Environmental sound classification (Human Data Analytics 2024)

Marco Furlan[†], Ivan Krstev[‡]

***Abstract*—Environmental Sound Classification (ESC) is a critical intersection of audio signal processing, machine learning, and environmental monitoring, gaining significance in the era of automated systems and smart technologies. ESC applications span from hearing aids to security systems. The renowned ESC-50 dataset, introduced by K. Pickzac, has witnessed state-of-the-art models like CLAP and BEATs achieving near-perfect accuracy (98%), surpassing human levels (81.30%). This paper employs high-complexity models, however, also focuses on simpler approaches with proper data augmentation and architecture exploration. Models include a TensorFlow remake of the original Pickzac model, sequential CNNs, parallel CNNs with Batch Normalization, classical machine learning models, and a Visual Transformer. The top performer, a parallel CNN, attains 91.0% accuracy on ESC50 and 97.0% on ESC10 but with increased memory and time consumption. Detailed model descriptions, training methods, and results are presented later in the paper.**

***Index Terms*—Sound Classification, Deep Learning, CNN, Transformers, Data Augmentation**

Link to repository

I. Introduction

Environmental Sound Classification (ESC) stands at the intersection of audio signal processing, machine learning, and environmental monitoring. As the world becomes more reliant on automated systems and smart technologies, the ability for machines to understand and interpret the sounds of their surroundings becomes increasingly vital. ESC, therefore, represents a pivotal research domain with applications ranging from hearing aids to security and surveillance systems.

One of the key datasets in the context of ESC is the famous ESC-50 developed by K.Pickzac [1]. Since its introduction innumerate models have been tackling its challenge, with state-of-the-art models such as CLAP [2] and BEATs [3] getting performances close to perfection (around 98% accuracy [1]), well above the human accuracy (81.30%).

The mentioned models are the highest level in terms of results. In this paper we will focus on simpler models, which put their focus on proper data augmentation and parameter selection, trying different architectures (feature extraction, CNNs, transformers) to get above-human-level results. Every model is tested on ESC10 and ESC50 datasets. The models we present are the following:

1) A loyal tensorflow remake of the **original Pickzac model** [4];
2) a **sequential CNN**, without data augmentation, which pre-selects the most intense 1.25s clip;
3) a **parallel CNN**, with Batch Normalization and proper data augmentation;
4) Classical Machine Learning models with **Feature Engineering** techniques as a baseline
5) **Visual Transformer** Implementation

The best model is the parallel CNN, achieving a test accuracy of 87.3% on the ESC50 dataset and 97.0% on the ESC10 dataset. It is also the most expensive in terms of memory and time consumption. All details about models, training and results are described later on in this paper.

II. Related Work

The challenge of ESC50 sound classification got sounding attention from many researchers, resulting in numerous models trained and tested on this task and leading to over-the-top performances. The main approach is to convert the audio files to spectrograms, but some works train their models on the raw waveform, or use both raw signal and spectrogram [5].

The current state-of-the-art deep learning models have performances above 98% [1]. For reference, the human baseline stands at 81%. The best performance at the moment [2] is held by a transformer-based model pretrained by natural language supervision. Out of the top 5 models, 4 of them use attention-based models, and all of them make use in different ways of pretraining on a different dataset.

Recent advancements in audio classification have seen the development of innovative models addressing various challenges and exploring diverse methodologies. In [6] HTS-AT is being introduced, a hierarchical token-semantic transformer for audio classification. Notably, HTS-AT achieves state-of-the-art results on AudioSet and ESC50 datasets, with an accuracy of 95.6% on the ESC50 dataset. Furthermore, it equals the state-of-the-art on Speech Command V2 while requiring only 35% of the model parameters and 15% of the training time compared to previous audio transformers.

The CLAP model (Contrastive Language-Audio Pretraining) is developed in [7], a paradigm shift in learning audio concepts from natural language supervision. CLAP sets a new benchmark for Zero-Shot performance across 16 downstream tasks in 8 domains, surpassing similar computer vision models despite being trained with significantly fewer audio-text pairs. Notably, CLAP achieves an impressive accuracy of 96.70% on the ESC50 dataset, showcasing its effectiveness and flexibility.

AemNet is introduced in [8], an end-to-end audio embeddings generator for acoustic scene and event classification.

