Environmental Sound Classification Human Data Analytics - Final Project

Gianmarco Lattaruolo[†], Leonardo Schiavo[‡]

***Abstract*—The remarkable developments in the field of machine learning have been carried out progressively increasing the complexity of the models and the computational power. However, in an educational viewpoint, we chose a divergent path for our project.**

We addressed audio recognition using the Environmental Sound Classification dataset by revisiting and reevaluating the potential of simpler models. This approach resulted in a comprehensive compilation-style work but however with some innovative insights. Our goal was to elucidate the circumstances that warrant the adoption of more complex models or enhanced computational resources, while also highlighting instances where such endeavors may be unnecessary.

In addiction to elementary non-backpropagation based models, we focused on the dense feed forward, convolutional and recurrent neural networks. Moreover, steadily believing in the importance of meaningful projection of data in an appropriate features space, we dedicated substantial effort to investigating the potential of simple autoencoder structures, exploring various architectural configurations. Our findings demonstrate the value of CNNs and emphasize the importance of hyperparameters exploration.

Source code for our work is available at GitHub.

***Index Terms*—Environmental Sound Recognition, Unsupervised Learning, Autoencoder, Feature Extraction.**

I. Introduction

In recent years, the field of sound recognition has witnessed numerous innovative developments, following the trajectory of other relevant machine learning domains. These advancements have expanded the boundaries and enhanced the effectiveness of earlier approaches, introducing increasingly intricate yet resource-intensive models. Undoubtedly, these innovations have also been driven by the wide spectrum of applications for environmental sound recognition, including surveillance, healthcare, security, manufacturing, and more [1].

The objective of this study was to steer away from excessively complex models and instead establish a continuum that spans from foundational models to relatively more sophisticated, albeit still rudimentary, models compared to those discussed in recent literature [2], [3]. With this objective in mind, we intentionally avoided conducting an exhaustive analysis of the results available in the literatur prior to our work, placing ourselves in a position of a first attempt study on the dataset. Clearly, this approach is absolutely wrong if the aim is to produce competitive results, but in our view and considering our great computational limitations, it can be considered reasonable.

Our exploration began with a comprehensive analysis of the available dataset and potential preprocessing strategies, delving into the relationship between mathematical principles and the inherent characteristics of sound. Subsequently, we conducted a series of empirical investigations using fundamental machine learning models within the realm of supervised learning. These encompassed a variety of algorithms, such as Logistic Regression (LR), Support Vector Machines (SVM), Decision Trees (DT), Random Forests (RF), and k-Nearest Neighbors (k-NN). Our research then extended to various neural network architectures, including Feedforward Neural Networks (FNNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs).

Leveraging a substantial repository of unlabeled data, we then examined the possibility of using autoencoders and unsupervised learning techniques to extract meaningful features from audios. Various architectural configurations were employed, incorporating combinations of dense and convolutional layers. Finally, this encapsulated knowledge was reused to retrain previously examined supervised learning models using encoded data. In this way, we aimed to bridge the gap between the two typical stages of model development for environmental sound recognition:

(1) designing and extracting an appropriate feature representation of the sound data;
(2) constructing a suitable classification model based on that representation,

without burdening model training and backpropagation entirely with the task of extracting high-level features.

Through this integrative approach, our goal was to address the task of sound recognition with renewed purpose, attempting to answer the question: is “more” always “better”? While we cannot provide a definitive answer to this question, our work demonstrates that:

- exploring the full range of hyperparameters is a demanding yet valuable task, potentially involving various types of grid searches;
- CNNs exhibit noticeable potential for improvement at an early stage of investigation, especially when compared to RNNs, despite the latter being specifically designed for sequential data;
- projecting data into a constrained feature space using rudimental encoders does not capture significant information from the data, motivating the exploration of

[†]Department of Mathematics, University of Padova, email: gianmarco.lattaruolo@studenti.unipd.it
[‡]Department of Mathematics, University of Padua, email: leonardo.schiavo.1@studenti.unipd.it

more complex autoencoders or alternative unsupervised learning approaches.

II. Related Work

The field's related work traces back to 1997, when Couvreur in [4] employed statistical pattern recognition techniques, including Gaussian Classifiers and Hidden Markov Models, to address noise pollution and develop an intelligent Noise Monitoring System. In 2015, K. J. Piczak's paper [5] marked a significant machine learning effort to establish a foundational reference point for sound classification research. The paper highlighted the limited availability of openly accessible audio datasets and code repositories. To tackle this challenge, he introduced ESC datasets, which were derived from the archive Freesound Project. The paper also defined benchmark performance levels for various supervised learning techniques, comparing their accuracies with human benchmarks.

In the paper [6] the same author delved into the feasibility of employing CNNs for sound classification, confirming their effectiveness. It recommended combining convolutional models with larger datasets, introducing data augmentation concepts into sound analysis.

Pushing these intuitions to the extreme, in 2016 the paper authored by Wei Dai et al. [1], proposes very deep CNNs for processing raw audios directly, achieving significant improvements in sound recognition. These networks, up to 34 layers deep, use batch normalization, residual learning and aggressive down-sampling in initial layers for efficiency. Models with 18 layers outperformed shallow models by over 15%, matching log-mel feature-based models' performances. This study demonstrates that, with no computational and energy limitations, good results can be achieved without relying on handcrafted features.

