Combining time-frequency and machine learning for drone audio classification

Raphaël Grisel
ENSTA Bretagne
Email: raphael.grisel@ensta.fr

***Abstract*—This paper explores the integration of time-frequency analysis and machine learning techniques for the detection and classification of drone audio signals. The rapid advancement of drone technology has raised significant security and privacy concerns, necessitating robust methods for identifying and categorizing drones based on their acoustic signatures. This study builds upon existing research by implementing a comprehensive framework that combines spectrogram representations with convolutional neural networks (CNNs) to achieve high accuracy in both detection and classification tasks. The framework is evaluated using a publicly available drone audio dataset, and the results demonstrate improved performance compared to previous studies. The findings highlight the effectiveness of spectrograms over Mel-scale spectrograms in capturing relevant frequency information for CNN-based classification. This work contributes to the development of reliable drone detection systems, crucial for safeguarding physical infrastructure and protecting society from potential threats.**

***Index Terms*—Drone, UAVs, Deep Learning, Time-Frequency Analysis, CNN, classification, Spectrogram.**

I. Introduction and related work

A. Context

Drone technology has rapidly advanced in recent years, leading to significant security and privacy concerns. As drones have become more affordable and accessible to the public, they also pose substantial risks, particularly when used for malicious activities. Authorities must address threats such as spying on private properties, monitoring critical areas, and transporting dangerous [1] or illegal objects.

Beyond illegal activities, drones also pose risks to cybersecurity and physical safety. Accidental incidents, such as losing connection between the drone and its remote controller, can result in unintended crashes, potentially causing severe damage. [2]

Given these challenges, the detection and classification of drones are essential for safeguarding physical infrastructure and protecting society. Accurate drone detection helps secure vulnerable areas, while effective classification allows for distinguishing between authorized and unauthorized drones.

B. Drone detection and classification

Several frameworks have been studied for detecting and classifying drones, as described in [3].

Radio-Frequency (RF) Detection: RF-based detection is a cost-effective method that can cover long distances and identify the type of drone. However, this method has significant limitations: it only works when the drone is actively transmitting signals, making it ineffective against autonomous flights.

Radar-Based Detection: Radar technology provides long-range detection and operates efficiently both day and night with minimal impact from weather conditions. However, radar systems are expensive and prone to false positives. Moreover, interpreting radar signals requires trained personnel.

Visual Detection: Vision-based techniques use high-resolution imaging to detect, classify, and track drones. However, this method is dependent on lighting conditions, has a limited range, and is susceptible to adverse weather. Additionally, it requires a direct line of sight to the target.

Acoustic-Based Detection: Acoustic detection relies on identifying the unique sound signatures of drones. It is relatively inexpensive and has low power consumption.

Furthermore, some studies have focused on acoustic-based drone detection using advanced signal processing and machine learning techniques. These techniques have demonstrated high accuracy and reliability in real-world scenarios.

C. State of the Art: Time-Frequency Analysis and Machine Learning

The combination of time-frequency analysis (TFA) and machine learning (ML), particularly deep learning (DL), has become a powerful approach for the classification and analysis of non-stationary signals. TFA enables the representation of a signal in the time-frequency domain, revealing features that are not easily visible in either the time or frequency domains alone. ML/DL, on the other hand, provides powerful tools for automatic feature extraction and classification, adapting to the complexities of real-world signals.

1) Principles of Time-Frequency Analysis: TFA encompasses a set of mathematical techniques that transform signals from the time domain to the time-frequency domain, generating time-frequency images (TFI) or spectrograms [4]. These visual representations illustrate the energy or power distribution of a signal over time and frequency, allowing detailed analysis of signal characteristics.

Several TFA methods exist, each with its own advantages and disadvantages:

- **Short-Time Fourier Transform (STFT)**: A classical method that computes the Fourier transform over short time windows.

- **Fourier-based Synchrosqueezing Transform (FSST)**: An improvement over STFT that provides better time-frequency resolution.
- **Wigner-Ville Distribution (WVD)**: A quadratic distribution offering high resolution but suffering from interference artifacts (cross-terms).
- **Smoothed Pseudo-Wigner Distribution (SPWD)**: A variant of WVD that reduces cross-terms at the cost of slight resolution loss [5].
- **Choi-Williams Distribution (CWD)**: Another distribution that mitigates cross-terms in WVD.
- **Continuous Wavelet Transform (CWT)**: A method using wavelets at different scales to analyze signals.

The choice of TFA method depends on the signal characteristics and application requirements.