[†]Masters in Data Science, University of Padova, email: marco.furlan.13@studenti.unipd.it, student ID: 2063577

[‡]Masters in Data Science, University of Padova, email: ivan.krstev@studenti.unipd.it, student ID: 2071991

[1]see table in https://github.com/karolpiczak/ESC-50

AemNet achieves high-quality and robust audio embeddings, boasting performance results of 91.60% on DCASE 2013, 83.55% on UrbanSound8k, and 92.32% on ESC-50, demonstrating its scalability and efficiency.

An extensive study on ensembles of classifiers for audio classification is performed in [9], employing Convolutional Neural Networks (CNNs) with various data augmentation techniques. The ensembles outperformed or performed comparably to state-of-the-art methods on benchmark datasets, including ESC-50, with an accuracy reported at 88.65% on this dataset. The study also highlighted the potential for further exploration, such as investigating the impact of different CNN topologies and parameter settings coupled with various types of data augmentation.

Finally, we mention the model proposed by K.J.Pickzac [4], which is reproposed in this paper with a tensorflow implementation.

III. Dataset

The datasets we are dealing with are the ESC50 dataset and its subset ESC10 dataset. The **ESC50 dataset** is a collection of 2,000 environmental sound recordings, with 40 samples for each of the 50 sound categories. Each audio sample is stored in a .wav file, 5 seconds long, 44.1 kHz, mono. It covers a diverse range of sounds including animals, nature, urban, and human-made sounds. It is widely used in research for tasks such as sound classification and event detection. The **ESC10 dataset** is a subset of the ESC50 containing 10 classes for a total of 200 samples. The 10 classes are: dog, rooster, rain, sea waves, crackling fire, crying baby, sneezing, clock tick, helicopter, chainsaw.

IV. Signals and Features

We present here briefly the theoretical background behind the approaches we used for obtaining and pre-processing features:

A. Spectrograms

A **log-scaled Mel spectrogram** is a representation of audio signals that combines the Mel scale, which approximates human auditory perception, with logarithmic scaling, as shown in Figure 2. It breaks down audio signals into short-time frames using the Fourier transform, converts the frequency scale to Mel scale (see Fig. 1) to better capture human perception of pitch, and computes the power spectrum of each frame. The resulting spectrogram is then transformed using a logarithmic function to compress the dynamic range and enhance the visibility of lower intensity components. This representation is commonly used in speech and audio processing tasks, such as speech recognition and sound classification, due to its ability to capture relevant acoustic features in a human-like manner.

B. Feature Engineering

Audio data is very simple and easy to collect. It comes very natural to how humans communicate, through talking to each other, especially naturally and spontaneously talking in an every-day manner. We extracted the acoustic features from the audio files using the Librosa library in Python.

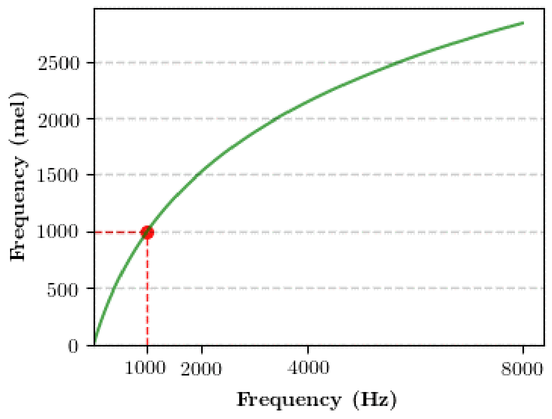

Fig. 1: The mel scale, which maps frequencies to mimic human perception.

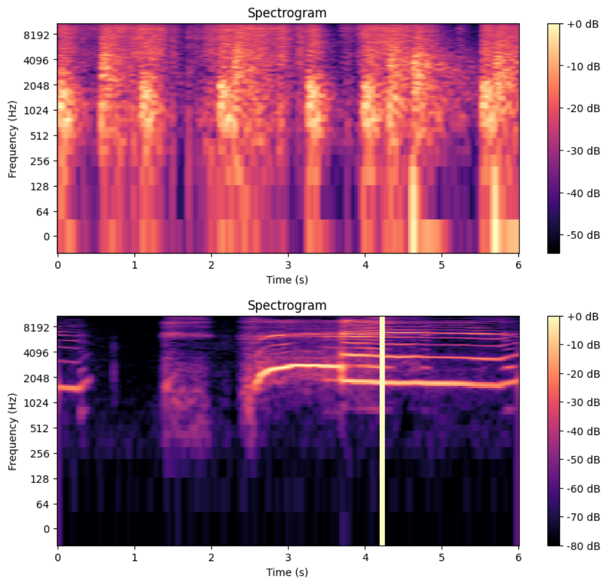

Fig. 2: Example Spectrogram without and with a mask applied.

