# "A Comparative Study of Input Features and Augmentation Techniques in CNN-CRNN models for Environmental Sound Classification "

Laura Legrottaglie[†]

***Abstract*—Environmental sound classification (ESC) is an expanding field with applications in surveillance, smart homes, and urban noise management. However, ESC faces unique challenges due to the diverse nature of environmental sounds and limited labeled data. This report presents a comparison of CNN and CRNN architectures that utilize different audio features, including log-mel spectrograms, MFCCs, and raw waveforms, to determine the most effective approach for ESC. The impact of various data augmentation techniques, such as mixup, white noise injection, and pitch shifting, on the performance of the models is also assessed. Results indicate that the CRNN model with log-mel spectrogram inputs achieve the highest accuracy of 74.80%, capturing intricate patterns across frequencies. Ensemble methods further enhance accuracy, with product-based ensemble surpassing individual models' performance and reaching an accuracy of 79.10%. Analysis of class-wise accuracy suggests that class-specific augmentation strategies may further optimize models' generalization ability. These considerations offer important perspectives in the field of sound recognition systems, contributing to ongoing progress in enviromental sound classification.**

***Index Terms*—Enviromental sound classification, Augmentation, Convolutional Neural Networks, Convolutional Recurrent Neural Networks, Mixup**

## I. Introduction

Environmental sounds encompass a wide range of everyday audio events that cannot be classified as either speech or music, such as the sounding of a dog barking or a glass breaking. Environmental sound classification (ESC) has received increasing research attention in recent years due to its potential applications in fields such as surveillance, home automation, urban noise management, wildlife monitoring, and intelligent sound systems. Unlike music and speech recognition, ESC faces unique challenges due to limited pre-existing knowledge about the temporal and frequency characteristics of environmental sounds. Additionally, the field is constrained by a lack of standardized datasets, methodologies, and evaluation criteria, as well as significant variability across different sounds [1]. These factors, combined with the difficulty of reproducing complex preprocessing techniques, make ESC a challenging and evolving area of study.

In recent years, Convolutional Neural Networks (CNNs) have proven to be highly effective for sound event recognition, due to their ability to capture complex, multi-layered features directly from audio data. This work aims to investigate and develop a range of architectures with the objective to effectively recognize the different environmental sounds. To achieve this, we will evaluate and compare various models, including Convolutional Neural Networks (CNNs) and Convolutional Recurrent Neural Networks (CRNNs), to determine the most effective architecture for accurate environmental sound classification.

The setup includes developing three distinct CNN architectures and one CRNN model. These include a CNN that utilizes log-mel spectrogram features, another that directly processes raw audio waveforms, and a third that uses the first 20 Mel-Frequency Cepstral Coefficients (MFCCs) as input. The first and third networks are shallow CNNs with 4 convolutional layers, designed to capture both low and high-level features from log-mel spectrograms and MFCC representations, respectively. In contrast, the CNN for raw audio is an end-to-end 11-layer fully convolutional model employing 1D convolutions, which draws inspiration from the work of Dai et al. in [2]. The CRNN model, instead, also uses log-mel spectrograms as input and employs two bidirectional Gated Recurrent Units with the aim of capturing complex patterns and relationships across frequencies.

To further enhance performance, the four models are ensembled by combining their predicted probabilities. Probabilistic ensembling is implemented by calculating both the product and the average of the predicted probabilities from each model. This enables exploration of two aggregation strategies: taking the product, which amplifies high-confidence predictions common across models, and applying soft voting, or averaging, which smooths out predictions by giving equal weight to each model's output. By ensembling these methods, the approach exploits the complementary strengths of the models, aiming for improved accuracy and robustness in the final classification results.

It should also be highlighted that these experiments were carried out deliberately avoiding the use of pre-training and fine-tuning techniques, seeking solutions that can perform well on the ESC task independently.

In this project, the performance of the proposed models is evaluated using the ESC-50 dataset, developed by Piczak [1], which is extensively used in environmental sound classification research. Given that the dataset is composed of only 2000 audio samples, various augmentation techniques, such as white noise injection, pitch shifting, and the mixup

[†]Department of Mathematics, University of Padua, email: laura.legrottaglie@studenti.unipd.it

approach [3], are applied to improve model generalization. To further analyze the impact of these techniques, the effects of noise addition and pitch shifting on prediction accuracy for each class will be examined, aiming to identify which sound types are most influenced by these transformations.

## II. Related Work

In recent years, following the success of deep learning in fields like computer vision and speech recognition, deep neural network models such as CNNs and RNNs have increasingly been applied to the ESC task. Unlike classical machine learning models, which struggle with the unstructured nature of environmental sounds, deep learning models have demonstrated superior performance by automatically learning complex patterns that traditional methods could not effectively capture [4]. Piczak pioneers the use of CNNs for ESC, achieving an accuracy of 64.5% on the ESC-50 dataset [5]. Rather than relying on manually extracted features, Tokozume et al. [6] introduce the first end-to-end stacked CNN model specifically designed for sound event recognition from raw waveforms, achieving comparable results. The concept of using very deep convolutional networks for raw waveforms is further supported by Dai et al. in [2], which demonstrates that such models outperform shallow ones, as they efficiently capture essential features in long input sequences.

