
POLITECNICO DI TORINO

# EXPLORING TRANSFORMER MODEL FOR ACOUSTIC SCENE CLASSIFICATION

Master's Degree in Mechatronic Engineering

**Supervisors:**

Prof. Marcello CHIABERGE

Prof. Luis CONDE BENTO

Prof. Mónica Jorge CARVALHO DE FIGUEIREDO

**Candidate:**

Federico PIOVESAN

July 2024

# Abstract

Sounds carry a large amount of information regarding the environment and events that take place in it. Deep learning architectures can be used to automatically extract and interpret these acoustic signals, an ability which is pivotal in numerous applications, such as multimedia retrieval, context-aware devices, robotics, and intelligent monitoring systems. Since its introduction, the Vision Transformer Architecture (ViT) has shown remarkable results in a diverse array of AI tasks, including those related to acoustics. The DCASE competition, with its acoustic challenges, has pushed research in the field, Moreover, over the last two years, DCASE Task1 has served as an effective benchmark for showcasing the performance of ViT models in Acoustic Scene Classification (ASC) problems. This thesis aims to explore the capabilities of the ViT model in this context, using the TAU Urban Acoustic Scenes Dataset. This work seeks to evaluate a Transformer-based solution that remains robust despite variations in recording conditions and devices, and assess the architecture's capabilities given the stringent data constraints and challenges of the task. The Transformer's substantial data requirements, large model size, intrinsic training difficulties, and the use of a consumer-grade GPU necessitated the adoption of efficient strategies to optimize performance and generalization. The role of the Transformer in a knowledge distillation framework was examined, where it served both as a teacher guiding a smaller model and as a student learning from a larger model. The importance of data augmentation and pre-training was also emphasized, confirming the significant challenges mentioned above. Results highlight the necessity of providing the Transformer with extensive data to fully leverage its capabilities and adopting an adequate training strategy. It also provides insights into future directions, including domain adaptation, to further enhance the model's robustness and applicability in diverse ASC scenarios.

# Contents

# List of Figures

# List of Tables

## Abbreviations

**ASC** Acoustic Scene Classification

**API** Application Programming Interface.

**ARM** Advanced RISC Machine

**NLP** Natural Language Processing.

**CBOW** Continuous bag of words

**CWT** Continuous Wavelet transform words

**CNN** Convolutional Neural Network.

**DeiT** Data Efficient Image Transformer

**KD** Knowledge distillation

**FT** Fourier Transform

**FFT** Fast Fourier Transform

**DNN** Deep Neural Network.

**FCN** Fully Connected Neural Network.

**DCASE** Detection and Classification of Acoustic Scenes and Events

**DCASE Task1** Low-Complexity Acoustic Scene Classification

**MLP** Multi layer perceptron

**LSTM** Long-short term memory

**KL** Kullback-Leibler.

# Chapter 1

# Introduction

### 1.0.1 Context and Motivation

Sound is a mean through which countless of meanings can be transmitted and expressed; it is one of the "fast track" of the words, its content can unleash in us several emotions; from the sense of danger or concern derived from hearing a thunder, to the reassuring sense of hearing words from a known and beloved voice. And the list can go on to infinity : sounds is one our guide in scarse condition visibility; if you want to fully catch attention from a stage, you will need images and sounds. Sounds enable us to understand more about our surroundings, and together with vision it gives us the full picture. The human brain can rely on the information carried by sound, it provides us several information, answering to query related to time, to space.

Can machine thinks? One of the questions that has accompanied the science of machine learning since its early steps; a question that hides also the will to understand how we think, in a machine-human parallelism from which knowing one side might enable to understand the other.How machine perceive and process sounds? Questions that are at the basement of a broader field that is the one composed by audio processing and machine learning, whose branches are Acoustic scene classification, sound detection, sound localization, sound captioning and the list can go on. Acoustic scene classification (ASC) involves assigning a predefined semantic label to an audio stream recorded in a certain environment, the latter being often the subject of this classification, while the audio itself with its feature is the mean by which this classification is performed. Generally,in fact, semantic labels stem from environmental sound categorizations that describe the ambiance of audio streams.

The usefulness of sounds for us, reflects also in the several applications that leverages abilities of context awareness,multimedia information retrieval context awareness. Abilities deployed for different scenarios ranging from bioacoustic scene analysis, acoustic abnormal monitoring; for smart devices which can leverage ASC to adapt functions to their surroundings.
.
An important clarification must be made: in the context of deep learning, the term "audio signals"

refers to the broad category of sounds, encompassing speech and musical signals. However, within the domain of Acoustic Scene Classification (ASC), the focus is specifically on the diverse range of environmental sounds. Unlike speech and music, which often contain repetitive patterns and structured elements, environmental audio signals in ASC exhibit greater diversity and complexity, making them more challenging to analyze and classify.

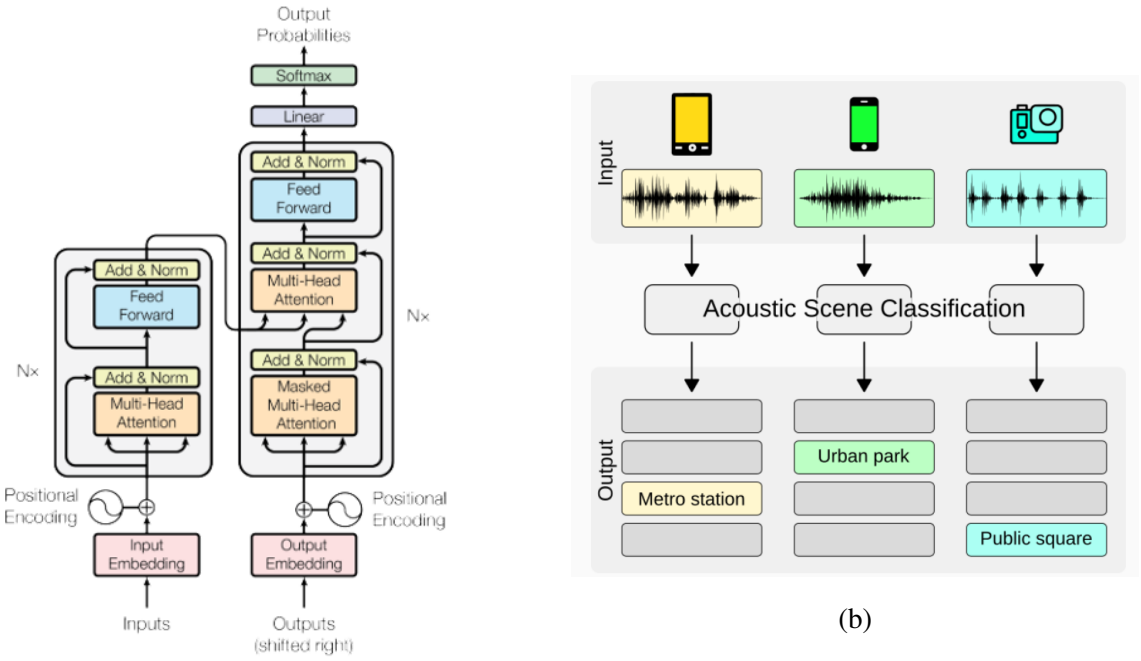
The general pipeline aimed at tackling ASC problem involves several (non-sequential ) steps among which there is data processing, feature acquisition, and modeling steps. The models are the characters,the bridge between the field of audio processing and machine learning. ASC has seen several kind of models in its decades long experience:
Supervised,unsupervised, self-supervised methods: several approaches were and are studied for ASC .The earlier ASC works employed conventional machine learning methods for modeling, such as K-Nearest Neighborhood (KNN), Gaussian Mixture Modelling (GMM), Support Vector Machines (SVM), random forest, and decision tree.As machine learning progressed, these methods were later complemented and advanced by deep learning approaches,recurring to architecture like Feedforward Neural Networks (FNN), Convolutioal Neural Network(CNN), Residual Network (ResNet) etc..

The challenges in ASC (Acoustic Scene Classification) and other branches of audio processing and machine learning are numerous, but the potential ,as just read, is vast. Within this context, the DCASE (Detection and Classification of Acoustic Scenes and Events) competition emerges as a proactive force. This competition, rooted in the fields mentioned above, acts as a significant promoter, driving research forward. It serves as a benchmark for various models and strategies, fostering innovation and advancing the state of the art in dealing with complex audio processing tasks.
In parallel, in completely different fields, the Transformer architecture, a recent advancement in deep learning, has made waves, becoming the state of the arts in several applications. Again, even more recently, Transformer architecture and ASC has crossed path,with the former already making a significant impact on the field. This works aim at exploring the capabilities and the needs of a Transformer based architecture in the field of Acoustic Scene classification.

### 1.0.2 Problem

DCASE Task 1, also known as Low-Complexity Acoustic Scene Classification, is a competition which requires participants to develop an AI architecture capable of classifying audio into one of ten acoustic scenes. Since its introduction in 2013, the task has consistently adhered to its core protocol: ten classes, constraints on the memory usage of the submitted model (aimed at mimicking the capabilities of an ARM4 architecture), an available training and test set for model training and validation, and a ranking based on model accuracy.

From the initial stages of this thesis, our study of the state-of-the-art and various trials revealed that the constraints were too stringent for a Transformer architecture. However, through




(a) (b)

Figure 1.1: a)Transformer Architecture as introduced in "Attention Is All You Need", 2017,extracted from [1]. b)Audios have to be classified in one among 10 classes; different recording devices were employed to build the Dataset.Extracted from [6]

Knowledge Distillation, we explored the possibility of integrating the Transformer into the training pipeline, potentially allowing for a student model to be submitted to the competition. The main objective of this thesis is not to submit a model to the competition, but to use the competition framework to investigate the capabilities of the Transformer in acoustic scene classification. The challenges presented by the competition are mainly the strict dataset,the different devices used for recording and the length of the audios,one second.

Delving into the competition's environment more accurately : DCASE Low-Complexity Acoustic Scene Classification, in short DCASE Task1, entails a classification task in which audios have to be classified in one of ten predefined classes,and those ten classes are shown in table 1.1

Regarding the constraints posed on the model size, the submitted model have to respect the maximum memory allowance of 128KB (Kilobyte). Moreover also a constraint related to the speed of inference : the number of MMAC(Milions Multiply-Accumulate Operations) has to be equal or less than 30. The competition offers a labeled dataset to train the model on: since this work's reference competition's rule are referred to the 2023 edition of DCASE, the datasets is "TAU Urban Acoustic Scenes 2022 Mobile, Development dataset". The training set comprises 139,620 one-second audio clips for training and 29 680 for testing :a training/validation split of 70%/30% results in 97,980 samples for training. Both the training and testing set of DCASE Task1 are class-

Table 1.1: Scene names and label names

| **Scene Name** | **Label Name** |
|---|---|
| Airport | airport |
| Indoor shopping mall | shopping_mall |
| Metro station | metro_station |
| Pedestrian street | street_pedestrian |
| Public square | public_square |
| Street with medium level of traffic | street_traffic |
| Travelling by a tram | tram |
| Travelling by a bus | bus |
| Travelling by an underground metro | metro |
| Urban park | urban_park |

wise balanced, meaning that there is an equal representation of audio samples for each class. This can not be said regarding the recording devices; each audio has been recorded with a recording device: in the training set there are audio recorded with device *[a,b,c,s1,s2,s3]*,while the test comprise also audios recorded with devices *[s4,s5,s6]*.If the testing dataset is class and recording-wise balanced, that is not the case for the training set, which sees a predominance of device *a*'s audios. One clarification has to be made about the recording devices. Devices *a,b,c* fall in the category of

| ***Device*** | **Equipment** |
|---|---|
| *Device a* | Soundman OKM II Klassik/studio A3 + Zoom F8 |
| *Device b* | Samsung Galaxy S7 |
| *Device c* | iPhone SE |

Table 1.2: Table of Devices and Corresponding Equipment

real device ,while devices from s1 to s6 are actually "simulated devices". An audio coming from a simulated devices is actually an audio recorded with device *a* which is then processed through convolution with a selected impulse response, and then processed with a selected set of device-specific parameters for dynamic compression. Each (not published) impulse response used for this convolution process defines a simulated recording device.

Finally, regarding the competition's metrics to award the best models, there are three metrics evaluated on another test dataset: Macro average accuracy(average of the class-wise accuracies), amount of memory needed by the memory, MACs for inference. Those three criteria define three different ranks,criteria, then the overall rank is calculated as a weighted average as of the the three rankings:

$$0.5 \times R_{\text{ACC}} + 0.25 \times R_{\text{MEM}} + 0.25 \times R_{\text{MAC}} \tag{1.1}$$

As previously said, the main objective of this work of thesis is not to build a model to submit; DCASE,since its introduction, has pushed the research in acoustic field related deep learning applications, proving to be an effective benchmark for asserting the capabilities of a model and its

training pipeline, in this branch of deep learning. Speaking about the main challenges that arises when dealing with the classification task proposed by DCASE Task1, they are:

- **Limited Dataset:** With an available Dataset which amounts of nearly 100 000 samples and the general knowledge that Transformer architecture is particularly data hungry, achieving good results is a challenge.
- **One second long audios:** The short length of the audio clips complicates classification tasks, as there is less information to differentiate between classes.
- **Recording devices unbalance:** The recording device information is embedded in each audio clip. While the model can leverage this information, it won't be robust because the test dataset comprises recordings from devices unseen during training, leading to potential performance issues.

These challenges reflect potential issues in various application scenarios, not just ASC. This thesis aims to deploy a Transformer-based architecture, seeking solutions to these challenges by examining the state of the art.

### 1.0.3 Research Questions

The research questions will serve as the foundation for the thesis and guide its direction throughout the research: at the end of the conclusion section we will answer to them. The proposed research questions can be categorized into four main areas: visual representation, the scale or size of the transformer architecture to be deployed, the effectiveness of data augmentation, regularization, and the training protocol techniques , and finally, transfer learning techniques. **Visual representation**

RQ1. **What are the optimal feature dimensions for representing acoustic scenes?**

> The size of the image in visual representations is crucial as it directly impacts resolution and computational resources. Larger images capture more details but require increased time and resources, posing a classic trade-off dilemma.

RQ2. **Are scalograms a viable alternative to spectrograms in the context of DCASE Task1?**

> Spectrograms are the most widely used visual representation tool in many fields. In contrast, scalograms, which offer a distinct visual representation, remain relatively unexplored. Scalograms shine in analyzing non-stationary, time-varying signals,and could offer a new vision to the problem, given the nature of the audio.

**Transformer Sizes**

RQ1. **Can Transformer architectures be optimized to meet the memory constraints of DCASE Task1?**

The competition sets a benchmark for exploring the Transformer's capabilities in acoustic scene classification. However, the role of the Transformer in the training pipeline needs exploration, considering the typical size constraints of these architectures.

RQ2. **Can the Transformer Size be Optimized? What is the Behavior of Different Sizes of Transformers?**

Transformer architectures are defined by hyperparameters such as depth, number of heads, and other parameters discussed in the section related to Transformer Architecture. Investigating the impact of these architectural choices on performance is crucial for understanding how to economize on space and avoid the deployment of superfluous layers or heads. Additionally, considering the context, what are the potential achievements with transformers of varying sizes?

**Data Augmentation, Regularization, and Training Protocol**

RQ1. **What is the effectiveness of different data augmentation and regularization techniques in improving the robustness and accuracy of Transformer models? What role do training protocols, such as learning rate scheduling, play in enhancing the performance and stability of Transformer model?**

The deployment of data augmentation seems compulsory given the limited size of the available dataset. Both domain-agnostic methods and those specifically tailored for acoustic scene classification are explored in this study. Transformer-based architectures are intrinsically difficult to train. This lead to the adoption of particular consideration for way to stabilize the training process, recurring to adapt schedulers.

**Transfer learning and compression technique**

RQ1. **What is the necessity of using a larger dataset for pretraining, and how much data is required to achieve optimal results?**

The small dataset provided by the competition contradicts the common understanding that Transformer models typically require large amounts of data to achieve even sufficient performances: by using pretraining we offer the models more data and an occasion for improving its generalization capabilities.

RQ2. **What are the results and improvement of Transformer Architecture trained with Knowledge distillation?**

The role of the Transformer in a knowledge-distillation paradigm has proved to be really efficient; moreover, since knowledge distillation enable to perform a

compression of the model size, it brings us closer to the idea of deployment on edge devices as the competition pushes to do.

# Chapter 2

# The Transformer Architecture

## 2.1 The Transformer concept

**Transformer architecture** was introduced in the seminal paper *"Attention Is All You Need"* (2017), originally in the domain of Natural Language Processing (NLP). Its remarkable and ongoing success, surpassing previous state-of-the-art methods, can be attributed to two key capabilities:

1. **Parallel Processing :** Referring to the field of NLP ,unlike traditional architectures such as Long Short-Term Memory (LSTM) networks, which process input words sequentially, treating each word as a step in time and building on a cumulative *hidden state*, the Transformer processes all input words simultaneously. In LSTM models, the hidden state is continuously updated with each input word, which is then used to inform the decoding process at the end of the sequence. In contrast, Transformers do not rely on this step-by-step accumulation, enabling them to handle dependencies more effectively and flexibly. The problem with LSTM by the time the encoder process the last word,the hidden state has already lost information regarding the firsts words the encoder processed: this means that those kind of architectures don't work well with long input phrases. The transformer eliminates this problem because it retains an architecture capable of working with all the words it is fed with in parallel.

2. **Lumped-Element Attention Mechanism:** One of the critical factors behind the success of the Transformer architecture is its Attention mechanism. This mechanism allows each input element to dynamically focus on, or "attend to," every other element in the sequence, enabling the model to capture complex dependencies and relationships within the input data efficiently.

A core concept of all the deep models that has to deal with words is how those words are translated, processed,before being fed the aforementioned models,since the latter are able to deal with number rather than words: the process that results in encoding those words into appropriates vectors is called Embedding. The basic approach is the so called "one hot encoding",where,given a vocabulary, each word is associated to a vector of dimension equal to the size of the vocabulary containing all zeros apart from only one spot occupied by a 1,meaning that at the end the position

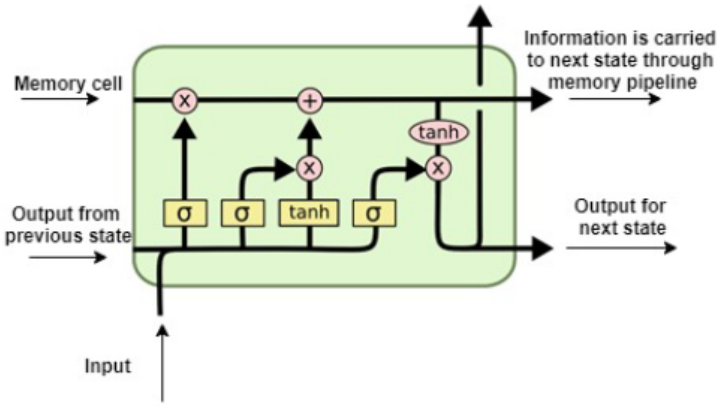



Figure 2.1: Long-Short-Term-Memory unit.Extracted from [36].

of this 1 will identify in a bijective way the associated word. One hot encoding is a solution, but usually not a good one: among the reasons is that,given that the average person knows 20 000 words in his/her native language, the associated one hot encoded vectors would be so large to make operations unfeasible. What it is usually done is building those one hot encoded vector,and then introduce a mechanism(the embedding) capable of "shrink" them to vectors of a feasible dimension,a suitable dimension which the model is able to handle, a suitable dimension capable of enabling the model to recognize,work with,telling apart each word. The concept at the base of the several possible techniques is the aim to create vectors, with each entry resembling a meaning. Which meaning? We don't know:it is the model which is supposed to know.

In Natural Language Processing (NLP), there are several way to perform this word embedding: an example is the algorithm of Word2vec,which consist of a shallow two linear layers; in one of the way it can be implemented, for example choosing the architecture of Continuos Bag-Of-Words, its embedding process involves target word given its context words within a sliding window. Drawing from a large corpus of meaningful text of data(books,articles...),in which each word is represented as an index of a vocabulary.For each word in the corpus, a fixed window size that incorporates its surrounding words. By building pairs of (context,target),where the context are the words encompassed by the window and the target is the word at the center of the window,the model is trained to predict the target word based on its surrounding context words: by doing this, CBOW assign values to the word embedding.

The input embedding of a transformer from the architectural point of view shares some similarities with the example shown above: a trainable linear layer.But usually ,that's all. What is it meant is that the Transformer architecture can be put through a pretraining session on a large corpus of unlabeled text using an objective(such an objective can be the same employed by CBOW),but usually,this is not the common approach; there is no hidden aim or mission that the model has to be performing, no pretext task,in order to adjust the weights: the input embedding is trained altogether with the transformer main task(translation,dialogue). If we then refer to the field of

the application of the transformer architecture we are dealing with, which is audio tagging, there is no need to pre-train the input embedding layers in order to build its own "sound vocabulary", although pre-training of the input embedding layers can be done but with other purposes.

A core aspect of the transformer architecture is the concept of attention: with “Attention” in the deep learning context the community refers to the mechanism with which an architecture is able to “focus”,to spends more resources on specific part of the input rather than others, dynamically. Even prior to the “Attention is all you need ” paper the concept of attention was carved on the mind of whoever wanted to tackle an object recognition/NLP problems: attention can be the ability of a model trained for object recognition on focusing on the important image rather than on the background,attention is the ability to highlight some words rather than others in a phrase and so on.

The ways to perform attention are several: sometimes,for the way they are constructed,model can offer a certain intrinsic degree of attention,as well as other models that have specific algorithms/ building blocks in charge of performing an attention mechanism. For example, in the paper “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention” by Xu et al.[57], a study on a caption generator equipped with attention mechanism is made: developing an algorithm to maximize the log likelihood of conditional probability of data they manage to compute some weights,each of which associated to a peculiar location in the images. Through those weights the model becomes aware on which location of the image to be focused on to generate the following word.

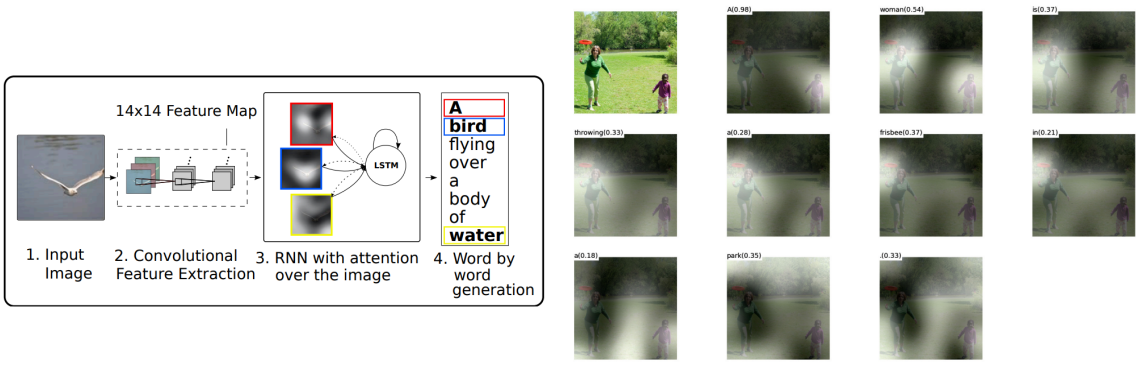



Figure 2.2: Attention weights computed by means of correlation are used to indicate to a RNN on which features coming from an image to focus, to produce the descriptive word.Extracted from [57]

In the paper:” Neural Machine Translation by Jointly learning to Align and translate”(2014)[4] the authors confronts themselves with Neural machine translation(English-French),employing a RNN(LSTM cell) as before equipped with other “modules” aimed at performing attention. In particular the hidden states(vectors computed by the LSTM cell given the input and the previous hidden states in a sequential manner) are fed to a feedforward NN(called alignment model,jointly trained with the overall system) that computes some weights which reflect which hidden state

across the sequence is more important for the concurrent translation operation,and since the hidden state j is(for how things work) more focused on the word j,this will lead to choosing a words rather than other in performing the next translation.

