STMAE project - Group 3 - Project 1: Environmental Sound Classification Using ESC-50

Galadini Giuliano, Lenoci Alice, Macrì Carlo, Messina Francisco

June 10, 2025

Abstract :

This paper presents an investigation into environmental sound classification using the ESC-50 dataset. We evaluate a convolutional neural network baseline on different audio representations (MFCCs and mel-spectrograms) and compare it to two hybrid architectures that combine convolutional feature extraction with a transformer encoder. One hybrid model is trained from scratch, while the other leverages a pretrained CNN backbone to accelerate convergence. Although spectrogram-level data augmentation did not yield additional improvements under our computational constraints, our results demonstrate that transformer-based models with pretrained backbones offer notable gains in both classification performance and training efficiency. These findings highlight the potential of combining transfer learning and attention mechanisms for robust environmental sound recognition and suggest that future work should explore richer augmentation strategies directly at the waveform level.

1 Introduction

Classifying environmental audio is a challenging yet increasingly relevant task within the broader field of machine learning for audio analysis. The aim of this project is to develop and evaluate a system that can automatically categorize short, isolated audio recordings into a set of predefined classes. These classes represent a wide variety of everyday sounds, such as animal noises, human activities, natural phenomena, and urban environments.

This problem sits at the intersection of signal processing and machine learning and has practical applications in a range of domains, including smart surveillance, context-aware systems, sound event detection, and interactive technologies. Automatically recognizing sound categories from short clips can enable more responsive and intelligent systems in real-world settings.

To address this task, the project adopts an exploratory approach that investigates multiple stages of the audio classification pipeline. Particular emphasis is placed on audio preprocessing and representation, where two time-frequency transformations—Mel spectrograms and MFCCs—are tested to extract informative features from raw waveform data. These representations are critical, as they determine how well the model can capture the relevant spectral and temporal characteristics of different sound events.

Another core aspect of the approach is data augmentation. Since real-world audio can vary widely due to background noise, microphone quality, or environmental conditions, it is important to simulate such variability during training. Augmentation techniques such as time masking, frequency masking, and background noise addition are applied to hopefully improve the model's robustness and generalization, particularly in low-data regimes. Ultimately, the goal is to implement a reliable and reproducible classification pipeline that takes a short audio clip as input and outputs the predicted sound category with high accuracy. Along the way, the influence of design choices—including feature extraction methods, model architectures, and augmentation strategies—is critically assessed to gain deeper insight into what contributes most to performance. You can directly check our work in our repository[1].

2 Background

The task of environmental sound classification (ESC) has been explored through various approaches, ranging from classical signal processing and machine learning pipelines to modern deep learning architectures. Early systems relied heavily on handcrafted features such as MFCCs (Mel-Frequency Cepstral Coefficients) and spectral representations derived from the Short-Time Fourier Transform (STFT), which were typically passed to shallow classifiers such as Support Vector Machines (SVMs) or k-Nearest Neighbors (k-NN).

However, these approaches often struggle to generalize to more complex or noisy audio environments. As a result, deep learning models have become the dominant paradigm, allowing feature extraction and classification to be learned jointly from data. Two notable architectures are the Convolutional Recurrent Neural Network (CRNN) and the Hierarchical Token-Semantic Audio Transformer (HTS-AT), that replaced the CRNN as a modern state of the art.

2.1 Convolutional Recurrent Neural Networks (CRNN)

CRNNs are hybrid models that combine the local pattern recognition abilities of Convolutional Neural Networks

[1] https://github.com/FranciscoMessina00/Selected-topics-exam-Group-3-P1

(CNNs) with the sequence modeling capacity of Recurrent Neural Networks (RNNs). In the context of audio classification, CNN layers are typically used to extract high-level time-frequency features from spectrograms, treating them similarly to images. These feature maps are then passed to RNN layers—often LSTM (Long Short-Term Memory) or GRUs (Gated Recurrent Units)—which model the temporal evolution of audio events across time (Xu et al., 2017).
This architecture is particularly well-suited for audio data, which is both spatial (in the form of frequency content) and temporal (changing over time). By decoupling the spatial and temporal components of feature learning, CRNNs can capture complex acoustic patterns while maintaining a relatively lightweight model. A commonly used implementation of this approach can be found in the PyTorch speech command recognition tutorial, which applies a similar architecture to keyword spotting tasks.