The following proposed features are shown to be very effective and descriptive in broad number of use-cases when it comes to audio data and Machine Learning, and describing the difference our classes of interest:

- Zero Crossing Rate (ZCR) - The number of times the amplitude of the signal changes its sign from positive to negative or vice versa in a given period.
- Spectral Rolloff (SR) - The frequency below a certain amount of energy of the spectrum is contained (85% by default for Librosa).
- Spectral Centroid - Localization of the mass spectrum in the spectrogram.
- Spectral Bandwidth (SB) - The value that characterizes the wavelength interval over which the magnitude of

the signal is not less than a specified fraction of the component of the signal having the maximal value.

- Chroma Frequencies (CF) - Define the harmonic and melodic characteristics of a given audio, in-dependently of the timbre and the instrumentation. They are closely related to the twelve different pitch classes
- Mel Frequency Cepstral Coefficients (MFCC) - The MFCCs are a system of values that adapt the frequencies of the sound to the way humans hear via a special referent system. They are computed in a sliding window fashion, which means the sound is divided into frames (windows) which can be overlapping or disjoint. Each of these windows is converted to the frequency domain using the FFT (Fast Fourier Transform) and each of the obtained spectrograms is mapped to the mel scale using the mel filterbank and is then logarithmized. Now, for each of the new spectograms, we perform Inverse Fourier Transform (Discrete Cosine Transform) and the MFCC coefficients are the amplitudes for each of the frames. Note that, the final results are neither in frequency nor in time domain. The newly obtained domain is called quefrency and here live the MFCCs. Only the first 40 mel coefficients were taken into consideration in our experiments.

The output of extracting the previously mentioned features is a matrix, due to the fact that we obtain one value for each frame in the audio, and this matrix can be the input of a Convlolutional Neural Networks, but as we also did experiments with Classical ML Models, that expect a vector as an input, we calculated the minimum, maximum, standard deviation, skew, median and mean of each of the dimensions (rows) of that matrix in order to obtain a 210-dimensional vector representation to feed the models.

C. Data Augmentation and Pre-processing of the audio signals

Data augmetation is an essential part of our processing pipeline. The following data augmentation techniques were applied on raw audio files:

- random time delays (see [4] for details),
- adding white noise,
- sound shifting,
- pitch shifting,
- time stretching.

And the following on the spectrograms:

- frequency masking,
- time masking. (Figure 2)

Different models use different combinations of data augmentations, or no data augmentation at all. Details are described per-model later on.

the **power** of an audio signal $x(t)$ is defined as

$$P = \frac{1}{T}\int_0^T (x(t))^2 dt$$

and it is a measure of the strength of the signal. In the discrete case, it is computed as:

$$P = \frac{1}{t}\sum_{t=0}^{T}(x(t))^2$$

If we represent x as a vector, such that the i-th entry is the audio signal strength at time i, then the power can be written in terms of the norm:

$$P = \frac{1}{T}|x|^2 \propto |x|^2$$

This implies that to compare the power of audio clips with the same length, as we do in Model 2, we can just compute their squared norms as dot product, $|x|^2 = x \cdot x$.

V. Learning Frameworks

In this section we describe the general architectures of the models employed in our study.

A. Classical Machine Learning

Five different Classical ML models were considered for the sound classification task, including Support Vector Machines (**SVM**), **Naive Bayes Classifier**, **Random Forests**, K Nearest Neighbours (**KNN**) and **XGBoost**.

Instead of training the models themselves, a pipeline, consisting of standardization and Principal Component Analysis (**PCA**) layer was trained. Both the standardization and the PCA dimensionalty reduction contributed to the improvement of the classifiers, and only the results obtained with these pre-processing techniques included in the pipeline will be discussed later.

Results yielded by the dimensionality reduction and the models heavily depend on hyper-parameters. So, in order to train accurate and precise models for classifying the sounds, tuning those hyper-parameters is crucial. For that purpose, Grid Search algorithm combined with 3-fold cross validation was used to optimize the classification models.