In 2018, a major advancement was achieved through the model proposed by Shaobo Li et al. in [2], which can be seen as a fair reconciliation between backpropagation-centric models and statistical approaches, integrated with complex preprocessing methodologies. This work introduces an ensemble stacked CNN architecture, featuring two parallel CNN models. One of these models is designed to process log-mel audio features, while the other directly manages raw audios. These models are fused using Dempster-Shafer evidence theory to build the DS-CNN model. The study demonstrates the effectiveness of combining log-mel and raw waveform features to enhance the sound recognition performance.

Recent publications, like [7] and [8], delve into more complex models. In the paper [7] the Audio Spectrogram Transformer (AST) has been introduced, a convolution-free, purely attention-based model. AST processes audio spectrograms and supports variable-length inputs, simplifying architecture design and convergence during training. Additionally, the paper presents an approach for transfer learning from ImageNet-pretrained Vision Transformers to AST, improving its performance. The paper [8] introduces the BEATS model, an iterative audio pre-training framework, which utilizes an acoustic tokenizer and an audio self-supervised learning model for discrete label prediction. This approach results in state-of-the-art performance on various audio classification benchmarks, including the one over our dataset.

Among the ever-growing accuracies in this papers trend, the one by David Elliott et al. [3] deserves a side discussion. It stands out for deploying data processing and machine learning models on edge devices, such as mobile phones and microcontrollers, with applications in areas like autonomous vehicles and security. Featuring a compact BERT-based transformer model trained on Mel spectrograms, they outperform a much larger CNN model on the Office Sounds dataset while using significantly fewer parameters, making it a promising choice for edge devices with limited computational resources.

III. Processing Pipeline

Our research can be divided into the following blocks:

1) exploration of the data and spectral analysis;
2) first comprehensive attempt at supervised learning;
3) autoencoder models;
4) supervised models on encoded data.

Commencing with the unprocessed auditory data, we executed three distinct preprocessing methodologies encompassing the extraction of Short Time Fourier Transform spectrogram (STFT), Mel spectrogram (MEL) and Mel-frequency cepstral coefficients (MFCCs). Including also the original raw audio data, the selection and training of our models were conducted concurrently across all four data types.

Subsequently, we systematically evaluated a range of supervised learning models, spanning from fundamental to moderately sophisticated ones. We started our experimentation with basic machine learning models such as k-NN, SVM and LR, before delving into more complex architectures such as FNNs, CNNs, and RNNs. We performed many exhaustive grid searches over the hyperparameters' spaces.

As a tangential approach, in the third block, we trained autoencoders, with the aim of exclusively utilizing their encoders for feature extraction. We explored various types of preprocessing, different code dimensions, and several architectural configurations, varying from the shallow models to the deeper ones. To alleviate computational demands, we implemented advanced grid search strategies, including Random Search, Bayesian Optimization and Hyperband, which will be discussed in detail later.

This provided us with multiple encoding modes for feature extraction, which were subsequently employed in conjunction with supervised models to assess their utility as high-level features extractors for classification. A comparative analysis was finally conducted to assess the outcomes before and after the feature extraction process.

Our overall processing pipeline reported in Fig. 1 has been applied to both supervised and unsupervised models.

The primary objective, as elucidated earlier, is to initiate our research with a blank canvas and gradually build more intricate models over time. This approach is grounded in the rationale that when basic models already produce satisfactory results, there is no compelling justification for introducing

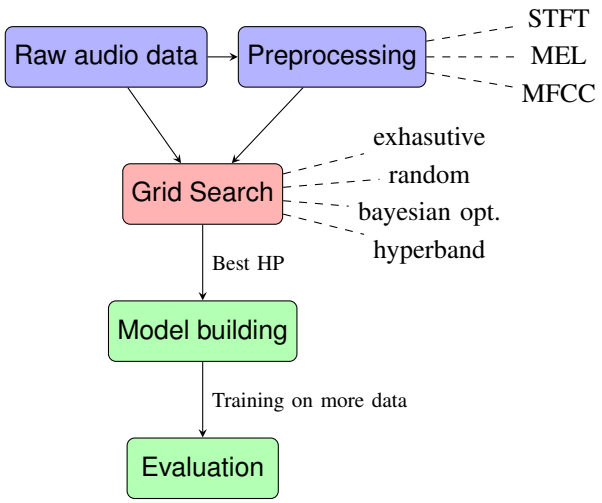


Fig. 1: Processing Pipeline for each model

more complex models. Conversely, if a thorough analysis of various configurations of the simplest models fails to yield substantial improvements, it may become reasonable to allocate additional resources or explore more intricate models. It is worth noting that while such a transition can be considered, it cannot be mathematically proven that the simplest models are inherently inadequate in all cases.

We will elaborate on each pipeline block in Sec. V, but first a brief description of the data we are using is in order.

IV. Dataset

We used two datasets: the Environmental Sound Classification dataset with 50 classes (ESC-50) and the Environmental Sound Classification - Urban Sound dataset (ESC-US). The ESC-50 consists of 2000 labeled audios and the classes are perfectly balanced, so we had 40 audios per class. A subset of 10 classes "well separated" from a human perspective, forms the ESC-10 dataset, which therefore is made of 400 labelled audios belonging to the following classes: `rooster`, `dog`, `helicopter`, `sneezing`, `crying baby`, `chainsaw`, `crackling fire`, `sea waves`, `clock tick`, `rain`. The ESC-US is an incredible larger dataset, made of 250,000 unlabeled audios, and it occupies more than 25Gb of memory disk. The audios are accompanied by some notes (tags) submitted by the original uploading users, which could be potentially used in combination with some natural language model or for weakly-supervised learning experiments. Each audio from both the dataset is 5 seconds long and is encoded in one channel with 44.1 kHz. The format is `.wav` for the ESC-50 and `.ogg` for the ESC-US (Ogg Vorbis codec with 192 kbit/s compression). In reading, each file thus resulted in a single array of length 220,500. In Fig. 2 a raw audio array from class `rooster` is shown. The labeled dataset was leveraged for all supervised learning phases, whereas the unlabeled dataset played a pivotal role in training the autoencoder. All our evaluations used 75% for training and the remaining 25% for testing, the latter eventually split in half to produce validation and test sets. The data at our disposal exhibited high quality, largely attributed to the work by [5]. All the data are available at Harvard Dataverse website.