2.2 Hierarchical Token-Semantic Audio Transformer (HTS-AT)

More recently, Transformer-based architectures have demonstrated strong performance on various audio tasks by leveraging self-attention mechanisms to model long-range dependencies in time and frequency. The HTS-AT model extends this paradigm by introducing a hierarchical structure for audio patch tokens, enabling the network to capture both local and global semantic relationships within spectrogram representations (Chen et al., 2022).
HTS-AT first divides the input spectrogram into patches, each of which is treated as a token. These tokens are then processed using a hierarchical attention mechanism that groups and merges them at multiple levels, allowing the model to construct higher-order semantic representations. Unlike CNNs, which are limited by local receptive fields, and RNNs, which are inherently sequential, Transformers can process the entire input in parallel, making them well-suited for identifying distributed acoustic cues across long time spans.
HTS-AT has achieved competitive results on large-scale audio benchmarks such as AudioSet and ESC-50, outperforming many CNN- and RNN-based models. The official implementation is publicly available via the HTS-AT GitHub repository, and it provides a flexible framework for experimenting with token-based attention models in audio classification tasks. These two architectures—CRNN and HTS-AT—were reviewed and considered as foundational references during the design of our own classification pipeline. They reflect two contrasting yet complementary approaches to modeling environmental audio: one rooted in sequential feature extraction and another in global attention and semantic abstraction.

3 Dataset

This project uses the ESC-50 dataset (Piczak, 2015), a curated collection of environmental audio recordings designed for benchmarking sound classification algorithms. It consists of 2000 labeled audio clips, each lasting 5 seconds, and sampled at 44.1 kHz in uncompressed WAV format. The dataset includes 50 distinct sound classes, making it one of the most diverse and balanced resources available for non-speech sound classification.
The sound classes are evenly distributed across five major semantic categories:

- Animals (e.g., *dog bark*, *rooster*, *frog*)
- Natural soundscapes and water sounds (e.g., *thunderstorm*, *sea waves*, *wind*)
- Human non-speech sounds (e.g., *coughing*, *sneezing*, *clapping*)
- Interior/domestic sounds (e.g., *door knock*, *dishwasher*, *clock tick*)
- Urban noises (e.g., *siren*, *car horn*, *engine idling*)

Each class contains exactly 40 examples, which makes the dataset well-balanced, a key factor for fair training and evaluation of classification models. ESC-50 also includes predefined 5-fold cross-validation splits, which help standardize performance comparisons across different models and experiments.
The clips are sourced from the Freesound archive and manually annotated to ensure label accuracy. However, despite the dataset's structure and balance, the task remains challenging due to factors such as background noise, overlapping sound events, and acoustic similarity between some classes.
Overall, ESC-50 is widely regarded as a benchmark dataset in environmental sound classification research and is particularly suitable for evaluating the performance and generalization capabilities of machine learning models.

3.1 Cross-Validation Splits

ESC-50 provides five predefined, stratified folds. We divided the data using a simple 80 %–20 % split for training and testing, without holding out any validation subset, so that we could maximize the number of training examples. To obtain a more reliable performance estimate, we repeated this process five times: in each run, a different subset (fold) served as the test set while the remaining 80 % provided the training data. After training on each run, we evaluated on its corresponding test fold and then averaged the results across all five runs to report the model's overall performance.

4 Methodology

In this section, we detail the steps taken to develop and evaluate our audio classification system. First, we describe how raw audio clips were preprocessed to obtain suitable time–frequency representations. Next, we outline the data augmentation strategies applied during training of the hybrid model. We then present the neural network architectures—both the CNN baseline and the CNN–Transformer hybrid—and explain their respective design choices. Finally, we summarize the training protocols, including optimizer configurations, and learning-rate scheduling, followed by the evaluation metrics used to quantify model performance. Each of these components is discussed in the subsections that follow.

