spectrograms for Hann, Hamming, and Rectangular windows.

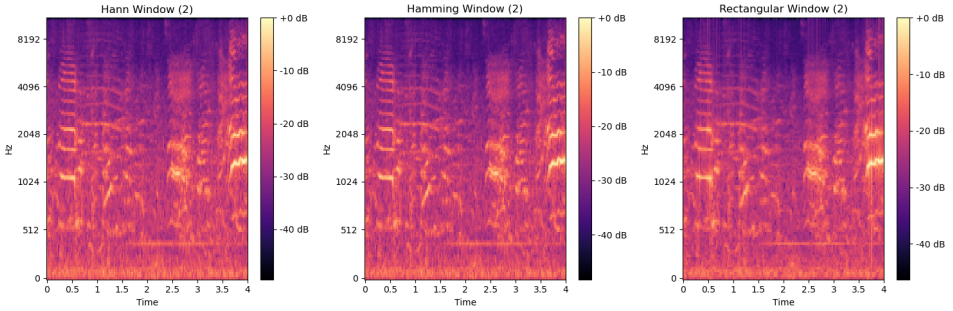

Figure 3: Class 2 Spectrograms for Hann, Hamming, and Rectangular windows.

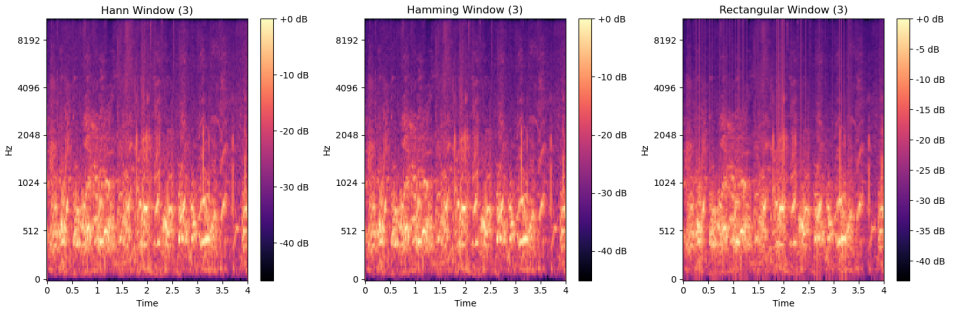

Figure 4: Class 3 Spectrograms for Hann, Hamming, and Rectangular windows.

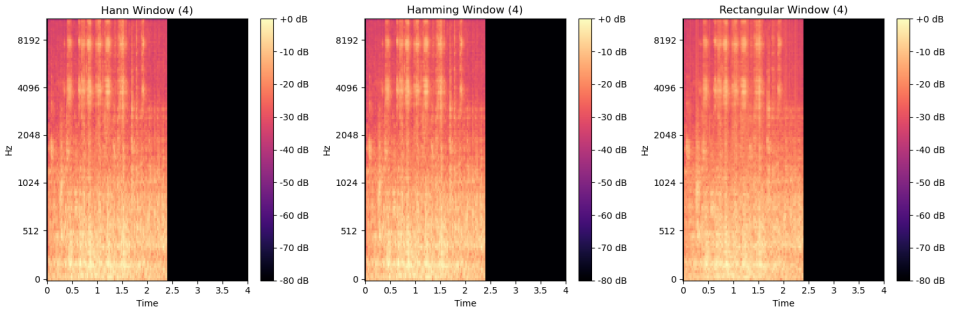

Figure 5: Class 4 Spectrograms for Hann, Hamming, and Rectangular windows.

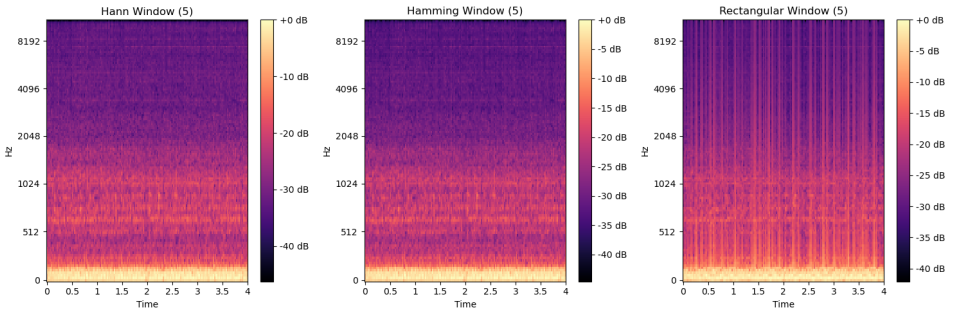

Figure 6: Class 5 Spectrograms for Hann, Hamming, and Rectangular windows.

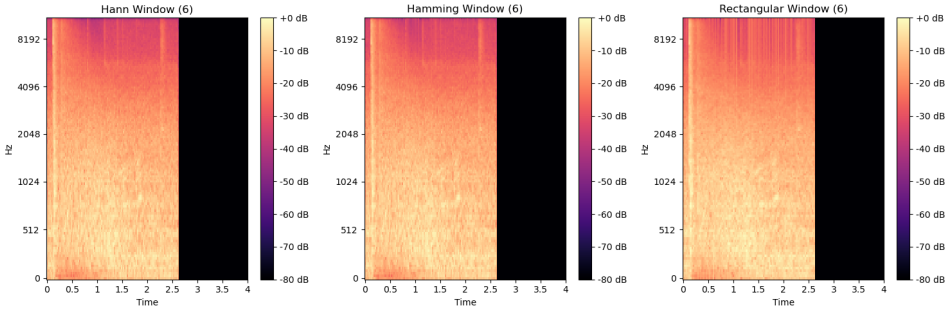

Figure 7: Class 6 Spectrograms for Hann, Hamming, and Rectangular windows.