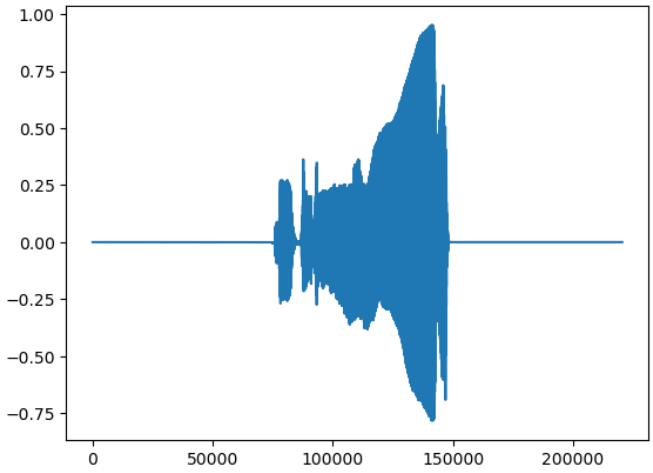


Fig. 2: Plot of a raw audio from `rooster` class.

In our work, we also defined a subset of 10 random classes to ascertain that even from an analytical point of view the 10 classes of ESC-10 are more distinguishable. The results of the basic machine learning models agreed with this supposition.

In the course of our research, we frequently encountered the necessity to use datasets of reduced size. The ESC-10 in particular has been exploited in almost every grid search made in the supervised learning frameworks.

V. Learning Framework

In this section, we provide an in-depth exploration of each step in the pipeline as outlined in Section III.

A. Preprocessing

The preprocessing techniques outlined in this section convert the one-dimensional audio vector into a matrix, which is visualized as a colored image with one channel, wherein the abscissa corresponds to time domain, the ordinate corresponds to frequency domain and the color corresponds to the intensity of the feature. Remarkably, these techniques reduce the dimensionality of each audio of at least one order of magnitude, which means from 220,500 values to less than 22,000.

1) Short Time Fourier Transform: STFT is a specialized Fourier transform technique that involves convolving the input signal, denoted as f, with a window function, typically referred to as w. This window function serves to localize the focus of the integral on a specific portion of the signal. This characteristic enables the STFT to capture not only the frequency variations of a signal over time but also its amplitude changes. The result is a two-variable function $\hat{f}(\tau, \omega)$ that for any fixed τ is the Fourier transform of $f(\cdot)\omega(\cdot - \tau)$:

$$\hat{f}(\tau, \omega) = \int_{-\infty}^{\infty} f(t) w(t - \tau) e^{-i\omega t} dt. \tag{1}$$

In the discrete domain, the formulation remains similar. We use the Hann window function, defined as $w(n) = 0.5 - 0.5\cos(2\pi n/(M-1))$, dividing the audio into segments of $20ms$ with a $10ms$ overlap. Considering the sample rate of $sr = 44,100Hz$ thus we have $M = sr \cdot 20ms = 882$. Given the audio signal $\boldsymbol{x} = (x_1, \ldots, x_N)$, $N = 220,500$ the resulting STFT $\hat{\boldsymbol{x}}$ is:

$$\hat{\boldsymbol{x}}(n,m) = \sum_{k=0}^{881} s_n(k) w(k) e^{-i\omega k}, \tag{2}$$

where s_n is the n-th segments from $\boldsymbol{x}$.

It's important to note that $\hat{\boldsymbol{x}}$ is a complex-valued matrix, therefore the actual matrices we used is the spectrogram $|\hat{\boldsymbol{x}}(n,m)|^2$ converted into decibel domain by the transformation $10log_{10}(\cdot)$. In Fig. 3 we can see the resulting matrix as a colored image.

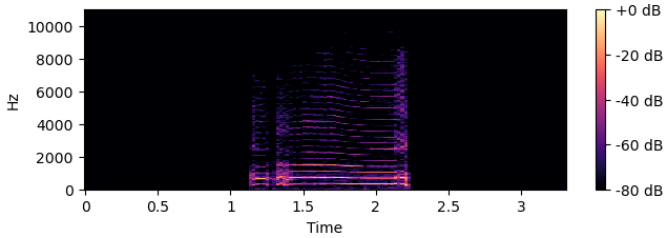


Fig. 3: Discrete STFT converted into decibel domain.

2) Mel Spectrogram: The Mel spectrogram relies on the same preliminary steps elucidated before for the STFT. The difference is that once the spectrogram $|\hat{\boldsymbol{x}}(n,m)|^2$ is computed, the values are converted into the Mel frequency domain through the function $2595\log_{10}(1 + \cdot/700)$. This empirical conversion attempts to project the frequency onto a scale that aligns with human auditory perception. Fig. 4 illustrates this transformation for the same audio sample used earlier, highlighting the subtle differences from the previous representation.