4.1 Preprocessing

Prior to model fitting, all audio waveforms were resampled from 44.1 kHz to 22.05 kHz and normalized so that each clip's maximum absolute amplitude equaled unity. Two complementary time-frequency representations were extracted and saved individually as NumPy (`*.npy`) files. This disk-based storage lets us load only the spectrogram or MFCC required at a given training step, keeping the full feature set out of RAM and markedly reducing memory overhead:

1. **Mel Spectrograms.** Each five-second clip was segmented using a Short-Time Fourier Transform (STFT) with a Hann window of length 1024 samples (approximately 46 ms) and a hop length of 512 samples (approximately 23 ms). A 128-band mel filterbank was then applied to the magnitude spectrum to obtain a mel spectrogram $\mathbf{M} \in \mathbf{R}^{128\times T}$, where $T \approx 217$ frames for a five-second clip. The mel spectrogram was converted to the decibel (dB) scale according to

$$\mathbf{M}_{\mathrm{dB}} = 20\log_{10}(\mathbf{M}+\varepsilon), \quad \varepsilon = 10^{-7}, \tag{1}$$

   and values below −80 dB were clipped. Each resulting $128 \times T$ array was saved to disk before the conversion to decibel scale, which will be done in each training step.
2. **Mel-Frequency Cepstral Coefficients (MFCCs).** From the same normalized waveform, 20 MFCCs per frame were computed using identical window and hop parameters as above. The resulting MFCC matrix $\mathbf{C} \in \mathbf{R}^{20\times T}$ was likewise saved for downstream use.

By precalculating and storing these features, data loading during training incurred minimal I/O overhead.

4.2 Data Augmentation

Data augmentation was applied exclusively when training the CNN–Transformer hybrid model. The smaller CNN baseline was trained on "clean" mel spectrograms without any synthetic distortions. For the hybrid model, each training sample underwent zero, one, or multiple augmentations at random (each with probability 0.3):

- **Additive Gaussian Noise.** Zero-mean Gaussian noise $\mathbf{N} \sim \mathcal{N}(0,\sigma^2)$ with $\sigma \sim \mathcal{U}[0,\,0.1]$ was added to the linear-scale mel spectrogram $\mathbf{M}$. After noise injection, the spectrogram was converted to dB and clipped at −80 dB.
- **Frequency Masking** Park et al., 2019. Up to 5 consecutive mel-frequency bins (out of 128) were randomly selected and zeroed, simulating missing spectral information.
- **Time Masking** Park et al., 2019. Up to 5 consecutive time frames (out of T) were randomly selected and zeroed, simulating short temporal dropouts.

No augmentations were applied during testing.

4.3 Model Architectures

4.3.1 Convolutional Neural Network (CNN) Baseline

A lightweight convolutional neural network, serving as the baseline classifier. Its architecture is summarized in Table 1.

Table 1
CNN Baseline Architecture

Layer	Operation	Parameters	Output Shape	Activation/Norm
1	Conv2d	$1 \rightarrow 32$, kernel 3×3, padding 1	$32 \times 128 \times T$	BatchNorm2d; ReLU
2	MaxPool2d	kernel 2×2	$32 \times 64 \times \frac{T}{2}$	—
3	Conv2d	$32 \rightarrow 64$, kernel 3×3, padding 1	$64 \times 64 \times \frac{T}{2}$	BatchNorm2d; ReLU
4	MaxPool2d	kernel 2×2	$64 \times 32 \times \frac{T}{4}$	—
5	Conv2d	$64 \rightarrow 128$, kernel 3×3, padding 1	$128 \times 32 \times \frac{T}{4}$	BatchNorm2d; ReLU
6	MaxPool2d	kernel 2×2	$128 \times 16 \times \frac{T}{8}$	—
7	AdaptiveAvgPool2d	size = (1,1)	$128 \times 1 \times 1$	—
8	Flatten	—	$B \times 128$	—
9	Linear	$128 \rightarrow 50$	$B \times 50$	— (logits)

The output logits are converted to class probabilities via softmax.

4.3.2 CNN–Transformer Hybrid

To capture both local spectral features and long-range dependencies, the hybrid architecture first extracts convolutional embeddings from the mel spectrogram and then processes them with a Transformer encoder. Tables 2 and 3 detail the CNN backbone and Transformer components, respectively.