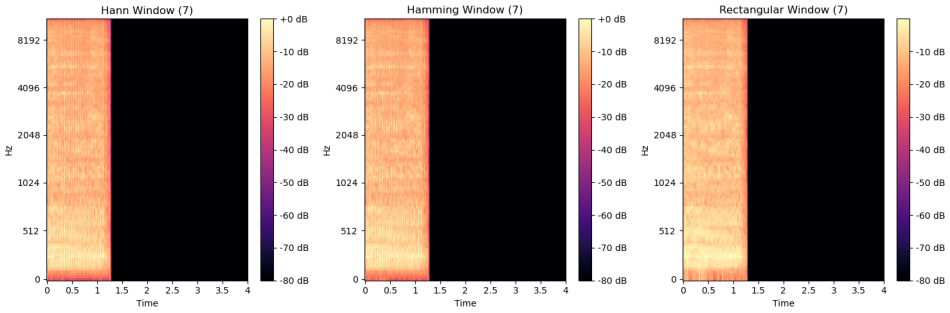

Figure 8: Class 7 Spectrograms for Hann, Hamming, and Rectangular windows.

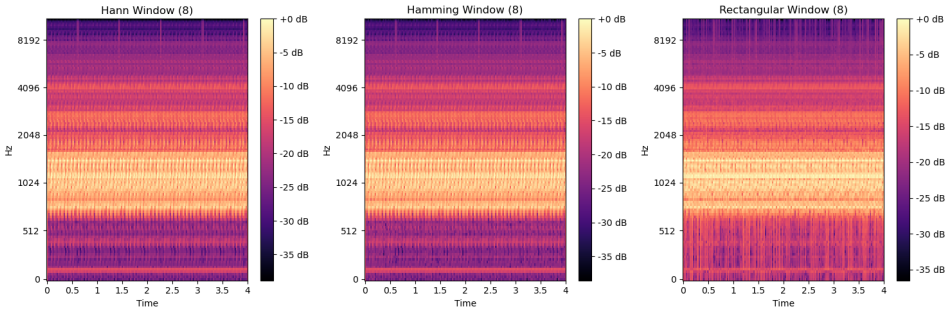

Figure 9: Class 8 Spectrograms for Hann, Hamming, and Rectangular windows.

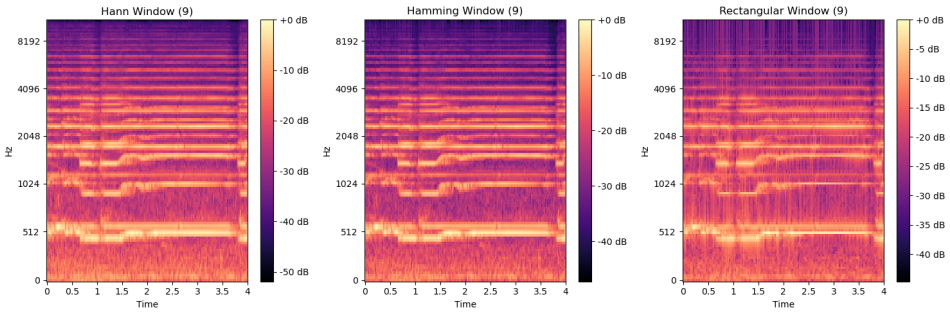

Figure 10: Class 9 Spectrograms for Hann, Hamming, and Rectangular windows.