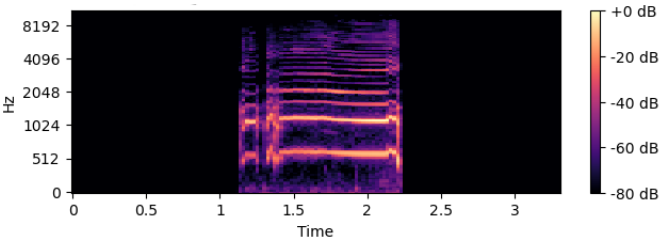


Fig. 4: Discrete STFT converted into Mel frequencies domain.

3) Mel-frequency cepstral coefficients: This last preprocessing technique is probably the most used one in the field of audio data analysis and for sure it's the most sophisticated one. A detailed explanation of this procedure has been covered during our course and is beyond the scope of this report. The high level intuition is that, still starting from the spectrogram $|\hat{\boldsymbol{x}}(n,m)|^2$, for each frame we convert the frequency spectrum into a set of coefficients that emphasize the most relevant frequency components, striving to mimic human auditory perception. The segments from the spectrogram are combined with functions from the Mel Filterbank, which consists of a set of α triangular-shaped functions. Subsequently, logarithmic functions and Discrete Cosine Transformations are applied. Finally, only a specific number C of coefficients is retained. Eventually, other set of values are computed, called *delta* and *delta-delta* coefficients, which are the first and second order differences between the previous coefficients. In our framework we set $\alpha = 50$ and $C = 40$ and in Fig. 5 we can see the plot of the resulting set of coefficients still for the same audio.

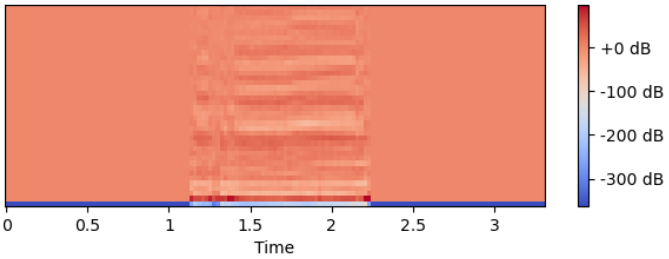


Fig. 5: Mel Frequency Cepstral Coefficients.

Finally, each set of data produced in these ways has been normalized by rescaling between -1 and 1.

B. Supervised Learning

As previously noted, we initiated our study with fundamental machine learning models. Being just a preliminary attempt, we limited ourselves to the study of the capabilities of these models over the raw audio data and the MFCCs preprocessed data. Our decision aligns with findings from prior research, such as [5], which indicated the effectiveness of training models with MFCCs data. To fine-tune the hyperparameters, we performed a comprehensive grid search, eventually incorporating preprocessing in the pipeline, using the ESC-10 dataset. Subsequently, we trained the best-performing models on the ESC-50 dataset. Remind that this process was replicated for each new class of models introduced. In the case of MFCCs preprocessed data, each matrix of coefficients was reshaped and treated as a vector.

Following this, we implemented FNNs and exclusively tested it with raw audio data. Due to the higher input dimensionality, we performed aggressive downsampling in the initial layer, resulting in a reduced number of units. The number of layers has been kept reduced as well. Despite these adjustments, training on the ESC-50 dataset pushed the limits of our computational resources due to the model's over 14 million parameters.

As established in the existing literature, convolutional networks have demonstrated their aptitude for image classification tasks. To leverage this, we introduced basic CNNs models. Using the images produced through our preprocessing techniques, we implemented a three-layer convolutional network with two-dimensional convolutions (of one channel),

each followed by a max-pooling layer. Several other aspects of the model architecture, such as the kernel size and number of units, has been fine-tuned.

Another notable approach involves the use of RNNs. Given the sequential nature of audio data, at first glance, RNNs initially appeared promising. However, using a simple recurrent layer on audio data faced challenges related to gradient propagation, especially with longer input sequences. To overcome this limitation, we turned our attention to Gated Recurrent Units (GRUs), a more adept choice for handling prolonged input sequences. This approach applied not only to raw audio but also to preprocessed audio representations like MEL, STFT, or MFCC. In these cases, the processed audio data can be seen as a sequence of temporal instances, in which the frequencies represent the features. Finally, using raw audio directly with RNNs would have required training recurrent models on sequences of $220,500$ length with just one feature, leading to excessively long epochs and not effective trainings. To mitigate this issue, we attempted to reduce sequence length using one-dimensional convolution, which, with more filters, provided a more manageable number of features for subsequent GRU layers.

C. Autoencoders

This constitutes the central theme of our work: with autoencoders (AEs) our aim was to leverage the large amount of unclassified audios at our disposal. To harness the benefits of working with highly informative features, we explored three distinct types of autoencoders using the Keras-Tuner library [9]:

- an AE that starts from raw audio and encodes it into a vector with significantly reduced dimensions compared to the original using only fully connected layers (hereafter we will refer to this model as "AE-fc");
- an AE that begins with preprocessed audio, reduces dimensionality through convolutional layers, and subsequently produces a vector-type code with a reshape and a fully connected layer (referred to as "AE-mix");
- an AE that operates on the preprocessed images but only applies convolutions to produce a three-dimensional code (referred to as "AE-conv").