Table 2
CNN Backbone for Hybrid Model

Layer	Operation	Parameters	Output Shape	Activation/Regularization
1	Conv2d	$1 \rightarrow 64$, kernel 3×3, stride 2, padding 1	$64 \times 64 \times \frac{T}{2}$	BatchNorm2d; ReLU; Dropout(0.2)
2	Conv2d	$64 \rightarrow 128$, kernel 3×3, stride 2, padding 1	$128 \times 32 \times \frac{T}{4}$	BatchNorm2d; ReLU; Dropout(0.2)
3	Conv2d	$128 \rightarrow 256$, kernel 3×3, stride 1, padding 1	$256 \times 32 \times \frac{T}{4}$	BatchNorm2d; ReLU

Transformer Encoder The Transformer encoder has four identical layers, each containing multi-head self-attention and a position-wise feed-forward sublayer Vaswani et al., 2017. Its configuration is summarized in Table 3.

Layer normalization and residual connections follow Vaswani et al., 2017. The final output $\mathbf{Z}^{(4)} \in \mathbb{R}^{N\times 256}$ is then globally pooled.

Table 3
Transformer Encoder Configuration

Component	d_{model}	# Heads	d_{ff}	Dropout	# Layers
Self-Attention & Feed-Forward	256	8	1 024	0.1	4
Positional Encoding	256	—	—	—	—

4.4 *Training Protocol*

All models were implemented in PyTorch Lightning Falcon, 2019 and trained on NVIDIA GPUs available from the Google Colab base plan with mixed-precision (16-bit). Table 4 summarizes optimizer and scheduler settings.

Table 4
Optimizer and Learning-Rate Configurations

Model	Optimizer	Initial LR	Weight Decay	Scheduler
CNN Baseline	Adam ($\beta_1 = 0.9,\ \beta_2 = 0.999$)	1×10^{-3}	0	None
Hybrid A (Scratch)	AdamW ($\beta_1 = 0.9,\ \beta_2 = 0.999$)	3×10^{-4} (max 1×10^{-3})	1×10^{-5}	OneCycleLR (max 1×10^{-3})
Hybrid B (Pretrained CNN)	AdamW ($\beta_1 = 0.9,\ \beta_2 = 0.999$)	3×10^{-4} (max 1×10^{-3})	1×10^{-5}	OneCycleLR (max 1×10^{-3})

CNN Baseline Training. Training used batch size 64, Adam (LR 1×10^{-3}), no augmentation, and no scheduler. We trained the model for 1000 epochs.

CNN–Transformer Hybrid Training. Two variants were conducted:

- **Hybrid A (From Scratch).** Random initialization of all parameters. AdamW optimizer and OneCycleLR (peak LR 1×10^{-3}). Mixed-precision, trained with 500 epochs.
- **Hybrid B (ResNet).** Since it took many epochs and time to train these models, the paper itself of the **HTS-AT** model, used existing checkpoints of RestNet18 to preprocess the data. AdamW optimizer and OneCycleLR (peak LR 1×10^{-3}). Mixed-precision, trained with 25 epochs, and on-the-fly augmentations (Section 4.2) applied with 0.3 probability.

A batch size of 64 was maintained for both hybrids. All training followed the splits described in Section 3.1.

4.5 *Evaluation Metrics*

To asses the performance of the models, we used the following metrics:

Accuracy: Defined as

$$\text{Accuracy} = \frac{\text{Number of correctly predicted test samples}}{\text{Total number of test samples}}. \tag{2}$$

Macro-averaged F_1 Score: Since ESC-50 is balanced across 50 classes, macro F_1 is

$$\mathrm{F}_{1,\text{macro}} = \frac{1}{50} \sum_{c=1}^{50} \frac{2\,\text{Precision}_c\,\text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}. \tag{3}$$

Confusion Matrix: A 50×50 matrix was computed to visualize inter-class confusions (rows: true labels; columns: predicted labels).

All metrics were computed on the stored checkpoints.

5 Results and Discussion

In this section, we present and analyze the performance of our models using multiple evaluation metrics. We first compare the CNN baseline trained on MFCC and mel-spectrogram inputs. Next, we examine two transformer-based hybrid models trained without any data augmentation (Hybrid A and Hybrid B). We then assess the impact of data augmentation on the hybrid model with the pretrained CNN backbone with ResNet18. Finally, we evaluate all models across the five predefined ESC-50 folds and summarize the key findings.

