CSL7770 : Speech Understanding

Assignment 1 Report: Speech Understanding

Task 2(a): Comparison of Windowing Techniques and Classifier on UrbanSound8K

Name: Anuj Rajan Lalla

Roll Number: B22AI061

Date of Submission: 02-02-2025

1 Introduction

This report addresses the use of different windowing techniques (Rectangular, Hann, and Hamming) for Short-Time Fourier Transform (STFT)-based spectrogram generation. The **UrbanSound8K** dataset is used, which consists of urban audio clips categorized into ten distinct classes. The objectives include:

- Understanding the mathematical basis of each window function.
- Comparing the resulting spectrograms qualitatively to observe how window choice affects frequency resolution and spectral leakage.
- Training a simple neural network classifier on extracted features to assess any performance variations due to window selection.

2 Dataset Overview

UrbanSound8K provides:

- **8,732** short audio samples (each up to 4 s).
- **10 classes** of urban sounds (e.g., *siren*, *car horn*, *dog bark*).
- **10 folds** for cross-validation, ensuring each fold can act as a test set once.

In this work, spectrogram features are extracted for training a CNN, using nine folds for training and the remaining fold for testing, repeated for all folds.

3 Windowing Theory & Mathematical Formulation

During an STFT, the continuous signal is split into overlapping segments (frames). Each frame is multiplied by a window function $w[n]$ to reduce discontinuities at the edges. The three windows studied here are:

3.1 Rectangular Window

The simplest window, sometimes called the *boxcar* or *uniform* window, is defined by:

$$w_{\text{rect}}[n] = \begin{cases} 1 & 0 \leq n < N, \\ 0 & \text{otherwise.} \end{cases}$$

It applies no tapering, so the raw segment is used. Abrupt discontinuities at segment boundaries can cause higher *spectral leakage*.

3.2 Hann Window

The Hann window (sometimes called Hanning) is given by:

$$w_{\text{hann}}[n] = 0.5\left(1 - \cos(\tfrac{2\pi n}{N-1})\right), \quad 0 \le n < N.$$

This function smoothly tapers to zero at both ends of the segment, reducing boundary discontinuities compared to the rectangular approach.

3.3 Hamming Window

The Hamming window is closely related to Hann but with a slightly different cosine coefficient:

$$w_{\text{hamming}}[n] = 0.54 \; - \; 0.46\,\cos\!\left(\tfrac{2\pi n}{N-1}\right), \quad 0 \le n < N.$$

It also reduces discontinuities, often providing a slightly different balance of frequency resolution and leakage suppression.

4 Implementation Details

4.1 Global Configuration and Parameters

A consistent set of parameters was applied throughout:

- **Sampling Rate (SR):** The original audio sampling rate from each file was used (`sr=None` in *librosa.load*), preserving native sampling.
- **N_FFT = 1024:** The number of frequency bins for each STFT frame. A size of 1024 is a common trade-off, providing moderate frequency resolution while maintaining reasonable time resolution.
- **HOP_LENGTH = 512:** The shift between consecutive frames is half the FFT size, resulting in 50% overlap.
- **N_MELS = 40:** The number of Mel filter-bank bins for final feature extraction. This compresses frequency information into a Mel scale before the CNN.
- **FIXED_TIME_FRAMES = 130:** The spectrogram is padded/truncated to 130 frames to ensure consistent input shape.
- **BATCH_SIZE = 8, EPOCHS = 5, LR = 1e-3:** Basic hyperparameters for training the CNN.

4.2 Caching and Speed Optimization

Feature extraction can be computationally expensive. To avoid recomputing the same STFT/mel-spectrogram multiple times, an internal cache was used: once a file's features were generated for a particular window type, they were stored in memory. This cache was cleared (`FEATURE_CACHE.clear()`) before processing each new window technique to ensure accurate comparisons while also improving speed within each pass.

4.3 STFT and Spectrogram Generation

For the **Rectangular** case, a "manual" STFT function was implemented, multiplying each frame by $w_{\text{rect}}[n] = 1$. For **Hann** and **Hamming**, *librosa.stft* was used with `window='hann'` or `window='hamming'`. Magnitude spectrograms were converted to a decibel scale for visualization. Mel-scale conversion was then applied for the classifier input.