For each type of autoencoder, we explored various configurations, including the size and number of filters, activation functions, number of units in the fully connected layers, learning rates, stride values, max-pooling sizes, the potential presence of batch normalization and dropout layers, and the code sizes. While the second and third autoencoders could have been trained on all three preprocessing methods, we chose to focus on STFT and MFCC preprocessing based on the results from our supervised learning experiments, where MEL spectrograms consistently underperformed. Moreover, given the extensive hyperparameter space search we attempted, a classical exhaustive grid search would have been completely impractical. Our analysis therefore shifted towards more sophisticated options[1], again offered by the Keras-Tuner library. In particular, following the suggestions in [11], the following grid search strategies were adopted for each hyperparameter:

- **Random Search**: this strategy randomly selects a combination of parameters without exploring all of them. It presents some advantages over the usual grid search: in case of large hyperparameter space a random search explores it more uniformly.
- **Bayesian Optimization**: this tuning strategy optimizes its parameter selection in each round according to the previous round score. Bayesian Optimization creates a probabilistic model, mapping hyperparameters to the probabilities of a certain outcome score[2].
- **Hyperband**: this hyperparameter tuning techniques is specifically aimed to reduce the time that a grid search takes in neural network models. It simultaneously starts the training of all the models with various combinations of parameters and, after a certain number of epochs, discards the majority of the models (in our case $2/3$) depending on their score up to that epoch. Then it repeats the procedure on the remaining ones, until only one remains.

All grid searches were performed on a subset of the ESC-US dataset. The preprocessed spectrograms have been resized to 64x128 to slightly decrease the computational burden. The best configuration for each hyperparameter was chosen based on consensus or by selecting the option with lower model complexity or greater reasonability when three or more tuners produced different values. Once an optimal configuration was identified, the selected autoencoder was trained on the entire ESC-US dataset.

D. Supervised learning on encoded data

With the top-performing autoencoders successfully trained, we utilized their encoders exclusively to extract relevant features from ESC-50 audio samples. Employing the encoders as non-trainable layers, we conducted smaller grid searches, considering the results obtained from the initial supervised learning block:

- using the encoder from the AE-fc model, we explored various combinations of fully connected layers;
- leveraging the encoder from the AE-mix model, which produces a code sequence, we trained RNNs with GRU layers;
- utilizing the encoder from the AE-conv model, we trained different CNN classifiers.

Subsequently, we trained the most optimal models on the ESC-50 dataset to build our final models.

E. Loss function and optimizer

Regarding the optimizer, we used the Adam optimizer for all the networks we trained, often tuning the learning

[1]For a short overview of HPO methods, we could refer to [10]

[2]For an in depth explanation of this HPO method please refer to [12]

rate in the grid searches. Whereas a distinction between the supervised and unsupervised setting is in order for what concern the loss function and the evaluation metrics. For all supervised models, we employed Categorical Cross-Entropy as the loss function, as we had encoded the classes into a one-hot-encoding representation. Given the balanced classes and the equal undesirability of all types of errors (true positive, false positive, etc.), we simply used accuracy as the evaluation metric. The choice of both loss function and evaluation metric for the autoencoders fell on the Mean Square Error (MSE) between the original image and the reconstructed one. More specialized loss functions for comparing two images has been analyzed as well. The most suitable for our task, the Structural Similarity Index Measure (SSIM), known for its capacity to accurately quantify disparities between images, significantly slowed down our training, leading to a tenfold increase in training time, and was therefore excluded from consideration.

VI. Results

A. Baseline

The results we had as baseline from the paper [5] on the ESC-50 dataset are the ones shown in Tab. 1, where ZCR stands for Zero-Crossing Rate, which is a feature commonly used in the analysis of audio signals. It measures how often the waveform of an audio signal crosses the zero amplitude line.

Model	Accuracy
MFCC & ZCR + k-NN	32.20 %
MFCC & ZCR + SVM	39.90 %
MFCC & ZCR + RF	44.30 %
CNN	64.50 %
Human	81.30 %

TABLE 1: Baseline from [5]

The CNN in the Tab. 1 has 2 convolutional and 2 fully-connected layers, mel-spectrograms as input, vertical filters in the first layer.

From here on we will discuss our results one by one, unfortunately they will often be unsatisfactory.

B. Basic Machine Learning results

The already fine-tuned by 5-fold cross validation models trained on ESC-50 reached the accuracies reported in Tab. 2:

The obtained results exhibit similarities to those presented in Tab. 1. We can appreciate a significant enhancement in outcomes when the same model is trained using MFCCs instead of raw audios. Nevertheless, a conspicuous issue of overfitting has come to our attention, warranting further investigation.

For what regards the FNNs trained on raw audio data the outcome was disastrous. We could expect this having only 40 audios per class and input with 220,500 features. The best model after the grid search had an accuracy of 86.67% on the training set and 4.40% on the test set, underscoring a pronounced issue of overfitting. Moreover, being this model with 14,126,834 parameters it weights more than 150 Mb and thus we can assert that this way to proceed is totally wrong.

Input	Model	Train accuracy	Test accuracy
Raw	k-NN	2.40 %	1.60 %
	SVM	NA	NA
	RF	100.00 %	35.60 %
	DT	14.86 %	3.00 %
	LR	100.00 %	5.00 %
MFCC	k-NN	100.00 %	32.80 %
	SVM	100.00 %	42.20 %
	RF	90.40 %	35.60 %
	DT	43.00 %	14.00 %
	LR	100.00 %	39.80 %

TABLE 2: Basic Machine Learning Methods results.

C. Convolutional Neural Networks results

As said before, with the CNNs, our experimentation initially focused solely on the ESC-10 dataset for grid searches purposes, with subsequent training of the optimal models on the ESC-50 dataset. In Tab. 3 we can see which hyperparameters performed better depending on the preprocessing.

hyperparams	CNN-1 on STFT	CNN-2 on MEL	CNN-2 on MFCC
activation	relu	relu	tanh
kernel size	(5,5)	(3,3)	(7,7)
n. filters	32	32	128
learning rate	0.001	0.001	0.0001

TABLE 3: Hyperparameters optimization results for CNNs.