Additionally, CRNN models have been proven to be effective for environmental sound classification. Sang et al. [7] demonstrate that extracting features using a CNN and performing temporal aggregation with an RNN significantly improves classification performance in ESC tasks.

Various researchers have also worked with the combination of neural networks considering different input features: the study by Li et al. [8] proposes a dual-path CNN architecture for ESC, exploiting the complementarity of the model based on log-mel feature input (MelNet) and the model based on learning features directly from raw waveforms (RawNet), combining the predictions using the Dempster-Shafer (DS) evidence theory.

Significant research has also focused on mitigating overfitting through data augmentation techniques. Zhang et al. in [3] present a deep convolutional neural network for the ESC task with a focus on mixup, an augmentation technique that constructs virtual training samples by linearly mixing pairs of training examples and their labels. This method enhances the diversity of the dataset and has shown to improve classification performance by regularizing the model and increasing generalization.

## III. Processing Pipeline

The dataset used for the environmental sound classification task, as already mentioned, is the ESC-50 dataset [1]. ESC-50 comprises 2000 labeled audio samples, sourced from publicly available recordings in the FreeSound project [9]. The audios are single-channel and exactly 5 seconds long, with shorter events padded to reach the target length. The clips are evenly distributed across 50 distinct classes, with each class containing 40 clips. The dataset provides a rich variety of environmental sounds: these include familiar sounds like snoring, clapping, and dog barking, as well as more unique ones such as crying baby or teeth brushing. Some classes capture subtle distinctions, like those between helicopter and airplane sounds. Additionally, the dataset is organized into five equal cross-validation folds, with each fold ensuring that clips from the same original audio source are grouped together, allowing more consistent training and evaluation.

The average human classification accuracy for the ESC-50 dataset is approximately 81.3%, highlighting the inherent challenge of distinguishing between diverse environmental sounds [1].

The followed preprocessing pipeline is depicted in Fig. 1 and begins by converting each audio file from 16-bit PCM format to floating-point. Each processed audio file is also linked to metadata from a summary file, including its category, fold, and filename. Following this step, augmentation techniques are applied. To reduce algorithm runtime, augmented datasets are created using simple methods like white noise injection

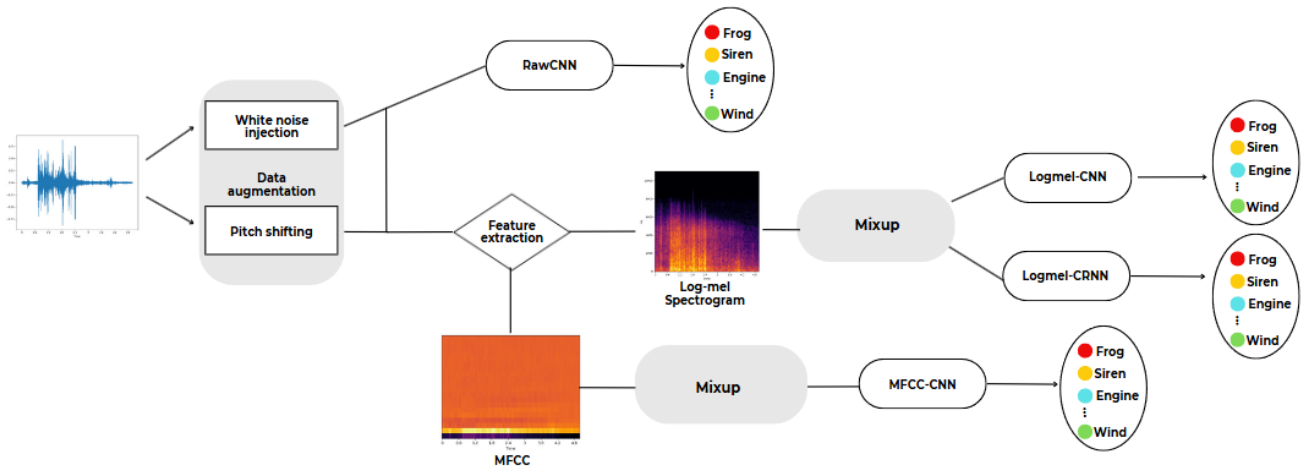


Fig. 1: Processing pipeline overview

and time shifting, despite an increase in memory usage. The augmented data can be already taken as input from the CNN network which process raw waveforms, while for the other three networks feature extraction is needed to produce the log-mel spectrograms and the MFCCs of the audio files. Subsequently, mixup data augmentation can be used to enhance the generalization of these three networks. It should be highlighted, in fact, that since log-mel spectrograms and MFCCs are structured, time-frequency representations, the combination can produce a plausible hybrid sound profile, which helps the model learn more generalized patterns. However, with raw waveforms, mixup might be detrimental. Raw waveforms represent the actual pressure variations over time, and linearly combining sounds can result in unnatural artifacts, such as phase cancellations or reinforcement in certain parts of the signal. Thus, this mixing does not preserve meaningful auditory characteristics, often resulting in distorted and unrealistic audio, which can mislead the model and negatively affect learning.