2) Machine Learning and Deep Learning for Signal Classification: ML/DL has revolutionized signal classification by enabling automatic extraction of relevant features from raw data. Deep neural networks (DNNs), particularly convolutional neural networks (CNNs) [5], [6]and recurrent neural networks (RNNs) , have demonstrated exceptional performance in various signal classification tasks. [6]

- **CNNs**: Excellent for extracting spatial features from TFIs, widely used in audio and radar signal classification.
- **RNNs**: Capable of modeling temporal dependencies, well suited for sequential signals such as speech and music.

Other ML algorithms, such as support vector machines (SVM), multilayer perceptrons (MLP), random forests (RF), and XGBoost, can also be used for signal classification.

3) Applications of Combined TFA and ML/DL: The integration of TFA and ML/DL has found applications in various domains:

- **Radar Signal Classification [7]**: TFA visualizes micro-Doppler signatures of radar targets, while ML/DL enables automatic classification.
- **Transient Stability Analysis in Power Systems**: TFA reveals the dynamic behavior of power system signals during disturbances, while ML/DL predicts system stability.
- **Speech and Music Recognition**: TFA captures spectral features of speech and music, while ML/DL facilitates recognition and classification.
- **Biomedical Signal Characterization [8]**: TFA analyzes EEG, ECG, and EMG signals, while ML/DL detects anomalies and diagnoses diseases.
- **Drone Detection**: Unique acoustic signatures of drones can be captured using TFA and classified using ML/DL algorithms for detection and identification.

4) Drone Detection using Time-Frequency Analysis and Machine Learning: Several studies have explored the combined use of TFA and ML/DL for drone detection based on their acoustic signatures.

- **Feature Extraction**: Extracted features from drone audio signals include Mel-frequency cepstral coefficients (MFCC), zero-crossing rate (ZCR), spectral centroid, spectral roll-off, and spectrograms.
- **Classifiers**: Utilized classifiers include Gaussian mixture models (GMM), CNNs, RNNs, and SVMs.

These studies demonstrate that ML/DL can effectively detect drones from their acoustic signatures, even in noisy environments.

5) Challenges and Future Directions: Despite significant advancements, several challenges remain in the combined use of TFA and ML/DL:

- **Selection of TFA Methods and ML/DL Architectures**: Automatically determining the appropriate time-frequency transformation and optimal deep learning architecture remains a challenge.
- **Data Overload**: The high dimensionality of TFIs can lead to data overload, requiring preprocessing, regularization, and hyperparameter optimization techniques.
- **Generalization**: Ensuring that ML/DL models generalize well to different types of drones, environments, and operating conditions is crucial.
- **Lack of Reference Datasets**: The absence of publicly available and standardized datasets hinders comparison and validation of different approaches.
- **Ranging**: Few studies have explored classification performance as a function of drone distance, which is crucial for risk assessment.

Future research directions include: Combining time-frequency representations with deep learning models has shown great performance for many reasons. Time-frequency analysis is particularly suitable for studying non-stationary signals—signals whose characteristics vary over time [4]—which is the case with drone audio. Time-frequency representations allow for the visualization of signals and the extraction of information that is not accessible in either the time or frequency domain alone [7]. Deep learning models, especially Convolutional Neural Networks (CNNs), are highly effective at extracting features from time-frequency images, making them valuable for classification tasks.

These combined approaches have demonstrated high performance and have been applied in electrical engineering, particularly in studying electrical network stability [7]. A combination of the Wigner-Ville representation and CNNs has also been used for detecting pathological voice disorders [8].

Finally, [6] demonstrates that combining spectrogram representations of drone audio with deep learning models is an effective approach for drone detection and classification. Among Recurrent Neural Networks (RNNs), Recurrent Convolutional Neural Networks (RCNNs), and Convolutional Neural Networks (CNNs), the latter has shown the highest accuracy for both detection and classification tasks.

The study conducted in this paper is based on the same drone audio dataset as in [6]. The contributions of this paper are manifold:

- Building a complete framework for audio drone detection and classification, accessible at GitHub Repository.
- Building a combining spectrogram and CNN method and comparing results with [6].
- Building a combining spectrogram and CNN method and comparing results with [6].

II. Dataset

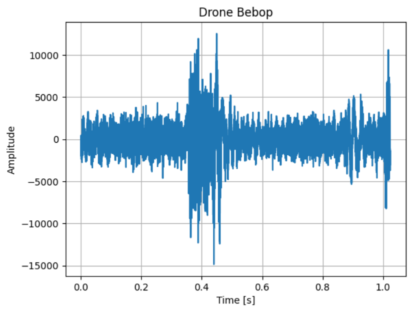
A. Dataset Description

The dataset used in our study was proposed by [6]. Due to the lack of publicly available drone audio data, [6] created an open-access drone audio dataset.