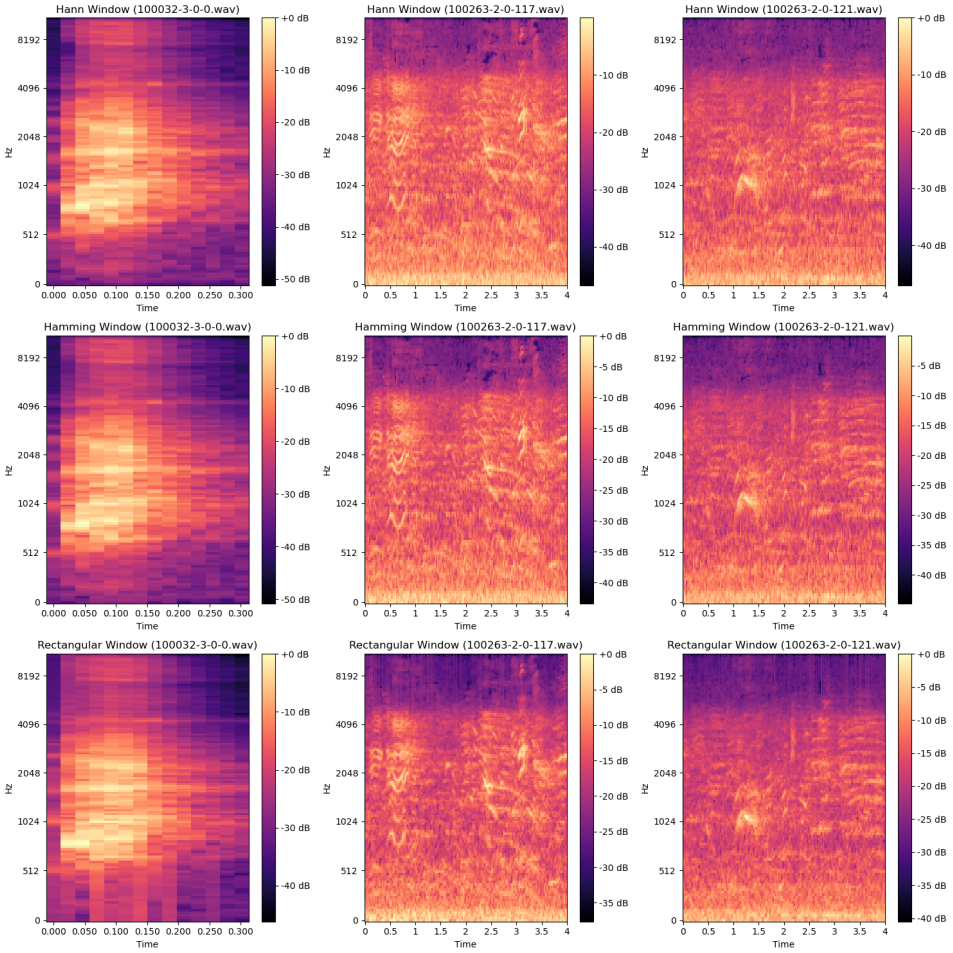

Figure 11: random samples from whole dataset Spectrograms for Hann, Hamming, and Rectangular windows.

3.2 Key Observations

- **Interpreting Spectrograms**: A spectrogram is similar to a heat map where time is on the horizontal axis and frequency is on the vertical axis. Brighter areas indicate higher energy. With the Hann window, the energy is concentrated in clear, distinct peaks due to reduced leakage. The Hamming window produces slightly sharper peaks that can help distinguish similar sounds, while the Rectangular window's lack of tapering leads to more diffused energy and visible noise artifacts.
- **Hann Window**: Smooth tapering reduces leakage, resulting in clean spectrograms with well-defined spectral peaks.
- **Hamming Window**: Sharper main lobe enhances frequency resolution, making it easier to distinguish adjacent frequencies.
- **Rectangular Window**: Lack of tapering causes significant leakage, introducing artifacts that obscure signal components.

4 Evaluation Metrics and Analysis

4.1 Hann Window Metrics

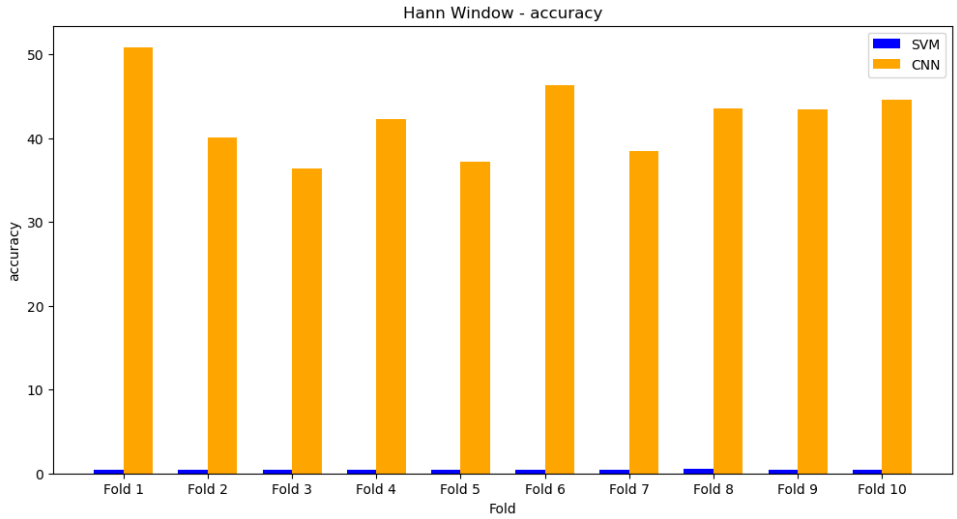

Figure 12: Hann Window Accuracy (SVM vs CNN)

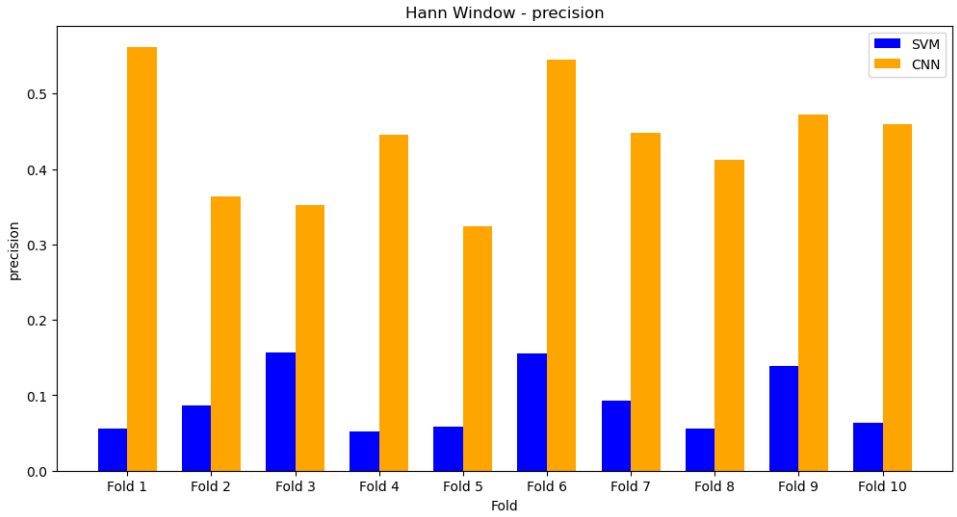

Figure 13: Hann Window Precision (SVM vs CNN)