4.4 Neural Network Classifier

A simple CNN (`SoundCNN`) was employed:

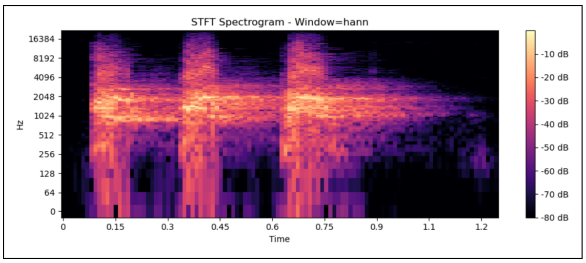
- 2 convolution layers (ReLU + max-pool).
- 1 fully connected layer + final output layer for the 10 classes.
- Dropout of 0.3 in the fully connected layer for regularization.

A 10-fold cross-validation was conducted, with each fold acting as the test set once, and the remaining folds used for training. Each experiment was repeated for Rectangular, Hann, and Hamming windows.

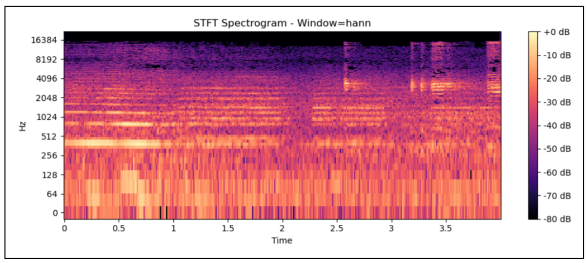
5 Results & Comparison

5.1 Spectrogram Examples Across Different Classes (2×2 Grid)

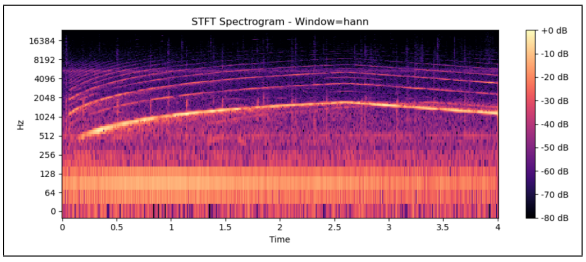
Four audio clips from the **UrbanSound8K** dataset were selected to illustrate the windowing effects. Each sub-figure applies one of the window techniques (Rectangular, Hamming, or Hann). Changes in windowing can affect how mid-to-high frequencies appear and the visual boundaries in time.



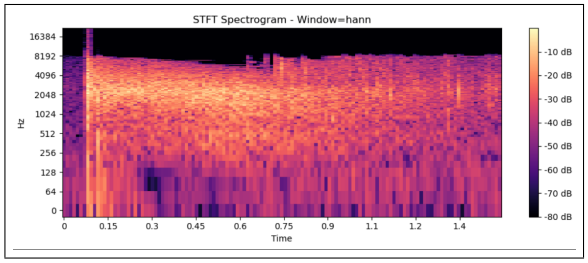
(a) Dog Bark (Hann Window)



(b) Car Horn (Hann Window)



(c) Siren (Hamming Window)



(d) Gunshot (Rectangular Window)

Figure 1: A 2×2 grid of spectrograms from four different classes, each applying one of the window techniques. Frames are truncated or tapered differently, influencing how frequencies appear in the time–frequency plane.

5.2 Spectrogram Comparisons for a Single Class (Siren)

A single "siren" audio clip was also processed under each window type (Rectangular, Hamming, Hann) to highlight the visual differences in detail:

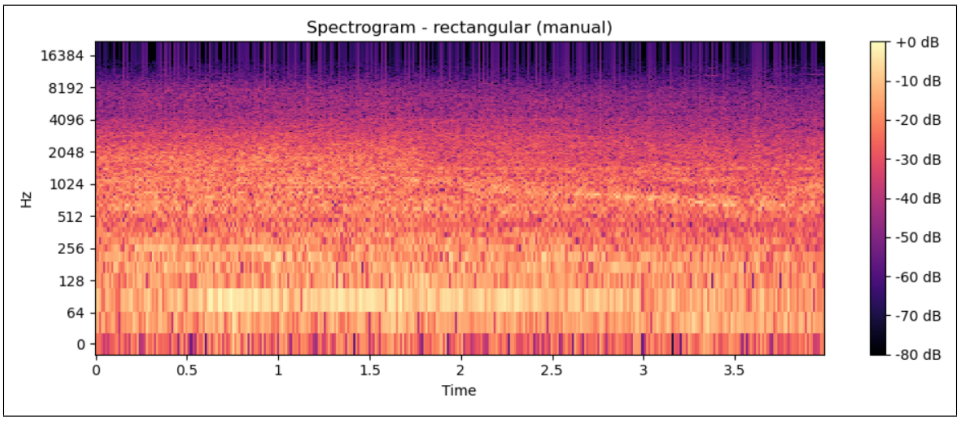


Figure 2: Spectrogram using a Rectangular (boxcar) window.

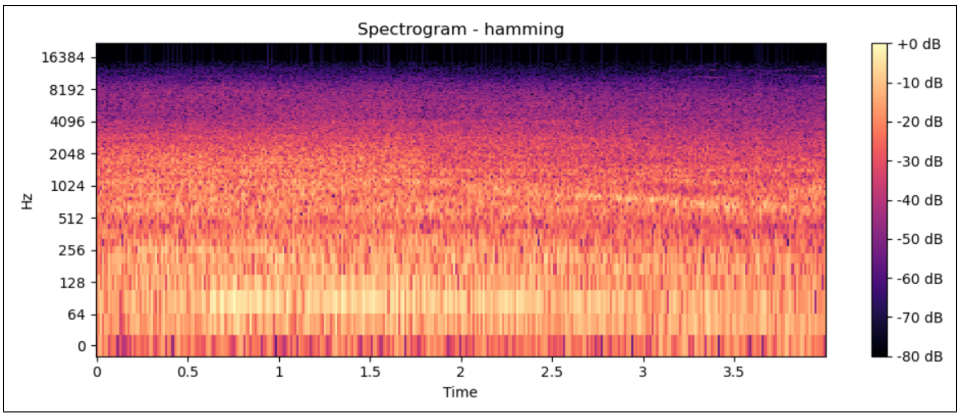


Figure 3: Spectrogram using a Hamming window.

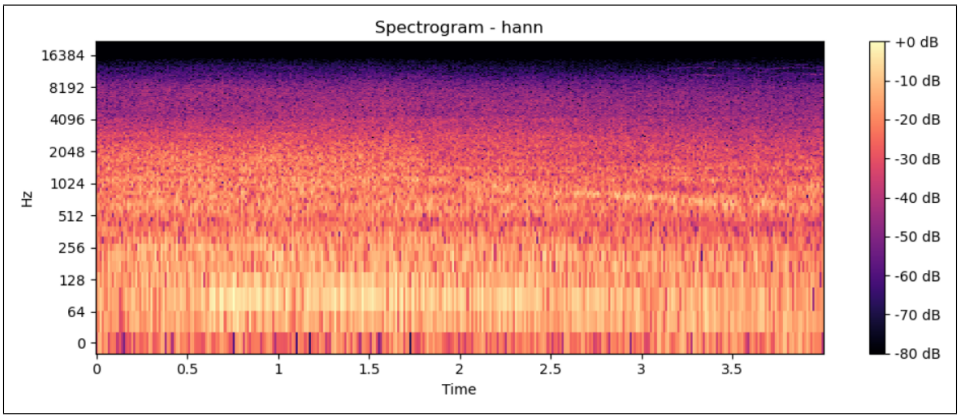


Figure 4: Spectrogram using a Hann window.

5.3 Impact of Windowing on Time Domain and Frequency Domain

The STFT splits audio into short frames. If each frame ends abruptly, it can induce "smearing" or "leakage" in frequency space. Tapered windows (Hamming, Hann) help fade the waveform near boundaries:

Rectangular Window (Manual)

- **Time-Domain Effect:** The signal remains at full amplitude until the edge of a frame, creating a sudden cutoff if the audio is not at zero amplitude.
- **Frequency-Domain Observation:** Such abrupt cutoffs introduce extra frequency components ("spectral leakage"). In spectrograms, this can appear as faint ripples or lines in higher frequencies.