## 2.2 The Transformer building blocks

In this section, we will dwell on the Architectural components of the Transformer: Given its block-based and repetitive nature, we can observe these building blocks starting from a broad overview and progressively delve deeper into their specific roles and functionalities within the Transformer model.

The overall architecture of a Transformer can be briefly described as follows:

- **Input stage** The input stage incorporates the necessary blocks able to perform the translation from the input into the kind of data structure that can be handled by the transformer: regardless of the nature of the input,whether it is inherently a divided sequence or a continuous whole,the transformer will process and divide it.

- **Stack of Encoders** The stack of encoders is a series of identical layers, each of them containing several sub-blocks, among which there is the core block that perform the attention . The main aim of the stack of Encoders is to process the input, extracting features, establishing the amount of relationship between its constituent part.

- **Stack of Decoders** A series of identical layers, equal in number with the stack of encoders, that present a nearly equal structure. If the ultimate goal of the previous macro layer is the extraction of information, here the ultimate goal is to generate something, based on the information extracted from the encoders, and the precedent generated output.

- **Output stage** The output stage transforms the final hidden states of the decoder stack into the desired output format. It typically includes linear layers, softmax layers. But its actual structure strongly depend on the task that the transformer have to perform,being translation, text generation, or classification

The context in which the Architecture was explained is that of NLP. But Transformers can actually handle different types of data, being that images, audio, and even structured data like graphs or tables. Nevertheless,the aim of the input embedding is always the same, and the way in which it perform it remain constant nevertheless the nature of the input data: projecting the input into a vectors by means of linear layers,layers that are generally trained while the Transformer performs its task,if no pretext tasks are provided. One crucial aspect to highlight is that the Transformer processes its input by breaking it down into smaller segments, such as splitting sentences into words for text data or dividing images into patches.

### 2.2.1 Input Stage

At this stage, the input is projected into vectors: if the input is a continuous whole,this part is in charge also of dividing it in equal sections, for then projecting each part separately. Being the input a phrase,an image,whatever, since the inputs are processed in parallel the concept of position is lost(the orders in which words are spoken/written, for example);in order to overcome this issue,a Positional embedding operation is performed: after being embedded each token is assigned with a numerical value meant to represent the position of that token among the whole input. Practically what it is done is building a bi-univocal function like the one shown in equation 2.1, based on the position of the element in the sequence that will give as output a vector(with the same dimension as the token after the embedding) depending on the position of the token;this is done for each element in the sequence.After that the generated vector(the positional encoding vector) is simply summed with its corresponding embedded token. The procedure is represented in figure 2.3.

$$\mathrm{PE}(\mathrm{pos}, 2i) = \begin{cases} \sin\left(\frac{\mathrm{pos}}{10000^{2i/\mathrm{dmodel}}}\right) & \text{if } i \text{ is even,} \\ \cos\left(\frac{\mathrm{pos}}{10000^{2i/\mathrm{dmodel}}}\right) & \text{if } i \text{ is odd.} \end{cases} \tag{2.1}$$

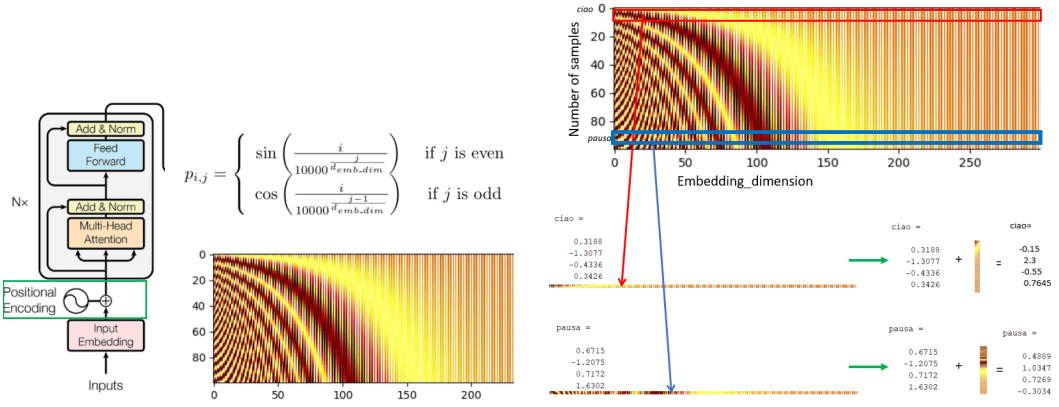



Figure 2.3: Definition of the bi-univocal function and addition for each token in the sequence.Modified from[30].

### 2.2.2 Encoder and Decoder structure

Once the divided input pass the input stage, it is a full-fledged token, and can be processed in parallel by the first encoder. Because the architecture actually provides a number $N$ of encoder(and an equal number of decoder); $N$ is a hyperparameter,it can be chosen. After going through the first encoder,the tokens are sent to the second encoder,and so on,up to the Nth encoder,in a sequential manner. The purpose of the stack of the encoder is to extract features: there are studies regarding the number N one should choose to achieve best results depending on the situation,but in general we can affirm that the more is N ,the more it is likely that complex features are extracted; a concept actually linked,to how much attention modules the transformer is able to supply[27]. The

extracted features at the end are then used to feed each decoder of the stack: each decoder takes as input the extracted features and the output of the previous decoder in the chain(apart from the first decoder,which takes as input the output of the whole stack of decoder). The aim of the decoder section is to generate:always referring to NLP field,this is the part in charge of generating the speech/translation after elaborating the extracted features.

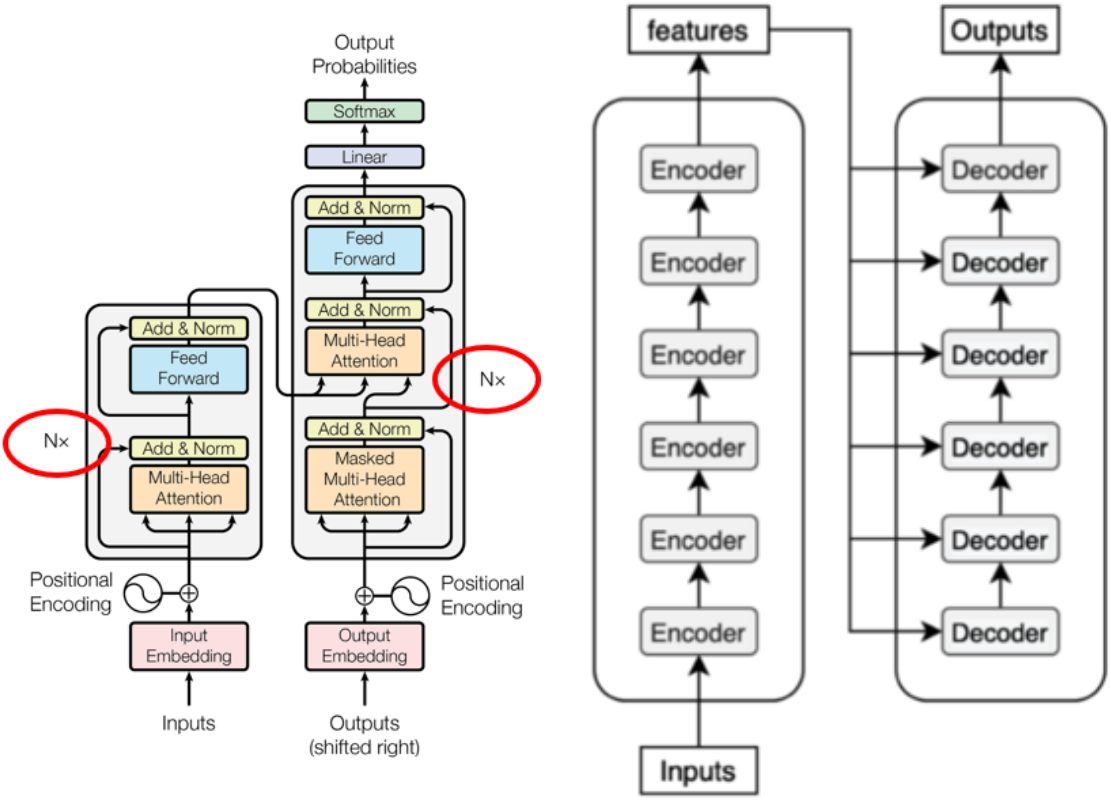



Figure 2.4: Features extracted from the encoder are fed to all the decoder's layers.Extracted from [21].

As we said before,the structure of the encoder will be repeated $N$ times;what will differ along the stack will be the weights,the parameters that are adjusted during the training. As it is possible to see looking at image (1),inside the encoder block those sub blocks can be found: a Multi head attention block,a multilayer perceptron model (MLP),normalization,and some residual connections. The MLP,at the end of each encoder, is none other than a FFN; it is composed of two fully connected layers interspersed by an activation function of the kind of ReLU or GELU, and usually equipped with normalization and Dropout layers. The first layer projects each token into a corresponding token of higher embedding dimension, which are then restored to their original size after passing through the second layer.MLP is meant to introduce a higher degree of non-linearity in the encoder,enabling the model to learn and represent intricate dependencies.

We won't delve into the specifics of normalization and residual connections or the reasoning behind their inclusion by the team who designed the Transformer architecture. However, it is

important to note that these components are indispensable for several reasons: they reduce sensitivity to initial conditions, mitigate the vanishing gradient problem, and effectively enable the deployment of deep architectures that would otherwise be challenging to train and optimize. Normalization ensures stable and efficient training by maintaining consistent activation distributions across layers, while residual connections facilitate gradient flow and simplify the learning process by allowing the network to bypass layers if necessary. The normalization block can be either placed before each sub-block,or after, but the latter seems to be grant more stability in terms of avoiding vanishing of gradients.

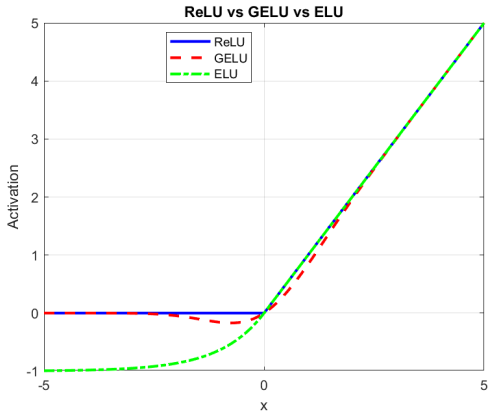



Figure 2.5: Typical activation function deployed in machine learnin.

$$\mathrm{MLP}(x) = \mathrm{GELU}(x \cdot W_1 + b_1) \cdot W_2 + b_2$$

$f$ : Frequency in Hertz (Hz)
$M(f)$ : Mel scale value

This architectural design, with its careful integration of these elements, makes it possible to build and train very deep networks that achieve state-of-the-art performance on complex tasks. Instead we won't skip a focus on the Multi-head attention sub-block,the core of the Encoder:where the "Attention" is performed.

## 2.3 Attention Mechanism: Query, key, value

At this stage,the whole input will assume three different roles at the same time : query,key,value. An example meant to show the reason behind the choice of those names could be a retrieval system,such as Youtube. The query is what we write on the research bar to look for a video; confronting those written words and the keys (title,channel of the videos themeselves in Youtube' servers),the provider will shows us some videos: those videos are the values.Not all the videos are shown,and some videos are shown before the others:this is decided after the compatibility of what we searched and the key parameters of the video.

In the context of Transformer,as we said,the input assumes all the three roles discussed above: the whole input sequence is channeled in the a learnable linear layer, one for each role: there will be so a linear layer that take the input and translate it into query, key,value . The whole procedure

begins with the whole input being considered the query... and at the same time being considered the key.

Prior to going on with how Attention is performed,it is necessary to linger on what is it meant on the Multi in the multi-head attention. Some parts of the structure of the Multi-head attention sub-block are organized in a parallel way;as we will see later( 2.7),there is a stack of “Scaled dot-product”.If we considered the dimension of each token at this stage,it is the embedding dimension. But at the moment each token pass through those parallel section, the token’s vector is divided in a number of sub-tokens,a number which is defined and equal to the hyperparameter "number of heads"of the transformer: so each sub-token’s vector has a dimension defined by the ratio of of Embedding dimension,by number of heads . Each sub-token generated by the corresponding token,undertakes the attention mechanism together with the other corresponding sub-tokens generated by the others tokens,but only with the sub-tokens generated by the same very dimensions of the original token. All the concept we will discuss later(query,key,value and others) remains valid: it is just important to keep in mind that those operations are performed in parallel layers, with each layers only including parts of the original tokens(the corresponding sub-token,for each token): nevertheless,at the end of the attention mechanism the results will merge together by means of concatenation.

We will now focus on the trip that one subsections of the original tokens will undertake, part that from now on we will call, input. The whole procedure begins with the input being channeled at the same time in the query and key linear layers,$W_q$ and $W_k$: exiting precisely as query and key at the same time. Then,a vector dot product between the elements of the query and of the keys is performed: the results are then contained in the Attention matrix. What happened is that each token(if we considered the global process of attention, considering all the parallel layers) is put in communication with each other tokens: dot product highlights the cosine similarity among vectors. So the aim of this dot product is to discover, to highlights the most valuable connection. After the dot product, there is a simple scale operation(a division): the elements of the attention matrix are divided by sqrt(); then a softmax is applied: the row of the attention matrix are normalized so that the sum of the elements in each row gives one. At this point, the Value joins the game. As before, the Value is non-other than the input itself, after passing through a linear layer $W_V$. The elements of the attention matrix are employed as weights that multiply the Values.

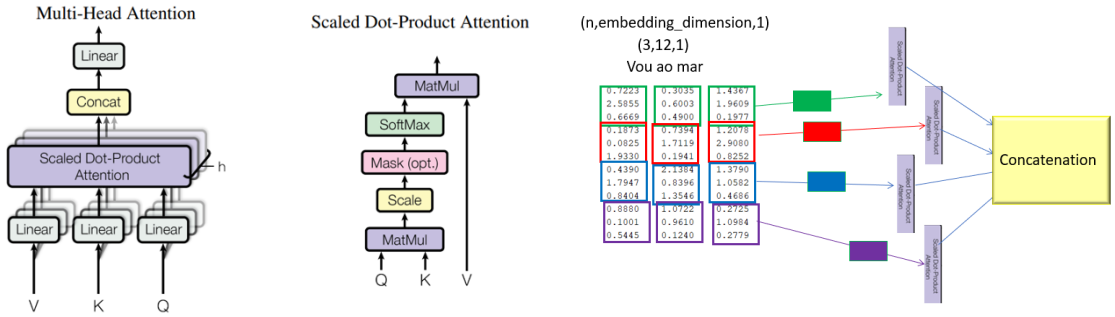
$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

So, in short, what happened is that after computing the Attention matrix (whose values represent the affinity in terms of meaning of each element of the input with all the other elements of the input and itself) those are used as weights to highlights or overshadow the very same input sequence. As we are dealing with only a part of each token, after going through this stage, there will be a simple concatenation meant to bring together all the information extrapolated from the parallelization: the output of the attention sub-block has a dimension equal to the input that enter

this sub block: there will be an output token for each token, weighted with the softmax-scaled elements of the attention matrix.

$$\begin{aligned} \text{MultiHead}(Q,K,V) &= \text{Concat}(\text{head1},\ldots,\text{headh}) \cdot WO \\ \text{where head}_i &= \text{Attention}(Q \cdot W_{Q_i}, K \cdot W_{K_i}, V \cdot W_{V_i}) \end{aligned} \tag{2.3}$$

After being subject to the Attention sub block, the input will be interested in all the other processes contained in the encoder (residual sums, batch normalization, FFN ). And as said before, once the output of the first encoder is ready, it will be passed through another encoder, until the Nth.

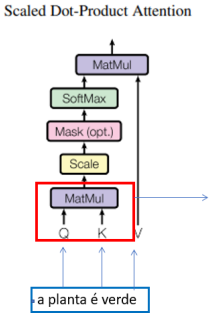



(a) (b)

Figure 2.6: a) Multi-Head attention.Extracted from[50].b)Schematized process of division in sub-tokens, Multi-Head attention and later concatenation.

The decoder structure is pretty similar to the encoder's one: we will focus on its 2 cores, both Attention sub-blocks. The very first decoder of the decoder stack takes as input the output the whole decoder generated previously:an output that underwent the very same stages that also the input experienced: an Embedding Positional encoding, deploying the same weights used in the input stage . After that, it goes into the first attention sub block: the decoder Masked Multi-head self attention. The situation is really similar, if not equal, to the one explained before in the Encoder's self attention sub block. One thing that changes are the characters: the previously generated output(in the case of the first decoder of the chain) or the results of the chain of decoder. But the main difference is the presence of another function in the computation of multi-head attention: masking. Masking is mostly applied only in the word processing field: in short, it involves shutting to -infinite some entries of the attention matrix, making the matrix a triangular one after soft max. The concept behind this is that the decoder generated the output word in order: for example, while producing the 3rd word of a phrase, he did not employ any information regarding the $4^{\text{th}}$ word generated, because this word was not already present. Masking is a way to prevent the "communication" between those words, since the generation of one did not comply to the generation of the other: remaining on this example, the $4^{\text{th}}$ word will compute a weight after the dot product with the $3^{\text{rd}}$ word, but not the other way around.

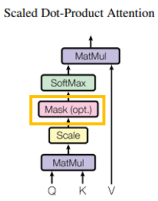
The second attention sub-block in the decoder involves as always the concept of query key value: this time, those don't share the same source. In particular, the output of the whole stack




**Mat Mul: Matrix multiplication**
**Attention Matrix**

| Dot Product | A | Planta | é | verde |
|---|---|---|---|---|
| A | 0.3 | 0.7 | 0.5 | 0.1 |
| Planta | 0.9 | 0.3 | 0.89 | 0.9 |
| é | 0.1 | 0.7 | 0.5 | 0.2 |
| verde | 0.5 | 0.92 | 0.28 | 0.6 |

(a)




| Dot Product | (A) La | (Planta) Pianta | (è) è | (verde) verde |
|---|---|---|---|---|
| (A) La | 0.3 | -inf | -inf | -inf |
| (Planta) Pianta | 0.77 | 0.3 | -inf | -inf |
| (é) è | 0.1 | 0.8 | 0.5 | -inf |
| Verde | 0.1 | 0.9 | 0.2 | 0.6 |

(b)

Figure 2.7: Example of computation of the attention matrix in the encoder a),and with the masked attention in b). Modified from [50].

of encoder is employed both as Query and Value, while the (after the normalization and residual connection) output of the first Attention module provides the Key. This means that for generating the next output, what has been already generated is used as a Key, a metric, in order to valuate, to give credit to what is it being treated as input: the encoder's results is query, and depending on the already generated context(the output treated as a key), parts of the encoder output (value) is highlighted or overshadow, to concur in generating the next output.

# Chapter 3

# Vision DNN for audio scene classification

The field of audio cross paths with the vision's one. This is due to the fact that the bridge that links the sound to a DNN, the mathematical representation that computer can understand, is often an image,meant to be representation of the original audio.This relationship holds especially true in Deep Learning, where the concept of images is deeply rooted in historical foundations,and there is wide range of models,each with its peculiarities.

## 3.1 Visual representation of sound

With the term **Spectrogram**, it is intended a visual representation aimed at showing the properties of a signal in terms of its spectrum of frequency. Spectrograms are tools deployed to represent graphically certain features of a signal, following a procedure whose foundation is the Fourier transform of the signal itself. They are used extensively in several fields of science, particularly in branches connected to sound (speech recognition, music, linguistics), with the common denominator being the analysis of a signal which varies with time.

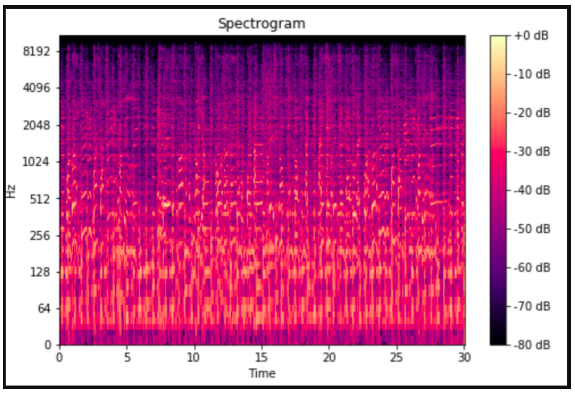



Figure 3.1: An example of a spectrogram.Extracted from [53]

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n} \tag{3.1}$$

$f$ : Frequency in Hertz (Hz)

$M(f)$ : Mel scale value

The groundwork for spectrograms was laid in the early 20th century with the advent of signal analysis techniques and the understanding of Fourier transforms. They were used for diagnostic and troubleshooting purposes for telegraph and telephone lines, for example.

Spectrograms can come in various graphical forms: waterfall plot, 3D plots, etc. The standard spectrogram layout is a graph with two geometric dimensions: time on the x-axis, frequency on the y-axis, and each point, with its intensity, represents some features within the grid defined by the two axes,so, a spectrogram is a one-channel image.

Essentially, a spectrogram represents the frequency content of a signal in the changing time. To understand what a spectrogram is telling us, with its brightness varying along the frequency and time axes, one must understand how it is constructed. As previously stated, the building block of a spectrogram is the Fourier Transform, specifically the Fast Fourier Transform (FFT).

This building block that is the Fourier Transform extract the frequency content of a sound, letting us discover the prominence of particular frequencies over others, the total absence of some frequencies. The question is: how is it possible to extract the time content of this audio? With the Fourier transform, we become aware of the frequency content and the magnitude, but not when these frequencies appear in the signal. The time aspect of an audio signal in a spectrogram is introduced by dividing the audio into overlapping segments. For each segment, the frequency content is extracted through the Fourier transform. Given a frequency range of interest ( the one limited by Nyquist frequency of the interested audio, for instance), and some measure frequencies that can be evenly spaced or follow a particular trend along this range, it is possible to proceed. The amplitude associated to the frequencies captured earlier with the FFT is indeed reported at these measure frequencies: number and spacing of those latter frequencies concur in defining the frequency resolution of the spectrogram.

This amplitude(its log, or its square) is the one defining the brightness of the points of the spectrogram: resuming, given a chunk of an audio(x axis), the magnitudes captured with the FFT associated to the measure frequencies are reported along the vertical axis of the plot. The spectrogram is the result of doing this operation for all the overlapping chunks of the interested sound, placing theme side by side. A windowing function(prior to the computation through the FT) like the Gaussian or Hamming window,shapes each chunk of audio to reduce edge effects and minimize spectral leakage.