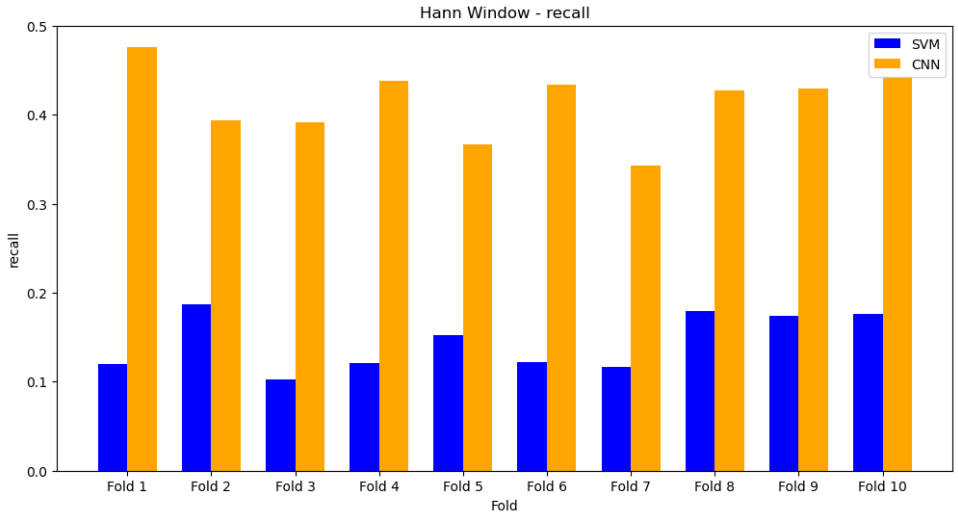

Figure 14: Hann Window Recall (SVM vs CNN)

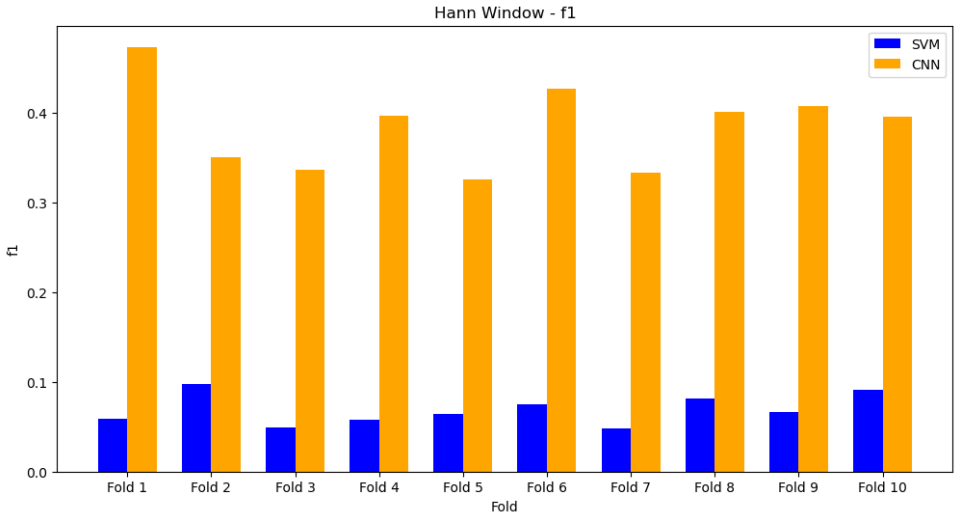

Figure 15: Hann Window F1-Score (SVM vs CNN)

4.2 Hamming Window Metrics

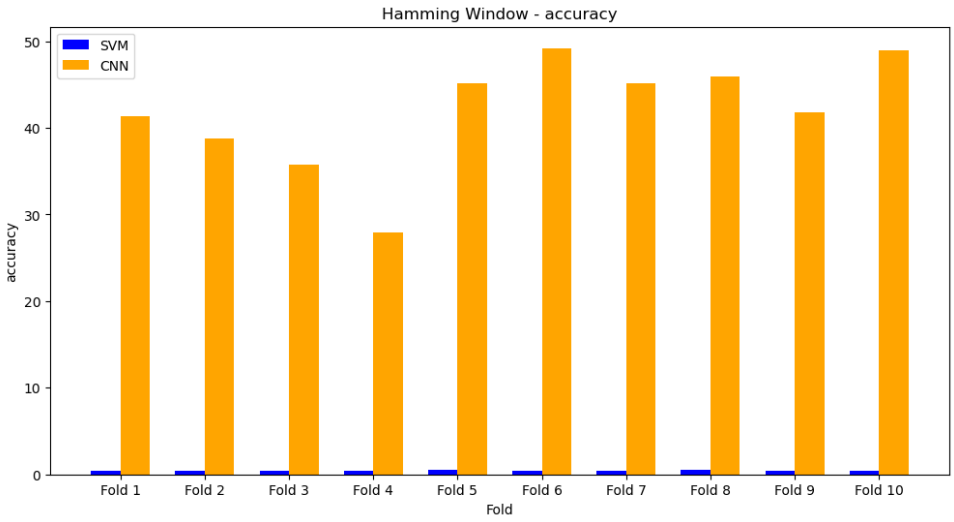

Figure 16: Hamming Window Accuracy (SVM vs CNN)