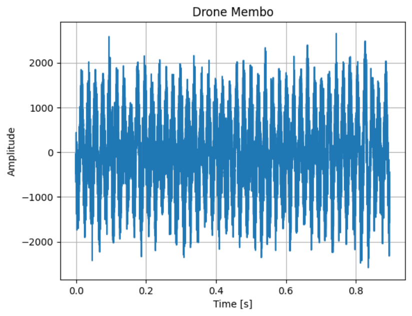
The dataset consists of recordings from two types of commercially available drones: the Bebop and Mambo models from Parrot. These are lightweight quadcopter drones primarily used for video filming. The recordings were made using a smartphone microphone while the drones were flying and hovering in a quiet indoor environment. For each drone, a continuous recording of 11 minutes and 6 seconds was captured. Each recording was then segmented into 1-second clips, with each segment labeled according to the corresponding drone type. To better reflect real-world scenarios, environmental noise was added to the drone audio recordings. Additionally, a separate dataset consisting only of environmental sounds was created to enable the model to distinguish drone sounds from background noise.
Two different datasets were constructed The first dataset is designed for the detection task and consists of two classes: drone sound (labeled ”yes drone”) and environmental sound (labeled ”unknown”). The drone sound class consists of augmented recordings from both drone types, totaling 1,333 samples. The environmental sound class contains more than 10,000 samples. This dataset allows the model to learn how to detect drone sounds among environmental noise. The second dataset is used for multiclass classification and includes three classes: Bebop class, Mambo class, and Unknown class, represented by 666, 666, and 10,335 samples, respectively.

B. Dataset Analysis

The final sample format is WAV, and the sample rate is 16 kHz. Before stating the specific tasks, we conducted a general sound analysis. The following figures show the temporal representations of the Bebop and Mambo drones without environmental sound augmentation:



(a) Bebop Drone



(b) Mambo Drone

Fig. 1: Temporal visualizations of the drone signals

We can analyze the characteristics of the signals for the two drones:

Bebop Drone Signal Analysis

- The signal exhibits high amplitude variations with significant fluctuations.
- Around 0.4s, a strong spike is observed, possibly indicating an event such as a sudden movement or external disturbance.
- The waveform appears to have a wider dynamic range, suggesting the presence of high-frequency components.
- There is a visible increase in intensity at different time intervals, indicating possible variations in drone activity.

Mambo Drone Signal Analysis

- The amplitude remains more uniform and consistent over time.
- The signal appears more periodic, displaying a regular oscillatory pattern.
- Fewer extreme peaks are present compared to the Bebop drone, implying a more stable operation.
- The signal is likely less affected by sudden disturbances, indicating more stable drone motion or environmental conditions.

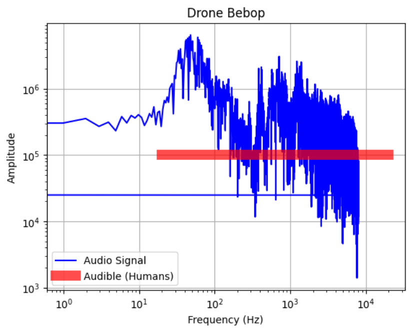
Applying the Fast Fourier Transform (FFT) can provide valuable insights into the frequency content of these signals.

The Fast Fourier Transform (FFT) is an efficient algorithm for computing the Discrete Fourier Transform (DFT), which converts a discrete-time signal from the time domain to the frequency domain:

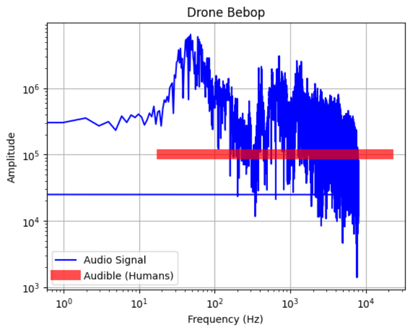
$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi kn}{N}}, \quad k = 0, 1, 2, \ldots, N-1$$

where:

- $X[k]$ represents the frequency-domain representation of $x[n]$.
- j is the imaginary unit, where $j = \sqrt{-1}$.
- $e^{-j\frac{2\pi kn}{N}}$ represents the complex exponential basis functions.



(a) Bebop Drone



(b) Mambo Drone

Fig. 2: FFT visualization of the drone signals

Once again, we observe that the magnitude of the Mambo drone signal is lower than that of the Bebop drone. This FFT analysis reveals that the drones produce significant noise in the audible frequency range, particularly between 100 Hz and 5 kHz.

Additionally, the frequency content of the Bebop drone at lower frequencies (below 1 kHz) is more prominent than that of the Mambo drone.