The process presented above is only actually one among the equivalent techniques that enable to draw a spectrogram: other techniques involve employing filter banks for example.The first process described was given a name, the Short-time Fourier transform.

### 3.1.1 Spectrogram Resolution

The width of each chunk(and so, the width of the window function) determines the time resolution of the spectrogram, which in turn determines also the frequency resolution of the visual representation: in fact a wide window gives better frequency resolution but poor time resolution,while narrower one gives good time resolution but poor frequency resolution; a concept related to the uncertainty principle.Given the spacing of the measurement frequency and how the spectrogram

is built, a the end the spectrogram will have the same resolutions (in time and frequencies) across all of it. Beyond the compulsory trade-off between time and frequency resolution, there is a major drawback of having a fixed resolution. Different parts of an audio signal might require different focus . For example, transient events (like a drum hit) require high time resolution to be accurately captured, whereas harmonic content (like a sustained musical note) requires high frequency resolution to distinguish between close frequencies. A fixed resolution cannot adapt to these varying requirements, and this is why other technique that entail multi-resolutional analysis were preferred over the spectrograms.

Human perception of sound is highly non-linear. We are more adept at discerning pitch differences at lower frequencies compared to higher ones. Moreover, the interaction between audio intensity and pitch plays a crucial role in shaping our auditory experience, and the anatomical structure of our ears influences how sound is perceived and interpreted as well. The MEL scale establishes a relationship between the frequency of a tone and its perceived pitch. The Mel scale,depicted in figure 3.2, was developed to better align audio signal processing with the way humans perceive sound. The term "mel" comes from "melody," reflecting the scale's focus on how pitch is perceived musically. The scale was built involving human subjects, with experiments aimed at understanding how they perceive sounds. Through those studies, it was possible to come up with an empirical relationship that formulated a logarithmic relationship between frequency in Hertz and perceived pitch in mels 3.2.

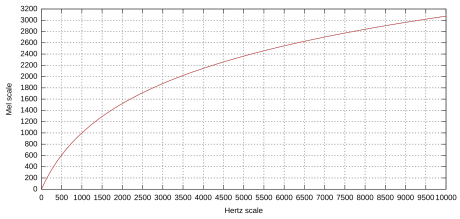



Figure 3.2: Mel Scale extracted from [55].

$$M(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{3.2}$$

$f$ : Frequency in Hertz (Hz)

$M(f)$ : Mel scale value

The pitch is the actual physical measurement of how many cycles of vibration occur per second. For example, the musical note C at a particular pitch is 65.41 Hz. How it is perceived is a different story:The difference between A2 (110 Hz) and a slightly higher note, say 120 Hz, is noticeable and perceived as a significant change in pitch.However, the difference between A4 (440 Hz) and a slightly higher note, say 450 Hz, is less noticeable even though the numerical change (10 Hz) is the same. So, the objective of the Mel Scale is to provide a perceptual scale of pitches or frequencies that reflects the way humans perceive sound: moreover, the implementation of this scale in the context of spectrogram helps overcoming the issue of equal resolution across the visual representation.

When in fact the measure frequencies or central frequencies in a spectrogram are spaced according to the MEL scale, our perception of pitch becomes less sensitive at higher frequencies. Central frequencies arranged according to the MEL scale are closer together at lower frequencies

and farther apart at higher frequencies, meaning that the frequency resolution at the bottom of the spectrogram is greater,since the space between bins is less.

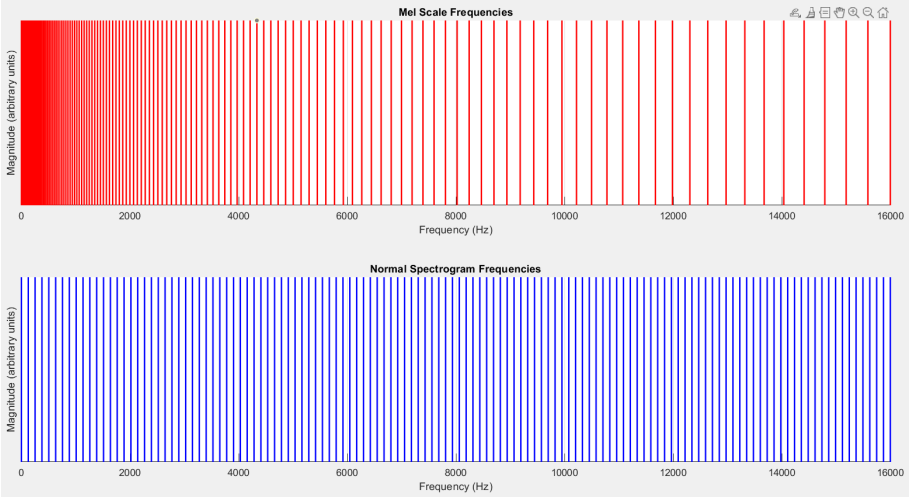


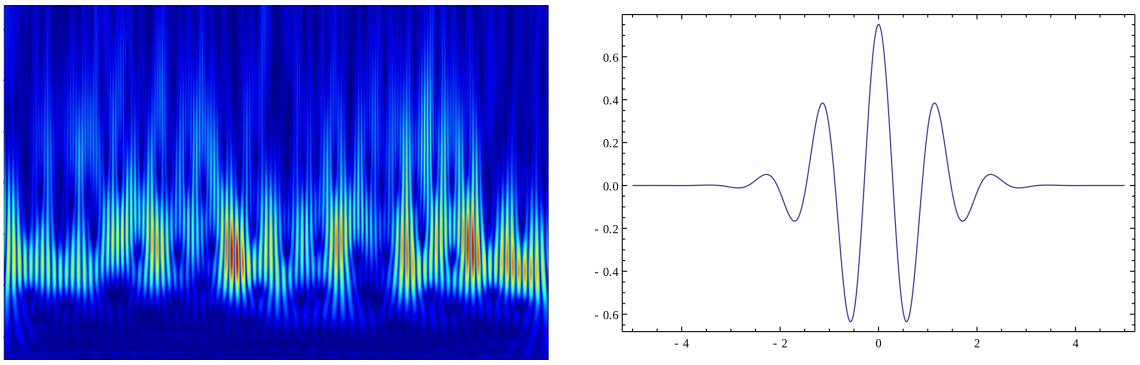

Figure 3.3: Distribution of measurement frequencies along the spectrogram for the Mel Scale Spectrogram and the general spectrogram

The process of building a Mel-Spectrogram involves other passages with the respect to the ones entailed with the Short Fast Fourier Transform: resuming the passages in for the "Normal" Spectrogram, after capturing the frequency contents throught the FT, a series of triangular filters are constructed. Each filter is centered at a specific Mel frequency and extends to adjacent frequencies. The shape of each triangular filter is determined by its center frequency and the bandwidth determined by the adjacent frequencies. These filters are usually overlapped to ensure smooth transitions and to capture the spectrum comprehensively. The overlap between adjacent filters is typically 50% . The energy within each triangular filter's frequency range is summed up, yielding a set of values that represent the energy distribution across the MEL frequency bands.

### 3.1.2 Scalograms

While the spectrogram, by using the FFT, automatically falls back into the category of spectral analysis techniques, the scalogram, recurring to the wavelet transform, falls back into the category of time analysis. Frequency is studied and displayed as well, but this time, time gains major prominence.

A scalogram is intended to depict the behavior of the frequency content of a sound over time, as in the case of spectrogram. Its usual representation is indeed similar to that of a spectrogram, with a time axis and a "pseudo" frequency axis. Even though the aim is similar and the axes are alike, a spectrogram and a scalogram of a sound representation will generally be different. By a practical point of view,the FFT is employed in the computation of the necessary coefficients of scalograms as well,but if the building block for spectrogram analysis was the FFT, the wavelet transform stands as the foundational building block for the scalogram. A wavelet is a mathematical function which describes a waveform localized both in time and frequency, imbued with other




(a) (b)

Figure 3.4: Mother Wavelets:row-wise: morlet wavelet,extracted from [56]

specific properties,like having zero mean.

$$\psi(t) = e^{-(t^2/2)} \cdot e^{i\omega_0 t}$$

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}} \psi\left(\frac{t-b}{a}\right)$$

Description of the general wavelet with parameters $a$ and $b$.

(3.3)

One of the primary distinctions in wavelet analysis is between discrete and continuous implementations, each serving different purposes and applications. The continuous wavelet transform provides a continuous, high-resolution representation of a signal. In contrast, the discrete wavelet transform (DWT) operates on discrete scales and translations, providing a more compact and efficient representation of the signal. From now on, we will focus on the continuous wavelet transform. An important aspect of scalograms analysis is the choice of the shape of the wavelet. There are in fact several kind of wavelets, each having its shape, and each of them determining differently other aspect of the wavelet analysis. In our thesis, we focused on the "Morlet" wavelet, the preferred choice in the continuos wavelet transform branch, since it provides an excellent trade-off between time and frequency localization, which is crucial for analyzing signals that vary in both domains.

It is important to stress out that the choice of the implementation(discrete or continuous) together with the choice of the mother wavelet,comply in defining the spacing between the scales, which are nonetheless pseudo-frequencies, having a similar role to what the measurement frequencies discussed earlier have for the spectrogram analysis.

$$W_x(a,b) = \langle x[n], \psi_{a,b}[n] \rangle = \sum_n x[n] \cdot \psi^*_{a,b}[n] \tag{3.4}$$

This is a major plus for scalograms: the fact that it provides multi-resolutional analysis is

intrinsic to the scalogram itself,whereas in the case of spectrogram was possible to implement recurring to the mel scale. The spacing promoted by this combination of design choice is good for us, since it is a logarithmic-type spacing that once again enhance the frequency resolution in the low frequency area. Again, in order to understand what a scalogram depicts, it is important to understand how it is built.Given a frequency range we are interested in, the wavelet is stretched or shrunken across this frequency range: when the scaling factor $a$ is small(higher frequency), the wavelet function becomes narrower in time duration and wider in frequency bandwidth: time resolution is enhanced, at the cost of a reduced reduction for frequency. Conversely, at higher scales (lower frequencies), the wavelet is stretched, resulting in a wider time duration and narrower frequency bandwidth.

This filtering operation promoted by the stretched/shrunken wavelet, is performed along all the audio: each sample participate in defining the wavelet coefficients. So the CWT transforms a 1D time series into 2D coefficients, ready to be spread across the scalogram.However, the computational cost of CWT is significantly higher compared to spectrogram computation. This is due to the continuous nature of wavelet scaling and the need to evaluate the wavelet across all time points and multiple scales. Each scale requires a separate wavelet analysis across the entire signal, leading to a more intensive computational process.

Spectrograms and Scalograms are both valid representation when dealing with audio. The choice between the two has to be made depending on the nature of the source. Scalogram enables to perform a multi-resolutional analysis depending on factors like the deployed wavelet and other parameters like the central frequency of the wavelet, offering another degree of freedom, enabling the capability to choose which range of frequencies prefer in term of resolution. Moreover, providing coefficients for each sample, it is able to provide more fined grained representations. Those benefit brought by the wavelet transform make the scalograms the preferred visual representation for non-stationary signals. Finally, spectrograms are build considering segments of audio: for tasks concerning event detection, spectrogram may smear out the transient event we are interested in.Scalograms,on the other hand excels in localizing transient event. Due to this properties, scalograms find large application in diverse fields,ranging from medical application like electroencephalography[25], to financial analysis[34].

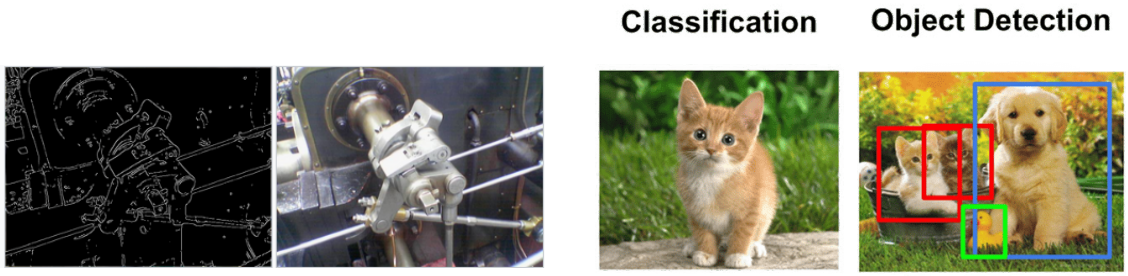
### 3.1.3 Computer Vision: Classification

Computer vision is a scientific disciple which entails processes for acquiring, processing, analyzing and understanding digital images, extracting features from them and taking decisions based on this: processes to be performed by a computer.

The roots of computer vision can be traced back to the 1950s and 1960s; and it is curious, that among the first approaches that were taken, there was a successful attempt of early artificial neural network, the Perceptron.

The period of 1970s and 1980 saw the development of algorithms aimed at detecting edges and extract features from images: significant contributions in this period are the Canny Edge Detector and the Hough Transform.

The next decade saw an incremented interest toward object recognition and scene understanding, recurring to Machine Learning:a notable early work is the paper by S. Geman and D. Geman, titled "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," published in 1984, which opened the implementation of Gaussian Mixture Model for segmentation. From that moment onward, machine learning became an integral part of computer vision: Support vector machine, the development of the API CUDA to leverage on the full potential of GPUs for the training of deep learning model, consacrated by the success of AlexNet, a deep convolutional neural network trained on GPUs, in the ImageNet competition, in which algorithms have to performed tasks associated with computer vision.




(a) (b)

Figure 3.5: a)Canny algorithm deployed to detect edges from an picture of a mechanical joint.Extracted from [54].Example of computer vision tasks.Extracted from [2]

Among the tasks embraced by computer vision there is Classification. In the context of deep Learning it refers to the process of categorizing data into predefined classes or categories using neural networks: this is the case of this thesis. Image classification has traditionally been dominated by CNNs due to their ability to efficiently learn and recognize patterns in visual data. The hierarchical feature extraction process in CNNs enables them to handle a wide variety of visual tasks by breaking down images into their constituent features and combining these features to form a comprehensive understanding of the image.

In recent years, the field has seen a significant shift with the introduction of Vision Transformers (ViTs), which offer a fundamentally different approach to image classification. Unlike CNNs, which focus on local features, ViTs process images globally by treating patches as sequences and applying the self-attention mechanism to capture relationships across the entire image.

### 3.1.4 CNNs

Convolutional Neural Networks (CNNs) have been the backbone of computer vision since their development by LeCun et al. in 1989 for handwritten digit recognition [26]. The modern resurgence of CNNs can be attributed to the success of AlexNet, a peculiar CNNs based architecture , which won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 and significantly outperformed other models in terms of accuracy [24].

CNNs are designed to exploit the spatial hierarchies in image data through the use of local receptive fields and weight sharing. The convolutional layers extract local features, which are then combined in subsequent layers to form more complex patterns. This hierarchical approach enables CNNs to efficiently capture spatial features such as edges, textures, and shapes, making them highly effective for image classification, object detection, and other vision tasks.

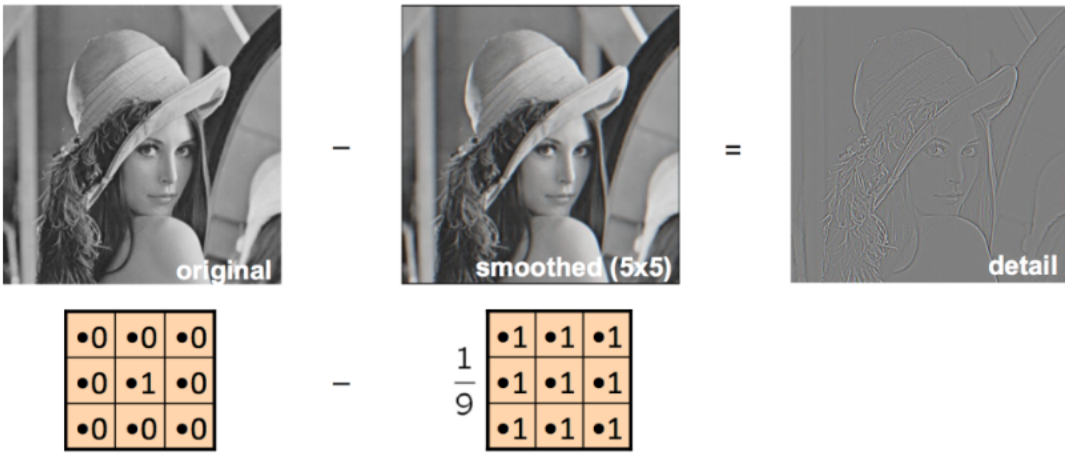



Figure 3.6: Stencil application to obtain details from an image;extracted from [60]

Over the years, various CNN-based architectures such as VGGNet[**?**], ResNet [16], and Inception [**?**] have been developed, each introducing innovations that improve the depth, performance, and efficiency of CNNs.

The operating principle of CNN is the base on the deployment of stencils,kernels,filters. The term "stencil" in the context of CNNs refers to the local region of an image that a filter (or kernel) covers during the convolution operation. Each filter is a small, fixed-size matrix that is systematically slid over the input image, computing dot products at each position to create a feature map. This process, known as convolution, enables the network to detect local patterns such as edges, textures, and more complex structures as the layers deepen.

Those stencils and filter are not new in the field of computer vision: they were already deployed in several Computer Vision algorithm prior to the introduction of CNNs. Those stencils or templates in traditional Computer Vision algorithms were manually crafted matrices or small grids of numbers. Each stencil represented a specific feature detector, such as detecting edges, corners, or textures, within an image. For example:

- **Edge Detection Stencil**: A 3x3 matrix that highlights areas of rapid intensity change, such as Sobel or Prewitt filters.
- **Corner Detection Stencil**: Typically based on local gradients, aiming to identify intersections of edges, such as Harris corner detector.
- **Texture Detection Stencil**: Patterns used to identify repeating textures or patterns in an image, like Gabor filters.

- **Blob Detection Stencil**: Used to identify regions of uniform intensity, often employed in image segmentation.

These stencils were manually designed based on expert knowledge of the desired feature characteristics. They were then convolved across the entire image to produce feature maps that represented the presence or absence of these features at different spatial locations.

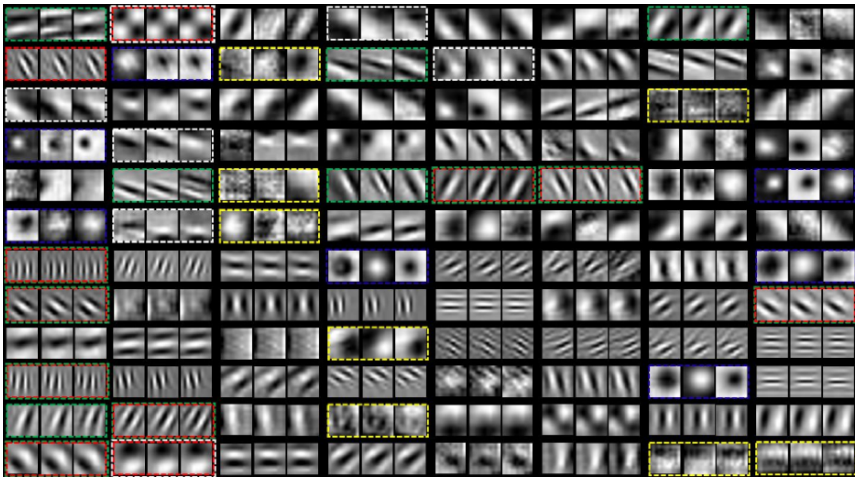


Figure 3.7: CNN filters from the first layer of the AlexNet network.Extracted from [29]

In contrast, CNNs automate the discovery of such stencils and kernels through a process of learning. Instead of relying on handcrafted filters, CNNs use multiple layers of learnable convolutional filters. These filters start with simple features (e.g., edges) in the initial layers and gradually learn more complex features (e.g., object parts or entire objects) in deeper layers, all without human intervention in the filter design process.

The use of stencils and weight sharing in CNNs significantly reduces the number of parameters compared to fully connected networks, which in turn lowers computational complexity and enhances model efficiency. This architecture allows CNNs to generalize well across different images, making them robust to variations in scale, orientation, and lighting conditions.

A typical full-fledged CNN architecture consists of several essential components:

- **Convolutional Layers**: These layers perform operations that extract features from the input data.
- **Activation Functions**: Applied after convolutional layers, these introduce non-linearity into the network.
- **Pooling (Subsampling) Layers**: These layers reduce the spatial dimensions of the feature maps.
- **Fully Connected (Dense) Layers**: These layers connect every neuron in one layer to every neuron in the next layer.

- **Normalization Layers**: Techniques like Batch Normalization standardize the inputs to a layer for each mini-batch.
- **Dropout Layers**: Used during training to randomly drop a proportion of neurons to prevent overfitting.
- **Flattening Layers**: Convert the 2D feature maps into a 1D vector before feeding into dense layers.
- **Output Layers**: The final layer responsible for producing the output prediction.

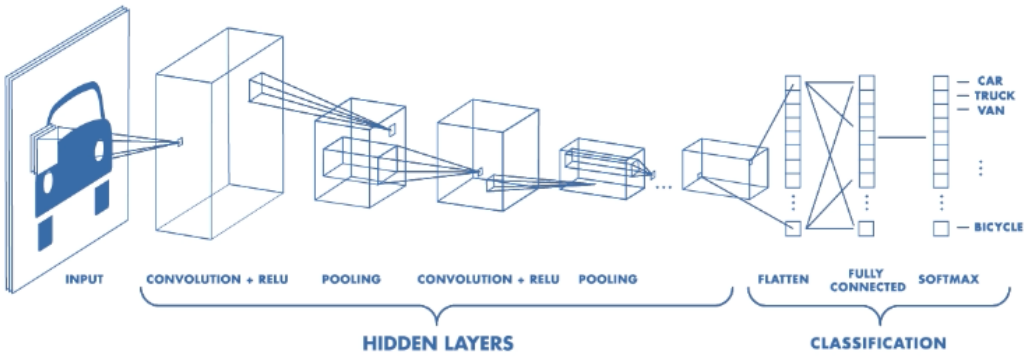



Figure 3.8: Convolutional Neural Network.Extracted from[49]

These components work together to enable CNNs to learn hierarchical representations of data, making them effective for tasks such as image classification, object detection, segmentation, and more in the field of computer vision.

The widespread deployment of CNN architectures across a multitude of computer vision tasks underscores their scalability and adaptability. Their extensive use in diverse applications justifies the existence and development of various versions and advancements over the years. Over the years, CNN architectures have indeed evolved, incorporating various innovations to improve performance, efficiency, and scalability. Some notable advancements include:

- **VGGNet:** Introduced by Simonyan and Zisserman in 2014, VGGNet demonstrated that deeper networks with small convolutional filters could achieve better performance by increasing the depth of the network [44].
- **ResNet:** He et al. introduced ResNet in 2015, which employs residual connections to alleviate the vanishing gradient problem and enable the training of very deep networks [16].
- **Inception:** Szegedy et al. developed the Inception architecture, which uses multi-scale processing within each layer to capture features at different resolutions [**?**].

Despite their successes, CNNs have limitations, particularly in capturing long-range dependencies due to their inherently local receptive fields. This has prompted researchers to explore alternative architectures, such as the Vision Transformer, which aims to address these limitations by providing a more global perspective on image data.

