Drone detect: A report by Anu Bazarragchaa

Introduction

In our study of the drone audio data, I found some interesting insights through spectrogram visualization. Utilizing the Librosa library in my code, I visualized the first 5 audio files within the “yes_drone” folder. The code generated 5 spectrograms, which you will see down below, with colorful representations illustrating sound intensity and frequency over time. Within these spectrograms, I saw multiple pitch-black sections above the 8192 Hz-and-up threshold. Moreover, I saw that the specific frequency bands where the intensity surpassed 80 decibels (dB), indicating loud sounds within those ranges. This might be potentially the drone sound’s most notable character - amidst background noise we can now successfully move onto the next parts of our research.

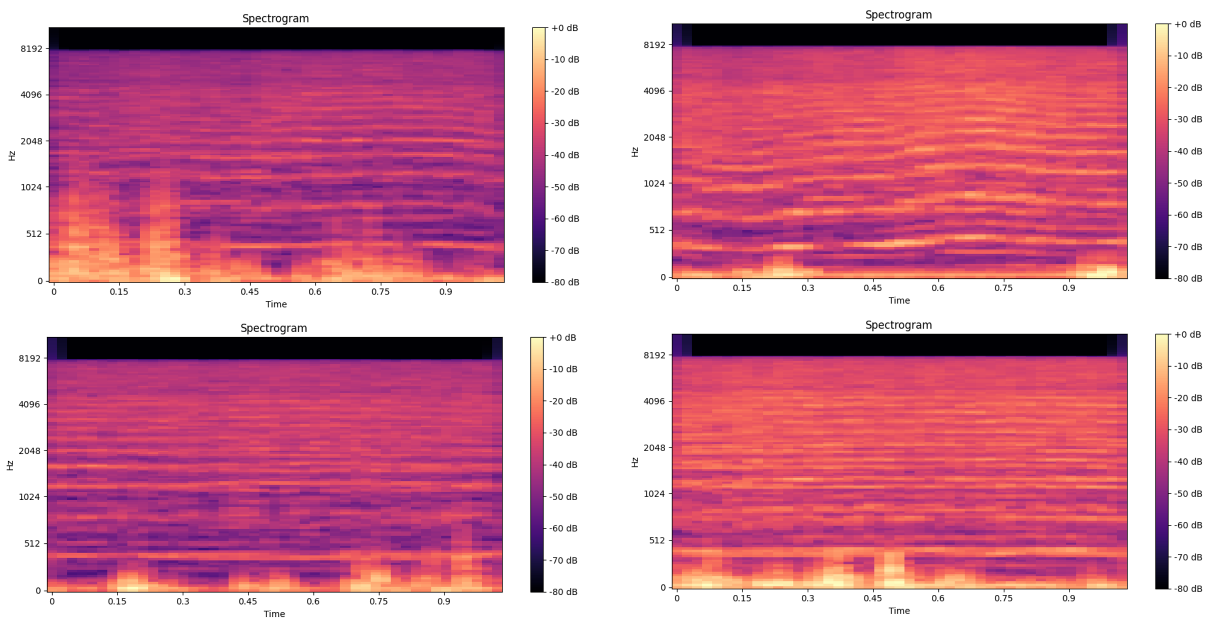


Figure 1: Spectrograms of the first 4 audio files with drone noise.

In these visualizations (Figure 1), Hz (Hertz) denotes frequency, reflecting the number of sound cycles per second. Different dB levels at specific Hz indicates different sound intensities. This means that higher dB values at certain frequencies showcase louder sounds, while lower or absent dB values depict quieter or silent frequency bands.

