Speech Understanding (CSL7770)
Assignment 1 - Question 2 Report
Shubh Goyal (B21CS073)

The code can be found here: GitHub Repository

TASK A

The task required us to make spectrograms of audio files with three different windows and perform classification using those spectrograms. The UrbanSound8K dataset had to be used for this which contains approximately 8300 audio files spread across 10 different classes of audio.

Methodology and Experiments

Windowing. Three windowing techniques were implemented; Hann, Hamming[1] and Rectangular using Numpy python library. The formulas implemented are as follows:

- Hann Window:

$$w(n) = 0.5\left(\sin^2\left(\frac{\pi n}{N-1}\right)\right), \quad 0 \leq n \leq N-1 \tag{1}$$

- Hamming Window:

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

- Rectangular Window:

$$w(n) = 1, \quad 0 \leq n \leq N-1 \tag{3}$$

Spectrograms. Next, spectrograms were generated using the Short-Time Fourier Transform (STFT) for all audio files with all three windows individually. The window size was kept as 1024 and the overlap was as 50% (512 samples).
The librosa [2] [11] and Scipy [3] python packages were used for implementation and the matplotlib [4] Python package was used for visualization and saving the spectrograms as png files.

Feature Extraction and Classifiction. Classification experiments were performed using two different methodologies that are as follows:

- **CNN based classification using spectrograms**
  The spectrograms generated from the audio files were directly used to train a Convolutional Neural Network. CNN had a resnet18-based backbone and two fully connected layers with the ReLU activation function in hidden layers. It was trained for 25 epochs with Adam optimizer and a learning rate of 0.001.

– **Siamese based Feature Extraction and Classification**
In here, self-supervised learning was used to learn an embedding space that can effectively represent the features from spectrograms. ResNet-18 was used as the backbone architecture of the model, and a 512-dimensional embedding space was created. The model was trained for 100 epochs using contrastive loss as the loss function and a learning rate of 0.001 for Adam optimizer.

After this, embeddings from all the spectrograms were used to train different models that included the following:

- ***Neural Network*** - four fully connected layers (512, 256, 128, 32, 10) with ReLU activation function in hidden layers was trained for 200 epochs using Adam optimizer with a learning rate of 0.001.
- ***Decision Tree*** - with max depth 20.
- ***Random Forest Classifier*** - with max depth 20 and 100 estimators in the forest.
- ***K-Neighbors Classifier*** - considering 5 nearest neighbors.
- ***Support Vector Classifier*** - polynomial kernel with degree 10.

The above mentioned parameters were obtained through manual hyperparameter tuning done for during the training phase.
All models were implemented and trained using PyTorch [5] and Scikit-Learn [6].

Window Correctness and Spectrograms

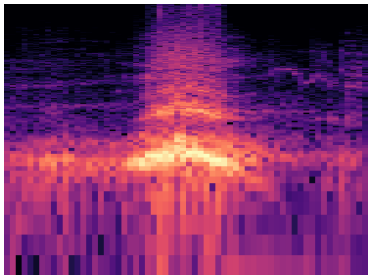

Fig. 1. Hamming

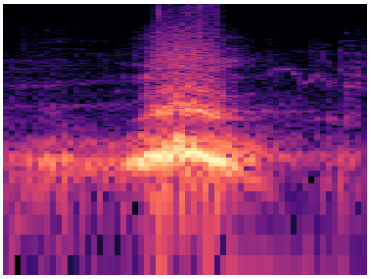

Fig. 2. Hann

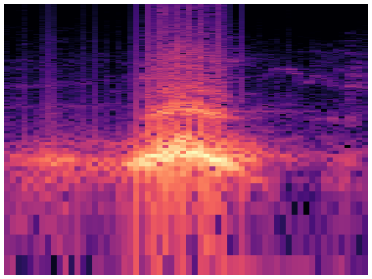

Fig. 3. Rectangular

Fig. 4. Spectrogram of Dog Bark

Observations. The spectrogram generated through Hann window seems to have more lower amplitude regions as compared to Hamming window spectrograms, inclining with the spectral leakage reduction offered by hann window over hamming window. Additionaly, the hann and hamming spectrograms seems to be smoother. And the spectrogram from rectangular window seems to have good amplitude even for higher frequencies and as can be seen the amplitude is high at many time ranges for many frequencies as compared to the hann and hamming spectrograms, suggesting the amount of spectral leakage caused by rectangular window. Additionally, the clarity of rectangular window spectrogram seems to be lower as there is a lot of mixing in between frequency components.

Window Correctness. Average RMSE of windowed against original audio signals for all the files were calculated and percentage error was calculated, for the three windows individually.