A comprehensive presentation of the outcomes is provided in Tab. 4. The obtained results exhibit inferior performance compared to the baseline, likely attributed to the baseline model's slight increase in complexity due to the incorporation of more dense layers after the convolutional ones. Finally notice that, even if not reported, we tested the same grid search for the MFCC including the *delta* and *delta-delta* coefficients. The better performing combination, whose result is in the Tab. 4, is the one including all the *delta* and *delta-delta* coefficients.

Input	Model	Train accuracy	Test accuracy	Size (Mb)
STFT	CNN-1	99.73 %	49.60 %	16.6
MEL	CNN-2	98.27 %	48.40 %	16.6
MFCC	CNN-3	99.80 %	54.00 %	5.9

TABLE 4: CNN results and models' weight.

D. Recurrent Neural Networks results

As expounded upon in Section VI-C, comprehensive grid searches were conducted on the ESC-10 dataset. The results of the hyperparameter tuning are reported in Tab 5. The "NA" value means that the hyperparameter was not involved in the building procedure of that model. Remind that for the RNN-1 we have reduced the input dimensionality through a 1-dim convolution and to have the correct input dimension for the subsequent GRU layer we were forced to use only one filter.

hyperpar.	RNN-1 raw	RNN-2 STFT	RNN-3 MEL	RNN-4 MFCC
activation	tanh	tanh	tanh	tanh
kernel size	256	NA	NA	NA
n. filters	1 fixed	NA	NA	NA
strides	128	NA	NA	NA
n. units	8	32	32	32
n. layers	NA	3	2	2
learning rate	0.001	0.001	0.001	0.001

TABLE 5: Hyperparameters optimization results for RNNs.

Analyzing the accuracies of these cross validated results, we concluded that STFT and MFCC were the only preprocessing types for which it was worthwhile to train the model with the entire dataset. The corresponding outcomes for the ESC-50 dataset are Tab. 6.

Input	Model	Train accuracy	Test accuracy	Size (Mb)
STFT	RNN-2	53.80 %	32.00 %	1.4
MFCC	RNN-4	54.67 %	32.80 %	0.35

TABLE 6: RNN results and models' weights.

We can say that a simple implementation of GRU layers for an RNN is not capable of handling an audio sequence properly, motivating further development such as bidirectional RNNs and attention mechanism.

E. Autoencoders results

As said before, to assess the effectiveness of the autoencoders, we employed the mean squared error, a suboptimal metric chosen due to the computational constraints associated with alternative options. The results of grid searches, already unified between the 3 different tuning methods presented in Sec. V-C, are shown in Tab. 7, where various hyperparameters influencing the models' architecture are separated from the standards hyperparameters regarding the model workflow. Moreover, we won't report the model score since it is of difficult interpretability. Thus in the same Tab. 7 the number of parameters and size on disk of the reulting model is included.

At a first glance, some of the results provided in Tab. 7 are quite wired: batch normalization seems to be never a good option, some strides and max pooling values seem to be very aggressive and indeed in the case of valid padding it forces a reduced number of layers and in general there is quite a lot of variability in the results. The AE for raw audio has a huge number of parameters and, listening to the reconstructed audio, very poor capabilities: again we can say that a good preliminary features extraction is mandatory. The reconstruction capability of the other autoencoders is quite poor as well as we can see in Fig. 6, which shows three example for the AE-conv trained on STFT transformed audios, but given the very reduced code size it is still appreciable[3]. Nevertheless, watching at the latent space of the same model, shown in Fig. 7, we can see how there is very little intra-class similarity and how between distinct classes the differences are not great. It does not seem the result of an effective training.

model input	AE-fc raw	AE-mix STFT	AE-mix MFCC	AE-conv STFT	AE-conv MFCC
hyperparameters for models' architecture					
n. units list	[64, 32, 16]	NA	NA	NA	NA
n. layers	3	1	3	2	2
batch norm	False	False	False	True	False
code chann.	NA	NA	NA	6	6
code size	16	32	32	NA	NA
hyperparameters for models' workflow					
activation	relu	elu	tanh	elu	elu
drop out	0.2	False	False	False	False
n. units	NA	16	16	128	16
kernel size	NA	(3,3)	(3,3)	(3,3)	(3,3)
strides	NA	(3,3)	(2,2)	(2,2)	(2,2)
max pooling	NA	(3,3)	(3,3)	(2,2)	(2,2)
padding	NA	valid	same	valid	valid
learn. rate	0.001	0.005	0.005	0.01	0.01
resulting model					
n. params	28,450,372	190,117	56,753	2,644	2,055
size (Mb)	341.8	1.6	1.2	0.77	0.37
code shape	(,16)	(,32)	(,32)	(,3,7,6)	(,3,7,6)

TABLE 7: Hyperparameters optimization results and models' sizes for autoencoders.