## IV. Signals and Features

### A. Features extraction

All audio files, originally in .wav format, are processed using the *pydub* library and resampled at a frequency of 22,050 Hz. Before feature extraction, each audio signal is normalized to ensure all sample values fall within the range [-1, 1]. As discussed earlier, beetween all the possible features that can be extracted for the ESC task, this project will be focused on the extraction of image-based features (log-mel spectrogram) and of cepstral features (MFCCs), in addition to the employment of raw audio waveforms (Fig. 2). Log-scaled mel spectrograms are generated with a window size of 1024, hop length of 512, and 60 Mel frequency bands using *librosa*. These spectrograms are represented as matrices of dimension (60, 216). To enhance temporal context, delta features, representing the rate of change in the spectrogram over time, are computed from the normalized spectrogram values using default *librosa* settings. The log-mel spectrogram and its delta are then combined to form a 2-channel input with a shape of (60, 216, 2), which is used as input to the first proposed CNN and CRNN models.

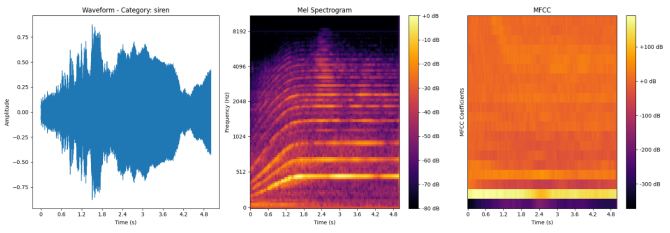


Fig. 2: Raw waveform, log-mel spectrogram and MFCCs of a clip belonging to the class *siren*

For the network that uses raw audio waveforms as input, each audio sample is normalized to ensure values fall within [0,1] range, resulting in an input shape of (110250, 1).
For MFCC extraction, the first 20 coefficients are used along with an additional energy coefficient, which represents the audio signal's amplitude and provides insights into its intensity over time. The resulting MFCC matrix is then combined with first and second-order derivatives ($\Delta$ and $\Delta$-$\Delta$ features), capturing both static spectral information and temporal dynamics (rate of change and acceleration) of the MFCCs. This process yields an input with dimensions (21, 216, 3).

### B. Data augmentation

To enhance the model's generalization ability and mitigate overfitting, various data augmentation techniques will be applied to expand the training set. The effectiveness of these augmentation methods will also be evaluated. Specifically, the data augmentation techniques being tested include:

1) Adding white noise. A small amount of random noise, generated from a standard normal distribution and scaled down by a factor of 0.05 to avoid overwhelming the signal, is added to the original audio.
2) Pitch shifting. The pitch of the audio samples is adjusted up or down while preserving their duration. Each sample is modified with pitch shifts of four semitone values: -2, -1, 1, 2.
3) Mixup data augmentation. Based on the work of Zhang et al. [3], in mixup augmentation method a new training pair of features and labels *(x,y)* is created by combining two randomly selected feature-label pairs from the original training set. Specifically, two samples $x_i$ and $x_j$ are chosen, with their corresponding one-hot encoded labels $\boldsymbol{y_i}$ and $\boldsymbol{y_j}$. The degree of mixing between these two samples is controlled by a factor, $\lambda$, which is determined by a hyperparameter $\alpha$ and drawn from a Beta distribution Beta($\alpha$,$\alpha$). When two samples are blended, the mixing factor $\lambda$ is applied to create a new combined sample according to the following equations:

$$\begin{cases} x = \lambda x_i + (1-\lambda) x_j \\ \boldsymbol{y} = \lambda \boldsymbol{y_i} + (1-\lambda) \boldsymbol{y_j} \end{cases} . \tag{1}$$

Using this approach a new "in-between" sample is created. This sample has a mix of both the original samples' features, and its label is also a combination of the two labels. For example, let's say we have two samples, one representing the log-mel spectrogram of a snoring sound and the other representing the log-mel spectrogram of a water drop audio, with their one-hot encoded labels given as $\mathbf{y_1} = [1,0,0,0,0,0,0,0]$ and $\mathbf{y_2} = [0,1,0,0,0,0,0,0]$, respectively. If we blend these two samples using $\lambda = 0.8$, the resulting label for this mixed sample would be $\mathbf{y} = [0.8, 0.2, 0, 0, 0, 0, 0, 0]$. This label reflects a mix where 80% of the target corresponds to the snoring class and 20% to the water drop class. By training on these mixed samples,

the model learns to recognize not only the exact training examples but also interpolated patterns that lie between them, making the model more robust to small oscillations in the inputs.

In this project, mixup is applied exclusively during the training phase to enhance the generalization of networks that use log-mel spectrograms and MFCCs as input. The augmented samples are generated dynamically on-the-fly using a custom *MixUpGenerator*, eliminating the need to store precomputed augmented data.

To prevent data leakage and have a fair estimation of the metrics of the models, a 5-fold cross validation is performed using the original fold division. In each iteration, three folds are used for training, one fold serves as the validation set for parameter tuning, and the remaining fold is used as the test set. Starting from an original dataset of 2000 samples, using these augmentation techniques the dataset will be expanded up to 10,000 samples, resulting in 9200 samples used for training, 400 for validation and 400 for testing.