III. Framework

In the following section, we describe the framework implemented for detection and classification tasks on time-frequency images of the described dataset.

A. Data Processing

Data processing is of great importance as it shapes the overall approach. First, we ensure that the labels are equally represented in the datasets. Using a dataset with class imbalance can lead to lower accuracy and a biased model in deep learning.

In both datasets, the class containing environmental sound samples is larger. Therefore, the first step is to randomly select a limited number of samples to ensure an equal number of samples for each class.

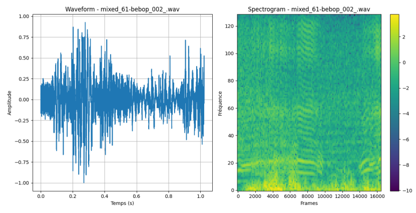
Then, the data processing implementation takes into account the chosen data representation.

1) Spectrograms: Spectrogram representation is a visually meaningful way to characterize a signal. A spectrogram is a visual representation of the frequency content of a sound signal over time. It is obtained by applying the Short-Time Fourier Transform (STFT) to the signal.

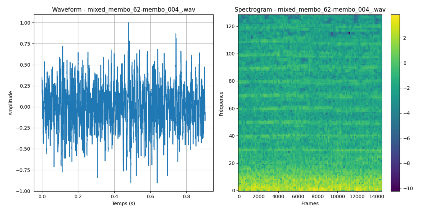
- **Start with a sound signal**: Represented as a time-domain waveform $x[n]$.
- **Apply the Short-Time Fourier Transform (STFT)**:
$$X(m,k) = \sum_{n=0}^{N-1} x[n]w[n-mR]e^{-j\frac{2\pi kn}{N}}$$
where a window function $w[n]$ is applied to extract short segments of the signal.
- **Compute the magnitude spectrum**: The spectrogram is obtained by taking the squared magnitude of the STFT:
$$S(m,k) = |X(m,k)|^2$$
- **Convert to decibels (optional)**: To improve visualization, the power is often expressed in dB:
$$S_{\text{dB}}(m,k) = 10\log_{10} S(m,k)$$

It was shown that spectrograms are good to extract features for CNNs model [6] [8]. That is why it will be the first time frequency representation we will study for our detection and classification tasks.

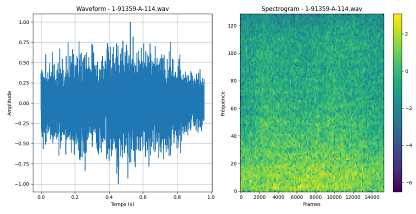
The spectrograms of drone sounds demonstrate that most of the energy is concentrated in the lower frequencies, whereas environmental sounds exhibit more energy at higher frequencies. Some patterns extend beyond 100 Hz, indicating the presence of higher-frequency harmonics, likely due to mechanical noise and environmental reflections.



(a) Bebop Drone



(b) Mambo Drone



(c) environemental sound

Fig. 3: Signal and their spectrogram

2) Mel Spectrogram: Our study also tested Mel Spectrogam as a way to represent drone sound. A Mel spectrogram is a spectrogram that uses the Mel scale to represent frequency. The Mel scale is a perceptual scale that approximates how humans perceive pitch, making it particularly useful for speech and audio analysis. It gives closer human perception representation : humans do not perceive frequency linearly; lower frequencies are more distinguishable. It will reduce high-frequency details which is coherent with audio dataset caracteristics. The mel Spectrogram is obtained by applying the steps :

- Start with a sound signal $x[n]$.
- Apply the Short-Time Fourier Transform (STFT) to obtain the frequency representation over time:

$$X(m,k) = \sum_{n=0}^{N-1} x[n]w[n-mR]e^{-j\frac{2\pi kn}{N}}$$

  where $w[n]$ is a window function, and R is the hop size.
- Compute the spectrogram by taking the squared magnitude:

$$S(m,k) = |X(m,k)|^2$$

Applying the Mel Filter Bank

- Convert the linear frequency scale (Hertz) into the Mel scale using the formula:

$$f_{\text{mel}} = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

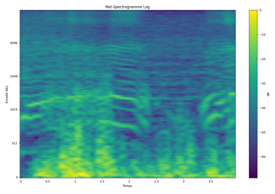
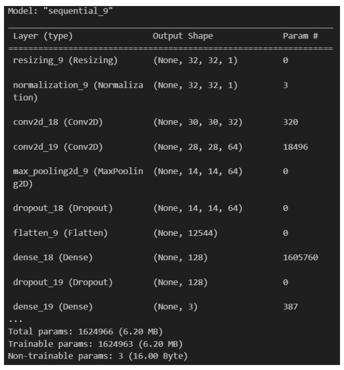