5.1 *CNN Baseline: MFCC vs. Mel-Spectrogram*

When evaluated on the held-out test fold, the CNN baseline model trained on MFCCs achieved an average test accuracy of approximately 66 % and a macro-averaged F_1 score of around 0.64. By contrast, training on mel-spectrograms yielded higher performance, with test accuracy near 74 % and macro-F_1 around 0.73. A notable difference in performance between the two models emerges when analyzing the classification of a few animal sounds (first classes of dataset) and some exterior and urban noises (last 10 classes of dataset). The model trained on Mel spectrograms demonstrates a significantly higher accuracy for these classes compared to the model that utilized Mel-Frequency Cepstral Coefficients (MFCCs). This is visibly represented in the confusion matrices (Figure 1), where the diagonal entries corresponding to the initial classes—which predominantly represent animal sounds in the ESC-50 dataset—are much stronger for the Mel spectrogram-based model. These results indicate that mel-spectrograms capture more discriminative time–frequency information for environmental sounds. Consequently, all subsequent experiments—including transformer models and data augmentation—were conducted using mel-spectrogram inputs.

5.2 *Transformer Architectures without Augmentation*

5.2.1 Hybrid A (From Scratch) Hybrid A combines a small CNN backbone (three convolutional layers) with a four-layer Transformer encoder (eight attention heads, $d_{\text{model}} = 256$), as described in Section 4.4. Trained from scratch on mel-spectrograms, Hybrid A required 500 epochs to train and achieved approximately 75 % accuracy and a macro-F_1 of 0.74 on average. While a consistent trend of the transformer model outperforming the CNN model in specific classes was not observed across all five folds, the transformer model generally exhibited higher overall accuracy. To illustrate this, consider Fold 2 of the dataset. In this particular fold, the CNN model showed a notable confusion between class 48 (footstep) and class 25 (fireworks), as you can see in Figure 2. The transformer model, however, effectively mitigated this specific confusion in Fold 2, demonstrating its ability to better distinguish between these two sound events in that instance. It is important to note that this particular misclassification pattern (footstep being confused with fireworks) was not consistently present in the other folds, highlighting the variability in performance across different data splits. The extended training duration reflects the need to learn both CNN and Transformer parameters without any pretrained initialization.

5.2.2 Hybrid B (Pretrained CNN Initialization) Hybrid B uses a ResNet-18 backbone, with the training proceeding in two stages (Section 4.4): the Transformer and classification head are trained at a learning rate of 3×10^{-4} and a OneCycleLR schedule with a peak learning rate of 1×10^{-3}. This approach allowed Hybrid B to converge in just 25 total epochs, in stark contrast to the 500 epochs required by Hybrid A. Hybrid B achieved approximately 77 % accuracy and a macro-F_1 of 0.77. These results confirm that initializing from a pretrained CNN backbone significantly accelerates convergence and improves both accuracy and recall/precision balance.

5.3 Data Augmentation on the Pretrained CNN Backbone

To determine whether data augmentation could further enhance performance, we applied additive Gaussian noise, frequency masking, and time masking (each with 30 % probability) to Hybrid B's mel-spectrogram inputs (Section 4.2). Despite these augmentations, test accuracy remained around 76 % and macro-F_1 approximately 0.76, showing no meaningful improvement over the non-augmented Hybrid B. One likely explanation is that, for computational efficiency, all spectrograms were precomputed and stored to disk, so augmentations were applied directly to mel-spectrogram arrays rather than to raw audio waveforms at each training cycle. Augmentations at the waveform level (e.g., time stretch or pitch shift) may introduce greater variability than spectrogram-level operations. Consequently, our augmentation pipeline may not have been sufficiently rich to improve the performance of the model. However, data augmentation did demonstrate a specific benefit in one instance. We observed that the model without data augmentation frequently confused class 48 (fireworks) with class 12 (crackling fire) in fold 5 (Figure 3). The transformer model trained with data augmentation successfully reduced this particular misclassification, indicating that while it didn't boost overall accuracy, it did help the model better differentiate between these two acoustically similar classes.