## 3.2 Vision Transformer

With the term ”Vision transformer” one refers to a category of transformer that are designed for dealing with images. In the specificity of this report, we are dealing about Vision transformers because even though in the context of acoustic scene recognition the input is sound (so, audio files), through the representation of the latter in specific graphs the classification problem is shifted from the sound domain to the image's domain. A really important consideration regarding vision transformer is that if the objective for which they are designed does not involve the generation of something, the decoder can be completely cut off from the structure; a possibility like that arises for example,when one deals(as in DCASE'case) with classification.

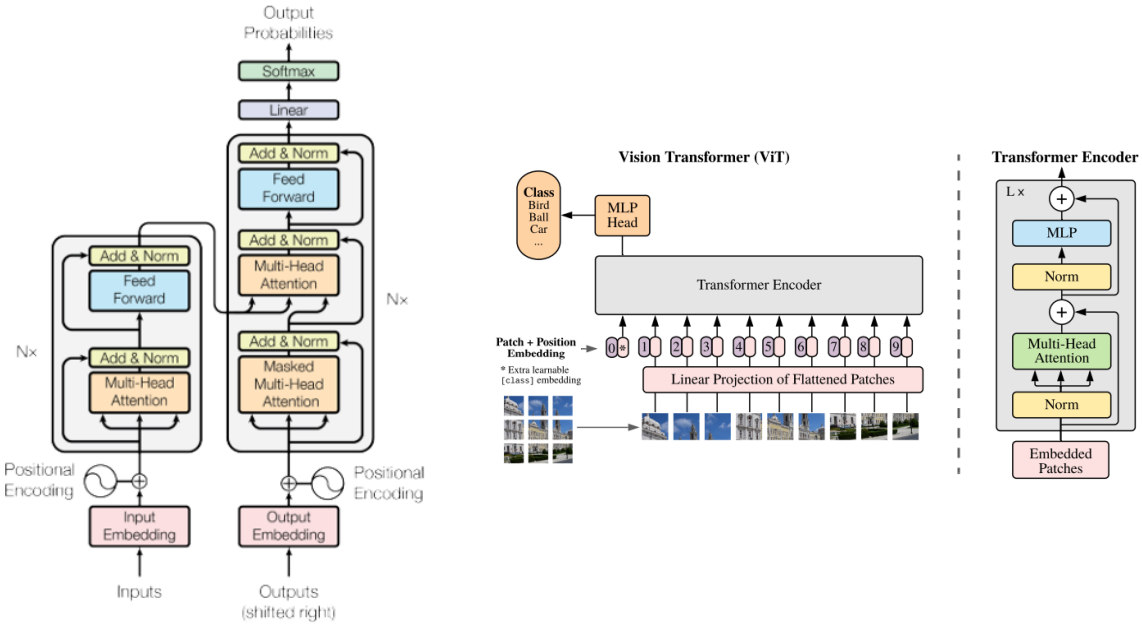



Figure 3.9: Original Transformer Architecture and Vision Transformer Architecture.Extracted from [10] and from [50].

The Vision Transformer (ViT), introduced by Dosovitskiy et al. in their seminal paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" in 2020, represents a groundbreaking advancement in computer vision. The ViT leverages the architecture and principles of the Transformer model, originally designed for natural language processing (NLP). Unlike traditional convolutional neural networks (CNNs), which rely on local receptive fields and

weight sharing to process visual information, the ViT applies the self-attention mechanism to image patches, treating them as a sequence akin to words in a sentence. This innovative approach allows the ViT to capture long-range dependencies and complex relationships within an image, thereby enhancing its ability to recognize intricate patterns and details. By dividing an image into a grid of non-overlapping patches, embedding each patch into a vector, and processing them with the Transformer's attention mechanisms, the ViT can effectively handle diverse visual tasks such as image classification, object detection, and segmentation. The introduction of ViT has demonstrated that Transformers, with sufficient training data and computational resources, can outperform or match the performance of state-of-the-art CNNs on various benchmarks, thus opening new horizons for applying Transformers in computer vision. With all this being said, it is clear that Vision Transformers and CNNs are foundational in modern computer vision.

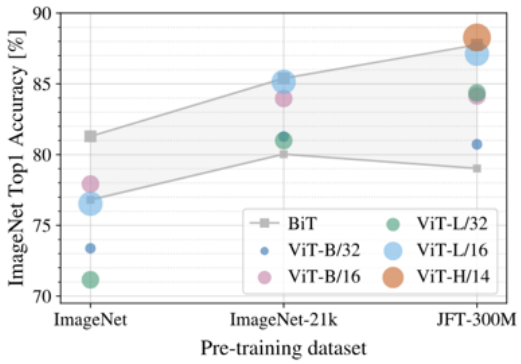
- **Vision Transformer:** With their recent introduction and attention-based mechanism, they have ushered in a new era in deep learning for computer vision, establishing themselves as the state-of-the-art under appropriate conditions

  **Convolutional Neural Networks :** With a legacy of accomplishments and a rich history, they remain at the forefront across various applications. The study on them has not stopped, as they continue to make significant contributions to advancements in computer vision. The concept of convolution and filters, so intrinsic to the understanding of images, make them fundamental tools for tasks such as feature extraction, pattern recognition, and image classification.

Both architectures have their strengths and are suited for different types of problems in computer vision.It is indeed useless to make comparison among the two,without taking in considerations other several factors into the equation.Among those factors there are certainly some figures of merit like the size of the model, the throughput, the robustness, the type of Data,the resources at disposal, the size of the Dataset, the complexity of the operations. Factors that tip the balance towards the CNNs, actually. In fact, ViT has been able to outsmart CNNs in several occasions, but at the cost of deploying a size,sometimes twice as the one of CNNs[10]. Vision Transformers are architectures prone to having a large number of parameters to accommodate their attention mechanisms, which currently limits their deployment on edge devices, where CNNs maintain their dominance.Another important issue to not underestimate is that ,again, ViTs have indeed outshone CNNs in several cases, but under the condition that the datasets were sufficiently large. Numerous papers have shown that ViTs struggle to match CNN models when datasets are not substantial enough.In several occasions ViT has shown their inherently Data-hunger[65]. Unite this to the intrinsic difficulties of training them[46], those make reasonable to still questioning nowadays the capabilities of ViT. Nevertheless, the promising results achieved until now, the fact that the model is recent and the fact that there is an intensive study to deploy it, make it worth the efforts.

| Network | #param. | image size | throughput (image/s) | ImNet top-1 | Real top-1 |
|---|---|---|---|---|---|
| Convnets | | | | | |
| ResNet-18 [21] | 12M | $224^2$ | 4458.4 | 69.8 | 77.3 |
| ResNet-50 [21] | 25M | $224^2$ | 1226.1 | 76.2 | 82.5 |
| ResNet-101 [21] | 45M | $224^2$ | 753.6 | 77.4 | 83.7 |
| ResNet-152 [21] | 60M | $224^2$ | 526.4 | 78.3 | 84.1 |
| ViT-B/16 [15] | 86M | $384^2$ | 85.9 | 77.9 | 83.6 |
| ViT-L/16 [15] | 307M | $384^2$ | 27.3 | 76.5 | 82.2 |
| DeiT-Ti | 5M | $224^2$ | 2536.5 | 72.2 | 80.1 |
| DeiT-S | 22M | $224^2$ | 940.4 | 79.8 | 85.7 |
| DeiT-B | 86M | $224^2$ | 292.3 | 81.8 | 86.7 |

(a)



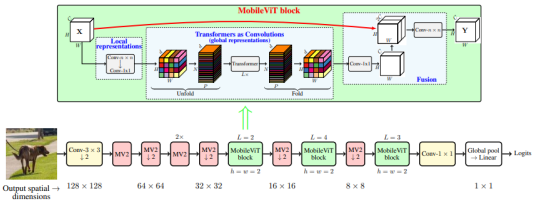

(b)

Figure 3.10: CNN-based architecture and ViT 's result in ImageNet.Extracted from [10].

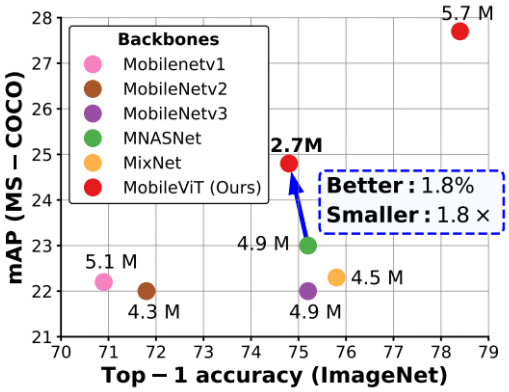
### 3.2.1 MobileViT

So, on one side there are the Convolutional Neural Networks: with their instrinsic spatial inductive biases, they are prone to learn representations with fewer parameters,making them the preferred choiche in edge devices. Their structure inherently focuses on capturing spatial hierarchies and relationship within local regions of an input image. On the other sides there are ViTs: they learn global representation,they lack the spatial inductive biases, and showed to achieve performance improvements, at the cost of model size and latency. Unlike CNNs, ViT shows substandard optimizability and are difficult to train; moreover when ViTs are downsized to match resource constraints of mobile devices , their performance don't match the achievements of light-weight ViT. Is it possible to combine the strenghts of the two architecture to build a new model for mobile vision tasks? A light-weight model, which retains athe global awareness of the ViT, as well as the spatial biases of the CNNs? MobileViT, introduced in 2021, is an architecture tailored to answer to the preceding questions: and its answer is yes. With 5-6 milions parameter, it outsmarted models of the calibre of MobileNet's family, CNNs based model designed for edge devices. MobileViT aims at extracting global and local information from the input, given a lower number of parameters in comparison with ViT. A smilar-MobileViT model have indeed been submitted to the competition [61].

The structure of MobileViT is composed of MobileNetV2[42] blocks (CNN based architecture designed to be effective for mobile and edge devices ) interpersed by MobileViT block: this peculiar blocks hosts the place where global representation feture extraction take place.It is indeed in the MobileViT block where the stacks of encoders of the MobileViT are found. Basically , given the feature map at the entrance of mobile vit block, this map having shape HxWxC is passed through convolutional layers that project into higher dimension , is unfolded into non overlapping patches, enters the stack of encoders, is folded back to the image-like layout. Other convolutional operations, as well as concatenation are performed. One thing that strike the eye is the "preparation" of the feature map before entering the stack of encoders , which resemble a lot the patch embedding that occurs into usual ViT What the author of MobileViT address is that, since the




(a)




(b)

Figure 3.11: a)MobileViT and focus on the MobileViT block.b)MobileViT metrics compared with other architecture designed for edge devices.Extracted from [31]

feature map is suddenly reorganized in the HxWxC shape typical of images after each MobileViT block, the sense of spatial sequence is preserved: the encoders in MobileViT don't need the positional encoding. Instead of adding explicit positional encodings, MobileViT leverages the fact that the patches are processed in a spatially coherent manner within each block. This means the spatial relationships are inherently preserved through the way patches are handled and reintegrated.

## 3.3 State of the Art

The interest in the discipline and the point of no return which is the Transformer with its abilities that can not be overlooked anymore, led to the development of efficacious Transformer based models, tailored for ASC. Those models tend to incorporate Transformer's concepts, such as the attention module ,due to the modularity that characterize it. Moreover, in those model Transformer is often side by side of CNNs based architecture: their synergy already proved to be winning.As already stated, DCASE arises as a guiding light in this context: many successful applications in those terms have passed through the ranking system of some challenge of DCASE,and validate themselves achieving state-of-the art results.

### 3.3.1 Ast and PaSST transformer

The synergy discussed above, during the first steps of the Transformer architecture in ASC's field, actually hides a lack of total trust on this architecture,thereby hindering its widespread adoption: the trend of building CNN-attention hybrid models was dominant. The first convolution-free, entirely based transformer architecture introduced in the sense of ASC was the AST(Audio Spectrogram Transformer)[13]: a Transformer-based architecture which deploy spectrograms to

tackle Audio Signal Processing tasks. The introductory paper of AST established the groundwork for future works, outlining the parameters for audio-spectrogram transformation, appropriate data augmentation techniques to be applied, and also some techniques that proved to be useful when dealing with Spectrograms.The AST transformer as it is introduced, is none other than a ViT_B;however, it is accompanied by a set of guidelines that position this model as foundational for addressing ASC with Transformers,among which there are:

- **Overlapping:** It promotes the effectiveness of overlapping patches: each patch is linked with neighboring patches to establish a prior expectation that there will be stronger correlations among adjacent tokens[62].

- **Need for pretraining:cross modality transfer learning** AST is meant to be evaluated on Audioset, a large corpus of audios. But audio dataset typically do not have a size able to cope with the Data-requirements of Transformer architectures. AST takes its weights from a ViT_B pretrained on ImageNet,a dataset of images: this cross-modality transfer (from image field to audio field) produced several improvements in comparison to a AST trained from scratch.

The AST, along with variant configurations such as ensembles of ASTs, has emerged as the state-of-the-art on AudioSet at the time of its introduction(2021), surpassing models that combine CNNs with limited attention-based modules. This underscores the capabilities of a fully fledged attention-based model in audio processing, especially in Audio Scene Classification (ASC).

PaSST is a peculiar structure of a ViT,introduced in ”Efficient Training of Audio Transformers with Patchout”(2021),designed to perform as best as possible in ASC:it is the natural evolution of an AST.The ”fuel” of this audio-based machine learning system are indeed Spectrograms. The possibility to establish a contact between each token during self attention comes at the cost of developing a complexity that rise quadratically with the input sequence length,both in terms of memory and computational load. In order to overcome this issue,a technique called PatchOut is employed: some patches,before being fed to the encoder,are completely removed: they are not masked for example with a uniform field of pixel,they are completely discarded. Thanks to the positional encoding the transformer would know which spots are missing. The concept is really similar to Dropout[45],and with the latter this method shares some benefit(generalization,reduced computational complexity). By the time of its introduction, PassT could effectively compete with state-of-the-art methods in audio scene classification (ASC), including CNNs and other transformer models. It highlights not only the achievement of high mean Average Precision (mAP) but also significant reductions in training time and memory usage when employing PatchOut.

PaSST is an advanced evolution of the Audio Spectrogram Transformer (AST). It builds upon the foundational concepts of AST, specifically focusing on improvements in handling spectrograms and leveraging overlapping techniques, and the necessity to use pretraining. The most important newelty introduced with PaSST are:

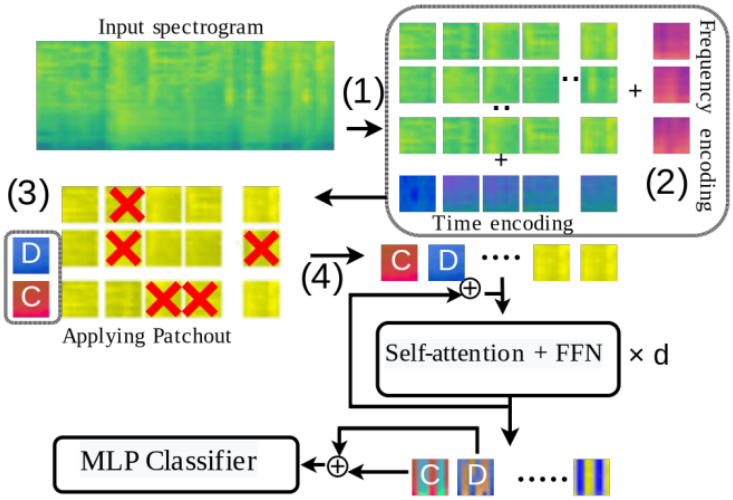



Figure 3.12: PaSST overall principle of action.Extracted from [23]

.

- **Patchout Technique** Patchout comes in two main variants: unstructured and structured, each offering different ways of dropping parts of the input sequence. In particular, the former involves dropping at random patches from the sequence, while the latter involves removing entire columns or/and rows, from the grid of patches.
- **Improved Positional Encoding:** PaSST disentangles positional encoding in both the frequency and time dimensions of spectrograms, facilitating the processing of audio files of varying lengths with greater ease.
- **Lower Computational Cost:** Optimizes computational efficiency, reducing training time and memory usage while maintaining high performance.
- **State-of-the-Art Performance:** Achieves superior results in audio classification, sound event detection, and other audio processing tasks, demonstrating significant improvements over traditional AST.

The techniques of PatchOut and the PaSST transformer are pivotal in this thesis, particularly for optimizing token sequence handling, especially with the computational demands of models like Large_ViT. Without these techniques, my NVIDIA GeForce RTX 3060 GPU would struggle to sustain the lengthy training sessions required. The importance of robust computational power becomes evident; GPUs such as the NVIDIA GeForce RTX 3080 or the AMD Radeon RX 6800 XT are more suited for such tasks. Their superior parallel processing capabilities and high memory bandwidth significantly reduce training times and enable efficient handling of complex models.

### 3.3.2 CP-JKU TEAM

As previously mentioned, the competitive environment makes implementing a Transformer-based architecture quite challenging. However, a groundbreaking achievement by two consecutive-year

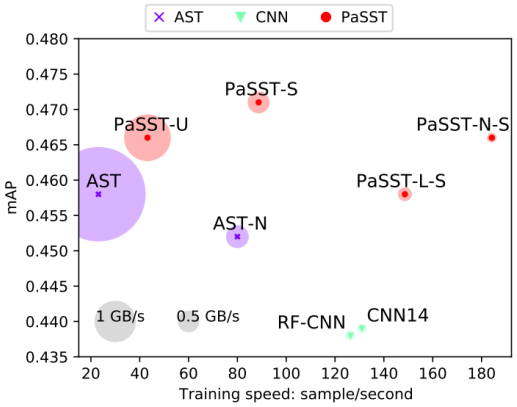



Figure 3.13: PaSST comparison with other ASTs and CNN-based architectures.

winners (2022,2023) decisively disproved this notion. There was no trick: Transformer' scales are not suited for the competition size constraints. But a transformer was indeed inserted in the training pipeline that brought a model to submit and win. This was thanks to the applications of a Knowledge Distillation framework, a compression technique in which a bigger and expert model is meant to train a smaller one. But as CNNs architecture proved to be good teachers for ViTs[48], it is not unlikely that ViTs can be good teachers for CNNs, if the logic is that each architecture compensates for the shortcoming of the other.This was the idea that crowned CP-JKU( Institute of Computational Perception of Johannes Kepler Universiy) as the two times consecutive winner of DCASE Task1.

Following this belief, the better results achieved during the competitions of 2022 and 2023 entailed a distillation framework in which the student (the submitted model) consisted of a CNNs based architecture able to fit into the competition's constraints,whereas the teacher consisted of either a single PaSST pretrained on Audioset,or an Ensemble of 6 pretrained PaSST(2022), or a single PaSST pretrained on Audioset,or an Ensemble of 6 pretrained PaSST and 6 convnets for a total of 12 models (2023). Results Each individual PaSST that comprises the ensemble is trained following a different configuration of parameters for the Data augmentation techniques applied. Moreover,each deployed PaSST, based on ViT_B/16, account for 85 M parameters, the same as the Large_ViT model studied in this thesis.Large_ViT takes is partially initialized with the weights coming from one of the PaSST mentioned above.

### 3.3.3 Attention Maps

Interesting and necessary tools to monitor the model's learning process as well as for understanding its way of thought are visualizations of the attention of the Transformer. Attention maps alone can already tell us several things about the transformer: its attention patterns, which reveal the importance each token reserve to each other token to the others in the sequence, and its the contextual understanding, meaning how the transformer subdivide its attention across the layers and the heads, how it specializes each of its head.

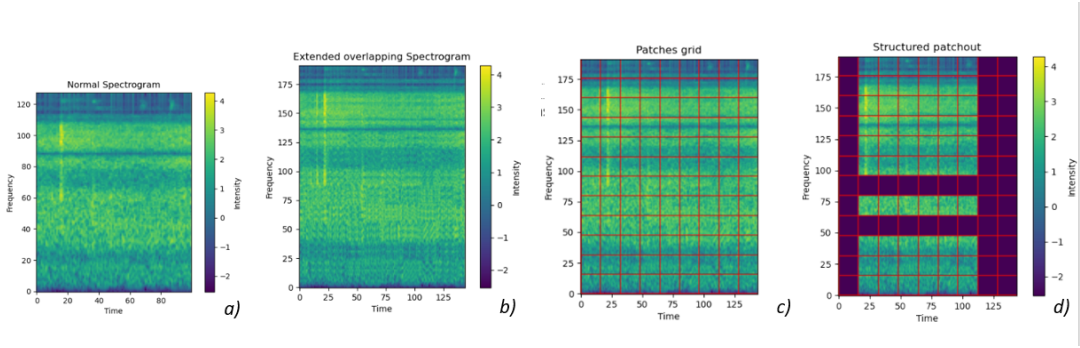



Figure 3.14: a)original spectrogram,b)spectrogram extended by the effect of overlapping,c)patch division,d)structured patchout along both frequency and time dimension applied.

| *PaSST Settings for downstream training* | **Accuracy** | **Log Loss** |
|---|---|---|
| $f = 0, p = 0.0$, *Mixup* ($\alpha = 0.3$) | 0.596 | 1.304 |
| $f = 4, p = 0.0$, *Mixup* ($\alpha = 0.3$) | 0.604 | 1.119 |
| $f = 8, p = 0.0$, *Mixup* ($\alpha = 0.3$) | 0.600 | 1.063 |
| $f = 8, \alpha = 0.8, p = 0.5$ | 0.602 | 1.064 |
| $f = 6, \alpha = 0.8, p = 0.5$ | 0.612 | 1.050 |
| $f = 4, \alpha = 0.8, p = 0.5$ | 0.612 | 1.077 |
| $f = 2, \alpha = 0.8, p = 0.5$ | 0.606 | 1.123 |
| $f = 6, \alpha = 0.8, p = 0.3$ | 0.612 | 1.054 |
| $f = 6, \alpha = 0.8, p = 0.7$ | 0.603 | 1.066 |
| $f = 6, \alpha = 0.6, p = 0.5$ | 0.611 | 1.050 |
| $f = 6, \alpha = 0.4, p = 0.5$ | 0.611 | **1.044** |
| PaSST_ENSEMBLE | 0.627 | 1.001 |

(a)

| *KD with varying temperatures* | **Accuracy** | **Log Loss** |
|---|---|---|
| **Baseline** | 0.524 | 1.418 |
| **T=1** | 0.541 | 1.216 |
| **T=3** | **0.567** | **1.146** |
| **T=5** | 0.559 | 1.183 |
| **T=8** | 0.554 | 1.219 |
| **T=12** | 0.551 | 1.265 |

| *KD with varying* $\lambda$ | **Accuracy** | **Log Loss** |
|---|---|---|
| $\lambda$**=10** | 0.555 | 1.231 |
| $\lambda$**=30** | 0.563 | 1.175 |
| $\lambda$**=50** | **0.577** | **1.138** |
| $\lambda$**=100** | 0.565 | 1.144 |
| $\lambda$**=200** | 0.562 | 1.160 |

(b)

Figure 3.15: CP-JKU's results from DCASE task1,2022. Extracted from [43].