B. Deep Learning Approaches

- **Model 1: Piczak's**. The project architecture can be found in [4]. Originally it was implemented in pylearn2, but we re-implemented it from scratch in tensorflow. Briefly, it consists of a double classic convolutional block (conv, relu, maxpool) and two fully connected layers.
- **Model 2: sequential CNN**. This model follows a common pattern where we repeatedly apply convolutional blocks (conv, relu, maxpool) to progressively reduce the dimension of the image, while we increase the number of features. The exact structure consists of 4 blocks of (conv, relu, maxpool) where the convolutional layers have filter size (3x3) and 32, 64, 128, 256 filters; the maxpoolings have (2x2) pool size. Finally we add a dense layer with relu activation function and 512 neurons, and a final dense layer with 50 neurons (for ESC50, 10 for ESC10) and a softmax activation function.
- **Model 2b: sequential CNN #2**. Different combinations of layers were tried to better the performance of the model above. Ultimately an improvement was found by

applying to each layer a pre-activation Batch Normalization layer. As we can see from 2, performance was slightly improved.

- **Models 3/3b: parallel CNN**. This model uses a different approach than the other two, parallelizing groups of 4 blocks of (conv, BN [2] , relu), whose convolutional layers have 1-dimensional filters of increasing length. All blocks start by encoding patterns in a single frequency band (horizontal filters), and then alternate vertical and horizontal to encapsulate relationships in-frequency and across-frequencies. This approach reveals to be extremely powerful and memory efficient, but requires longer training time and proper data augmentation.
- **Model 4: Vision Transformer** inspired by the success of transformer architectures in natural language processing, have emerged as a powerful paradigm for image understanding [11]. Unlike traditional Convolutional Neural Networks (CNNs) that rely on localized convolutional operations, visual transformers leverage a self-attention mechanism to capture global contextual relationships within an image. This self-attention mechanism enables each pixel or patch to weigh the importance of others, facilitating the modeling of long-range dependencies. Visual transformers typically split an input image into fixed-size patches, linearly embed them, and process them through multiple transformer layers, where self-attention is applied to capture intricate spatial dependencies, as demonstrated in Figure 3. The transformer architecture excels at handling diverse and complex visual patterns, making it well-suited for tasks such as image classification, object detection, and segmentation. The success of visual transformers has sparked a new era in computer vision research, showcasing their versatility and effectiveness in learning hierarchical representations from visual data.

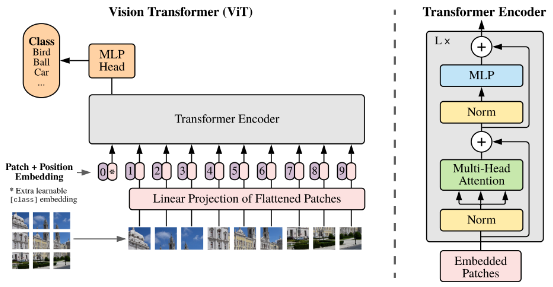

Fig. 3: General Architecture of the Visual Transformer Model.

VI. Processing Pipelines

Preprocessing is a crucial part of this task, with a special focus on data augmentation given the limited amount of samples per-class and the high number of classes. Here is a per-model breakdown of data preprocessing:

- **Model 1: Piczak's** [4]. The dataset is resampled to 22.05k. Random time delays are applied to the ESC50 dataset, for a total of 5 augmented copies, while the ESC10 has class-specific data augmentation. Audio files are normalized and converted into log-scaled mel-spectrograms (512 hop length, 1024 fft, 60 filters). Spectrograms are normalized and then divided into 10 equally spaced segments of 0.95s each, which have a $\sim$ 50% overlap. Silent segments are discarded in the process. The input to the model are the segments and their deltas stacked on the features dimension (input size (60x41x2)). We do a train-test split of 0.8 - 0.2 for consistency with the original model. Optimizer, learning rate, epochs, loss function and dropout coefficients are consistent with the values presented in the original paper.
- **Models 2/2b: sequential CNN** . This model does not rely upon data augmentation, but on carefully choosing the most significant clip from each sample. This is done by splitting the audio signals in 207 equally spaced overlapping 1.25s clips, and selecting the one with the highest power. The clips are then converted into log-scaled mel-spectrograms (108 hop length, 2048 fft, 256 filters, resulting in (256x256) input).
  Train-test split is 0.8-0.2. For training it uses AMSGrad optimizer with 10^{-4} learning rate. Batch size is 10, 50 epochs, loss function is categorical crossentropy.
- **Model 3: parallel CNN**. This model heavily relies on data augmentation: on top of the raw dataset, 4 copies are made, augmenting with white noise, sound shifting, time stretching, and a combination of white noise and sound shifting or time stretching. White noise consists in adding a standard normal distribution multiplied by a random uniform between 10^{-4} and $5 \cdot 10^{-3}$. Sound shifting is done by shifting a relative amount $\frac{1}{i}$ with i randomly chosen in $\{2, 3, 4, 5, 6\}$. Time stretching has a random rate between 0.8 and 1.2. Then we convert the entire waveform to log-scaled mel-spectrograms (128 hop length, 1024 fft, 128 filters) resulting in an input size of (128x1723).
  For memory constraints, we use partial learning on one augmented dataset at a time, 20 epochs per augmentation, totaling 100 epochs, plus 20 final epochs on the original dataset. The model is trained on a generator which randomly outputs one of the following:
  - raw spectrogram
  - spectrogram with frequency masking
  - spectrogram with time masking
  - spectrogram with frequency and time masking.