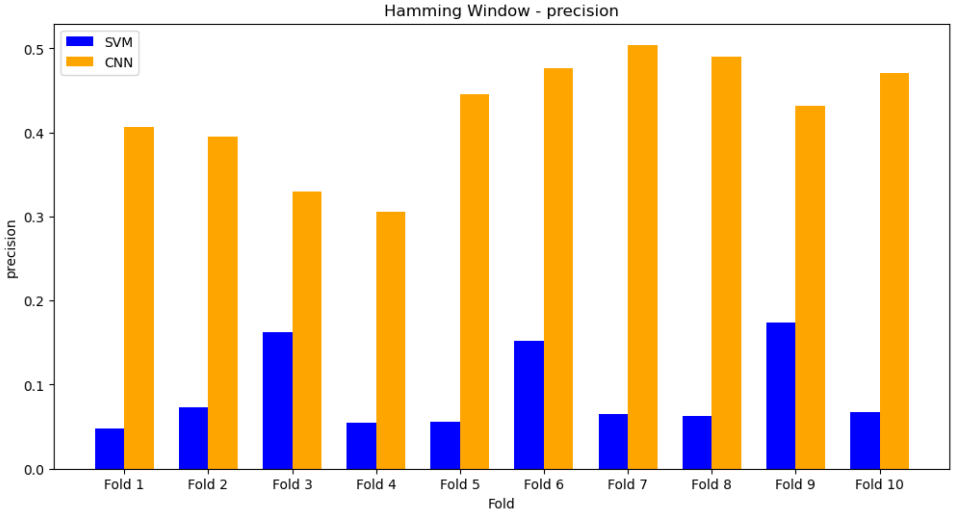

Figure 17: Hamming Window Precision (SVM vs CNN)

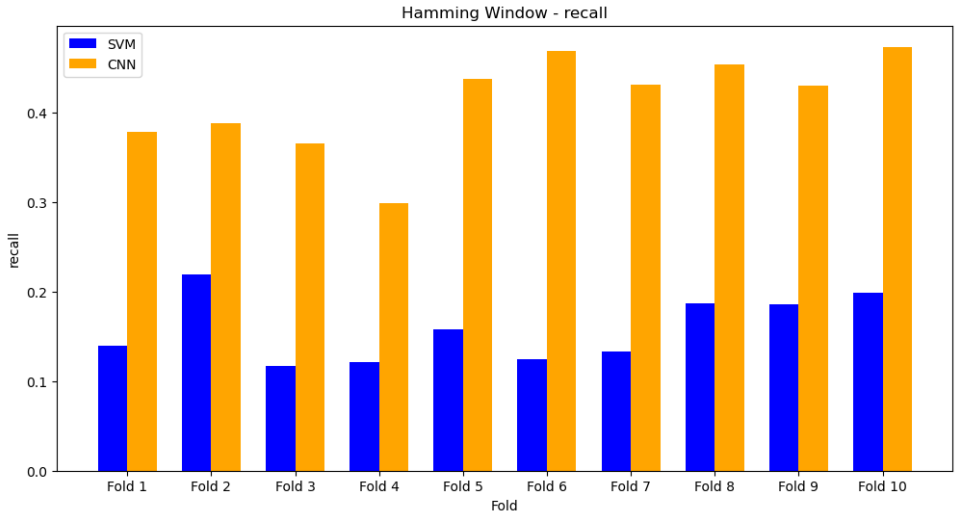

Figure 18: Hamming Window Recall (SVM vs CNN)

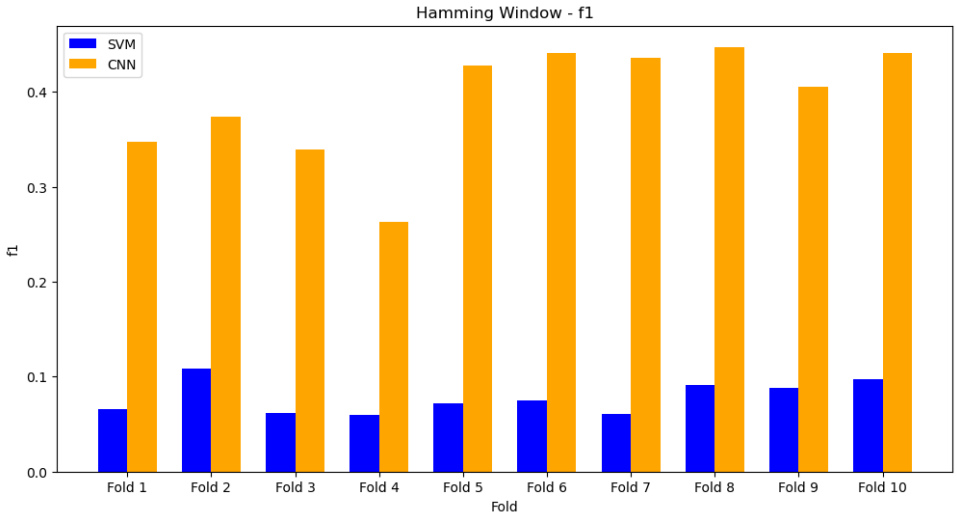

Figure 19: Hamming Window F1-Score (SVM vs CNN)

4.3 Rectangular Window Metrics

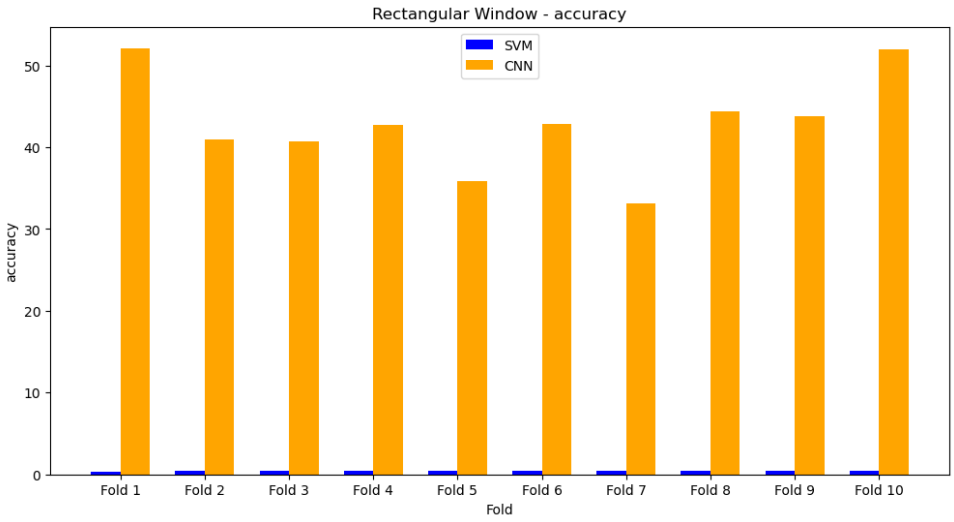

Figure 20: Rectangular Window Accuracy (SVM vs CNN)