By "visualizing the attention" one refers to the procedure of comparing the weights in the attention matrix, spatially locating them on a grid meant to recall the image that activate those weights.This usually translate in building heat maps that can be overlaid on the image, with the intensity decided by the magnitude of the corresponding weight in the attention matrix. The latter is a square matrix whose values are computed through the cosine similarities of query and key values. There is not a single way to create those heat maps: which weights are chosen?The attention matrix has a number of values equal to the square of the length of the patches/tokens of the input,plus one or more row and columns for each added token like the class token and the distillation token.Moreover, there are several attention matrix in each Attention block, one for each head, and this pattern repeat for all layers. One way to produce this attention representation is recurring to each head separately(without performing averages among them for example) [14]. For each head, in the case of classification , the matrix row related to the class token is extracted: it represent how the class token(query) attends to all the other tokens in the token(keys),how much attention is being paid to the patches of the sequence.

If instead one prefer to adopt a procedure that entails an overall visualization of the attention across all layers and heads in one image, the visualization techniques proposed by [1] proved to be effective.Attention Rollout aggregates attention scores across multiple layers of the Transformer,

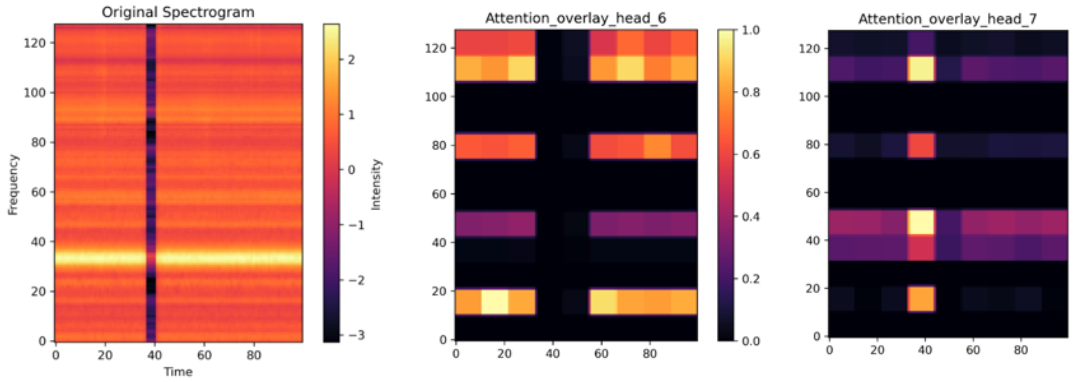



Figure 3.16: Attention heatmaps with Patchout: the sixth head ignores the time masked portion of the spectrogram, while the seventh focus on it.

providing to visualize how information flows from input token to the last layer.It is defined by the recursive process, in which $A$ is the average attention matrix across the heads; $l$ indicates the layer:

$$\begin{aligned} A^{l}_{\text{rollout}} &= \left(A^{l}+I\right)A^{l-1}_{\text{rollout}}, \quad \text{for } l>1 \\ A^{1}_{\text{rollout}} &= A^{1}+I \end{aligned} \tag{3.5}$$

Once again, since the obtained shape is that one of an attention matrix, usually the first row compelling to the class token is chosen for the heat maps values. Whenever Patchout and or Overlapping were applied, countermeasure were taken:in the case of overlapping, since its effect is like presenting to the model an image whose sizes are augmented with respect to the original one, the image is reshaped to resemble the original size.When Patchout is applied, the heat map is reconstructed assigning correctly the attention weights to the associated position in the image: the dropped patches are depicted as black ones in the resulting image.

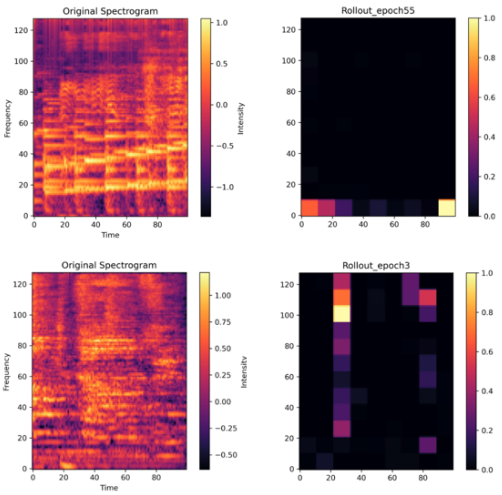



Figure 3.17: Attention rollout representations across the epoch.

# Chapter 4

# Materials and Methods

## 4.1 Data Augmentation Introduction

One significant challenge identified in the seminal Vision Transformer (ViT) paper [10] is that transformers do not generalize well when trained on an insufficient amount of data. This limitation is especially pronounced due to the inherently weaker inductive bias of transformers, which, unlike Convolutional Neural Networks , lack properties like translational equivariance and invariance. This means that ViTs may require larger and more diverse datasets to achieve robust generalization and learn the properties that CNNs have by default. An exemplary study, the Data-efficient Image Transformer (DeiT)[48], has demonstrated that state-of-the-art results can be achieved by employing techniques such as knowledge distillation and taking care of the data augmentation, rather than relying on massive datasets. The DeiT approach highlights the critical role of innovative methods that compensate for the lack of extensive data, thereby enhancing the performance of vision transformers even with limited data resources. Again, in other papers such as "How to Train Your ViT? Data, Augmentation, and Regularization in Vision Transformers,"[46] it is addressed that the absence of a large dataset can be compensated for by adopting an effective data augmentation scheme, which, if implemented correctly, can be equivalent to increasing the dataset size tenfold. This demonstrates that with robust data augmentation strategies, it is possible to achieve performance gains comparable to those obtained with much larger datasets.

In our context, we are constrained by the size of the training dataset, which consists of the DCASE task 1 training set with fewer than 100,000 unique samples and a small subset of the Audioset dataset,as it will be explained later in 4.2.3. The limited data availability necessitates the application of extensive data augmentation and regularization techniques to boost the model's performance and generalization capability. The unique context provided by the competition, where audio is transformed into images, necessitates that we pay attention to data augmentation techniques not only in the field of audio but also in the field of vision. This dual focus ensures that we can leverage the strengths of both domains to improve the overall quality and effectiveness of the training data. Therefore, in this chapter, we delve into various data augmentation strategies and regularization techniques employed to mitigate the data insufficiency challenge acting on the

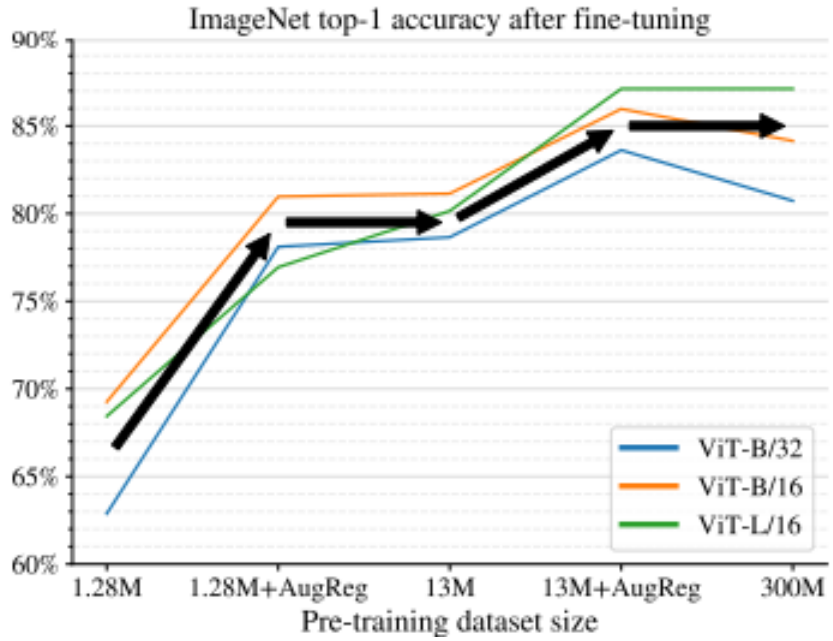



Figure 4.1: Improvement of ViT models thanks to data augmentation and/or a larger dataset. [46]

input when it is an audio, and on the input when it is an image.

### 4.1.1 Offline data augmentation techniques

With offline data augmentation, we refer to techniques that are applied to the dataset prior to the training phase. This involves generating augmented versions of the data which are then saved and used as part of the training dataset. In contrast, online data augmentation refers to the transformations applied dynamically during the training process. These augmentations are performed on-the-fly, generating new data variations in each training batch.

Among the data augmentation techniques employed there is **time shifting**, which entails moving the audio signal slightly forward or backward in time. This is particularly beneficial in scenarios where the timing of audio events varies. For instance, in a public square, the sounds of people talking, footsteps, and background activities might occur at different times in each recording. By shifting the timing of these events, the model becomes more adaptable to variations in when specific sounds occur, enhancing its ability to correctly identify the scene. Moreover it seems a harmless way to expand the dataset. As some portion end up out of the established boundary, Temporal shifting can be performed in two ways: discarding the samples that roll beyond the first or last position, or reintroducing the samples that roll beyond the first or last position are re-introduced at the last or first. The major drawback of employing this strategy is the production of artificial discontinuities,abrupt transitions at the points where the audio loops. Anyway, given that the goal is not to locate a precise event in time, the benefits overcome the drawbacks.

**Pitch shifting** is a common and effective data augmentation technique used in audio processing, a method involving altering the pitch of an audio signal without changing its duration. While

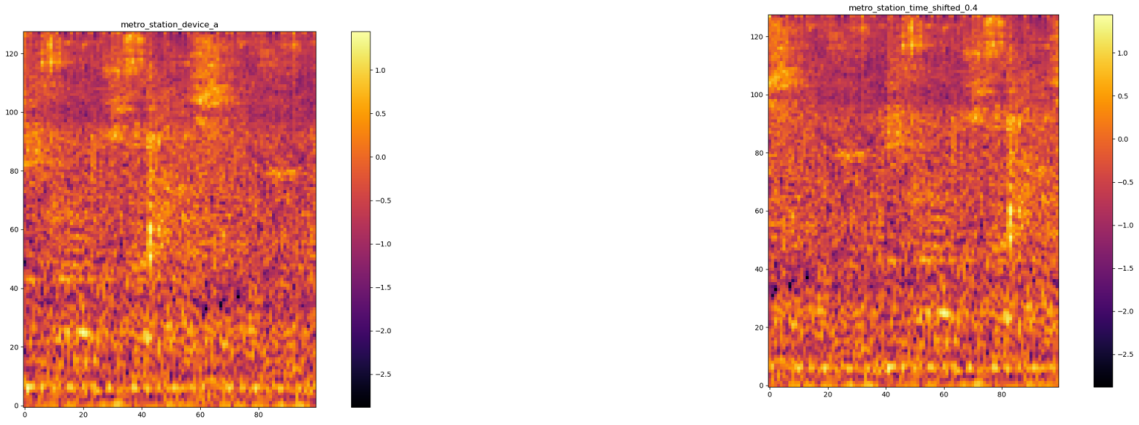



Figure 4.2: Time shifted spectrograms

pitch is a perceptual attribute, it can be estimated by analyzing the fundamental frequency of the audio signal using various algorithms. By adjusting the pitch, it is possible to create variations of the original audio sample, thereby increasing the diversity of the dataset and helping the model generalize better.

Musicians and producers often use pitch shifting to change the key of a song to better fit a vocalist's range or to create harmonies. For example, if a song is recorded in a key that is too high for the singer, the entire track can be pitch-shifted down by a few semitones, making it easier for the singer to perform. Additionally, pitch shifting can be used to create harmonies by duplicating a vocal track and shifting the pitch of the duplicate up or down, thereby generating a harmony that complements the original melody.

Sounds like footsteps, door slams, or bird songs can have slight variations in pitch depending on the context. By pitch-shifting the training samples, the model can learn to recognize these sounds more effectively, regardless of minor pitch differences. This improves the generalization capability of the model, making it more accurate in real-world applications where pitch variations are common.

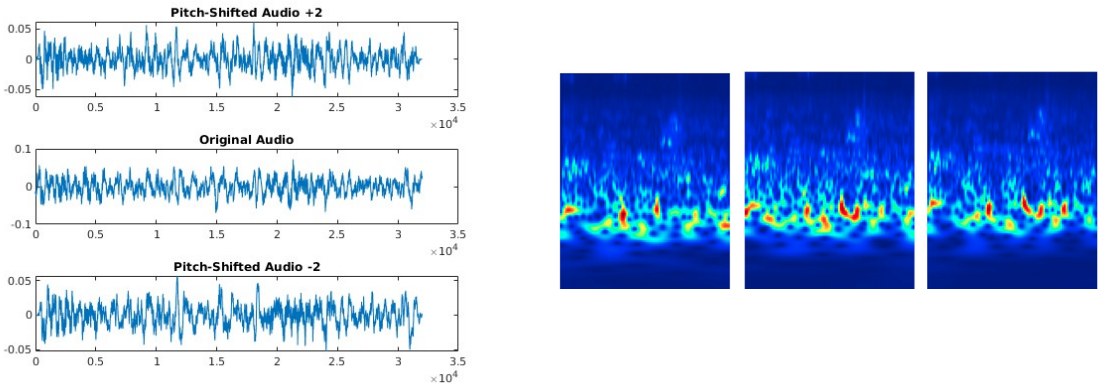



Figure 4.3: Pitch-Shifted audio and effects on its scalogram representation.

Another data augmentation technique regarding audio is **Time stretching**, which adjusts the speed of the audio while preserving the pitch. This can either enlarge (stretch) or shrink (compress) the audio's temporal duration. Enlarging the audio means playing it more slowly: the length of the audio is increased,but the exceeding part is removed. Time shrinking on the other hand involves playing the audio faster. In environments such as a metro station or on a tram, sounds can change speed due to varying speeds of vehicles or crowd movements. Stretching the audio to be slightly faster or slower helps the model learn to identify these scenes despite differences in the speed of sound events, making it more resilient to temporal variations. It is interesting to address the effect that this kind of data augmentation have on the associated visual representation like the one in spectrogram;in particular

- **Audio Stretching:** By elonging the length of the audio, spectral components are spread out over time. As a result, the sharpness and the definition(the magnitude) of spectral peaks is reduced, since the spreading promotes wider distribution around each peak , leading to a less precise representation of individual frequencies. As a final results, time resolution is also influenced since rapid changes or transients in the audio signal may appear more blurred or spread out in time, making it harder to distinguish quick events or percussive sounds.
- **Audio Shrinking(Compression):** Its effects on the visual representation are the opposite of the Stretching case: time compression shortens the duration of the audio signal, which compresses the spectral components: this can lead to overlapping or clustering of frequency components,resulting in spectrograms that may show increased density or concentration of energy in certain frequency bands.Rapid changes or transient events may appear more pronounced or distinct, making it easier to analyze fast-paced changes in the audio signal.

Those drawbacks can be limited recurring to low compression/stretching factor, like we did; in this case in fact the benefits hold. Additionally, **time masking** is applied, which involves randomly

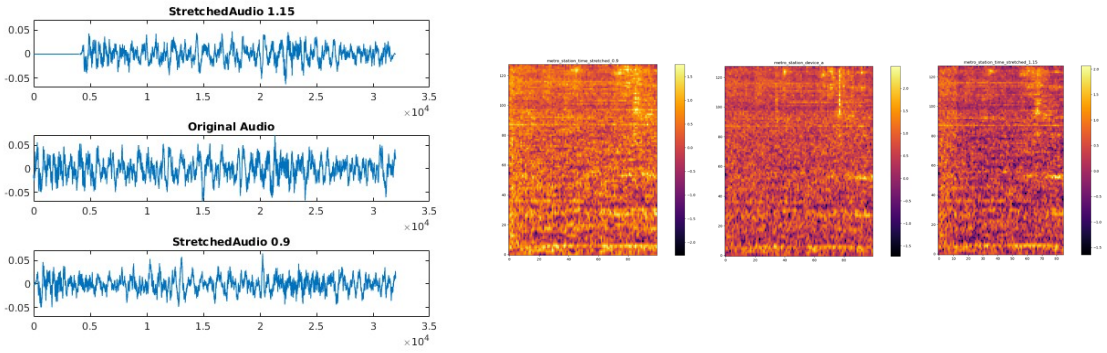



Figure 4.4: Effects of time stretching on audio and corresponding spectrogram representations

covering parts of the audio: either with silence (zeroing the values) or by filling the interested area with the medium value enclosed by the area. By exposing the model to audio with missing segments, time masking helps the model become more resilient to occlusions and interruptions;

moreover masking prevent the model from relying too heavily on specific temporal patterns: the model is encouraged to learn useful information from the remaining unmasked segments. Also, the ability to extract features from a non conitnuos sequence in enhanced in this way. And as stated for the other techniques, variability in the training data is introduced As always, some drawback arise from the implementation of this technique:possible masking of relevant parts,creation of artificial boundaries between masked and unmasked segments. Nevertheless, the nature of the ten acoustic scenes make the drawback less heavy to handle: we are not dealing with phoneme recognition, or event time recongnition.

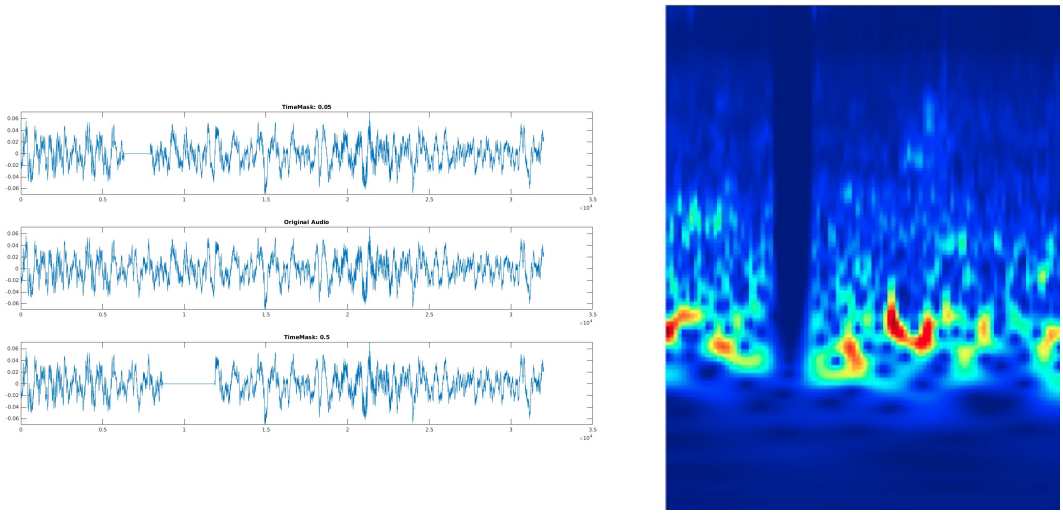



Figure 4.5: Time masking with different windows and corresponding scalogram.

As already stated, among the main challenges brought by the task is the fact that the audios are recorded with different devices: Training a model on audio signals recorded with a small number of different microphones can make generalization to unseen devices difficult. To address this issue, two main strategies are typically employed: either by suppressing device-specific characteristics of the audio signal, or by enhancing device diversity within the training set through techniques such as device balancing or augmentation. The **Device-Impulse-Response (DIR)** [33]approach belongs to the second category, focusing on increasing the diversity of devices in the training set to ensure better generalization across different recording conditions. In order to create the audio recorded with simulated devices in DCASE training and evaluation sets, some audios recorded with device *a* where convoluted with the impulse response of different recording device: each impulse response giving birth to a "simulated device". As a matter of fact, the latter is a really common practice in audio design Convolution involving two sounds in fact simulates the effect of combining those two audio signals in one. This process entails the mathematical operation of correlating the spectra of the signals, often resulting in a new audio signal that resembles the characteristics of both original inputs. This technique is widely used to apply effects such as reverb, echo, and spatialization to audio recordings. When audio signals are convolved, the resulting sound can simulate the acoustics of different environments or manipulate the spatial characteris-

tics of a recording. For instance, convolving a recorded impulse response of a environment with an audio signal can make it seem as though the audio was recorded in that particular environment.

Convolution in the time domain involves integrating the product of the input $x(\tau)$ and the impulse response $h(t-\tau)$ over all time $\tau$, producing the output $y(t)$ 4.1. In the frequency domain, this operation corresponds to multiplying the Fourier transforms $X(f)$ and $H(f)$ of $x(t)$ and $h(t)$ 4.2, respectively. Impulse response of a dynamic system describes the output of the latter when presented with a very short, ideally instantaneous input signal known as an impulse. It can provide comprehensive information about the system itself.Simulation of acoustic environments,but also, simulation of recording devices: this convolutional practice regard both the cases:

- **Impulse Response of a recording device:**This describes how a microphone or other recording equipment responds to an impulse signal. By convolving this impulse response with an audio signal, it is possible to simulate how that specific device would capture and reproduce that audio signal. .
- **Impulse Response of an environment:**Describes how sound behaves within a physical space, including reflections, reverberation, and frequency response characteristics: the response have to be long enough to capture those phenomena . By convolving an audio signal with the impulse response of an environment, it is possible to simulate how the sound would be altered by that environment. This is used in applications like virtual acoustics, where different room or concert hall simulations can be applied to recordings or live sound to change the perceived spatial qualities.

**Convolution in Time Domain:**

$$y(t) = (x * h)(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)\,d\tau \tag{4.1}$$

**Multiplication in Frequency Domain:**

$$\mathcal{F}\{x * h\}(f) = X(f) \cdot H(f) \tag{4.2}$$

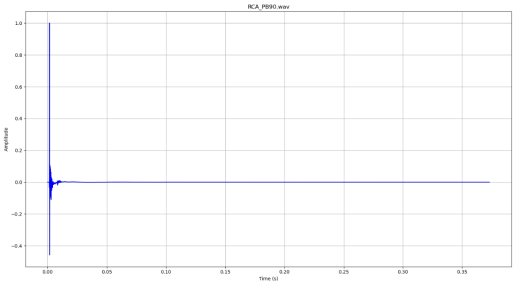
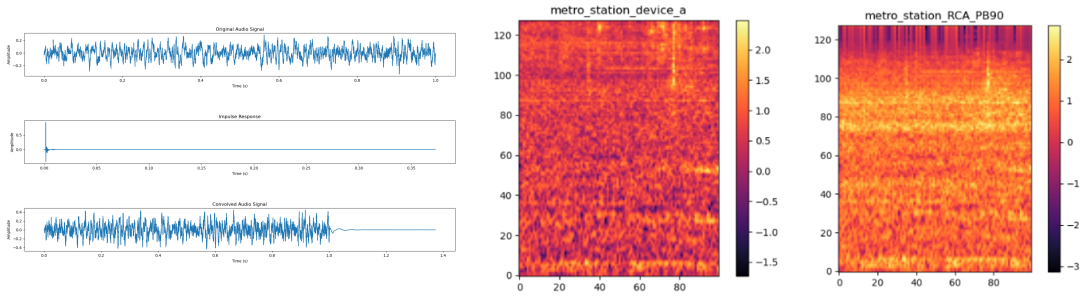


Figure 4.6: Impulse Response of a microphone taken from [47]

This practice leverages the principles of signal processing where convolution integrates the spectral components of two signals, blending their characteristics to create a modified audio output. In essence, convolution in audio engineering facilitates the enhancement, modification, or simulation of acoustic environments and spatial effects.