## V. Learning Framework

### A. CNN applied to Log-Mel Spectrograms

The first network proposed takes as input the log-mel spectrograms of the audios. It is a shallow CNN composed of 4 convolutional layers, with 64,128, 256 and 256 filters respectively (Fig. 3). This design allows the network to capture simple, low-level features in the initial layers, like texture or rhythm, and more complex, higher-level features in the deeper layers. All the layers have kernel size equal to (3,3) with ReLu as activation function and are followed by a batch normalization and a max pooling layer with kernel size equal to (2,2). The last convolutional layer is followed by a Global Max Pooling layer which allows the model to aggregate features from different frequency bands and time frames, focusing on the overall patterns of the sound, rather than the exact timing of individual events, effectively capturing the most dominant features. The information is then passed through a dense layer, which consists of 256 neurons with ReLU activation function, and a final output layer composed of 50 neurons as the number of classes to predict with softmax as activation function. To reduce the risk of overfitting, each convolutional layer is followed by a dropout layer with a dropout rate of 0.2, while a higher dropout rate of 0.5 is applied after the dense layer.
The network is trained over 50 epochs with a batch size of 50 samples, utilizing the ADAM optimizer set to a learning rate of 0.001.

### B. CNN m11 applied to raw waveforms

The second network implemented is the CNN *m11* from [2]. This model is a fully convolutional, end-to-end stacked CNN architecture specifically designed for sound event recognition from raw audio waveforms, effectively removing the need for manual feature extraction. By relying exclusively on 1D convolutions along the entire network path, it operates directly on the one-dimensional time-series nature of audio signals, bypassing fully connected layers to maintain an efficient structure. The network (Fig. 4) is composed of 11 layers: the first convolutional block applies a Conv1D layer with 64 filters, a kernel size of 80, and a stride of 4. This is followed by batch normalization for stabilization and a ReLU activation to introduce non-linearity. A MaxPooling layer with a pool size of 4 reduces the feature map dimensions, preserving essential features while minimizing computational load. This pattern of Conv1D layers with batch normalization, ReLU activations, and MaxPooling layers is repeated with an increasing number of filters (64, 128, 256, and 512) in successive layers, deepening the network and enhancing its ability to capture complex patterns in the data. While in the first layer a large receptive field is used to mimic the effect of bandpass filters, capturing a wide range of frequencies initially, in all the subsequent layers, the kernel size is set equal to 3 to focus on more local temporal patterns. After these blocks, a Global Average Pooling layer is used to aggregate features across the time dimension. Finally, the model concludes with the output layer consisting of 50

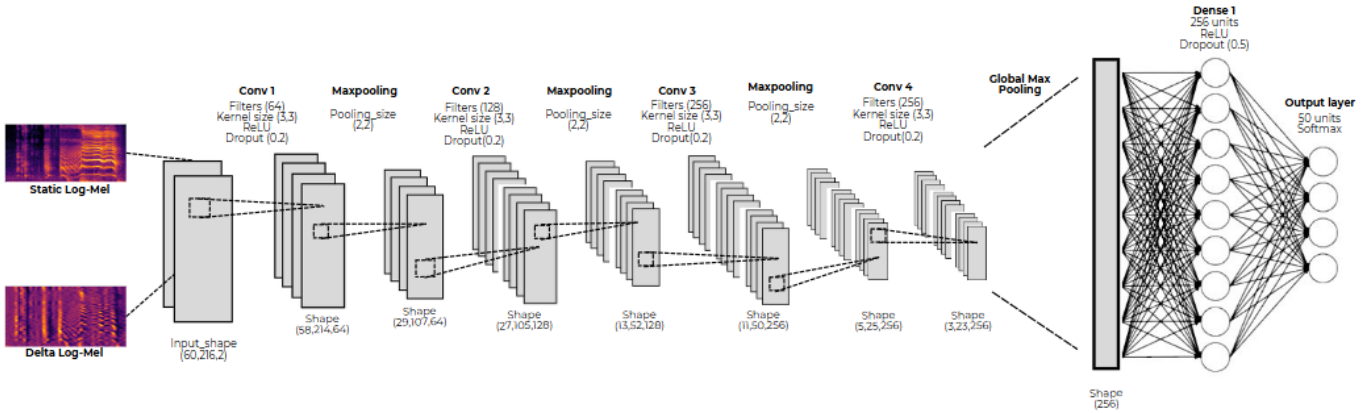


Fig. 3: CNN architecture applied to Log-Mel Spectrograms

units and softmax activation function to output classification probabilities for each sound class.

The network is trained over 50 epochs with a batch size of 32 samples, utilizing the ADAM optimizer set to an initial learning rate of 0.001. As suggested in [2], glorot initialization is used to avoid exploding or vanishing gradients. In addition, in this model a learning rate reduction strategy is employed through the *ReduceLROnPlateau* callback, monitoring the validation accuracy during training. If the validation accuracy does not improve over a defined patience period of 10 epochs, the learning rate is reduced by a factor of 0.5, down to a minimum threshold of 0.0001.