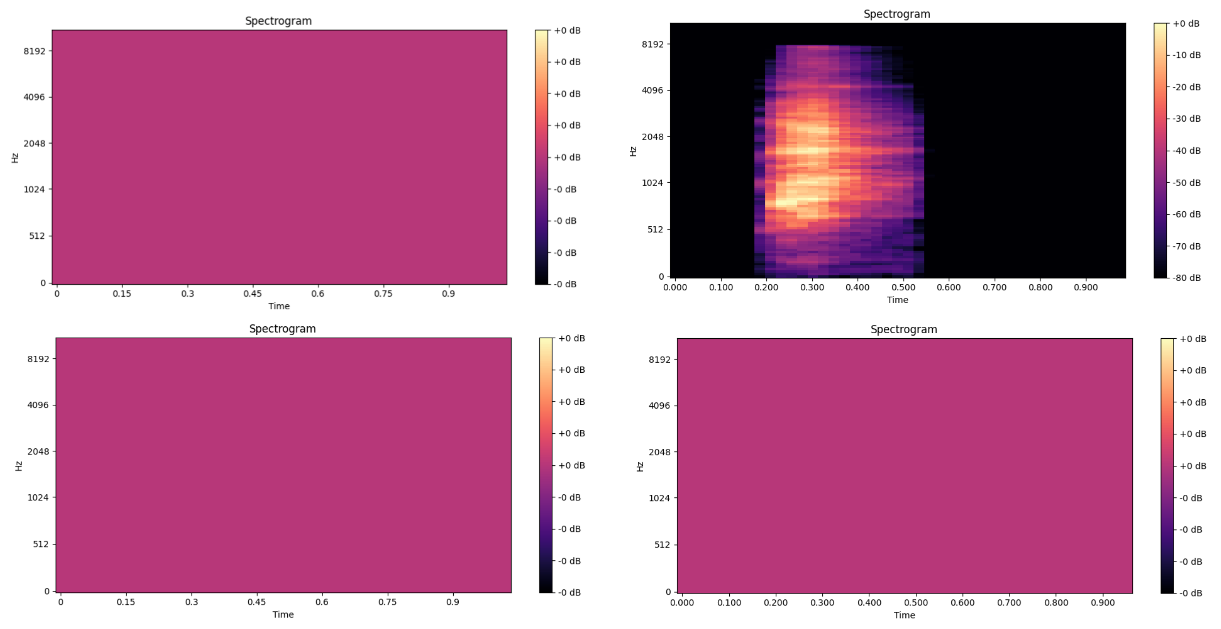


Figure 2: Spectrograms of the first 4 audio files without drone noise.

In Figure 2, I visualized the first few files of the “no-drone” dataset. We can see that there is also a general absence of the specific region described before - most of the dataset is silent (0db). I also noticed that in one of the audios, there was a very loud noise file among the first 5 files. This also helps us classify the drone data further: If the whole dataset is noisy, even though it has the same region we are looking for in the drone dataset, we should also check the rest of the frequency for the presence of loud sounds.