Fig. 4: Mel Spectrogram for Bebop drone



```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resizing_9 (Resizing)       (None, 32, 32, 1)         0

 normalization_9 (Normaliza  (None, 32, 32, 1)         3
 tion)

 conv2d_18 (Conv2D)          (None, 30, 30, 32)        320

 conv2d_19 (Conv2D)          (None, 28, 28, 64)        18496

 max_pooling2d_9 (MaxPoolin  (None, 14, 14, 64)        0
 g2D)

 dropout_18 (Dropout)        (None, 14, 14, 64)        0

 flatten_9 (Flatten)         (None, 12544)             0

 dense_18 (Dense)            (None, 128)               1605760

 dropout_19 (Dropout)        (None, 128)               0

 dense_19 (Dense)            (None, 3)                 387

...
Total params: 1624966 (6.20 MB)
Trainable params: 1624963 (6.20 MB)
Non-trainable params: 3 (16.00 Byte)
```

Fig. 5: Model Summary

- Create a Mel filter bank, which consists of overlapping triangular filters that are spaced according to the Mel scale.
- Multiply the spectrogram by these filters to obtain the Mel spectrogram.

Convert to Decibels

- Apply a logarithm to better visualize intensity variations:

$$S_{\text{mel, dB}}(m,k) = 10 \log_{10} S_{\text{mel}}(m,k)$$

With this Mel spectrogram representation, the lower frequency content is emphasized, as shown in the figure.

To compute these representations in our implementation, we used the signal library from TensorFlow, which contains tools for signal analysis.

B. Model

For the Deep Learning model, we implemented a simple Convolutional Neural Network for classification , the following figure describes it:

This model summary describes a convolutional neural network (CNN) architecture, which is commonly used for image processing tasks. Let's break down the components and their purposes:

- **Resizing Layer**
  - **Type**: Resizing
  - **Output Shape**: (None, 32, 32, 1)
  - **Purpose**: Resizes the input images to a fixed size of 32x32 pixels with a single channel, ensuring consistency for batch processing in neural networks.
- **Normalization Layer**
  - **Type**: Normalization

- **Output Shape**: (None, 32, 32, 1)
  - **Purpose**: Scales pixel values to a specific range (typically 0 to 1), stabilizing and speeding up training by maintaining a consistent input data distribution.
- **Conv2D Layers**
  - **Type**: Conv2D (Convolutional)
  - **Purpose**: Applies convolutional operations using filters (kernels) to extract image features. The first layer has 32 filters, while the second has 64, enabling deeper pattern learning.
- **MaxPooling2D Layer**
  - **Type**: MaxPooling2D
  - **Purpose**: Performs down-sampling by selecting the maximum value in each patch of the feature map. Reduces spatial dimensions, decreasing computational complexity and preventing overfitting.
- **Dropout Layers**
  - **Type**: Dropout
  - **Purpose**: Randomly deactivates a fraction of input units during training to prevent overfitting, encouraging robust feature learning.
- **Flatten Layer**
  - **Type**: Flatten
  - **Purpose**: Converts 2D feature maps into a 1D feature vector, preparing data for fully connected layers.
- **Dense Layers**
  - **Type**: Dense (Fully Connected)
  - **Purpose**: Performs classification through weighted sums of inputs followed by activation functions. The first dense layer has 128 units, and the final output layer has 3 units, corresponding to three output classes.

This is a lightweight CNN model. To make it more complex, it is possible to add additional convolutional or fully connected layers. The chosen model is simple but incorporates strategies to mitigate overfitting.

C. Training

The training loop is designed to efficiently train a deep learning model while optimizing data loading and learning rate scheduling. It starts by preparing the training and validation datasets using TensorFlow's cache and prefetch operations. Caching speeds up data retrieval by storing elements in memory, while shuffle randomizes the training set to prevent order-based biases. The prefetch function improves performance by allowing the CPU to prepare the next batch while the GPU processes the current one.

- The loop defines two different learning rate (LR) schedules :
  - Cosine Decay Scheduler: This reduces the learning rate smoothly using a cosine function, starting from 0.01 and gradually decreasing to nearly zero by the end of training. This approach helps the model converge efficiently by taking larger steps initially and smaller steps as it approaches a minimum. The decay is computed as:

    $$\alpha_t = \alpha_{\text{initial}} \times 0.5 \times \left(1 + \cos\left(\frac{t}{T}\pi\right)\right)$$

    where t is the current step and T is the total number of decay steps.
  - Exponential Decay Scheduler: This decreases the learning rate exponentially with each step using the formula:

    $$\alpha_t = \alpha_{\text{initial}} \times \text{decay rate}^{(t/\text{decay steps})}$$

    Here, the initial LR is 0.001, and it decays by a factor of 0.96 every 100,000 steps in a staircase fashion (discrete jumps instead of smooth transitions). This ensures stable learning as training progresses.