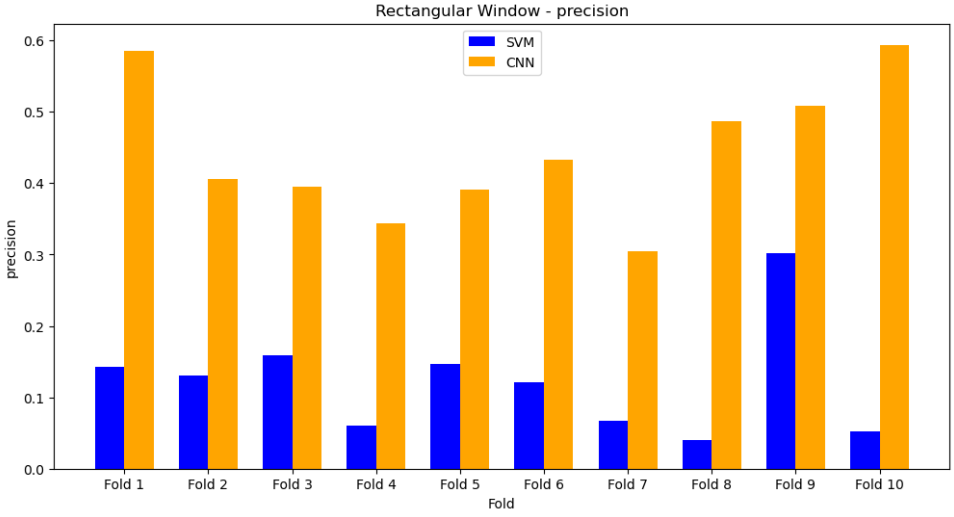

Figure 21: Rectangular Window Precision (SVM vs CNN)

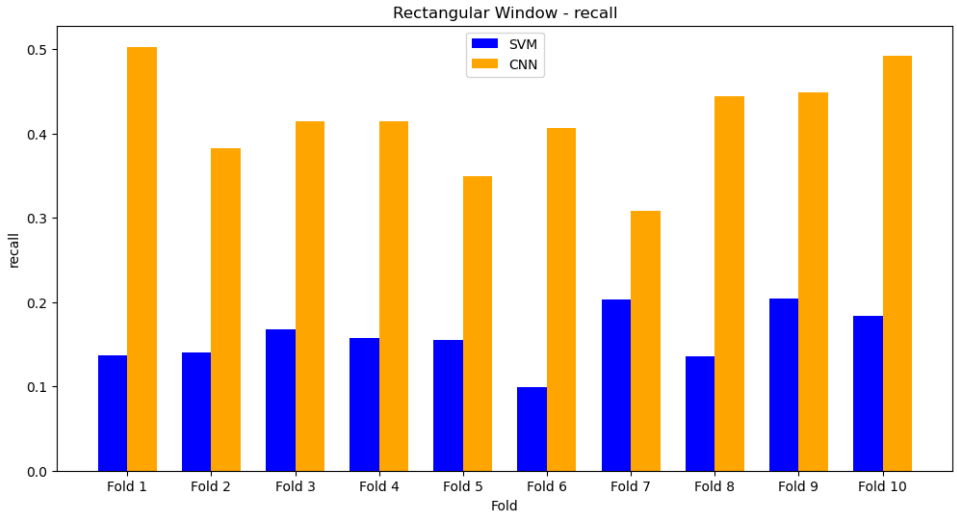

Figure 22: Rectangular Window Recall (SVM vs CNN)

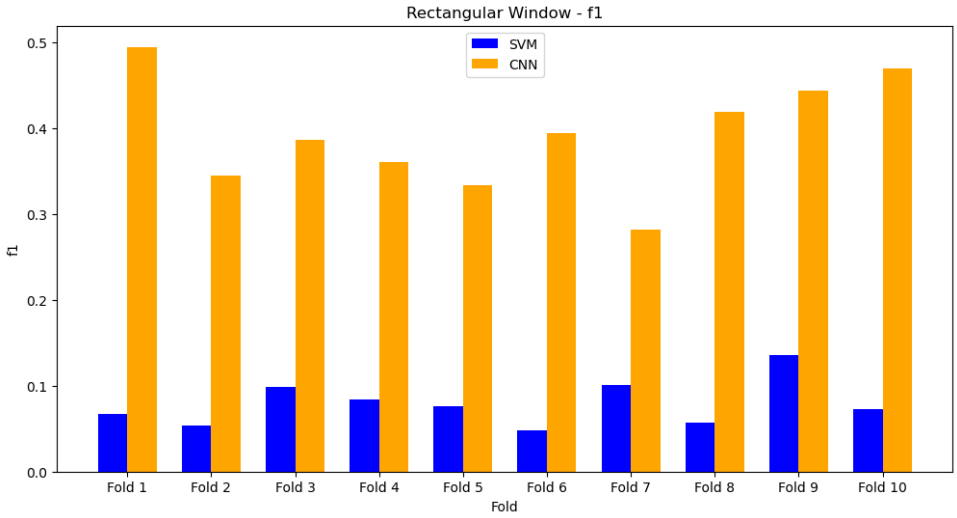

Figure 23: Rectangular Window F1-Score (SVM vs CNN)