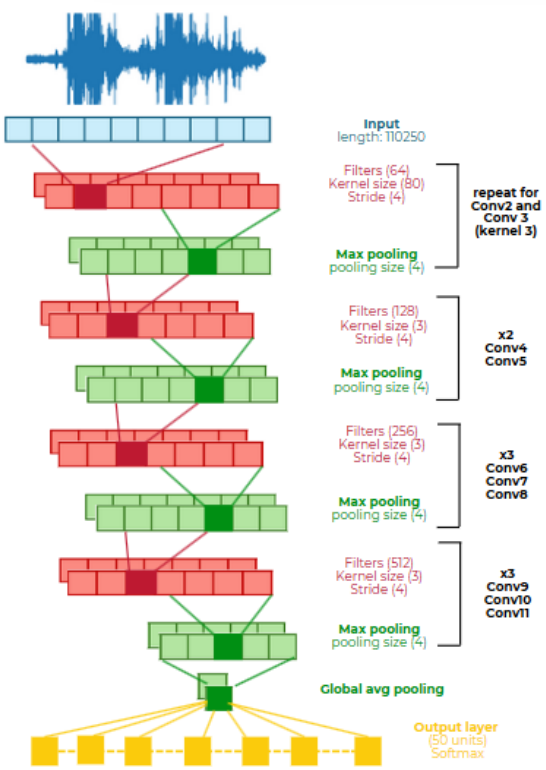


Fig. 4: m11 architecture applied to raw waveforms

### C. CNN applied to MFCCs

The third proposed network is a CNN designed to process MFCCs as input. Its architecture is similar to the network used for analyzing log-mel spectrograms, with the following modifications:

- The four convolutional layers use 32, 64, 128, and 256 filters, respectively.
- Each of the last three convolutional layers is followed by a regularization step with a dropout rate of 0.3.
- To further reduce overfitting, L2 regularization with a coefficient of 0.001 is applied to the weights of each layer.

The network is trained over 150 epochs with a batch size of 50 samples, utilizing the ADAM optimizer with a learning rate of 0.001.

### D. CRNN applied to Log-Mel Spectrograms

The last network proposed is a Convolutional Recurrent Neural Network (CRNN). This approach combines the strengths of CNNs and RNNs since it can identify both spatial and temporal patterns in the audios. The CNN component serves as the initial feature extractor. By applying 2D convolutions, it processes log-mel spectrogram inputs to identify local spatial patterns. Following feature extraction, instead of processing sequential time steps, bidirectional RNN layers are employed to capture frequencies dependencies, processing the sequence of frequencies in both forward and backward directions simultaneously. This allows each frequency bin to benefit from contextual information across the full range of frequencies. The network is structured as follows:

- The convolutional layers resembles the structure of log-mel CNN.
- The last global max pooling layer reduces the dimensionality of the convolutional output and specifically it performs max pooling over the time dimension (axis=2) of the input tensor. This allows the model to aggregate temporal information, resulting in a representation that highlights relevant frequency features. This choice has been done in order to preserve critical frequency components that are essential for distinguishing between different sounds.
- After this step, two bidirectional Gated Recurrent Unis (GRU) with 256 units each are used to learn relationships across frequencies. To mitigate overfitting, dropout regularization with a rate of 0.3 is introduced after each GRU layer.
- Finally, the model outputs class predictions through a dense layer with 50 units and a softmax activation function.

The network is trained over 150 epochs with a batch size of 50 samples, utilizing the ADAM optimizer set to an initial learning rate of 0.001. Also in this case, a learning rate reduction strategy is employed using the *ReduceLROnPlateau* callback reducing the rate by a factor of 0.2 after 10 epochs of no improvement in validation accuracy, with a minimum rate threshold of 0.0001.

## VI. Results

The results presented are obtained by implementing the described architectures in TensorFlow. All experiments are conducted in the Kaggle environment, utilizing a P100 GPU with 32 GB memory. As already mentioned, a 5-fold cross-validation approach is employed to ensure comprehensive performance evaluation. The performances of LogMelCNN, MFCCNet and RawCNN are first examined, comparing outcomes obtained with the standard dataset to those achieved through different augmentation techniques. Subsequently, the impact of these augmentation methods

on the accuracy of each class is analyzed, with a focus on identifying the strengths and weaknesses of the applied approaches.
Then, a comparative analysis of the results of the models is presented. They are evaluated based on the following metrics: accuracy, precision, recall, F1 score, and training time. As previously stated, the dataset is balanced; therefore, these metrics provide an equitable assessment across all classes without the need for adjustments for class imbalance.
The number of parameters and the memory usage for each architecture are reported in Tab. 1. As shown in the table, the CNN processing the MFCCs has the fewest parameters, while the RCNN model has the most. This is due to the bidirectional recurrent layers with 256 units, which significantly increase the parameter count.

| Model | Parameters | Memory (Mb) |
|---|---|---|
| LogMelCNN | 1,041,778 | 3.97 |
| RawCNN | 1,811,442 | 6.91 |
| MFCCNet | 468,978 | 1.79 |
| LogMelRCNN | 2,961,010 | 11.30 |

TABLE 1: Parameters and memory from each architecture.