  We tried both schedulers for training and Exponential Decay Scheduler showed better performances.
  Loss Function: Sparse Categorical Crossentropy The model is compiled with the Sparse Categorical Crossentropy Loss, which is widely used for multi-class classification with integer labels. It is defined as:

  $$L = -\frac{1}{N}\sum_{i=1}^{N} \log P(y_i)$$

  where $P(y_i)$ is the predicted probability for the correct class. The model outputs raw logits (unnormalized scores), and TensorFlow automatically applies the softmax function internally to convert them into probabilities before computing the loss.
- Optimizer: The Adam optimizer is used with the learning rate scheduler. Adam is a widely used optimizer that combines the benefits of Adaptive Gradient Algorithm and Root Mean Square Propagation by adjusting learning rates per parameter using first-moment (mean) and second-moment (variance) estimates.
- Early Stopping : It monitors validation loss and stops training if the loss does not improve for 5 consecutive epochs. This prevents overfitting (where the model memorizes training data instead of generalizing) and saves computational resources. It works by keeping track of the best validation loss and stopping when no improvement is detected.

D. Evaluation

To evaluate the performances of the detection and classification tasks, we decided to compute accuracy metrics, F1-score and to use a confusion matrix to visualize classfication error.

1) Accuracy and F1-score: Accuracy is a simple but effective metric for evaluating the overall performance of the model. It is defined as the ratio of correctly predicted instances to the total number of tested samples:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

where:

- TP (True Positives): Correctly classified positive examples.
- TN (True Negatives): Correctly classified negative examples.
- FP (False Positives): Negative examples incorrectly classified as positive.
- FN (False Negatives): Positive examples incorrectly classified as negative.

We compute the accuracy metrics over the dataset of test by comparing the true labels with predictions after applying `argmax` to extract the predicted class.

The F1-score is a metric that balances precision and recall into a single score. It is particularly useful when dealing with imbalanced classes. It is defined as the harmonic mean of precision (P) and recall (R):

$$F1 = 2 \times \frac{P \times R}{P + R} \tag{2}$$

where:

$$P = \frac{TP}{TP + FP} \tag{3}$$

$$R = \frac{TP}{TP + FN} \tag{4}$$

2) Confusion Matrix: The confusion matrix is a powerful tool for visualizing classification model performance. It helps identify which classes the model confuses the most and provides insights into classification errors.

A confusion matrix is a table where each element $C_{i,j}$ represents the number of instances of class i predicted as class j. It is defined as follows:

$$C_{i,j} = \text{Number of samples from class } i \text{ predicted as class } j \tag{5}$$

It is computed using `confusion_matrix` from `sklearn`:

It is then normalized to express values as percentages: Finally, a heatmap is generated using `seaborn` to visualize it.

IV. Binary Classification

As an initial implementation, we decided to train the CNN model for a detection task. The model will learn to distinguish drone sounds from environmental sounds.

A. Training configuration

We trained the CNN model described on a training dataset of spectrogram images, followed by training on a dataset of Mel-scale spectrogram images. The model was trained over 10 runs of 50 epochs, and we computed the mean accuracy and mean F1-score.

The plots below show that both the validation loss and training loss decrease over epochs. Similarly, both validation accuracy and training accuracy increase over epochs. The model is neither prone to overfitting nor underfitting.

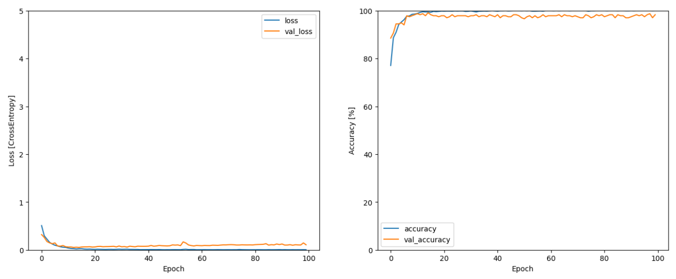


Fig. 6: Training loss and accuracy over epochs

B. Experimentation results

After the 10 runs of 50 epochs, we computed the mean accuracy and obtained 97.62%, along with an F1-score of 98.1These metrics show that our model performs better than the one implemented by [6]. Indeed, for a CNN model trained for a detection task, the authors achieved a mean accuracy of 96.38% and an F1-score of 95.90