DIR data augmentation involves augmenting the training set by creating audios following the approach related to the impulse response of devices case: taken an audio, it is possible to generate another audio by convolution of the original audio with the impulse response of another recording devices. The more impulse response at disposal, the more “simulated device” are introduced in the training set. In this way the model is less encouraged to focus on the feature generated by the




(a) (b)

Figure 4.7: Audio convolution with an impulse response of a microphone, and corresponding effect on the spectrograms.

device , which are inevitably embedded in the visual representation of the audio. As the competition is held annually, it undergoes regular updates to stay aligned with the latest advancements in the field.One of the most important update was made in 2022, regarding the Dataset: a shift from "TAU Urban Acoustic Scenes 2020 Mobile, Development dataset" to TAU Urban Acoustic Scenes 2022 Mobile, Development dataset. The major and only difference between the two datasets,is that the former contains 10 long audio data,while the second one contains 1 second long audio data,extracted by the previous version: from each audio in the old dataset configuration, 10 non overlapping chunks were extracted.Competition until 2021 foreseen inference over 10 second long audio; as already stated,we are dealing with one second long audio, since we are deploying the 2022 update dataset.
Since the metadata provide an identification for each audio snippet, which enable us to reconstruct each 10 second long original audio, many teams deployed the following data augmentation strategy:

- **1) Reassembling of the original 10 second long audios**
- **2) Divide again the audio in 10 segments ,but after applying a time shift:** For instance, instead of starting dividing the audio from second 0, division process is started from second 0.5. The resulting audio chunk cover a time range that is already covered by the first two chunks in the original division, but in only one audio.
- **3) Increase Dataset Size :** Incorporate the outcomes of the aforementioned processes into the existing 1-second audio dataset, effectively doubling its size.

Until now we have talked about data augmentation technique applied to the audio's waveform. A common technique for data augmentation in the field of images instead,is masking some area of the image itself. **Frequency masking** involves covering specific(randomly chosen) frequency bands of a spectrogram with a mask, essentially occluding certain frequency ranges while leaving others intact. This technique introduces a controlled form of data variation that compels the model to learn to classify scenes even when key frequency information is missing or altered. In

acoustic scene classification that entails visual representations like Spectrograms and Scalograms, frequency masking is a pivotal technique for enhancing model robustness and generalization. By obscuring specific frequency bands in the data, this method trains the model to identify key features and patterns despite missing or altered frequency information. Frequency masking forces the model to learn from incomplete data, ensuring it does not rely on specific frequency details but instead focuses on the broader acoustic signatures of different scenes. This enhances its ability to classify environments accurately, even when parts of the audio spectrum are obscured or noisy. Furthermore, introducing frequency masks creates a diverse set of training examples, which helps the model generalize better across different recordings of the same scene. By learning to handle variations in the frequency domain, the model becomes more adept at recognizing a wide range of acoustic environments, from the constant hum of an airport to the dynamic sounds of a busy street or urban park.


Figure 4.8: Frequency masking of a spectrogram and a scalogram,averaging the interested zone

### 4.1.2 Online Data Augmentation and regularization techniques

#### 4.1.2.1 Mixup

Large,deep neural networks exhibit undesirable behaviours such as memorization, or sensitivity to adversarial examples: **mixup**[63] is a learning technique in which the model is trained on convex combinations of pairs of examples and their labels. The theoretical foundation of this action resides in the differences between Empirical Risk Minimization and Vicinal risk minimization approach in the training of a machine. The authors of the paper, address that the former is not suitable: ERM allows a machine to memorize the training data even in the presence of strong regularization; this leads to drastical drop when the machine is evaluated on examples just outside the training distribution.

$$R_\delta(f) = \int \ell(f(x),y)\, dP_\delta(x,y) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i),y_i). \tag{4.3}$$

where:

- $R_\delta(f)$: Represents the expected risk under the empirical distribution $P_\delta$.
- $\ell(f(x),y)$: Denotes the loss function measuring the discrepancy between the predicted value $f(x)$ and the true label $y$.
- $dP_\delta(x,y)$: Stands for the measure under the empirical distribution $P_\delta$, approximating the true distribution $P$ using training data.
- $\frac{1}{n}\sum_{i=1}^{n}\ell(f(x_i),y_i)$: Represents the empirical risk, computed as the average loss over the training examples $(x_i,y_i)$.

This principle translates into minimising this loss function:when considering models with a number of parameters comparable to $n$ this loss function can be easily minimized by memorizing the samples. Vicinal risk minimization is an approach in machine learning which aims to improve the generalization performance of models by incorporating information from nearby points in the feature space around each training example, pointing out that following the Empirical Risk Minimization's path in scenarios(when the dataset is limited, for example) where the training data not fully represent the underlying distribution from which the data is drawn may not suffice. Vicinal risk minimization proposes to draw virtual examples from the vicinity distribution of the training examples to enlarge the support coming from having more samples to train on.First of all, it is possible to express an approximation of the distribution of the data P, $P_v$, 4.4 recurring to the a vicinity distribution $\nu$ that measures the probability of finding the virtual feature-target pair (~x, y~) in the vicinity of the training feature-target pair (xi, yi) . Those virtual features are potentially generated or sampled based on the neighborhood or vicinity distribution $\nu(\tilde{x},\tilde{y} \mid x_i,y_i)$. This distribution characterizes the likelihood of observing virtual feature-target pairs $(\tilde{x},\tilde{y})$ around each training example $(x_i,y_i)$. By sampling from $\nu$, hypothetical instances $(\tilde{x},\tilde{y})$ that mimic variations or perturbations around $(x_i,y_i)$ are created.

$$P_\nu(\tilde{x},\tilde{y}) = \frac{1}{n}\sum_{i=1}^{n}\nu(\tilde{x},\tilde{y}|x_i,y_i) \tag{4.4}$$

By assuming a type for the vicinity distribution(like the Gaussian), it is possible to build a dataset $D_\nu := \{(\tilde{x}_i,\tilde{y}_i)$ whose distribution is represented by $P_v$: Empirical Risk Minimization on this new dataset would be more effective since it enhances the training process with additional data points that reflect the presumed distribution around each observed sample. This approach can improve the model's ability to generalize by providing a more comprehensive representation of the underlying data distribution, thereby potentially reducing overfitting and enhancing predictive performance on unseen data.

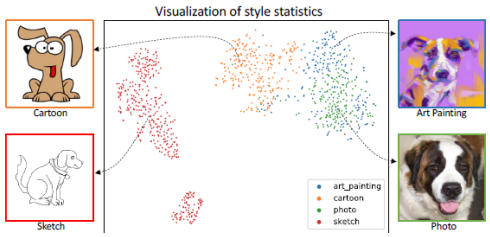
The proposal of a particular function for the vicinal distribution[63] is equivalent to extract virtual feature-target pairs as in 4.5

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1-\lambda)x_j, && \text{where } x_i, x_j \text{ are raw input vectors} \\ \tilde{y} &= \lambda y_i + (1-\lambda)y_j, && \text{where } y_i, y_j \text{ are one-hot label encodings} \\ \lambda &\sim \text{Beta}(\alpha, \alpha), && \text{for } \alpha \in (0, \infty). \end{aligned} \tag{4.5}$$

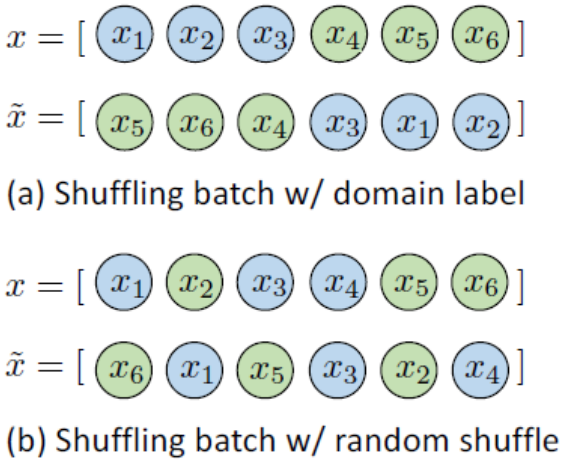
The advantages of this form are numerous:this is dataset-agnostic data augmentation is dataset-dependent; moreover, it does not assumes that examples in the vicinity share the same class as classical data augmentation do, since it acts also on the labels. The strong assumption behind mixup is the prior knowledge that linear interpolations of feature vectors should lead to linear interpolations of the associated targets,in this way extending the training distribution. Moreover, this assumption forces the model to behave linearly in between training examples: authors argue that his linear behaviour reduces the amount of undesirable oscillations when predicting outside the training examples.

#### 4.1.2.2 MixStyle

In the context of computer vision, the term "domain" refers to a specific distribution or set of characteristics that define the nature of the input data. It encapsulates several qualities of the input, such as quality, visual style, and more. A good example of this concept is the difference between a picture of the same subject, like a dog, taken with a camera versus drawn as an illustration. These two examples highlight different visual domains: camera one, and sketch one. However, the concept of a domain can sometimes be more subtle. Consider a photograph of an outdoor environment: variations in weather conditions or the time of day when the picture is taken can create different domains. If a model is not trained to handle these variations, its performance may degrade when confronted with such changes. Basically, the visual domain is closely related to the image's style Precisely under this last assumption, MixStyle[64], a domain generalization and data augmentation technique, was introduced to counter the limitations posed by domain-specific variations in data. MixStyle aims to enhance the robustness and generalization capabilities of models by addressing the distributional shift across different domains. It does so by mixing the styles of data from various domains, effectively augmenting the training set and forcing the model to learn more invariant features. One thing to stress out is that this technique like mixup is Domain agnostic. The technique itself is of easily implementation since it is applied to batches during training. The first operation involves preparing another batch from the original one by means of shuffling: this is done so that in this way when the two batches will be put aside, there will be a probability that different domain data end up side to side. The following operations involve a first instance normalization of data (Normalizing feature tensors with instance-specific mean and standard deviation has been found effective for removing image style in style transfer models):given a batch of images, each image's mean across width and height,as well as variance are extrapolated.




(a)




(b)

Figure 4.9: a)Visualization of different style,same conceptual subject.b)Shuffled batch construction.Extracted from [64].

This is done for both the batches, and the captured means and variances,altogether with other hyper-parameters, all controls the extent of the core operation: the mixing 4.8.

$$\sigma_{\text{mix}} = \lambda\sigma(x) + (1-\lambda)\sigma(\tilde{x}) \tag{4.6}$$

$$\mu_{\text{mix}} = \lambda\mu(x) + (1-\lambda)\mu(\tilde{x}) \tag{4.7}$$

$$\text{MixStyle}(x) = \sigma_{\text{mix}}\left(\frac{x-\mu(x)}{\sigma(x)}\right) + \mu_{\text{mix}} \tag{4.8}$$

This is equivalent to give to the normalized original batch a new variance and mean by the combination of the previous variance an mean computed, with a grade of mixing decided by the hyper-parameter $\lambda$.
With its agnosticism, MixStyle found application in several case and field; strong of its results, it has inspired advancements in image generation and style transfer tasks.

1. **Which are the domains in the DCASE dataset?**

   In the context of the DCASE training dataset, the device sources can be treated as domains, since each of the recording devices seem to leave an imprint over the audio. Though the imbalance in terms of devices,lead to a less effecting mixing action.

The strength of MixStyle is it's data agnosticism: but several attempts have been made to report this technique specially in the field of spectrogram or similar visual representations. DCASE Task1 has indeed pushed several teams to develop strategies in this sense. It has been recognized that the imprint left by the recording device over visual representations of spectrograms can be found

along the frequency:frequency magnitudes along the time dimension differ among the devices[39]. The work of [40] provides an estimation of the mutual information between the dimension-wise statistics (frequency, time,depth) of a CNN-based architecture processing spectrograms, and the devices and scenes label of Dcase task1. The estimated plot across the layer are shown in figure 4.10.It is important to note the correlation between the frequency statistic and the device and scene label, with respect to the time. Given the high correlation of the former,it is evident that the frequency component play a predominant role carrying a huge amount of information concerning the device and the scene: conversely, this also mean that the device information is imprinted on the frequency dimension.

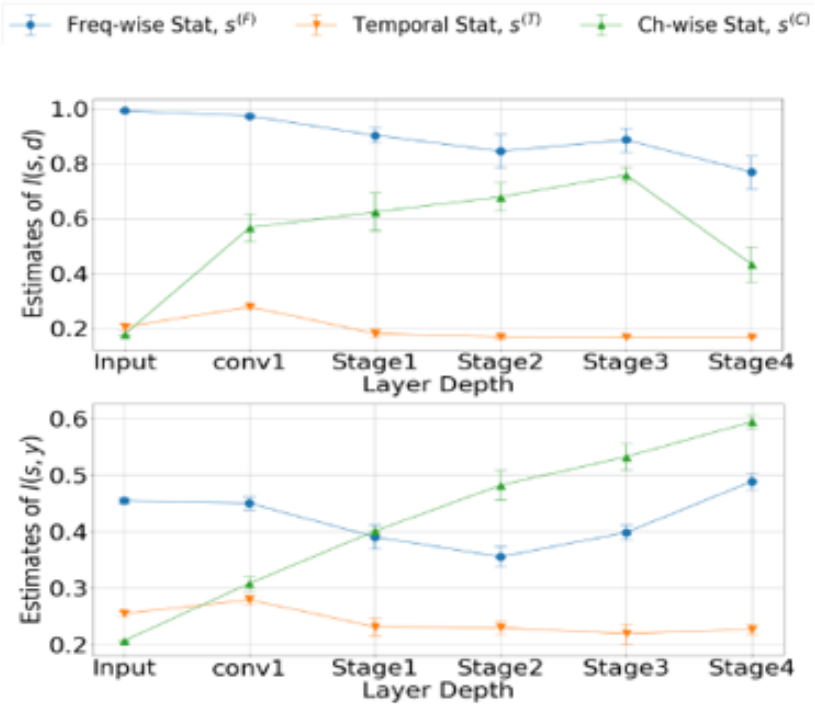



Figure 4.10: Mutual information between of dimension-wise statistics with label and scene.Extracted from[22].

Ulterior studies on CNNs and ViTs have also revealed that while this device-specific information persists through the hidden layers in CNNs, it tends to dissipate earlier in transformers[39]. Consequently, the application of the two previously mentioned strategies (suppressing device-related information or augmenting it) should be performed on the input data rather than on the features extracted by the model. Those studies along the presence of MixStyle, led to the application of a peculiar technique taylored for ASC: **Freq-MiXStyle**,a frequency-wise centering operation for spectrograms to remove recording device characteristics at an early stage. If in general MixStyle the $\sigma$ and $\mu$ of the batch of samples are computed along the width and height of the image, in the case of Freq-MixStyle they are computed obly with respect of the frequency dimension.

## 4.2 Transfer Learning

In the context of deep learning, the term "Transfer Learning" refers to a field of research and practice that focuses on utilizing a pretrained model on one task and adapting it to perform a different but related task. Transfer Learning in particular embrace the whole procedure, which entails both the pretraining and subsequent fine tuning stages. The pretraining-fine tuning paradigm has became ubiquitous in several field of deep learning: above all computer vision, natural language processing, Vision-language fields[35], which, as projects grow in complexity and scope, can't but benefit from the advantages brought up by transfer learning techniques; sometimes, they can't help recurring to transfer learning, which becomes a crucial stage in the traininig pipeline.

The term "Pretraining" can be viewed as a synecdoche of the term Transfer Learning:nevertheless, the pretraining and fine tuning stages in a Transfer learning framework are strictly linked, so in several occasions, when dealing with pretraining, we will end up discussing about fine tuning as well, and so , to the whole procedure of Transfer learninig itself.

Pretraining can be seen under different lights: the relationship between the tasks in pretraining and fine tunining, the nature of the datasets, the availability of labels,are all elements that concur in defining a different type of pretraining. A common approach in computer vision is to train a model on a larger Dataset, of the kind of ImageNet, and then fine tuning on smaller one which enclose the real meaningful information for the real task: the two "training procedure" (being that object recognition,image classification or others ) of the two stages pretraining and fine-tuning are often equal.[66] Example of the kind can be the pretraining on in fact Imagenet, and a fine tuning which involves identifying specific species of flowers in a botanical classification task. In both stages, the model undergoes a similar training process, but the fine-tuning phase allows the model to adapt its learned features to the nuances of the target dataset.[16]
This is the simplest case of Pretraining(from a conceptual point of view): perform training on a bigger dataset (with all the positive aspects of the case) in order for the model to acquire the necessary biases and knowledge to tackle the specific task,which often dispose of a smaller dataset that would make difficult for the model to effectively learn if it were trained solely on this limited amount of data.

To be more precise, that is the simplest case of Pretraining when the labels of the datasets are available: with the continue growing of projects and subsequent necessity of more data, it becomes difficult to dispose of a full-fledged labeled dataset, since the process of labelling is time consuming,resources expensive. This latter problem led to the development of pretraining strategies which not rely at all on having labels; the availability of labels enable a categorization about the nature of those strategies:

- **Supervised Pretraining:** Again, a synecdoche: Supervised Pretraining is no less the concept of Supervised learning applied to the transfer learning framework. Supervised learning

can be seen as a training process in which the model maps the input to the output based on labeled data. [7]

- **Unsupervised Pretraining :** The human intervention by providing labels to guide the model, to supervise it, is not contemplated. Instead, unsupervised pretraining involves training the model on unlabeled data, allowing it to learn underlying patterns and structures without explicit supervision. This approach relies on the data itself to provide the signals for learning. This is also why this category has to rely on pretext tasks; neveretheless recurring to unsupervised approach also comply having at disposal ,even if not labelled, a huge amount of data to draw from.[51]

- **Self-Supervised Pretraining:** A subset of unsupervised learning, where models are trained on a pretext task with automatically generated labels from the data itself, such as predicting missing parts of an image or the next word in a sentence [9].

#### 4.2.0.1 BERT

The fields of vision-language processing, natural language processing, and machine translation heavily rely on unsupervised and self-supervised transfer learning frameworks: ubiquity of pretrained models, prepared to tackle even multiple tasks[35]. The reason why the above listed machine learning fields deploy the unsupervised or self-supervised approaches is soon said. The tasks involved (Image Captioning,Question answering, Multilingual translator etc etc) require a sheer volume of data: large collections of text documents, such as books, articles, websites, and social media posts, images video. Such data is readily available in huge quantities on the internet but often lacks annotations or labels.
An important example that comply Unsupervised Learning, Pretraining with pretexts tasks, and transformer architecture in the context of Natural Language Processing is the case of BERT[8],introduced in 2018. BERT(Bidirectional Encoder Representations from Transformers) is a deep learning model which inherits the encoder based module of the Transformer Architecture, and presents itself in two forms., *Bert base* with 110 M parameters, and *Bert_Large* which accounts for 340 M parameters. As it happens with the standard Vision Transformer, since BERT is equipped with encoders and doesn't have any module resembling a decoder it can not 'generate': but by adding an additional output layer tailored for the objective task,pretrained BERT model can be fine-tuned to tackle a wide range of NLP tasks, such as question answering and language inference. The "Bidirectional" in Bert accounts for the fact that this model is able to consider both left and right context for each word in a phrase.
At the time of its introduction and for a significant period thereafter, BERT set the standard as the state of the art in natural language processing,and became the preferred choice in terms of baseline in this deep learning fields: several model leverage on the pretrained weights of Bert to accomplish countless tasks.[8] and it is precisely the particular pretraining procedures adopted that have made BERT perform so well.

In particular, those two pretext tasks were used for BERT, enabling it to gain its bidirectional ability are:

- **Masked Language Modelling:** A percentage of the input tokens sequence (sequence of word) is masked: meaning that during training stage, 15% of the input sequence, chose at random, and each token of this chosen sequence is substitute either with a MASK_TOKEN,either a random token or less probable it is left as it is. The pretext task associated is for the encoder to predict the content of those masked tokens.

- **Next Sentence Prediction :** This pretext task aims at making BERT able to understand relationship between phrases: given two sentences, BERT will seek to determine whether the second sentence is the successor of the first one.

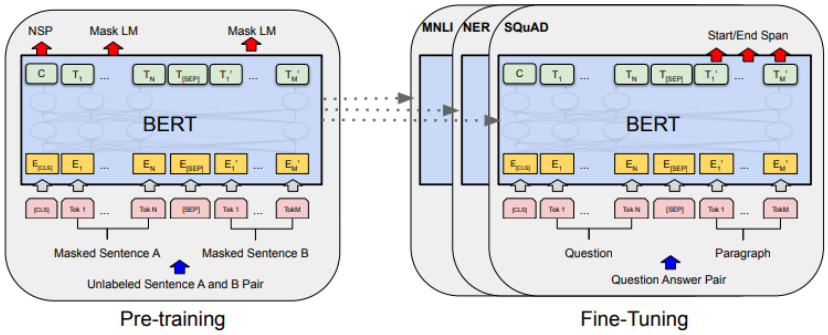



Figure 4.11: BERT's pretext tasks.Extracted from [8].

While the exact mechanisms behind how BERT achieves its capabilities are still being understood, significant importance is placed on its pretraining strategies[35] Other fields revolving around the input type of the images have their strategies and pretext tasks when dealing with unsupervised and self-supervised Pretraining. Similar to BERT's Masked Language Modelling, in the field of computer vision there is Masked Image Modelling, which works by randomly masking regions of an input images, training the model to predict those missing regions of pixels[20] Other techniques like contranst learnining in the field of self-supervised learning[5], are deployed as well, leveraging on the concept of discerning dissimilarities among the provided samples, to build classes.

### 4.2.1 Why Pretraining is useful

Pretraining is intuitively valuable due to the common challenge of acquiring sufficient data for effective learning. However, it's important to note that fine-tuning can sometimes jeopardize the effectiveness of the pretrained model if attention is not taken in consideration[66]. The benefits

and objectives of this process should be carefully understood, particularly since in a pretraining-fine-tuning framework, the initial phase typically consumes a significant portion of the overall time required. Among the fortes of Pretraining, there are:

- **Provide profitable universal inductive biases:**Pretraining allows models to learn general features and patterns from large datasets, which can accelerate learning on specific tasks.
- **Improve generalizability :** By learning from diverse data during pretraining, models often exhibit better generalization to new, unseen data in various domains.
- **Faster fine tuning and reusability :** Pretrained models can be adapted to new tasks with relatively less data and computation compared to training from scratch, thereby improving efficiency and reducing resource requirements.

### 4.2.2 Pretraining in CNNs and Transformers

The concept of pretraining is closely tied to the size and nature of the available dataset. For instance, AlexNet, a milestone model with 60 million parameters, achieved top performance in the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012), which featured 1.2 million images across 1,000 categories. AlexNet did not require pretraining; instead, its weights were utilized directly for the task. Similarly, the original Transformer model, introduced in the context of natural language processing (NLP), successfully tackled the WMT 2014 English-German and WMT 2014 English-French datasets. These datasets consist of 4.5 million and 36 million sentence pairs, respectively. The Transformer did not undergo pretraining in these instances. These examples highlight that while pretraining can significantly enhance model performance under suitable conditions, it is not necessary for every application or dataset. Different is the situation in which this Transformer capabilities in ASC exploration takes place: we are dealing with. The dataset in question comprises approximately 100,000 samples. Smaller datasets are often encountered, especially in medical studies, where the limited number of cases restricts dataset size. However, given the task's nature and the Transformer model's inherent appetite for data—typically trained on datasets in the millions—there arises a significant need for pretraining.