*A. Data augmentation on the RawCNN*

The following results present the accuracies and training times obtained by the RawCNN network under various dataset configurations: the standard dataset with no augmentation (2000 samples), a dataset with 10,000 samples augmented by adding white noise, and a dataset with 10,000 samples augmented through pitch shifting.

| Augmentation | Accuracy | Training time (s) |
|---|---|---|
| No aug | 60.9 $\pm$ 0.03 | 209 |
| White noise | 60.2 $\pm$ 0.03 | 940 |
| Pitch shifting | 60.6 $\pm$ 0.02 | 931 |

TABLE 2: Results of Raw-CNN for different types of augmentation techniques

It's possible to notice that augmentation techniques affect slightly negatively the performances of the model. This might be imputed to the fact that white noise and pitch shift directly affect the waveforms and can diminish the distinctions between sounds, potentially confusing the model since it hasn't yet decomposed the signal into higher-level features.

*B.Data augmentation on LogMelCNN and MFCCNet*

This analysis explores how data augmentation techniques influence models with Log-Mel spectrogram (Logmel-CNN) and MFCC (MFCC-CNN) input representations. The tables (Tab. 3) and (Tab. 4) contain the accuracies and training times of the two models in case of no augmentation (original dataset of 2000 samples), white noise injection (10,000 samples) and pitch shifting (10,000 samples). Additionally, the performance metrics for mixup learning are reported, where 9200 training samples are observed at each epoch by combining 1200 instances from alternative three folds of the original dataset. Finally, results are provided for a configuration using all augmentation techniques combined, yielding 9200 training samples per epoch by mixing instances from a pitch-noise augmented dataset.

| Augmentation | Accuracy | Training time (s) |
|---|---|---|
| No aug | 63.8 $\pm$ 0.02 | 47 |
| White noise | 67.4 $\pm$ 0.04 | 173 |
| Pitch shifting | 68.9 $\pm$ 0.02 | 175 |
| Mixup | 71.4 $\pm$ 0.02 | 244 |
| All | 69.3 $\pm$ 0.03 | 292 |

TABLE 3: Results of Logmel-CNN for different types of augmentation techniques

| Augmentation | Accuracy | Training time (s) |
|---|---|---|
| No aug | 58.0 $\pm$ 0.03 | 50 |
| White noise | 60.8 $\pm$ 0.05 | 204 |
| Pitch shifting | 62.9 $\pm$ 0.03 | 173 |
| Mixup | 65.7 $\pm$ 0.04 | 249 |
| All | 63.6 $\pm$ 0.04 | 246 |

TABLE 4: Results of MFCC-CNN for different types of augmentation techniques

From the results, it's possible to highlight that the model benefits from additional data, though at the cost of longer training. Pitch shifting has the highest single-augmentation accuracy, implying that it adds valuable variation to the data, more effectively than white noise. Mixup alone reaches the highest score showing that it is very effective in enhancing model generalization. It's also worth to notice that combining all augmentation techniques results in slightly lower performance than using mixup alone. This might happen because when mixup combines two original samples, the resulting synthetic samples closely resemble natural variations within the data distribution, allowing the model to learn meaningful patterns from realistic transitions. However, when mixup is applied to already augmented data, such as samples with added noise or pitch shifts, the synthetic samples may include unnatural artifacts. These hybrid samples can introduce distortions that deviate from the original data distribution, potentially confusing the model. In this case, the benefit of added data diversity is offset by the drawback of generating less realistic samples that the model may struggle to interpret effectively.

*C. Class wise accuracy comparison*

This section presents an analysis of how white noise injection and pitch shifting affect accuracy based on the specific