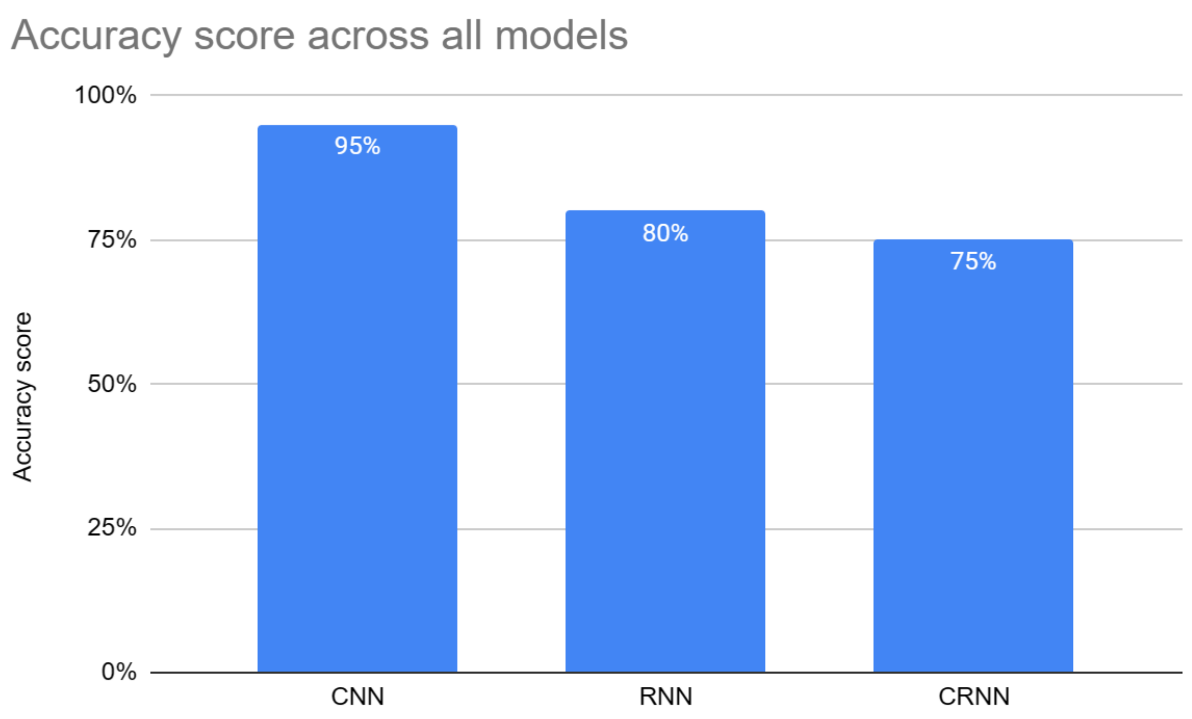
Training

During the data preparation, we divided the datasets into training (80%), validation (10%), and testing (10%) sets. This is different from the 70,15,15 ratio that was mentioned in the source publication, which kind of suggests that our results will vary vastly – and the data handling is to blame.

I used PyTorch to give the data some structure in order to proceed with the model training. In order to do this, I created a class called ‘AutoDataset’, extracted its Mel-spectrogram features, and converted them into PyTorch tensors. This makes it easier for the upcoming steps of training.

The project was done using the following libraries: NumPy, PyTorch, Librosa and MatPlotLib for visualization.

While reading the publication, I saw that the team benchmarked three models: CNN, RNN, and CRNN. The CNN model was able to capture spatial features from spectrogram-adjacent data with 95% accuracy. The RNN and CRNN models, on the other hand, had considerably lower accuracies of 80% and 75%.



Conclusion

The model's accuracy being at such vastly different levels helps us reflect on what went differently than that of the original publication and the [17]th publication the authors have quoted. The first thing that comes to mind, is of course, the data. There may have been issues with data preparation, such as the splitting might have been different from the original publication. In this project, we did not, in fact, use the 70,15,15 ratio for the training, validation and testing.

A deeper look into CNNs might suggest that they are particularly good at capturing spatial features in spectrogram-adjacent data, while RNNs are often sought out for sequential data analysis. The architecture and hyperparameters such as the number of layers in question, hidden units, learning rate, etc. could all have had a huge impact on performance.

In the publication they create CPU-Time, Accuracy, Precision, Recall and F-score evaluation metrics. During the course of this experiment, I faced computing capacity challenges (linked to outdated hardware), hence the report only includes a miniature amount of points to make the process faster. Accuracy is the fraction of properly categorized cases among all instances. This was the only

Analysis

It is clear that there is a trade-off between model performance and computing efficiency. Our experiment highlights potential trade-offs between training duration and performance, providing insights into model selection concerns for practitioners. Finally, the comparison analysis emphasizes the significant influence of experimental sets on model performance. It highlights the need of balanced data splitting, selecting appropriate models based on data properties, and the crucial role of architectural and hyperparameter considerations. These findings have significance for academics and practitioners working to improve drone detection approaches that use auditory characteristics.

Key Metrics

Over the course of this project and additional reading, I've gathered that the following key metrics are the most important when assessing accuracy and effectiveness of models.

1. Accuracy - This is a crucial metric that shows the model's performance. Because it measures overall correctness, as this should be available for **ALL** models.
2. Precision-recall: Shows false positives, and false negatives respectively. It's needed because when one error type is heavily affecting your model's accuracy, you can tell it here.
3. F1 score: This balances precision and recall. It provides a more comprehensive performance metric.
4. Loss function: Shows the gap between anticipated and actual values is quantified and used to steer the model's learning process. Used in order to improve predictive accuracy during training.
5. Computational Efficiency: Time and resource use are measured, which is critical for real-time or resource-constrained applications. This helps count for resource allocation and helps make cost calculation easy.