5.4 Epoch Count Discussion

The disparity in required training epochs between Hybrid A and Hybrid B is substantial. Hybrid A, trained from scratch, needed approximately 500 epochs to reach a good accuracy, reflecting the necessity of learning all parameters without prior knowledge. In contrast, Hybrid B, with a pretrained CNN backbone, allowed to train the model for only 25 epochs—highlighting the benefit of transfer learning. By reusing a well-trained ResNet-18 backbone, Hybrid B not only achieved higher accuracy but also drastically reduced computational cost.

5.5 Summary of Findings

Our experiments lead to the following key conclusions (Table 5). First, mel-spectrogram inputs outperform MFCCs for CNN-based classification, achieving 75 % accuracy and 0.74 macro-F_1 compared to 66 % accuracy and 0.64 macro-F_1 with MFCCs. Second, transformer models benefit substantially from initializing the CNN backbone with pretrained weights: Hybrid B reaches 77 % accuracy and 0.77 macro-F_1 in 25 epochs, whereas Hybrid A, trained from scratch, attains 75 % accuracy and 0.74 macro-F_1 only after 500 epochs. Third, applying augmentations directly to precomputed mel-spectrograms does not improve performance, suggesting that richer waveform-level augmentations may be necessary for further gains. Lastly, multifold evaluation confirms that all reported results are stable across different data partitions and not artifacts of a single train/test split.

Table 5
Comparison of Models on ESC-50 (average over 5 folds)

Model	Accuracy (%)	Macro–F_1
CNN Baseline (MFCC)	66	0.64
CNN Baseline (Mel-Spectrogram)	74	0.73
Hybrid A (From Scratch)	75	0.74
Hybrid B (Pretrained CNN, no data augmentation)	77	0.77
Hybrid B (Pretrained CNN, with data augmentation)	76	0.76

6 Conclusions

In this work, we have demonstrated that a CNN–Transformer hybrid model with a pretrained ResNet-18 backbone (Hybrid B) achieves superior performance on the ESC-50 environmental sound classification benchmark, reaching an average accuracy of 77 % and a macro-F_1 score of 0.77 across all five folds. Compared to a CNN trained on mel-spectrograms (74 % accuracy, 0.74 macro-F_1) and a CNN–Transformer trained from scratch (75 % accuracy, 0.74 macro-F_1), the pretrained backbone yields both higher accuracy and dramatically faster convergence (25 vs. 500 epochs). These results confirm that transfer learning is a highly effective strategy for audio classification when computational resources permit. Although we applied data augmentation directly to precomputed mel-spectrograms without obtaining further gains, future work should explore waveform-level augmentations (e.g., time stretch, pitch shift) to introduce richer variability. Implementing such augmentations would likely require additional computational resources (e.g., GPU memory on Colab or dedicated servers) but could further improve generalization. Overall, our findings support the adoption of transformer-based architectures for environmental sound recognition and highlight the practical benefits of backbone pretraining and targeted data augmentation for continued performance gains.

References

Chen, Y., Q. Kong, Y. Wang, et al. (2022). “HTS-AT: A Hierarchical Token-Semantic Audio Transformer for Sound Classification and Detection”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).*

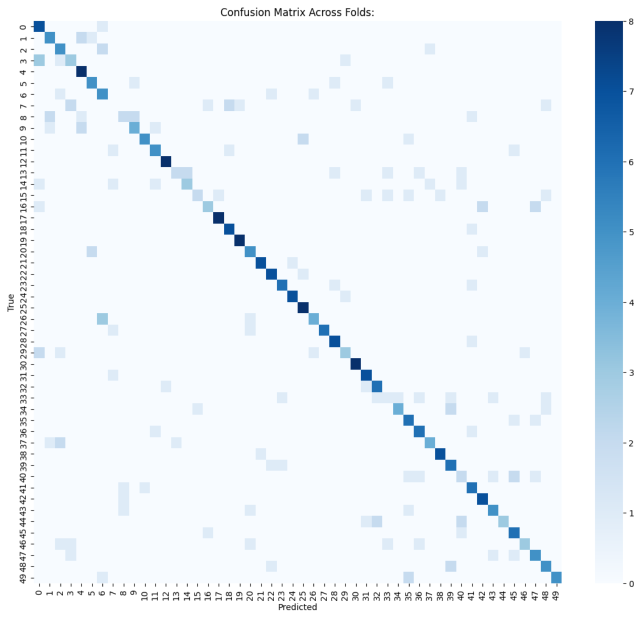
Falcon, William (2019). *PyTorch Lightning: A Lightweight PyTorch Wrapper for High-Performance AI Research.* `https://github.com/PyTorchLightning/pytorch-lightning`. GitHub repository.