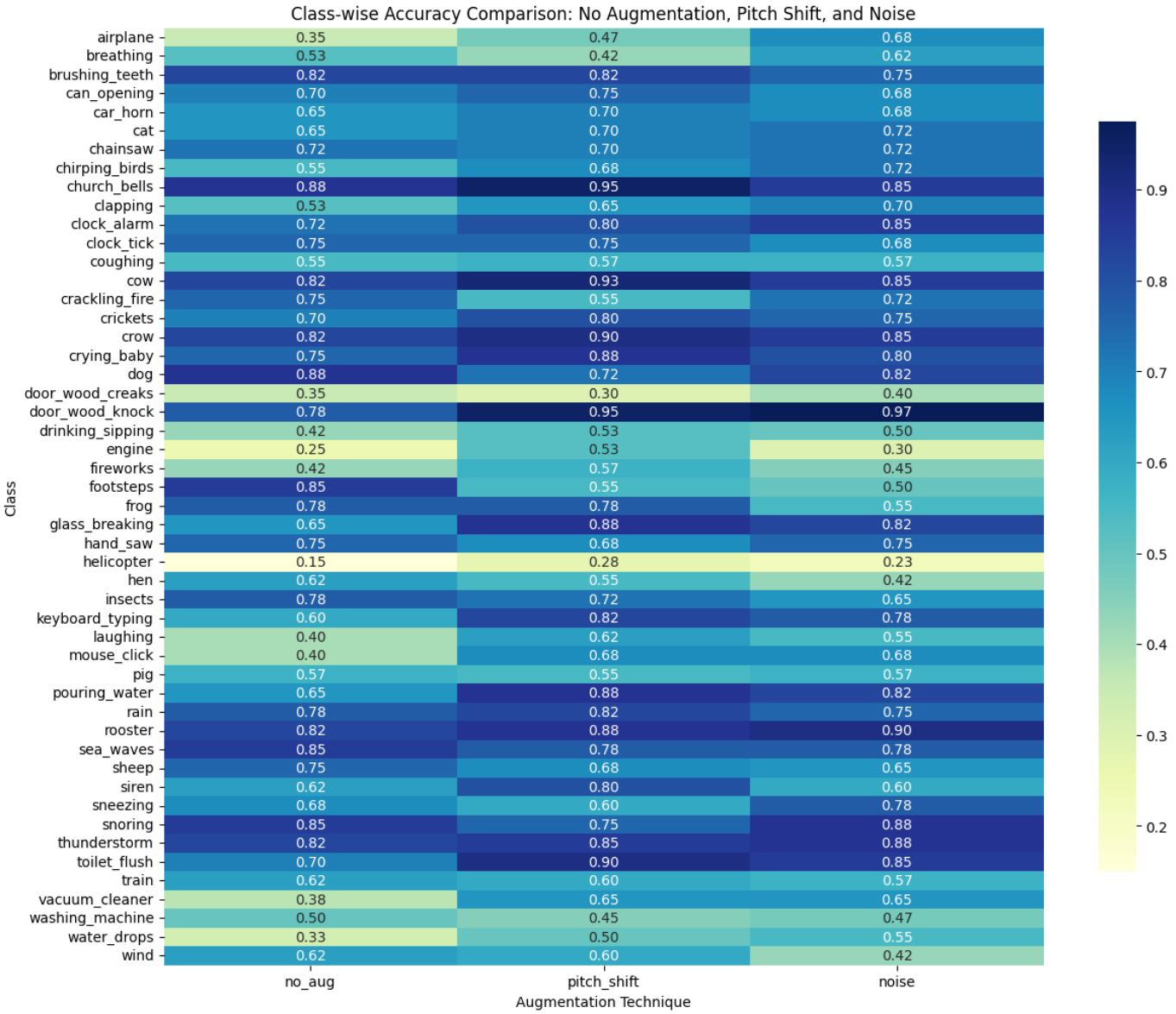


Fig. 5: Classwise comparison of the different augmentation techniques used - predictions obtained from LogMelNet

characteristics of audio's profile. From the heatmap in (Fig. 5), some classes, like *door_wood_knock* exhibit high accuracy consistently across all augmentation techniques, while other like *helicopter* show relatively low accuracy, indicating that this class may be challenging to classify regardless of augmentation.

Pitch-based is particularly beneficial for sounds that have a dominant pitch or frequency that is stable and easily identifiable like a single musical note, such as *church_bells*, *siren*, and *engine* noises. In the real world, these sounds naturally vary in pitch depending on their source or environmental context. For example, an engine's pitch changes with its speed, and siren frequencies can vary slightly between different emergency vehicles. Pitch-based augmentation simulates these natural variations during training, helping the model become more robust to real-world diversity in sound characteristics.

In contrast, noise augmentation is helpful for sounds that naturally occur in environments with various background noises, like *airplane*, *clapping*, *sneezing* or *breathing*. Introducing noise during training enables the model to focus on the core features of these sounds, effectively filtering out irrelevant background elements.

### *D. Comparison of the proposed models*

A comprehensive comparison of the proposed models is presented to assess their performance and highlight the strengths and limitations of each architecture. The evaluation considers multiple metrics to provide an extended perspective

| Model | Accuracy | Precision | F1-score | Training time (s) |
|---|---|---|---|---|
| LogMelCNN | 71.40 ± 0.02 | 75.05 ± 0.03 | 70.50 ± 0.03 | 244 |
| RawCNN | 60.95 ± 0.03 | 62.84 ± 0.04 | 59.79 ± 0.03 | 209 |
| MFCCNet | 65.70 ± 0.04 | 71.61 ± 0.03 | 64.88 ± 0.04 | 249 |
| LogMelCRNN | 74.80 ± 0.03 | 76.99 ± 0.03 | 73.96 ± 0.03 | 2102 |
| Ensemble (Average) | 78.65 ± 0.02 | 80.67 ± 0.02 | 77.67 ± 0.03 | 2804 |
| Ensemble (Product) | 79.10 ± 0.03 | 81.13 ± 0.03 | 78.25 ± 0.03 | 2804 |

TABLE 5: Results of the model tested

on each model's effectiveness as it's possible to see in Tab. 5. It's worth to mention that the models considered are the ones that have achieved the best results between all the augmentation techniques considered: simple mixup for the LogMelNet, MFCCNet and LogMelCRNN and no augmentation for the RawCNN.