4.4 Overall Dataset Metrics

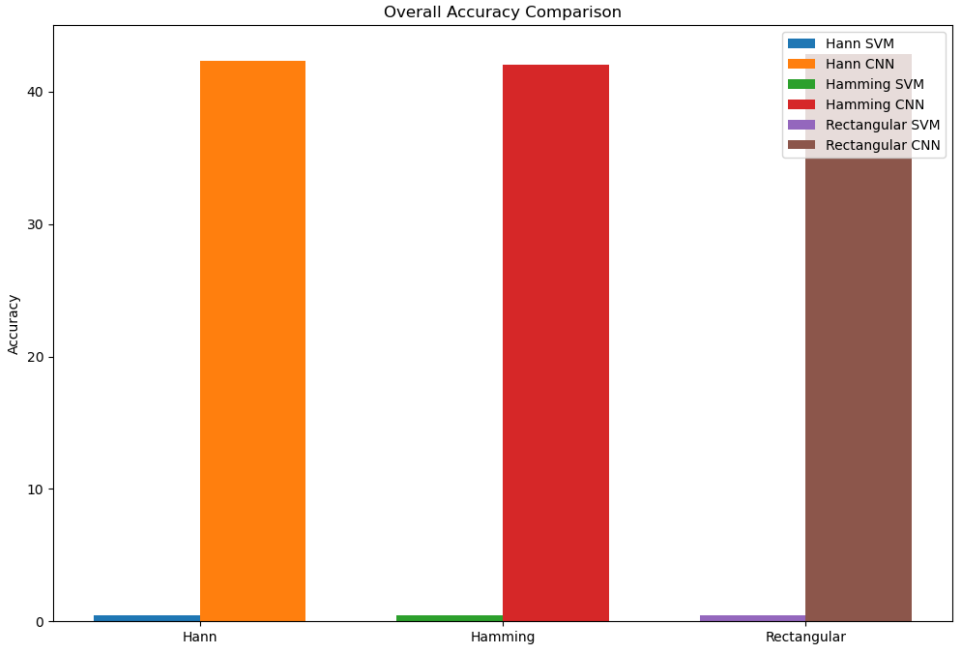

Figure 24: Overall Accuracy (SVM vs CNN)

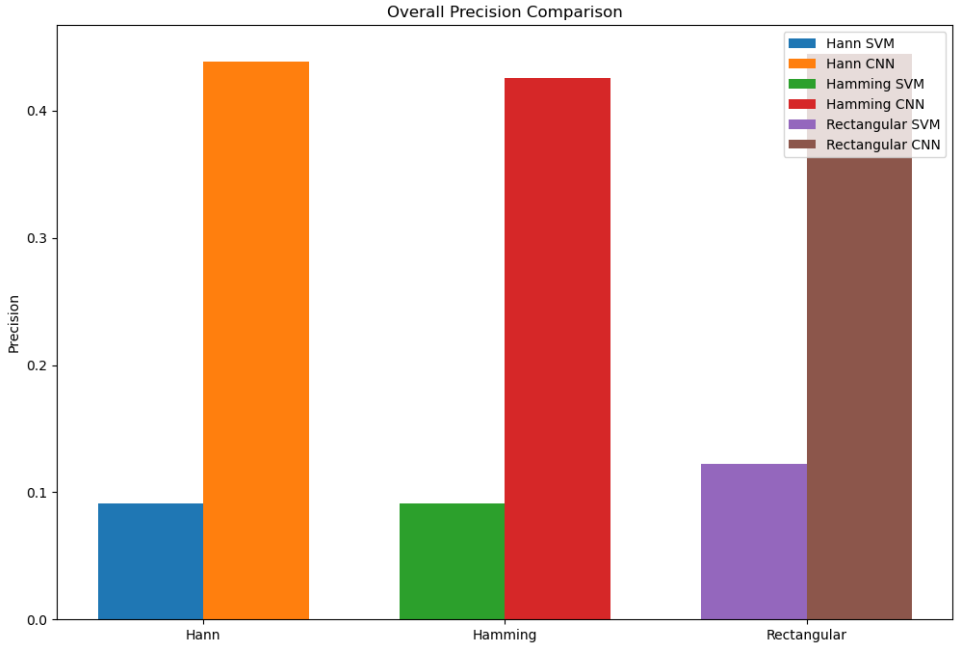

Figure 25: Overall Precision (SVM vs CNN)