Pretraining has proven beneficial in numerous instances; this work does not deal with models of the scales of BERT . Focusing on the examples in which pretraining proved to be useful especially, at the scale this thesis field of Transformers, we have:

- **ViT:** All the Vision Transformers proposed in the orgininal paper, to perform specific tasks,were pretrained on different scales of Imagenet,ranging from 1.3 M images to 303 M images.
- **DeiT:** A variant of ViT that achieves top performance using significantly less data for pretraining compared to traditional ViTs, leveraging advanced data augmentation techniques
- **PaSST :** A peculiar case is the one of the Vision Transforme of PaSST: taylored to deal with spectrograms, they tackled the data hunger by being pretrained on ImageNet, a dataset containing photo, not spectrograms, an example of cross-domain pretraining.

As stated in the research questions, we are interested on the response of ViT from pretraining. We ended up deploying different approach related to pretraining depending on the scale of the explored Transformer:

- **Small_ViT:** Due to the principle of over determination [19], pretraining was deemed unnecessary for a model with such a small number of parameters. As a result, Small_ViT was excluded from the pretraining processes.
- **Medium_ViT :**Even if more rare,those kind of ViT with this size are present in literature: but due to the fact that as regard the token dimension (altogether with the depth, is the weightiest hyperparameter in deciding the size of a transformer) I chose an unconventional value, plus there seems not to be available medium size ViT pretrained on a useful Dataset like Audioset, I decided to "pretrain from scratch" instead of searching for already pretrained architecture to draw from, recurring to a subset of Audioset to pretrain the model and then use DCASE Task1 dataset for fine tuning.
- **Large_ViT:** Given the large number of parameters in Large_ViT models, pretraining on vast datasets such as AudioSet and ImageNet is essential. Pretrained ViT models of the interested size on these extensive datasets are readily available, which is crucial because the enormous size of these datasets and the significant time required to pretrain make it impractical to start from scratch with our resources. Given the better results achieved by pretraining on Audioset over ImageNet in the context of ASC, already pretrained ViT on the former Dataset were deployed, in order to fine tune them on the DCASE dataset.

All the cases explored in this thesis concerning Transfer Learning fall under the category of Supervised Learning; moreover, no pretext tasks were deployed, meaning that the model during pretraining was subjected to a task that coincide to the one in fine-tuning stage: image classification.

### 4.2.3 Audioset

Alongside the Competition dataset, other dataset were deployed as well to enhance the capabilities of the model in ASC; the competition itself enable each team to use any other dataset, at the condition for it to be public prior to a deadline that preceeds the day of the model's submission, in order for every team to decide whether to deploy that resource or not.

AudioSet[12] is a large-scale dataset that has been a significant resource for various audio-related research and applications. It is primarily used for training and evaluating machine learning models for sound recognition and audio event classification. AudioSet was introduced by Google in 2017 to facilitate the development of machine learning models that can understand and categorize sounds. Unlike other datasets that focus on specific domains (e.g., music, speech), AudioSet covers a wide range of sounds from various environments and activities: you can have chirp, electronic music, sounds of engine, and so on. AudioSet contains over 2 million 10-second audio

clips sourced from YouTube videos. The dataset is labeled with 527 audio event categories such as "dog barking," "car honking," "music," and "speech." AudioSet uses a hierarchical taxonomy where labels are organized into categories and subcategories. Among those categories there are for example:

- **Human Sounds:** Includes speech, laughter, singing classes and others.
- **Animal Sounds:** Encompasses barking, meowing, chirping classes and others.
- **Vehicle Sounds:** Consists of engine noise, horn sounds, sirens classes and others.

This structure conceals a limitation: for example, the sound of a car is typically linked to the noise produced by its engine. Since both those classes ("car" and "engine")are categorized in Audioset, it's not uncommon for the same audio samples to be associated with multiple classes. Each audio clip often has primary labels and additional labels as well. Many audio samples, for instance, fall under the categories of Speech and Music, which encompass a broad range of meanings and interpretations. The Dataset can be retrieved in two ways:

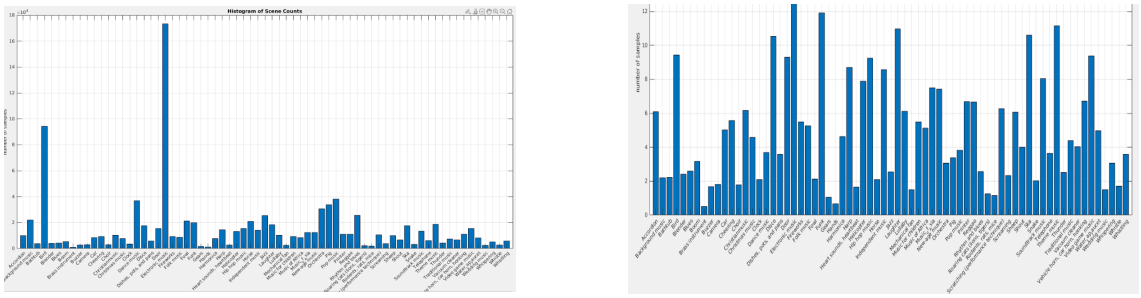
- **From YouTube:** A .csv file provides the YouTube URLs of the relevant audio clips along with their corresponding labels. However, this method has drawbacks, such as the lengthy retrieval process and the risk of YouTube videos being removed or made inaccessible.
- **Already Extracted PCA Features:** An efficient model has extracted 128-dimensional features from each audio clip, in order for them to be used by other models. We opted not to use this method because the feature dimensions would constrain the Transformer architecture design. Additionally, despite the primary focus not being on competition rules, using feature-extraction models counts towards the total model size,making following this path already having a disadvantage.

So, the chosen approach was the first one.The difficulties of downloading all the data, the fact that an audio can be be found in several classes due to the hierarchical structure of the dataset,led me to choose only a subset of audio, comprising 70 classes, striving for avoiding repetitions of audios; That led to a subset of Audioset consisting of nearly 200 000 samples,that after the splitting in one second long audio, reached a size of 2 M samples, 30 % of which used for validation purpouses.

Taken the whole Audioset picture, the distribution of samples per class is heavily skewed. A few classes dominate the dataset, specially the one involving music and human speech, due to the easier availability .Even choosing the 70 classes, it was inevitable to create a heavily unbalanced subset,given the first goal this whole procedure of pretraining has: necessity of a considerable amount of data to make up for the size of DCASE dastaset.

Models that tackle the Audioset are big: even thought samples repeat themselves, even though the classes are unbalanced , the number of samples is more of a strength,and those models are capable

of overcoming those issues thanks to their size and their organization. Classification performance on AudioSet is usually evaluated by a simple average over per-class metrics, meaning that performance on rare classes is equal in importance to the performance on common ones.Medium_ViT does not seem to behave the same way. In particular, while using the highly unbalanced version of the chosen created subset as a pretraining Set, the model achieved only 22% of validation accuracy : basically,the model was always asserting that a sample belonged to the highest sample class, resulting totally biased by the latter class.
Balancing,in my case has to be performed: recurring to an operation of total balancing classeswise is not so useful and it is not generally needed,moreover,it could be even harmful in the context of Audioset[32].Those reasons lead me to operate a balance with a fewer degree of action.




(a) (b)

Figure 4.12: Subset of 70 classes from Audioset,prior(a) and after(b) being balanced through offline data augmentation

AudioSet exhibits a long-tail distribution where a small number of classes have a large number of samples, and a large number of classes have a small number of samples. This long-tail nature makes it challenging for models to learn effectively across all classes without specialized techniques.
To resolve this problem without diminishing the learning potential by decreasing the data from the more represented classes, the offline data augmentation techniques discussed in section ... were applied only to low represented classes: moreover, in order to give equal importance to the underrepresented classes, class weighting technique was applied to adjust the loss.
More specifically, class weighting is a technique used to address class imbalance in machine learning by adjusting the loss function to assign higher importance to underrepresented classes, ensuring that the model pays adequate attention to all classes during training. This is typically achieved by weighting the loss inversely proportional to the frequency of each class. Also in my case, weight were computing making them inversely proportional to the number of samples per class, leading to the model incurring to higher loss if it misclassifies these underrepresented classes. Since even after the targeted data augmentation operation, the gap between the most undereppresented class and the most represented one low represented classes, in order to avoid overcompensation for rare

classes and unstable gradients update, a threhsold was posed: classes having a number of samples < 10 000 were treated as being composed exactly of 10 000 samples in the computation of their correspective class weight.

## 4.3 Knowledge Distillation

### 4.3.1 Compression Techniques

Are deeper models intrinsically better than their smaller counterparts? This question has driven research, seeking to understand the underlying factors that contribute to the general fact that deeper models have shown superior performance. Researchers have delved into various aspects, such as the ability of deeper models to capture more complex patterns and representations, their improved generalization capabilities, and the benefits of increased parameter capacity.[3]
Nevertheless, deeper models have seldom access to the world of edge devices due to their sizes, even if the interest in their deployment ,to thicken the ranks of AR/VR,IoT,autonomus driving and mobile assistant applications, never ceases to grow.[41]

These needs have pushed the research in developing techniques to counteratc the natural tendency of models "to go deep" . These methods cover a wide range of approaches, aimed at managing model complexity and computational demands more effectively. They encompass a broad spectrum of techniques. Famous is the case of the family of "MobileNet" [18], which revolutionized the well established and immutable concept of CNNs layers, proposing a version of them, more suitable for low-resource devices.
But there are several others examples:instead of acting on the structure itself, is it possible to act on the weights . Pruning for example is a technique in which connections (weights) are removed from a trained model if deemed to be redundant or less important, reducing the overall amount of parameters and improving inference speed: all striving to maintain the original achievement of the model to be pruned.
Quantization is instead a technique in which the weights of the model (which are typically stored as 32-bit floating-point numbers (float32) ) are mapped to a smaller representation (e.g int8)[15]. Neural networks are sensitive to small changes in weights: quantization involves operations of calibrations of scaling factors and zero-points to ensure that the quantized model performs as closely as possible to its full-precision counterpart.
And the list can go on, with Low Rank factorization,weight Sharing and more.

Those techniques are addressed in the field of Deep Learning as "Compression Techniques",since their aim is to compress the ability of a model into a smaller model in terms of size,capable of keeping up with its uncompressed version. Among the Compression Techniques, there is also Knowledge Distillation[17].

### 4.3.2 Knowledge Distillation Concept

Knowledge distillation is a technique in machine learning where a smaller, more compact model (student) is trained to mimic the behavior and predictions of a larger, more complex model (teacher). The goal is to transfer the knowledge encoded in the teacher model to the student model, achieving similar performance while reducing computational complexity and memory footprint. The concept of distillation can be traced back to 2015, where ensemble methods and model compression techniques were explored to reduce model size and computational cost:these methods often involved training simpler models to mimic the outputs or decisions of larger ensemble models.

Even if the concept of "Distillation" has deeper roots, the seminal paper by Hinton, Vinyals, and Dean, 'Distilling the Knowledge in a Neural Network' [17] has profoundly shaped the way we know and deploy Knowledge Distillation nowadays:it introduced several innovations that have since become foundational for anyone interested in deploying a knowledge distillation framework. The reasoning of this technique, starts by the awareness that the "logits" 4.9 coming from the inference of a model in what can be a classification task, say a lot regarding how this model "thinks". Since those logits have an intrinsically greater entropy with respect to ground truth labels, they can convey more information:a smaller model can effectively harness this information, leveraging the diverse knowledge embedded in the logits to refine its own predictions and decision boundaries.[17].

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{4.9}$$

- $z_i$: Logit associated with the $i$-th class.
- $T$: Temperature parameter controlling the sharpness of the probability distribution.

In the basic form of Knowledge distillation , the relationship between Teacher and Student is conveyed through the logits of the teacher; and the whole process of distillation leverage on the form of the loss of the student. The loss of the student toward the ground truth labels as in standard procedure, and the loss of of the student generated by considering the logits of the teacher: those two components comply in defining the total loss function that the studet will deploy to improve itself during the training procedure:

$$\mathcal{L}_{\text{KD}} = (1-\lambda)\cdot \text{CE}(y_{\text{true}}, \hat{y}_{\text{student}}) + \lambda \cdot \text{D}_{\text{KL}}(q_{\text{teacher}} \| q_{\text{student}}) \tag{4.10}$$

$\text{CE}(y_{\text{true}}, \hat{y}_{\text{student}})$ :hard distillation loss (cross-entropy loss between the true labels and the student's predictions)

$\text{D}_{\text{KL}}(q_{\text{teacher}} \| q_{\text{student}})$ :soft distillation loss (Kullback-Leibler divergence between the teacher's and student's logits).

The hyper-parameter $T$ in equation 4.9 is used to soften the probability distribution output by the teacher model. It is called the "Temperature". With a high $T$, the softmax function produces probabilities that are less sharp, providing more information about the relative similarities between different classes to the student model. The weighting parameter $\lambda$ instead serves to balance the importance of the components of the loss function. Often, the balance is heavily weighted towards the distillation loss.

### 4.3.3 DeiT: Data Efficient Image Transformer

In the previous section, the role and importance of the loss as a guidance for the student during training was hightlighted. In many Knowledge distillation frameworks it is indeed the only expedient to implement this compression technique. However in many cases knowledge distillation extends beyond focusing solely on output logits. It can includes distilling features extracted from intermediate layers of the teacher model for example. These layers capture complex representations of data that are crucial for making accurate predictions. For instance, in natural language processing, intermediate layers may encode syntactic structures or semantic meanings of sentences. By distilling these features, the student model not only learns to mimic the teacher's outputs but also gains a deeper understanding of the underlying data representations, enhancing its performance on similar tasks.

Furthermore, strategies involving multiple teachers may further enrich the distillation process. Each teacher may specialize in fact in different aspects or tasks, providing diverse insights and knowledge perspectives to the student model. For example, in computer vision applications, one teacher might focus on object detection while another specializes in facial recognition. By aggregating knowledge from multiple teachers, the student model integrates comprehensive information, improving its adaptability and performance across various related tasks.
A noticeable case of Knowledge distillation in the context of the field of vision , which entails the deployment of a Transformer model with a peculiar and new approach, is the case of DeiT(Data efficient image Transformer)[48].
DeiT's benchmark is image classification: the latter is considered a core task in computer vision,often being used as a standard to verify the model's understanding in the field of images. DeiT is no less than a ViT:maintains the same scale and architecture as the original ViT, with variants like DeiT-Tiny, DeiT-Small, and DeiT-Base corresponding to the ViT configurations,allowing

for a direct comparison betweenn the two approaches. Unlike original ViT, which was initially pre-trained on the expansive JFT-300M dataset, DeiT demonstrates its effectiveness using the much smaller ImageNet dataset. DeiT incorporates data augmentation strategies such as RandAugment, Mixup, and CutMix. These techniques enhance the diversity and quality of the training data, improving the model's robustness and generalization capabilities. So,the "data efficiency" in its name stands for the fact that this model is able to reach state of the art even if managing smaller datasets with respect to its predecessor, challenging the common belief that transformer vast amount of data to excel.

The reason we are discussing about DeiT in a section dedicated to Knowledge distillation is soon said: the state of the art results achieved by DeiT alone,are surpassed by a DeiT involved in a knowledge distillation framework. Specifically, a Knowledge distillation paradigm that sees a DeiT as a student, a convnet like the RegNetY-16GF as a teacher, a new Knowledge distillation strategy that leverages on the transformer architecture, is able to surpass the very same convnets, the DeiT, and another Knowledge distillation paradigms that sees DeiT architectures for both the teacher and the student.
It is important to note that the purpose of this knowledge distillation was not aligned with model compression; the DeiT achieving state-of-the-art results remains large.
The astonishing fact is that the better results were achieved using a convnet as a teacher: correlational studies brought up the fact that a DeiT, instructured by a convnet teacher, at the end seems to think more like a CNN rather than a ViT, as if the CNN transmitted its inductive biases to the ViT.
The contribution brought by DeiT regarding Knowledge Distillation technique is proposing an approach which leverages on the Vision Transformer architecture. Vision Transformer takes an image as an input, divide the latter in patches that will be treated as tokens once projected, and add to this sequence of tokens, a class token: a token that will enter in contact with all the others during attention module ;a token upon which (in image classification) the encoder'head will make the classification. Deit propose the addition of an ulterior token to the sequence, the distillation token: it will undergo the very same path of the class token, but it will be the base upon which an ulterior encoder's head will make the classification,but a classification based on the teacher's logits,rather than the ground truth labels as in the case of the class token. This innovative approach effectively enhances the conventional method solely reliant on loss functions, and it is pretty easy to integrate to the standard ViT architecture.

Moreover,DeiT propose an alternative to the usual knowledge distillation's loss computation. It proposes indeed, the Hard label Distillation loss,in which computation of the distillation loss is performed recurring to the teacher's hard predictions as ground truth labels, foregoing the use of softened logits that contain more nuanced information. However, empirical evidence suggests that the conventional soft distillation approach often outperforms the Hard Label Distillation method.

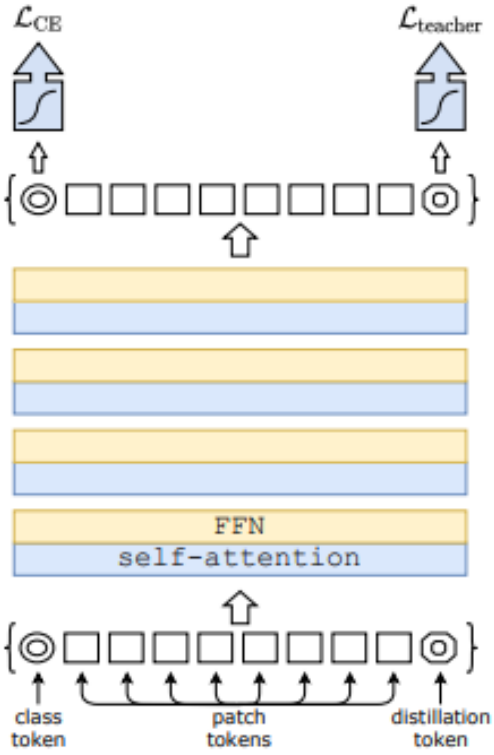



Figure 4.13: DeiT,with its characteristi distillation token inserted in the sequence of tokens.Extracted from [48].

**Soft Distillation:**

Soft distillation combines cross-entropy loss with knowledge distillation using softened logits from the teacher model:

$$L_{\text{global}} = (1-\lambda)\mathcal{L}_{\text{CE}}(\psi(Z_s), y) + \lambda\tau^2 \text{KL}(\psi(Z_s/\tau), \psi(Z_t/\tau)), \tag{4.11}$$

where:

- $\mathcal{L}_{\text{CE}}(\psi(Z_s), y)$ is the cross-entropy loss between the predictions $\psi(Z_s)$ of the student model and the ground truth labels $y$.
- $\tau$ is the temperature parameter.
- KL denotes the Kullback-Leibler divergence.
- $Z_s$ and $Z_t$ represent the logits from the student and teacher models, respectively.
- $\lambda$ controls the trade-off between the two terms.

**Hard-label Distillation:**

Hard-label distillation uses the hard decisions (argmax) of the teacher model as true labels:

$$L_{\text{hardDistill}} = \frac{1}{2}\mathcal{L}_{\text{CE}}(\psi(Z_s), y) + \frac{1}{2}\mathcal{L}_{\text{CE}}(\psi(Z_s), y_t), \tag{4.12}$$

where:

- $\mathcal{L}_{\text{CE}}(\psi(Z_s), y_t)$ is the cross-entropy loss between the predictions $\psi(Z_s)$ of the student model and the hard labels $y_t$ (argmax of $Z_t$).

In equation (3), $y_t = \arg\max_c Z_t(c)$ represents the hard decision of the teacher.

DeiT's innovative Knowledge Distillation framework with the added distillation token was deployed in this thesis work: this method proved to be in fact more effective that the standard approach.

### 4.3.4 Featured Based Knowledge Distillation: ViTKD

Even if DeiT's knowledge distillation paradigm leverages on the architecture of the model, it is still a logits-based Distillation approach. Other strategies involve the sharing, the transfer of other or additional information from the teacher to the student. It is the case of Feature-based Knowledge distillation, which entails the transferring of the intermediate representations (features) extracted by the teacher model to the student model, during the training of the latter.

If in logits-based, the student was incentivized to align its predictions with those of the teacher model influencing the student's representations across the layers as a consequence of the back-propagation, in feature based this approach is more direct, encouraging the student to learn, to mimic representations of the teacher across the layers.
An interesting work in this sense entails the appliance of feature-based techniques to a knowledge distillation framework where both the teacher and the student are DeiTs[59]. VITKD,this is the name of the technique, thickens the ranks of the feature based KD for ViTs,leveraging on the structure : the majority of techniques in this sense are inherently structure-independent[59].

VITKD's foundations are based on several observation made on the attention maps of teacher and student(not yet in the KD-framework) trained and tested on the same dataset. The conclusions is that if the attention maps in the attention blocks of the shallow layers show a similar diagonal pattern present both in the model that will become the teacher and the model that will be treated as the student, the same can not be said regarding the last layers, where the attention maps show that the two models attend to different patches.
Extensive experimentations lead to the definition of two pretext tasks for performing the distillation action,adequate to deal with the misalignment between teacher and student attention maps that happens with a different degree and phenomenology across the layers: one of this proxy objective is mimicking, and it is meant to be applied to the features extracted at the shallow layers, while the other, suited for deep layers instead, is generating.

Given the student's and teacher's features $F^S \in \mathbb{R}^{N \times D_S}$ and $F^T \in \mathbb{R}^{N \times D_T}$, where $D$ account for the embedding dimension and $N$ stands for the number of patches, a linear layer to align the dimension of the student's $D_S$ and the teacher's $D_T$, the loss relative to the mimicking is computed as follow:

$$L_{\text{lr}} = \sum_{i=1}^{N} \sum_{j=1}^{D} \left(F_{i,j}^{T} - f_c(F^S)_{i,j}\right)^2, \tag{4.13}$$

This method requires the teacher's embedding dimension being an integer multiple of the student's one; the work proposes an alternative in case this condition is not met: since our application satisfies this condition, the other alternative was not explored. The features are extracted at the exit of the MLP layer, at the end of the encoder relative to the layers interested in the mimicking process: the first 2 layers. Teacher and student do not have to posses the same number of layers.

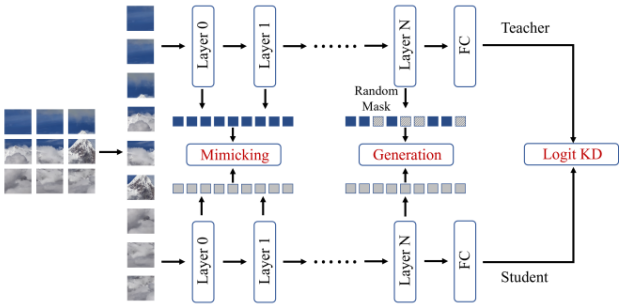



Figure 4.14: ViT_KD: Mimicking is performed at the shallow layers, while Generation is done at the deep leayers.Extracted from [59].