The Hann window and Hamming window had an error of 1.9% and 8.26% respectively, while the Rectangular window gave an error of 915%. This suggests the amount of spectral leakage possible in the three windows.

Results

Figure5 shows the accuracy comparision of different models across all the three window types.

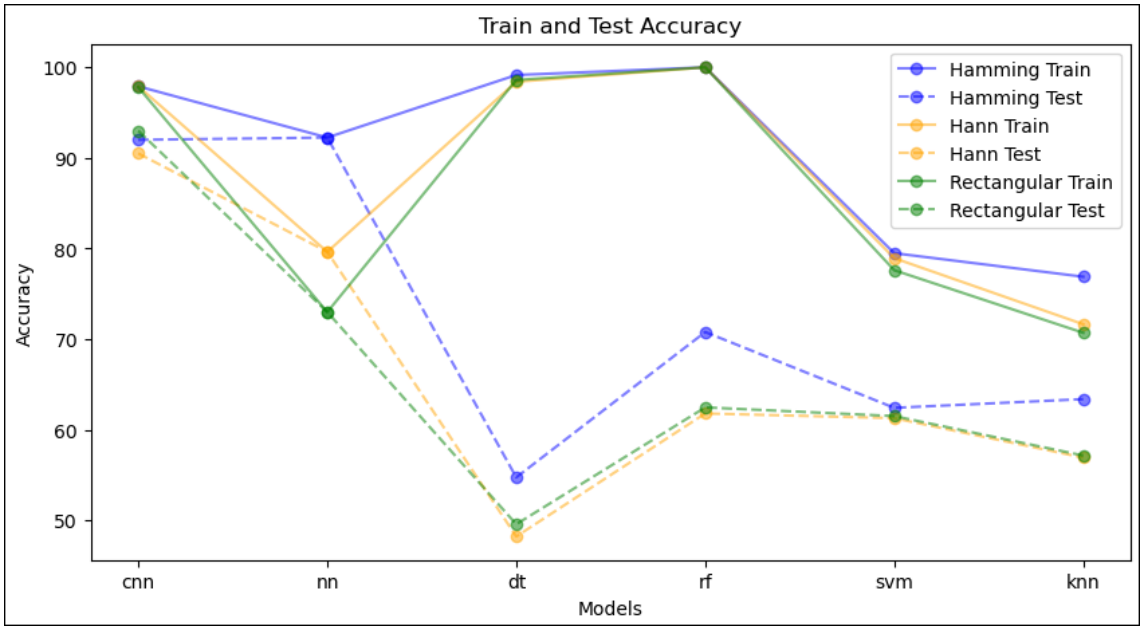

Fig. 5. Accuracy plot for all three windowing techniques across different models.
Solid lines represent training accuracy, while dotted lines represent test accuracy. Model abbreviations used in the plot: CNN - Convolutional Neural Network, NN - Neural Network, DT - Decision Tree, RF - Random Forest, SVM - Support Vector Machine, KNN - K-Nearest Neighbors.

Observations. It can be observed that CNN generalized better on the spectrograms for all the three windows and gave decent accuracies (greater than 90%) for test data for all the three windows.

Additionally, neural networks trained on the extracted features also generalized very well giving similar accuracies on training and testing data for all the three windows. The accuracy obtained for Hamming window was around 92%, but slightly lower for other two windows.

For simple machine learning classifiers, the accuracies didn't vary much for the three windows. But, decision tree classifiers and random forest classifiers clearly over-fitted, giving very low accuracy on test set in all the three cases.

Additionally, in case of knn classifier and support vector classifier, the generalization was better as compared to the previous two, but the overall accuracy was lower than random forest classifier on test data.

Overall, Neural Networks seemed to perform better. And, results obtained were slightly better for **'Hamming'** window across all the classifiers suggesting it is better suited for classification tasks.

TASK B

The task involved plotting and analyzing a spectrogram of four songs each from a different genre.

Spectrograms were generated using the Short-Time Fourier Transform (STFT) with a 'Hann' window of size 1024 and a 50% overlap (512 samples). The librosa [2] Python package was used for implementation and the matplotlib [4] Python package was used for visualization.

The four songs used and the observations made are as follows:

- **POP - Perfect** by Ed Sheeran [7]
  There is a consistency in the lower and mid frequency ranges of the spectrogram. There are regular high-low amplitude transitions which might be chorus or the verse changes in the song. The intensity variation seems to be pretty smooth as in case of pop songs. And the presence of slightly higher frequency may suggest the use of guitars.