The LogMelCRNN performs best among the individual models, thanks to its ability to capture complex frequencies patterns through the RNN layers achieving the highest accuracy of 74.80%, followed by the LogMelCNN model and the MFCCNet. The model which performs worst is RawCNN, reaching only an accuracy of 60.95%. It's also interesting to look at the performance difference between LogMelCNN and MFCCNet. It can be attributed to the distinct information captured by their respective input features: log-mel spectrograms and MFCCs. Log-mel spectrograms retain detailed frequency information over time, which appears to be particularly advantageous for environmental sound classification. This richer time-frequency representation allows the network to capture subtle variations essential for distinguishing between diverse environmental sounds. In contrast, MFCCs are designed to compress spectral information which can lead to the loss of finer details, critical for distinguish complex environmental sounds.

Regarding ensemble methods, they provide even better performances, with the product-based ensemble achieving the highest scores, indicating that combining models with different input features and architectures enhances the system's ability to correctly classify sounds. The trade-off between training time and performance is also evident, with the CRNN and ensemble models requiring longer training times but delivering higher accuracy, precision, and F1-scores.

## VII. Concluding Remarks

In conclusion, this project explored multiple deep learning architectures for environmental sound classification, a task with unique challenges due to the complex nature of environmental sounds. By experimenting with distinct types of feature representations (log-mel spectrograms, MFCCs, and raw waveforms), as well as different model architectures (CNNs and CRNNs), the study demonstrated how specific input features and model structures affect classification performance. The results showed that the CRNN model with log-mel spectrograms as input achieved the highest accuracy among individual models due to its ability to capture intricate patterns across the frequencies using RNN layers. In comparison, models that used compressed representations, such as MFCCs, or raw waveforms performed less effectively, highlighting the importance of preserving detailed frequency information in this classification task.

Notably, the performances obtained by the ensemble models were comparable to the average human classification accuracy of 81.3% for the ESC-50 dataset.

The project also emphasized the importance of data augmentation, especially for small datasets like ESC-50. Techniques such as mixup and pitch shifting were effective in enhancing the model's ability on unseen data, although mixing augmented samples may introduce unrealistic hybrid samples. In addition, the analysis of the influence of each augmentation method on class-wise accuracy suggests that model performance could be further improved by using class-specific data augmentation strategies.

These considerations offer valuable insights in the field of sound recognition systems. While state-of-the-art models in environmental sound classification have achieved impressive accuracy, particularly with the use of attention mechanisms and pretrained models, challenges remain. They include developing models that can maintain high performance in real-world, noisy environments and achieving an optimal balance between model complexity and computational efficiency for deployment on edge devices.

### *A. What I have learned*

This project, in addition to the labs, has given me the possibility to translate theory into practice. I have gained valuable experience in modeling complex architectures using TensorFlow and addressing the challenges posed by audio data and their preprocessing techniques. Additionally, this was my first time writing a formal report in LaTeX, and I enjoyed learning in detail how to write a scientific paper while also mastering the use of this powerful tool.

### *B. Difficulties encountered*

I have encountered several difficulties while developing this project, but a trial and error approach has always a solution as its goal. Initially, I attempted to implement the models referenced in the papers provided in the project slides. However, probably due to the complexity of the preprocessing techniques and the lack of details regarding

the augmentation strategies used, the results I obtained were not comparable to those reported. As a result, I decided to shift my focus to more "original" network architectures, which yielded better results and required less training time. Another significant challenge was implementing the 5-fold cross-validation approach. It was crucial to avoid data leakage and maintain the integrity of the original fold divisions to ensure a fair evaluation of the models. However, additional computational resources were needed to achieve this level of rigor and also to implement data augmentation techniques. Therefore, I started implementing the code on Colab and then switched to the Kaggle environment, which offers access to greater computational resources.

## REFERENCES


[1] K. J. Piczak, "ESC: Dataset for environmental sound classification," *Proceedings of the ACM International Conference on Multimedia*, 2015.

[2] W. Dai, C. Dai, S. Qu, J. Li, and S. Das, "Very deep convolutional neural networks for raw waveforms," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 421–425, 2017.

[3] Z. Zhang, S. Xu, S. Cao, and S. Zhang, "Deep convolutional neural network with mixup for environmental sound classification," in *Pattern Recognition and Computer Vision: First Chinese Conference, PRCV 2018, Guangzhou, China, November 23-26, 2018, Proceedings, Part II*, 2018.

[4] A. Bansal and N. K. Garg, "Environmental sound classification: A descriptive review of the literature," *Intelligent Systems with Applications*, vol. 16, p. 200115, 2022.

[5] K. J. Piczak, "Environmental sound classification with convolutional neural networks," *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2015.

[6] Y. Tokozume and T. Harada, "Learning environmental sounds with end-to-end convolutional neural network," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2721–2725, 2017.

[7] J. Sang, S. Park, and J. Lee, "Convolutional recurrent neural networks for urban sound classification using raw waveforms," in *2018 26th European Signal Processing Conference (EUSIPCO)*, pp. 2444–2448, 2018.

[8] S. Li, Y. Yao, J. Hu, G. Liu, X. Yao, and J. Hu, "An ensemble stacked convolutional neural network model for environmental event sound recognition," *Applied Sciences*, vol. 8, no. 7, 2018.

[9] G. R. F. Font and X. Serra., "Freesound technical demo," *Proceedings of the ACM International Conference on Multimedia*, pp. 411–412, 2013.