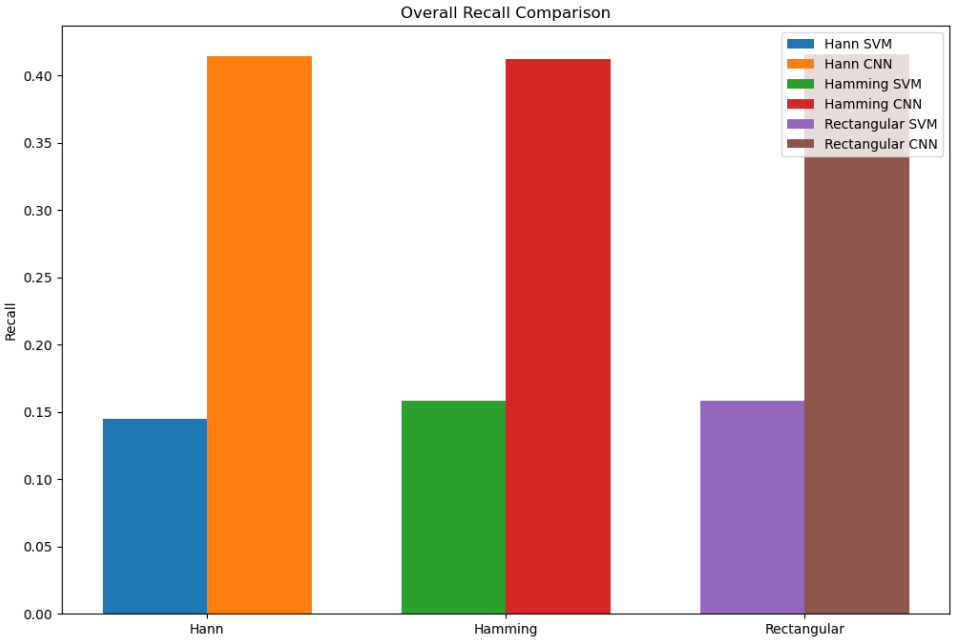

Figure 26: Overall Recall (SVM vs CNN)

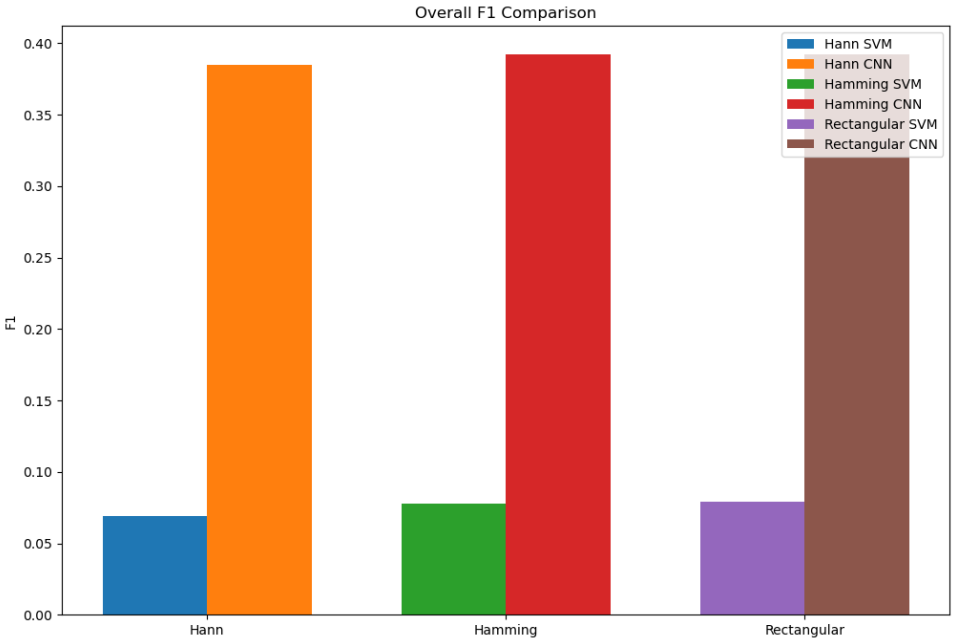

Figure 27: Overall F1-Score (SVM vs CNN)

5 Model Comparison

5.1 Performance Summary

Model	Hann	Hamming	Rectangular	Overall
CNN (Accuracy)	44.0%	42.0%	42.8%	43.0%
SVM (Accuracy)	1.9%	1.8%	1.2%	1.5%

Table 1: Comparison of accuracy across window functions and overall dataset.

5.2 Key Insights

- **Overall Performance**: The CNN consistently shows higher accuracy and better overall performance compared to the SVM. This suggests that deep learning methods are more effective at handling the complexity of urban sound signals.
- **Window Function Effects**: Among the three windowing techniques, the Hann window provides the best balance between reducing spectral leakage and maintaining important signal details. The Hamming window offers comparable performance, while the Rectangular window's increased leakage results in lower model accuracy.
- Rectangular window's leakage significantly impacts SVM performance, but CNN remains robust.

6 Further Improvements

While the current study provides valuable insights into the impact of windowing techniques and model performance, several avenues for further improvement exist:

- **Adaptive Windowing**: Investigate adaptive windowing strategies that adjust the window function dynamically based on the local characteristics of the audio signal.
- **Data Augmentation**: Implement advanced data augmentation techniques (e.g., pitch shifting, time stretching, and noise injection) to enhance the robustness of both the CNN and SVM models.
- **Advanced Architectures**: Explore more complex neural network architectures such as residual networks or attention mechanisms to capture finer details in spectrogram representations.
- **Feature Engineering**: Experiment with additional feature extraction techniques beyond MFCCs for the SVM, including chroma features or spectral contrast. Also try more iterations to reach a better minima.
- **Hyperparameter Optimization**: Conduct a systematic hyperparameter search to further optimize model performance.

Implementing these improvements could lead to enhanced classification metrics.

7 Conclusion

This report demonstrates that the method of converting audio into spectrograms—especially the choice of window function—has a significant impact on the performance of sound classification

models. Experiments with the UrbanSound8K dataset show that a CNN, which learns directly from high-quality spectrograms, outperforms an SVM that depends on engineered features. Future work could explore adaptive windowing techniques, more advanced data augmentation, and deeper network architectures to further enhance classification accuracy.

8 Bibliography

References

[1] UrbanSound8K Dataset. Available at: https://www.kaggle.com/datasets/chrisfilo/urbansound8k

[2] Audio Classification with CNN. Available at: https://www.kaggle.com/code/swayamshah09/audioclassification-cnn

[3] CNNs for Audio Classification. Available at: https://medium.com/towards-data-science/cnns-for-audio-classification-6244954665ab

[4] Audio Classification Using CNN Coding Example. Available at: https://medium.com/x8-the-ai-community/audio-classification-using-cnn-coding-example-f9cbd272269e