- **RAP - Love The Way You Lie** by Eminem [8]
  The spectrogram is more segmented as compared to others, suggesting abrupt intensity on all frequency levels. And the frequent variations suggests the fast paced lyrics. It has more regular intensity in the lower range as compared to others, suggesting use of drums.

- **BOLLYWOOD - Sapna Jahan** from Brothers [9]
  The spectrogram contains comparatively larger number of silent or low amplitude blocks (at around 0:30, 2:00, 3:30, 4:20) suggesting long and short vocal pauses included to give an emotional element. Amplitude variations seems to be more gradual, as compared to Rap.

- **INDIAN CLASSIC - Ghoomar**, a Rajasthani Folk [10]
  The patterns in the spectrogram are rhythmic, it does not contain very abrupt shifts in frequencies, rather it contains more steady variations. Rhythms can be observed through the recurring peaks at the same frequencies. Additionally, it does not seem to focus very much on higher frequencies, but on low frequencies, indicating more natural instruments, rather than electronically synthesized ones like in Rap.

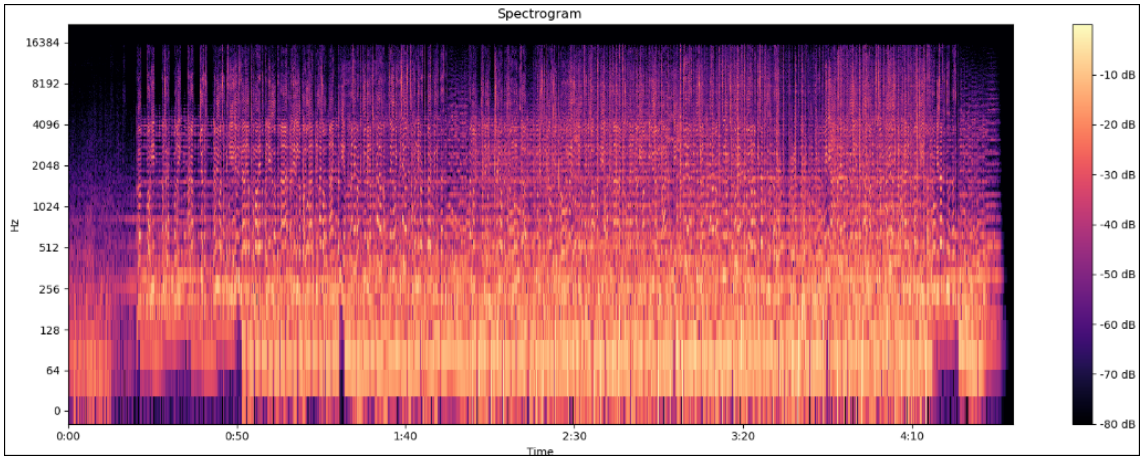

Fig. 6. Spectrogram of **Pop Song** - ***Perfect*** by Ed Sheeran

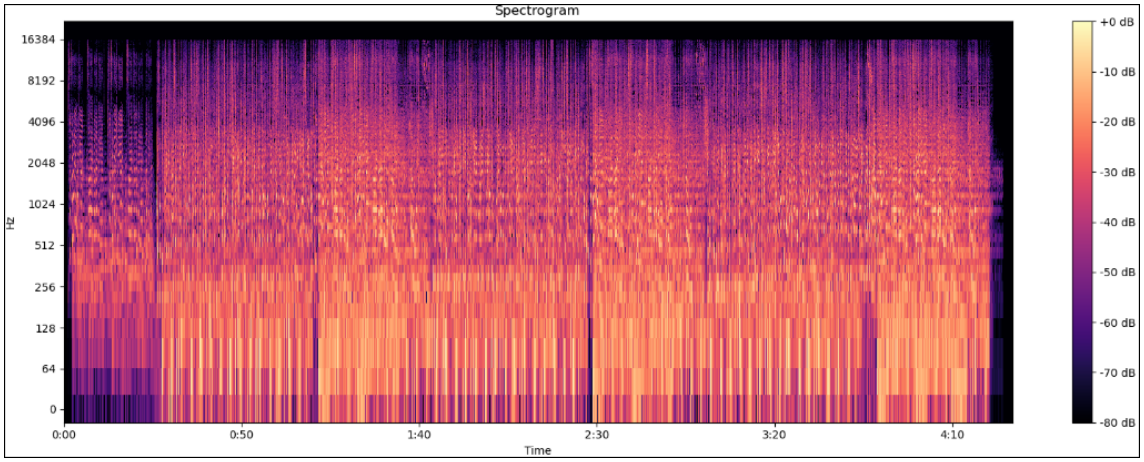

Fig. 7. Spectrogram of **Rap Song** - ***Love The Way You Lie*** by Eminem

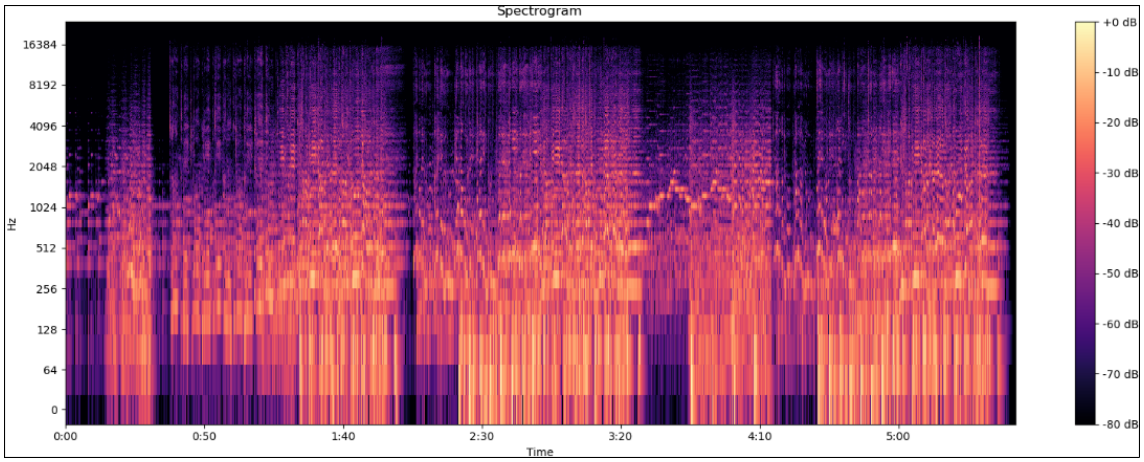

Fig. 8. Spectrogram of **Bollywood Song** - ***Sapna Jahan*** from Brothers

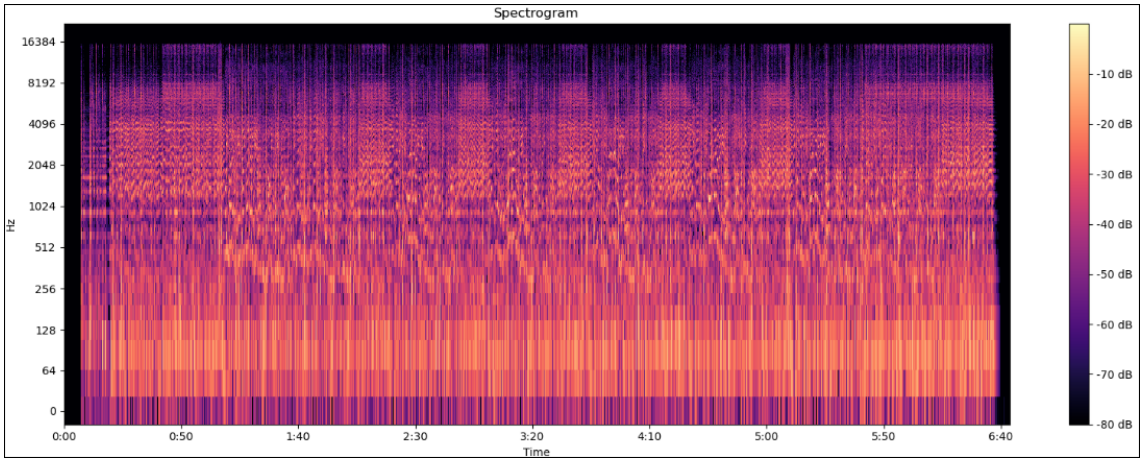

Fig. 9. Spectrogram of **Indian Classic Song** - ***Ghoomar***, Rajasthani Folk

References

1. https://www.mathworks.com/help/signal/ref/hamming.html.
2. https://librosa.org/doc/latest/index.html.
3. https://scipy.org/.
4. https://matplotlib.org/.
5. https://pytorch.org/.
6. https://scikit-learn.org/.
7. https://www.youtube.com/watch?v=2Vv-BfVoq4g.
8. https://www.youtube.com/watch?v=uelHwf8o7_U.
9. https://www.youtube.com/watch?v=Uvj827SqHak.
10. https://www.youtube.com/watch?v=nHhRWgkkpMk&list=RDQMFaGfjevYPQY&start_radio=1.
11. 2020. https://khareanu1612.medium.com/audio-signal-processing-with-spectrograms-and-librosa-b66a0a6bc5cc.