Hamming Window

- **Time-Domain Effect:** Gently tapers to zero amplitude at frame edges.
- **Frequency-Domain Observation:** Reduces leakage significantly, yielding cleaner upper-frequency regions and smoother transitions.

Hann Window

- **Time-Domain Effect:** Similar tapering to Hamming, but uses a slightly different cosine coefficient.
- **Frequency-Domain Observation:** Also minimizes leakage, often striking a balance between resolution and artifact reduction.

5.4 Summarizing Observations

Rectangular windowing preserves the full waveform within each frame but can end each segment abruptly, resulting in extra "tails" in the spectrogram. Hamming and Hann windows gradually reduce amplitude to zero at the boundaries, decreasing energy leakage into adjacent frequency bins. This typically appears as:

- **Less artificial spreading of frequencies ("leakage").**
- **Smoother transitions between frames.**
- **Fewer faint lines** in the upper frequency regions.

These improvements can translate into slightly higher CNN classification accuracy, though the difference among the three windows may remain moderate.

5.5 Cross-Validation Results

A 10-fold cross-validation was performed on *UrbanSound8K*. Each fold was held out as a test set once, with the remaining nine used for training. Table 1 lists the accuracy for each fold under each window, plus the average (Mean) and variability (Std Dev). Higher mean accuracy indicates better overall performance; lower standard deviation suggests more consistent results across folds.

Table 1: CNN 10-Fold Accuracies by Windowing Technique

Fold	**Rectangular**	**Hann**	**Hamming**
Fold 1	0.455	0.496	0.585
Fold 2	0.501	0.500	0.467
Fold 3	0.463	0.478	0.387
Fold 4	0.498	0.475	0.501
Fold 5	0.599	0.600	0.631
Fold 6	0.544	0.542	0.554
Fold 7	0.548	0.575	0.532
Fold 8	0.573	0.541	0.542
Fold 9	0.533	0.609	0.522
Fold 10	0.570	0.584	0.536
Mean	0.528	0.540	0.526
Std Dev	0.046	0.048	0.063

Observations:

- **Rectangular** yields a mean accuracy of 0.528. Some folds (e.g., Fold 5) do well, but overall it slightly lags behind Hann (0.540).
- **Hann** achieves the highest mean (0.540), suggesting that it provides a clearer frequency representation.
- **Hamming** ends at 0.526, close to Rectangular but with the highest fold-to-fold variation (Std Dev 0.063). It excels in certain folds (Fold 5: 0.631) but dips in others (Fold 3: 0.387).

These findings align with the visual inspection of spectrograms: Hann and Hamming mitigate abrupt boundaries, leading to cleaner frequency data, and thus generally yield slightly better CNN performance than Rectangular.

6 Conclusion

Three windowing techniques for STFT—Rectangular, Hamming, and Hann—were examined on the *UrbanSound8K* dataset. Rectangular windowing, while straightforward, often introduces more spectral leakage. Hamming and Hann windows taper the edges of each frame, reducing leakage and producing smoother spectrograms.

The 10-fold cross-validation results show a modest but consistent advantage for Hann (mean accuracy of 0.540) and a similar performance for Hamming, compared to Rectangular (0.528). Although the differences are not large, selecting a tapered window is typically advisable for clearer frequency representation and slightly better classifier results. Future investigations could explore additional windows (e.g., Blackman) or more advanced neural architectures, but these experiments confirm that window choice can significantly influence spectrogram quality and subsequent classification accuracy.

References

[1] **UrbanSound8K Dataset:** https://urbansounddataset.weebly.com/urbansound8k.html

[2] **GitHub manual STFT reference:** https://github.com/aluchies/stft

[3] **B. Yang,** "A study of inverse short-time Fourier transform," in *Proc. of ICASSP*, 2008. (Added as author of GitHub repo has requested to do so if anyone uses his code)

[4] **Spectral Leakage Explanation, YouTube:** https://youtu.be/tCWU9C-LdJQ?list=PLKQAlM7GPLR2HnGtlOHZmKyjm6QzX_l1g

[5] **Kaggle Notebook on UrbanSound:** https://www.kaggle.com/code/prabhavsingh/urbansound8k-classification

[6] **Libraries used:** librosa, numpy, pandas, matplotlib, PyTorch.