Park, Daniel S. et al. (2019). “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition”. In: *Proceedings of Interspeech.* Graz, Austria, pp. 2613–2617.

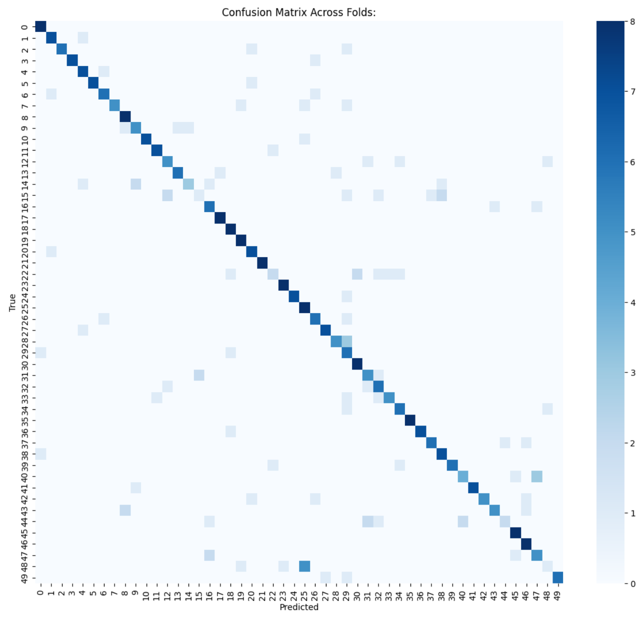
Piczak, Karol J. (Oct. 13, 2015). “ESC: Dataset for Environmental Sound Classification”. In: *Proceedings of the 23rd Annual ACM Conference on Multimedia.* Brisbane, Australia: ACM Press, pp. 1015–1018. ISBN: 978-1-4503-3459-4. DOI: `10.1145/2733373.2806390`. URL: `http://dl.acm.org/citation.cfm?doid=2733373.2806390`.

Vaswani, Ashish et al. (2017). “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems.* Vol. 30. Long Beach, CA, USA, pp. 5998–6008.

Xu, Y. et al. (2017). “Convolutional Gated Recurrent Neural Network incorporating spatial features for audio tagging”. In: *Proc. Int. Joint Conf. on Neural Networks (IJCNN).*

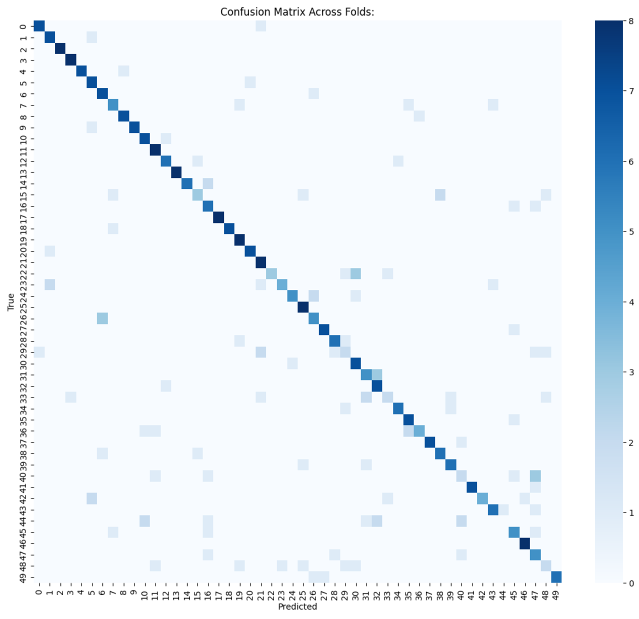


(a) Confusion Matrix (MFCC Model)