  Train-val-test split is 0.75-0.15-0.10, and the validation loss is used at each training cycle to choose the best model. We used AMSGrad optimizer with $5 \cdot 10^{-5}$ learning rate. Batch size is 16, loss function is categorical crossentropy.

[2]BN stands for Batch Normalization, a layer introduced in 2015 [10] which reduces internal covariate shift by normalizing the input with a running estimate of the mean and the standard deviation computed per-batch.

- **Model 3: parallel CNN #2**. The model above without time and frequency data augmentation; performance is similar.
- **Model 4: Visual Transformer**. In refining our visual transformer model for environmental sound classification, we've meticulously adjusted critical parameters to address overfitting and optimize performance. Notably, our $model\_config$ dictionary outlines these tailored configurations. Unlike the original settings, we've strategically modified the patch size, setting $patch\_size\_height$ to 16 and $patch\_size\_width$ to 37. This deliberate alteration aims to capture specific temporal and frequency characteristics present in environmental sound data. We've chosen a moderate depth and heads to balance model complexity and generalization, while expanding mlp_dim to enhance expressive capabilities. To avoid excessive overfitting, extensive data augmentation techniques are applied to both raw audio signals and spectrograms. With a batch size of 64, computational efficiency is maintained during training. Further, L2 regularization on the MLPs and the utilization of the AdamW optimizer contribute to model stability and convergence speed.

VII. Results

The Tables 1, 2 and 3 show all results concerning the models we tried. Results of Tables 2 and 3 are relative to a single train-val-test iteration.

The **feature-based models** are the baseline for our research, and they show a pretty good performance considering their simplicity. The SVM is the best classifier with a 0.79 accuracy on ESC10 and a 0.55 accuracy on ESC50.

The **Pickzac model** is one of the pioneers in the ESC50 challenge, but at this time it does not hold the competition with newer models, being the worst performing among our convolutional models.

The **sequential CNN** is a solid, fast architecture which gives reasonable results in fast times with very simple data processing and no data augmentation. It can be used to get quick, accurate results in smaller datasets such as the ESC10. It is not recommended for bigger datasets such as the ESC50 because of the low performance.

The **parallel CNN** is the best performing model out of all of them, with an above 90% accuracy on the ESC50 dataset. This model is also the one which requires the most data augmentation and the longest training time. It may not be the best pick for smaller datasets such as the ESC10 since a model like the sequential CNN can get pretty good results in much faster training times, but it is the recommended one for big datasets such as the ESC50. It outperforms human accuracy both on ESC10 and ESC50.

Finally, the **ViT** transformer has bad results, which are due to overfitting. Despite transformers being the state-of-the-art in the ESC50 challenge, a properly trained transformer needs massive amounts of data and/or serious pretraining, which usually requires great computational resources. Since we did not use pretrained models, the performance of our trained transformer without external datasets nor days-long training is to be expected.

VIII. Concluding Remarks

Environmental sound classification on the ESC10 and ESC50 datasets has reached almost the peak, with state-of-the-art models getting closer and closer to 100% accuracy. With this paper we brought our contribution to this well-known challenge, and achieved above-human performances without using external datasets nor pretrained models. We tried simple feature extraction, convolutional models, and transformers, and tackled the problem with different feature selection and data augmentation techniques. Ultimately, we got a 91% on ESC50 dataset. Potential works for the future may include improving further the sequential CNN model, or building a similar model which improves performance while keeping training low training times on smaller datasets, or training more in-depth a transformer architecture with more data augmentations or on external data sources and see if we can get a strong performance improvement.