The following figure shows the confusion matrix obtained after 50 epochs of training of the detection pipeline :

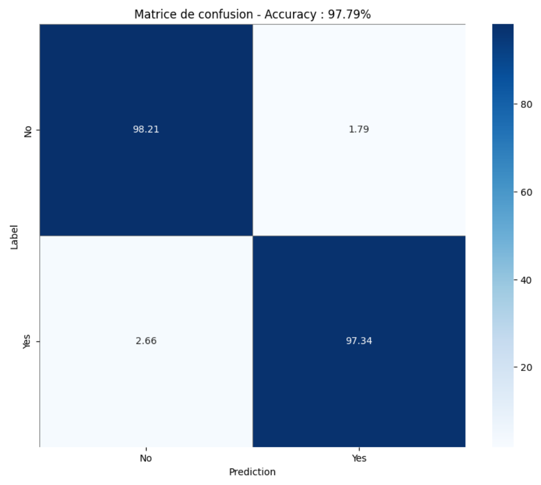


Fig. 7: Confusion matrix for detection task

The model achieves an accuracy of 97.79%, which is quite high, indicating strong overall performance.

- True Negatives (Top-left: 98.21%): 98.21% of the actual ”Unknown” class samples were correctly classified as ”Unknown”.

- False Positives (Top-right: 1.79%): 1.79% of the actual ”Unknown” samples were misclassified as ”yes_drone”. This represents false positives (FP), meaning the model incorrectly predicted the presence of the positive class.

- False Negatives (Bottom-left: 2.66%): 2.66% of the actual ”yes_drone” (positive class) samples were incorrectly classified as ”unknown” These are false negatives (FN), meaning the model failed to detect the positive class.

- True Positives (Bottom-right: 97.34%): 97.34% of the actual ”yes_drone” samples were correctly classified as ”yes_drone”

What we can understand from this confusion matrix is that the model has a low error rate as both false positive (FP) and false negative (FN) rates are below 3%. The confusion matrix indicates a balanced performances across both classes, meaning no class imbalance issues in prediction.

C. Mel-Spectrogram dataset

We also trained the detection model on a dataset of Mel-scale spectrograms. While the Mel-spectrogram is more suitable for analyzing signals with significant content in the low frequencies, it appears to be less effective for training the CNN. The model achieved an accuracy of 97.14%, which is very close to the performance achieved with standard spectrograms.

V. CLASSIFICATION TASK

A. Training configuration

The second implementation focused on the classification task. In this phase, the model had to learn not only to distinguish drone sounds from environmental sounds but also to classify drone sounds into two categories: Membo and Bebop. We trained the CNN model to predict three classes: Membo, Bebop, and Unknown.

The model was trained for 10 runs over 50 epochs. It successfully converged, with no signs of underfitting or overfitting observed.

B. Experimenation results

After 10 runs of 50 epochs, we computed the mean accuracy and obtained 96.05% accuracy and a 95.9% F1-score, whereas the authors in [6] achieved 92.94% accuracy and a 92.63% F1-score.

The figure below shows the confusion matrix after training the classification model for 50 epochs:

Here are the key observations from this confusion matrix:

- The model correctly classifies 98.46% of Bebop sounds and 95.90% of Unknown sounds.
- The Membo class has the highest misclassification rate.
- Membo sounds share features with both Bebop and Unknown, making classification more challenging.

To improve the classification model, potential strategies include increasing the training data for Membo sounds to help the model learn better class distinctions. Additionally, adjusting hyperparameters or exploring different architectures could enhance the classification of Membo samples.

C. Mel-Spectrogram dataset

We trained the model on a Mel-Spectrogram dataset of the drone audio dataset. The model achieved lower performance, with an accuracy of 87.4%.