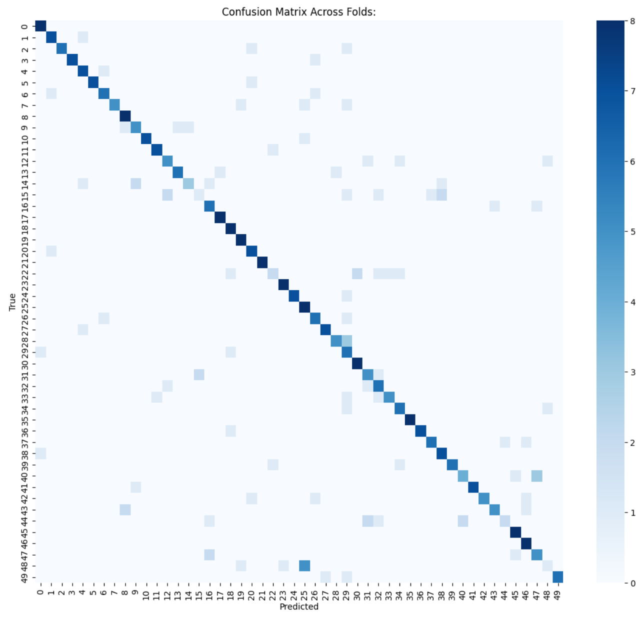


(b) Confusion Matrix (Mel Spectrogram Model)

Figure 1. Comparison of confusion matrices for the two models in fold 2 of the dataset.

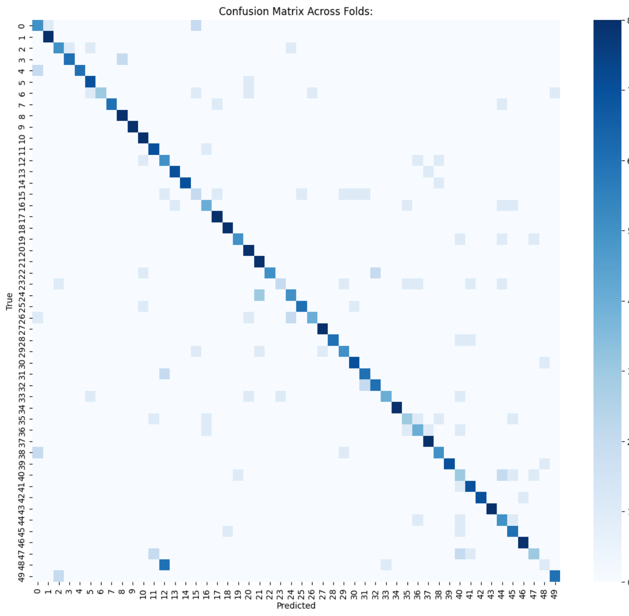


(a) Confusion Matrix (Transformer Hybrid A Model)

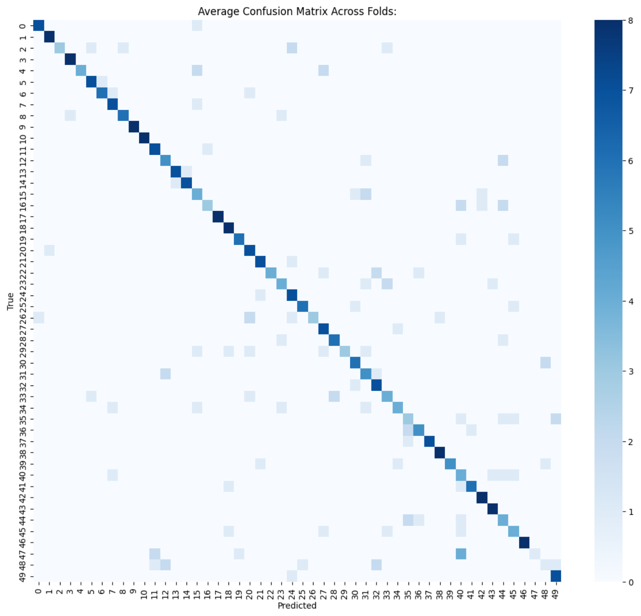


(b) Confusion Matrix (Mel Spectrogram Model)

Figure 2. Comparison of confusion matrices for the two models in fold 2 of the dataset.



(a) Confusion Matrix (Hybrid B Model without data augmentation)



(b) Confusion Matrix (Hybrid B Model with data augmentation)

Figure 3. Comparison of confusion matrices for the two models in fold 5 of the dataset.