References

[1] K.J.Pickac, "Esc: Dataset for environmental sound classification," in *the 23rd ACM International Conference on Multimedia*, (Brisbane, Australia), 2015.

[2] B. Elizalde, S. Deshmukh, and H. Wang, "Natural language supervision for general-purpose audio representations," *submitted to IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

[3] S. Chen, Y. Wu, C. Wang, S. Liu, D. Tompkins, Z. Chen, and F. We, "Beats: Audio pre-training with acoustic tokenizers," *arXiv preprint arXiv:2212.09058*, 2022.

[4] K.J.Pickac, "Environmental sound classification with convolutional neural networks," in *in Proceedings of the IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, (Boston, MA), 2015.

[5] 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 Science, vol. 8, no. 1152*, July 2018.

[6] K. Chen, X. Du, B. Zhu, Z. Ma, T. Berg-Kirkpatrick, and S. Dubnov, "Hts-at: A hierarchical token-semantic audio transformer for sound classification and detection," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 646–650, IEEE, 2022.

[7] B. Elizalde, S. Deshmukh, M. Al Ismail, and H. Wang, "Clap learning audio concepts from natural language supervision," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, IEEE, 2023.

[8] P. Lopez-Meyer, J. A. del Hoyo Ontiveros, H. Lu, and G. Stemmer, "Efficient end-to-end audio embeddings generation for audio classification on target applications," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 601–605, IEEE, 2021.

[9] L. Nanni, G. Maguolo, S. Brahnam, and M. Paci, "An ensemble of convolutional neural networks for audio classification," *Applied Sciences*, vol. 11, no. 13, p. 5796, 2021.

[10] S.Ioffe and C.Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *Proceedings of ICML (2015)*, 2015.

[11] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[3]Human performances from Piczak's paper [1]

ESC10: Feature Engineering					
#	Name	Accuracy	Precision	Recall	F1
1	SVM	0.79	0.79	0.79	0.77
2	Bayes Classifier	0.70	0.68	0.69	0.66
3	Random Forests	0.74	0.74	0.75	0.73
4	K-NN	0.66	0.58	0.59	0.57
5	XGBoost	0.70	0.71	0.69	0.69

ESC50: Feature Engineering					
#	Name	Accuracy	Precision	Recall	F1
1	SVM	0.55	0.57	0.55	0.53
2	Bayes Classifier	0.50	0.51	0.48	0.47
3	Random Forests	0.48	0.49	0.50	0.46
4	K-NN	0.45	0.55	0.44	0.44
5	XGBoost	0.47	0.47	0.47	0.45

TABLE 1: Performance measures on ESC10 and ESC50

ESC10					
#	Name	Accuracy	Precision	Recall	F1
	Human [3]	0.957	-	-	-
1	Piczak's	0.788	0.807	0.788	0.777
2	sequential CNN	0.85	0.885	0.85	0.852
2b	sequential CNN #2	0.862	0.885	0.862	0.863
3	parallel CNN	0.97	0.976	0.97	0.968
3b	parallel CNN #2	**1.0**	1.0	1.0	1.0
4	ViT	0.687	0.716	0.687	0.690

ESC50					
#	Name	Accuracy	Precision	Recall	F1
	Human [3]	0.813	-	-	-
1	Piczak's	0.488	0.48	0.488	0.456
2	sequential CNN	0.578	0.599	0.578	0.576
2b	sequential CNN #2	0.6	0.634	0.6	0.6
3	parallel CNN	**0.91**	0.921	0.91	0.906
3b	parallel CNN #2	0.9	0.922	0.9	0.9
4	ViT	0.412	0.442	0.412	0.405

TABLE 2: Performance measures on ESC10 and ESC50

ESC50					
#	Name	Parameters	Memory	Training time	Testing time
1	Piczak's	26.934.130	102.75 MB	*3m 06s**	*<1s**
2(b)	sequential CNN (#2)	26.104.114	99.58 MB	3m 20s	2s
3(b)	parallel CNN (#2)	1.268.786	4.84 MB	4h 6m	4s
4	ViT	1.680.394	6.4 MB	22m	5.207s

TABLE 3: Model stats on ESC50. **note that this model was trained and tested on a local computer with GPU NVIDIA GeForce RTX 3080 Ti, so the time statistics are much better. The others were trained and tested on Google Colab GPUs. Running the model in Google Colab for comparison was not feasible because of the high batch size.*