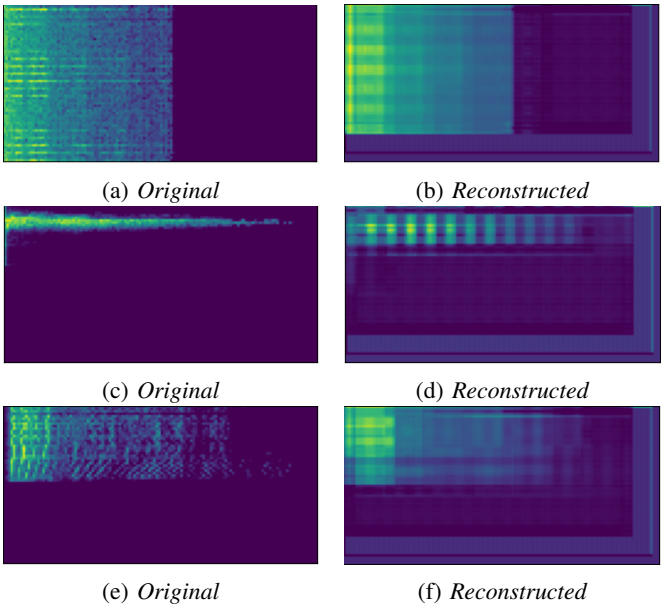
Solely for presentational purposes, we also trained an AE-mix to produce a code of size 2 in order to understand how far the classes were projected in a distinguishable manner on a plane. The result, shown in Fig. 8, is not so good but cherry picking two classes they are well clustered.

In general, everything in this section easily explains the unsatisfactory results of the next.

F. Supervised learning on encoded data results

After training the autoencoders, we constructed models in which we froze the encoder from the previous section and subsequently trained some classifiers. As always, the 5-fold grid searches have been performed on ESC-10, but the results are not interesting, since the final performance is definitely worse than the ones obtained in the first supervised learning

[3]We do not plot the original and reconstructed MFCC images since they are not very informative even in the original one.



(a) *Original* (b) *Reconstructed*

(c) *Original* (d) *Reconstructed*

(e) *Original* (f) *Reconstructed*

Fig. 6: AE-conv for STFT reconstruction capabilities.

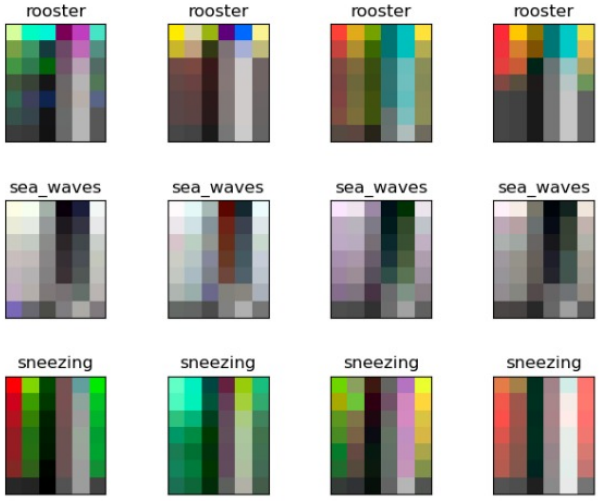


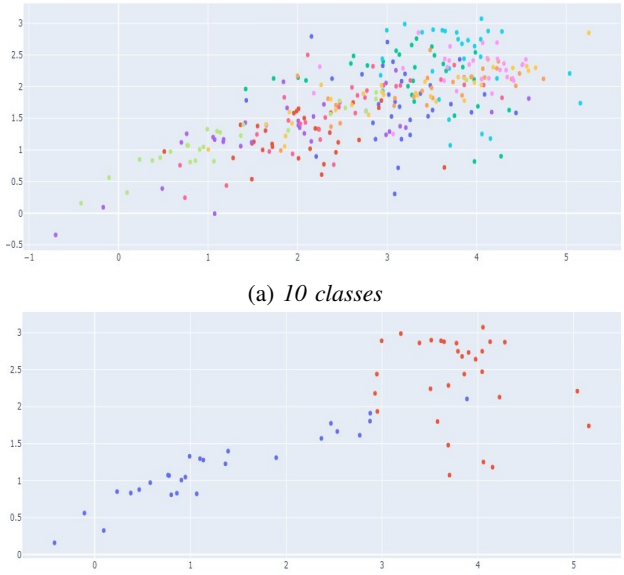
Fig. 7: Comparison between codes produced by the AE-conv for different classes.

block with the same classifiers, as we can see in Tab. 8, containing the achievements on the ESC-50 dataset.

Despite being quite unsatisfactory results, however, from a more general point of view, considering the large amount of work presented, some interesting conclusions can be drawn.

VII. Future Work and Concluding Remarks

Certainly, our work would not end here; it would have to extend until the connection with the most recent machine learning results or, at least, the inclusion of all the fundamental ideas still unexplored. We have not gone so far as to use residual connection, attention, transformers and other more advanced neural networks. We neither implement some transfer learning has done in [7], that would certainly be worthwhile. With the latter in mind, let us now analyze



(a) *10 classes*

(b) *2 well separated classes:* `crakling_fire`, `sneezing`.

Fig. 8: Plot of AE-mix's latent space if code size is 2.

Input	Encoder from	Model	Train accuracy	Test accuracy	Size (Mb)
raw	AE-fc	GRU	19%	16%	181
STFT	AE-mix	GRU	23.73%	21.20%	31
	AE-conv	CNN	58.80%	32.80%	1.1
MFCC	AE-mix	GRU	17.53%	14.00%	29.8
	AE-conv	CNN	78.33%	22.00%	2.2

TABLE 8: CNN results and models' weight.

unexplored ideas that could adjust some critical points in our work. Occasionally, some of these ideas have been explored by trying them out for just a few models, just to see if it would be the right way to go. Often we could not pursue these ideas because of questions of computational time, disk space, or other obstacles, but some additional results are provided.