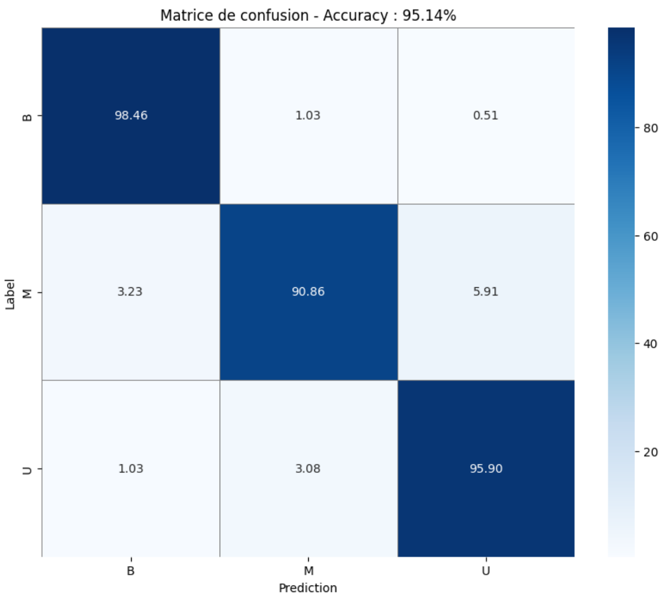


Fig. 8: Confusion matrix for the classification task

VI. ANALYSIS

Our study reveals two major findings: the detection and classification models we implemented perform better than the one studied in [6]. The second finding is that the best time-frequency representation is spectrograms rather than Mel-scale spectrograms. Better performances than the paper Spectrogram better than mel-scale spectrograms

A. Better performances than the existing work

The improved performance of the detection and classification algorithms can be attributed to both the preprocessing pipeline and the CNN model design.

In the preprocessing pipeline, we selected only a portion of the environmental sound samples to prevent class dominance, which can lead to poorer performance in deep learning models. Additionally, the normalization of spectrograms helped the model converge more effectively and improved accuracy.

Regarding model implementation, the careful selection of convolutional layers, max-pooling layers, and dense layers contributed to better performance. Furthermore, incorporating dropout layers helped the CNN model retain only the most relevant neurons for feature extraction throughout the training process.

B. Spectrogram is the best TF representation

The difference in performance between training a CNN on regular spectrogram images versus Mel-scale spectrogram images is due to how each representation captures frequency information and how this affects the CNN’s ability to learn patterns.

Spectrograms use a linear frequency scale, whereas Mel-scale spectrograms apply a nonlinear frequency scale that compresses higher frequencies. This compression can lead to a loss of fine details, particularly in high-frequency components, which might be crucial for distinguishing classes.

Initially, we assumed that since most of the frequency content is concentrated in the lower frequencies, it would be beneficial to use a representation that emphasizes them. However, our findings suggest that the higher frequencies played a key role in differentiating the classes.

Moreover, Mel spectrograms apply a filter bank to group frequencies, which smooths out the frequency representation. This smoothing can lead to a loss of sharp frequency transitions and harmonics that a CNN might otherwise exploit.

VII. Conclusion

In this paper, we introduced existing methods for the detection and classification of drone sounds. We also described approaches that combine time-frequency analysis with deep learning. Specifically, we presented a method using a CNN model trained on two different time-frequency representations: spectrograms and Mel-scale spectrograms.

Our study compared the results with those from the study conducted in [6]. The implemented algorithm demonstrated better performance than the previous study in both detection and classification. Additionally, our findings indicate that spectrograms are more suitable than Mel-scale spectrograms for CNN training in detection and classification tasks.

Finally, we provided a complete pipeline, including data preprocessing, different time-frequency representations, and CNN models, for detecting and classifying drone sounds.

VIII. Futur Work

Our study focused on representing sounds as images and applying a CNN model to perform specific tasks, such as detection or classification.

Extracting meaningful features from time-frequency analysis combined with machine learning models can highlight important time-frequency properties for sound classification. Viewing sound as a time series of varying frequencies over time and studying them as time series is one approach. However, studying other methods that combine features extracted from time-frequency analysis with machine learning models could provide further insights and potentially enhance classification performance.

References

[1] T. Guardian, "P. daniels, venezuela's nicolas maduro survives apparent assassination attempt," *https://www.theguardian.com/world/2018/ aug / 04 / nicolas - maduros - speech - cut - short - while -soldiers-scatter.*, 2018.

[2] wnyt, "Drone strikes iowa toddler on playground," *https : / / wnyt . com / news / drone-strikes-toddler-on-playground-iowa/5010493/.*, 2018.

[3] B. TAHA and A. SHOUFAN, "Machine learning-based drone detection and classification: State-of-the-art in research," *IEE*, 2019.

[4] R. B. Constantin CONSTANTINESCU, "An overview on sound features in time and frequency domain," *Sciendo*, 2023.

[5] S. v. E. Christonasis Antonios Marios1 and P. Duin, ""seeing sound": Audio classification using the wigner-wille distribution and convolutional neural networks," *arXiv*, 2019.

[6] A. M. A. A.-A. Sara Al-Emadi, Abdulla Al-Ali, "Audio based drone detection and identification using deep learning," *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, 2019.

[7] M. Parlak, "Use cases for time-frequency image representations and deep learning techniques for improved signal classification," *arXiv:2302.11093v1*, 2023.

[8] V. C. Georgopoulos, "Advanced time-frequency analysis and machine learning for pathological voice detection," *12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, 2020.