A. Data Augmentation

The ESC-50 dataset with 2,000 elements and only 40 5-second audios per class is too small for satisfactory training. To overcome this problem, a data augmentation approach can be taken. The main ideas of data augmentation for audios, following what is presented in [13], are: fade-in, fade-out, trim, spectrogram coverage with windows. We tried the last technique: the ESC-50 dataset preprocessed by STFT spectrograms has been covered with rectangular windows over certain frequencies or on a specific time interval, as shown in Fig. 9. We trained the best CNN model resulted from Sec. VI-C, which on ESC-50 with STFT obtained 52.40% accuracy on the test set. The model trained on the augmented dataset went as high as 89.60%. This result, which was definitely unexpected, goes to compare with more recent papers and more complex models, confirming that the data augmentation approach is definitely valid.

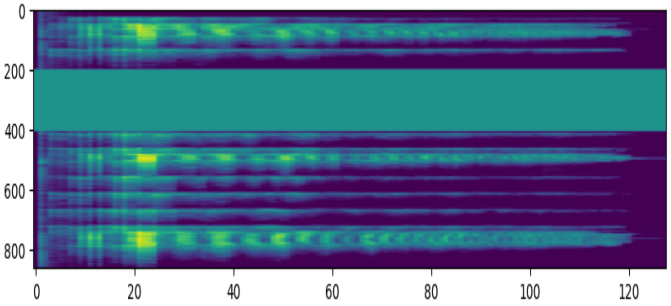


Fig. 9: Time masked STFT

B. Structural Similarity Index Measure (SSIM)

In autoencoder training we were obliged for reasons of computational capacity to use the mean squared error, however, this metric would not be the best metric to measure the difference between images. A more recommended metric is the SSIM which can also capture, for example, that two translated images are similar unlike MSE. To test whether SSIM would really bring a performance improvement, we re-trained the AE-mix model on STFT data with SSIM as loss function and next compared the performance of a simple SVM trained on this code and on the code provided by the previous AE-mix trained with MSE. The results were very similar, in fact with MSE the accuracy on the test set was 25.80%, while with SSIM we got 20.20% actually worsening the performance. We can deduce that either the SSIM does not particularly improve the performance of our models or our encoder structure is totally uncapable to capture meaningful features. In our opinion, using MSE is fine and empirically saves at least an order of magnitude in computational burden.

C. Unsupervised Learning

The low performance of our autoencoders could be due to the poor quality of the ESC-US dataset compared to the ESC-50 one, indeed many audios show excessive or otherwise dominant noise. To solve this issue, one approach would be to use noise reduction techniques with filters. The tags from the audios in the ESC-US dataset can be exploited for semi-supervised learning, as mentioned in Sec. IV, possibly combined with some Large Language Model so that consistent and usable features could be extracted. Finally, changing the dataset to AudioSet, which in more recent papers has been widely used for pretraining algorithms, could be a solution. This dataset is a classified dataset with more than 500 classes and 2 million audios of various lengths.

Of these ideas, we tried exploiting a subset AudioSet, called Balanced Train, to train the AE-mix on STFT data and comparing the classification results of an SVM between the autoencoder trained on AudioSet and the original autoencoder trained on ESC-US. The results were not satisfactory as well, and this reinforces the idea that the intrinsic capacity of autoencoders made of simple neural networks is very low. Moreover, the final confirmation that the structures of our autoencoders were too simple (or inadequate) came when we tried to train the convolutional STFT autoencoder with flat code on the augmented dataset. In that case, we then trained an SVM to classify ESC-50 as usual for comparison purposes. The results were very poor with an accuracy on the test set of 22.80%, which pales in comparison to the astonishing 89.60% reached by the simple CNN.

In conclusion, we have laid the baseline for the Environmental Sound Recognition study, demonstrated that convolutional networks work well and that the approach of data augmentation is very interesting. We encountered many difficulties in computational and spatial capacity. The results we got were an ascending climax in supervised learning and were very disappointing in autoencoders, which confirms the need for more cunning approaches and increasingly complex models.

References

[1] M. Cowling and R. Sitte, "Comparison of techniques for environmental sound recognition," *Pattern Recognition Letters*, vol. 24, no. 15, pp. 2895–2907, 2003.
[2] J. H. G. L. X. Y. J. H. Shaobo Li, Yong Yao, "An ensemble stacked convolutional neural network model for environmental event sound recognition," 2018.
[3] S. W. E. M. David Elliott, Carlos E. Otero, "Tiny transformers for environmental sound classification at the edge," 2021.
[4] C. Couvreur, "Environmental sound recognition: a statistical approach," *Doctorat en sciences appliqu ees, Facult e Polytechnique de Mons, Mons, Belgium*, 1997.
[5] K. J. Piczak, "Esc: Dataset for environmental sound classification," 2015.
[6] K. J. Piczak, "Environmental sound classification with convolutional neural networks," 2015.
[7] Y. Gong, Y.-A. Chung, and J. Glass, "Ast: Audio spectrogram transformer," *arXiv preprint arXiv:2104.01778*, 2021.
[8] C. W. S. L. D. T. Z. C. F. W. Sanyuan Chen, Yu Wu, "Beats : Audio pre-training with acoustic tokenizers," 2022.
[9] K. Team, "Kerastuner api," 2023.
[10] A. Bissuel, "Hyper-parameter optimization algorithms: a short review — by alois bissuel — criteo rd blog — medium," 2019.
[11] K. Team, "Getting started with kerastuner," 2023.
[12] W. Koehrsen, "A conceptual explanation of bayesian hyperparameter optimization for machine learning — by will koehrsen — towards data science," 2018.
[13] "Audio data preparation and augmentation tensorflow i/o," 2022.