As regard generating instead, it also involves a first step of alignment of the features dimension of teacher and student.After that, a mask is computed given the hyperparameter threshold $\lambda$: it is meant to mask the tokens randomly, given $r\_i$ a random number uniformly distributed in [0, 1]. This prepares the ground for the actual pretex task:after masking,the tokens are fed into a generative block $G$ (whose nature can be that of CNNs based architecture, or an attention's one) and they are meant to generate as best as possible the corresponding patches in the teacher sequence of tokens.

$$\hat{F}_i^S = \begin{cases} \text{masked token, if } r_i < \lambda \\ \text{original token, otherwise} \end{cases} \qquad M_{\text{ask}_i} = \begin{cases} 1, & \text{if } r_i < \lambda \\ 0, & \text{otherwise} \end{cases} \qquad G(\hat{F}^S) \to F^T. \tag{4.14, 4.15}$$

The student is encouraged to generate its own representations of the teacher's features,through the back propagation of the resulting generative loss,which quantifies the discrepancy between the teacher's features and the student-generated features. This loss, shown in equation, altogether with the mimic loss computed at the shallow layers and the loss relative to the ground truth label concurs in defining the total loss deployed by the student during training.

$$L_{\text{gen}} = \sum_{i=1}^{N} \sum_{j=1}^{D} M_{\text{ask}_i} \left(F_{i,j}^{T} - G(\hat{F}_{i,j}^{S})\right)^2 \tag{4.16}$$

$$L = L_{\text{ori}} + \alpha L_{\text{lr}} + \beta L_{\text{gen}} \tag{4.17}$$

This technique has demonstrated its effectiveness in enhancing the capabilities of DeiT models on ImageNet. Various configurations are propose, such as using DeiT-B as the teacher and Deit-Ti as the student. This configuration parallels the scale of the Large_ViT and Medium_ViT models used in my thesis: the effectiveness of this technique was studied, given the different conditions with respect to the presentation's paper; namely, the ASC context, and the ristricted available dataset in comparison with ImageNet.

# Chapter 5

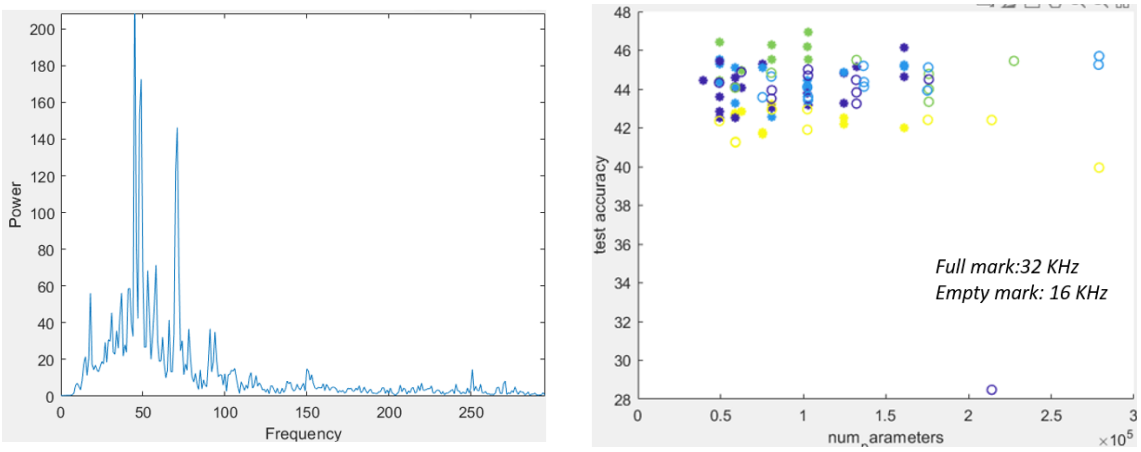
# Experimental finding on Transformers for ASC

This chapter presents the main findings and experiments conducted during this research: its structure try to retrace the Research Question's Layout. First, the focus is on the results concerning the main visual representation adopted throughout the experiments.We then proceed to examine the performance and needs of different size of Vision Transformer, verifying the effectiveness of data augmentation and regularization techniques . Finally, we discuss the improvement brought by the employment of Pretraining and Knowledge distillation Frameworks. It is important to address that those sections are linked: the effectiveness of a technique with respect to another may be appreciated better in the context provided by a another subsection.

### 5.0.1 Visual Representation results

In this work, concerning the visual representation adopted, the focus was posed on two different kind of audio-image representations, namely the Mel-Spectrogram and the Continuos Wavelet Scalogram. The first interrogative, nevertheless the nature of the visual representation, was the resolution of the image itself.CNN-based architectures often need to be adjusted when dealing with images of different sizes. Instead, Vision Transformer,with its patching operation, is able to process different size images without major structural changes. If a Vision Transformer were to be fed a different input image size, the major difference would be the patches sequence length, and the only major adjustment would involve altering the positional embedding vector according to the new configuration. But this first interrogative of the image size is actually strictly linked to the nature of the visual representation. In the case of Spectrogram, the window size and the number of frequency bins both influences the size of the resulting image and the resolution in terms of frequency and time resolution. On the other hand for Scalograms,the computation of the wavelet coefficients involve each sample,meaning that at a first instance, the width(the axis corresponding to time) is totally determined by the length of the audio and the chosen sampling rate, being the of the two. The size related to the pseudo frequencies(height) is determined in a similar way to that of Spectrograms, defining a similar number of bins. Scalograms need also a resizing

process since the aspect ratio of the resulting image without it would be totally unsuitable, and unfeasible given the number of parameters to be deployed in the training procedure. So, the very first choice that has to be made, is the entity of the sample rate:audio is provided in a single-channel 44.1kHz 24-bit format, but this sampling rate seems to be more than necessary, once the average spectrum of the dataset is seen in figure 5.1. Subsequent test on different sampling rate values, led us to confirm and adhere to the standard approach proposed by the seminal works of Audio Spectrograms Transformers [13]: in terms of sampling frequency, this choice entails recurring to a sample frequency of 32 KHz. Moreover, a more strong adherence to the most deployed approach and the not good results achieved ,led us to shift from the initial chosen dimension of 70x50 for scalograms and 50x50 for spectrogram, to the uniform size of 128x100 for both the visual representations: 128 frequency bins,window size of 0.025 s with 0.01 the stride or the hop.




(a) (b)

Figure 5.1: a) Average spectrum of DCASE task1 dataset.b)Study on sampling frequency effectiveness.

The necessary steps to prepare the Scalograms(involving the computation of the wavelets coefficients and the resizing operation) open the doors to various choices: the number of frequencies per octave defining the frequency resolution, and the resizing shape(based on average,interpolation, others) are the two most influential factors in this process, leading to various trials in deciding the right scalogram configuration. As explained in section 3.1.2, we deployed the *Morlet* wavelet to compute the wavelet coefficients.In order to apply direct comparison, after the computation of the scalogram,it was resized to 128x100. In the end, three factors influenced our decision to opt for spectrograms over scalograms:

- **Preliminary Results**: Since the early stages of trials comparing scalograms and spectrograms results, it was obvious the superiority of the latter representation.
- **Knowledge and Trend**: Scalograms are less popular than spectrogram,which are the preferred choice in ASC applications, and have proved better results in general[38]. Consequentially, the deployment techniques for scalograms are less familiar and established, and

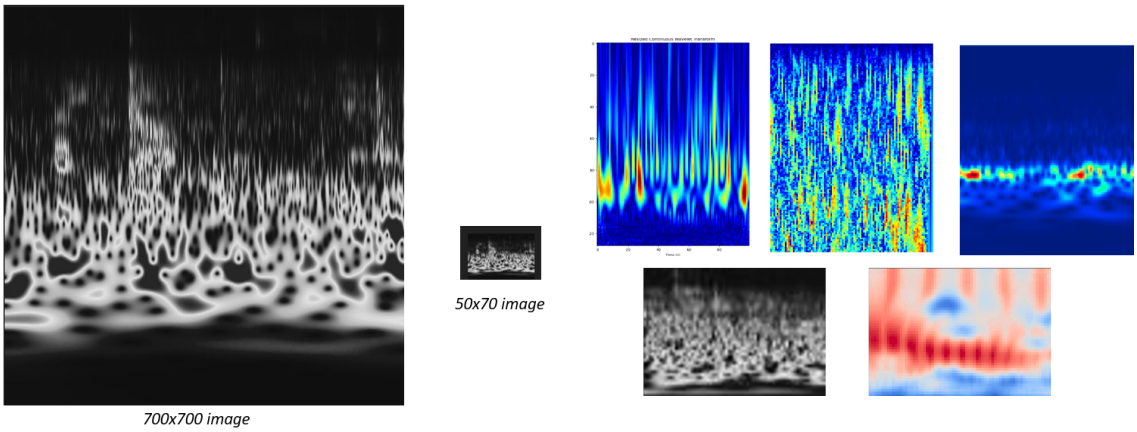
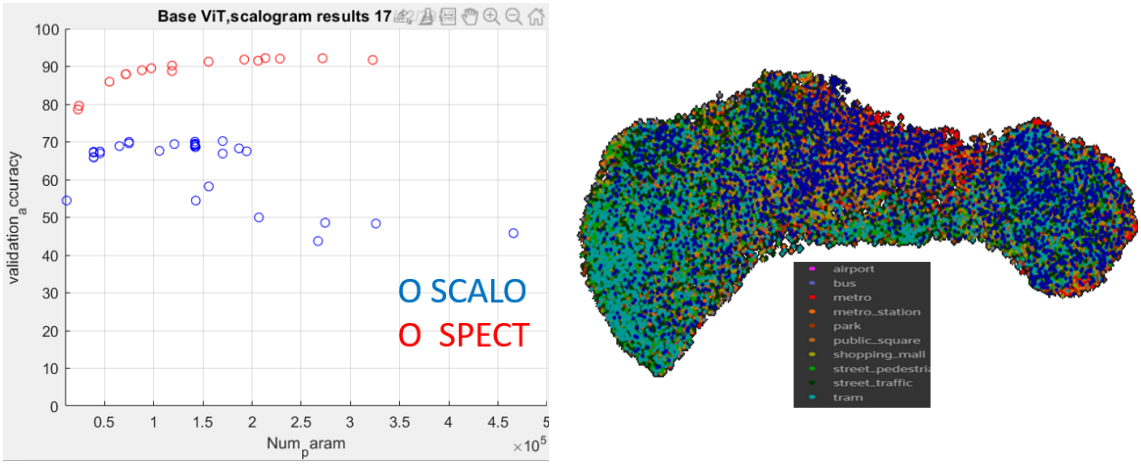



Figure 5.2: Resizing of scalogram,and the various scalograms tested.

require more thorough investigation.

- **Computational aspect**:The generation of scalograms is significantly more computationally expensive compared to spectrograms. For a given dataset, preparing scalograms can take up to 50 times longer than preparing spectrograms [38]. In my case, it took days to generate scalograms compared to just a few hours for spectrograms, highlighting the less flexible nature of a Scalogram-based training approach.




(a) (b)

Figure 5.3: a)Scalograms versus spectrograms.b) Representation of the embeddings of a scalogram set of DCASE task1,provided by FiftyOne[52],an open-source tool for visualizing the presence of hidden structure in the data.

### 5.0.2 Transformer Sizes

As discussed in chapter 3, the shift from audio to image domain, typical in the competition and ASC, prompted the adoption of ViT-based variant of the Transformer architecture. The original ViT paper introduced three variants ViTs , ranging from the smallest with 86 million parameters to the largest with 632 million parameters. In contrast, DeiT introduced variants such as DeiT-Ti, DeiT-S, and DeiT-B, with 5 million, 22 million, and 85 million parameters respectively. This thesis focused the exploration of three different scales of Vision Transformer, for different reason, being them the

- **Small_ViT:** This model was designed to have a number of parameters that fit within the competition constraints,leading to a ViT' size of an order of magnitude fewer than the smallest ones found in the literature.However, it ultimately proved insufficient for the task.
- **Medium_ViT:** This model's parameter count matches that of the smallest state-of-the-art fully attention-based ViT. With Medium_ViT, in this work it is referred to a series of ViTs having number of parameters in the range 5-8 M. Using this architecture, we demonstrated the effectiveness and necessity of techniques such as knowledge distillation, pre-training, and data augmentation.
- **Large_ViT:** With 86 million parameters, this model adheres closely to the original Vision Transformer architecture. It achieved superior results but required more extensive training. It was employed as the teacher in the knowledge distillation framework.It also defines the threshold in terms of ViT' scale I could deploy given the GPU used for experiments. This model takes part of its weight from the pretrained-on-Audioset version of the PaSST deployed by the winning team of DCASE Task1 2022 and 2023(not fined tuned on DCASE).

Table 5.1: Employed ViT architectures

| Model | # Layers | Hidden Size | MLP Ratio | # Heads | # Parameters | Comparable Models |
|---|---|---|---|---|---|---|
| Small_ViT | 4 | 60 | 1 | 6 | 103,150 | MobileAST Light |
| Medium_ViT | 7,8,9,12 | 256 | 2-4 | 4,8 | 5M-8M | MobileViT, DeiT_Ti |
| Large_ViT | 12 | 768 | 4 | 12 | 85M | ViT-B/16, DeiT-B |

The size of the ViT dictates the scope of pathways that can be explored within it. Models with smaller memory footprints, resulting in fewer layers and dimensions, can be incorporated into a training pipeline with hyperparameter search frameworks like Optuna[37], thereby enhancing their capabilities. The latter is the case of Small_ViT, for which an hyperparameter search was executed, tuning the parameters related to data augmentation,learning rate, architecture. But as expected, Small_ViT is not suited for the competition: even tough its training is way easier, the model seems not to appreciate the Data Augmentation techniques applied, which seems to lead the training toward instability. The parameter range listed for Medium_ViT in the table reflects variations made to explore the contributions of key components of the transformer architecture, the

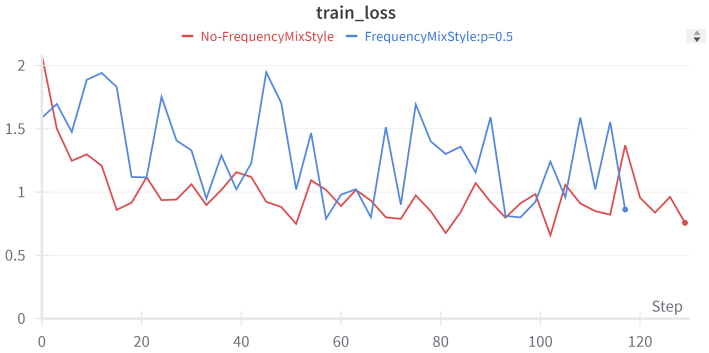



Figure 5.4: Effects of FreqMixStyle on Small_ViT.

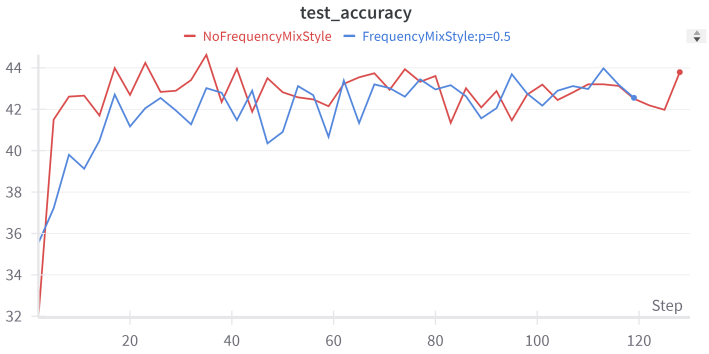



Figure 5.5: Small_ViT unstable training.

attention heads, the multilayer-perceptron, the layers itself. More heads in the multi layer blocks means a Vision Transformer able to capture diverse relationships and dependencies across different parts of the sequence of tokens;but comes at the cost of building relationship based upon smaller dimension vectors. Also the layers participate in thickening the ranks of different patterns that the Vision Transform can understand regarding the relationship between tokens[27]. The general "rules of thumb" extracted from those trials, Again, employing Optuna, an hyper-parameter search in the sense of the architecture was made using a small set of 22 classes from Audioset.

### 5.0.3 Data Augmentation and Regularization techniques

In assessing the outcomes of applying data augmentation techniques, some key considerations can be made. Regarding what we addressed as offline data augmentation, the techniques, as well as the parameters defining their degree of action(e.g, how much masking to apply,or the probability with which they are applied) have already been tested in the competition, providing good results while deploying CNNs base architecture. Instead of testing the effectiveness of each singular technique alone, three global approaches have been followed in building the overall training dataset:

- **Device balance approach:** Offline data augmentation techniques are applied to all the audios except the ones recorded with device a, in order to introduce a sort of balance in the sense of the

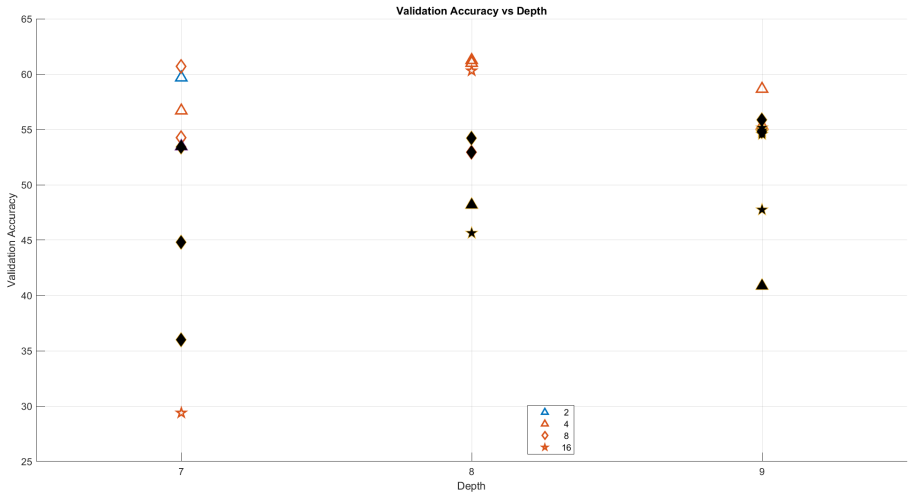



Figure 5.6: Results of Medium_ViTs on a subset of Audioset, given then depth, the number of heads,and the mlp ratio.

recording devices. This, plus the reconstruction of the original 10 second long audios,followed by a shifted split, produced a dataset of 440 000 training samples.

- **Data Hunger Approach:** Offline data augmentations are applied to all audios indiscriminately, producing a training dataset of 586 560 samples.
- **IRD Dataset** To verify the effectiveness of Impulse Device Response data augmentation, a dataset was built following the same approach as the Data Hunger one, plus the introduction of IDR data augmentation reserved to audios recorded with device a,with an incident probability of 50% for each audio belonging to that category.This gave birth to a training set of 621 840 samples.

A fast glimpse to the different effects of deploying those approaches can be seen in the following table,in which the results of Large_ViT using the above datasets are displayed.

Table 5.2: Comparison of Large_ViT Models on different training sets.

| Model | Offline Data Augmentation Approach | Shifted Audio Addition | IDR Data Augmentation | Training Set Size | Test Accuracy |
|---|---|---|---|---|---|
| Large_ViT | Device-balance | ✓ | ✗ | 440,000 | 49.95% |
| Large_ViT | Data-hunger | ✗ | ✓ | 586,560 | 51.26% |
| Large_ViT | Data-hunger | ✓ | ✓ | 621,840 | 48.45% |

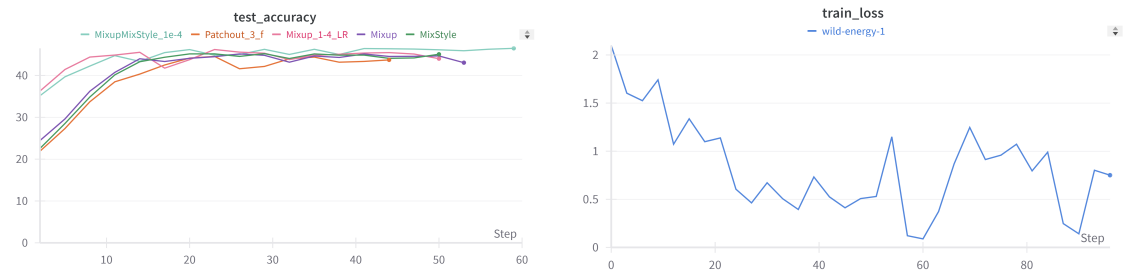
A check on the contingency tables that relates the accuracy per class and device

Another important key consideration has to be made concerning Patchout, the regularization technique which drops part of the input sequence. Various trials made on Medium_ViT confirmed that this technique is not suited for this model size. Various degree of structured patchout,both in the frequency and time dimension, were tested: nearly all the trials without recurring to Patchout consistently surpassed those that included this regularization technique, and the appliance of heavy

patchout(50% of the sequence dropped) conducted to unstability. On the other hand, given the constraints posed by the deployed GPU,Patchout seems the only way to implement Large_ViT: for this model, we followed the Patchout configuration proposed by the 2023's DCASE Task1 winning team; apply frequency-wise structured patchout,dropping six rows from the grid of patches. Given the other configurations of patch size and stride, this translated in dropping 54 patches out of a sequence of 108 patches, resulting in 54 patches handled by the model.

Regarding the effectiveness of the online techniques MixUp and Freq-MixStyle, various permutations were tested: applying each technique individually, applying both simultaneously during training, and alternating between them within batches, taking into consideration that both have a probability of 50% of being applied to a batch during training. Overall, the combinations yielded similar results, with Freq-MixStyle showing a slight advantage,as shown in figure 5.7.

However, Freq-MixStyle is responsible for the peaks observed in the training loss curves: these peaks tend to alternate between epochs, potentially due to the 50% application rate per batch: if early batches in an epoch have higher loss due to augmentation, subsequent batches might also experience challenges in learning effectively, potentially leading to higher overall loss for that epoch. So the rise can be due to the case when Freq-MixStyle is applied with more frequency to the early batches. Regarding the effectiveness of IDR Data augmentation, tables containing the



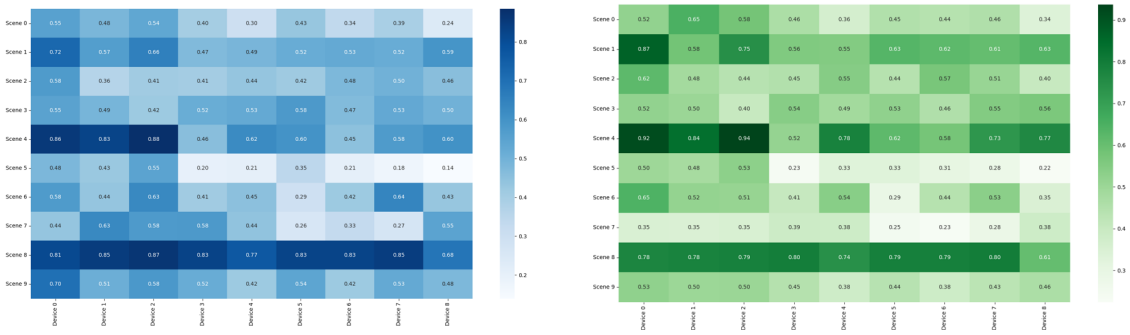

(a) (b)

Figure 5.7: a)Different online data augmentation applied to Medium_ViT.b) Effect of FreqMixStyle on the loss of Large_ViT during fine-tuning.

test accuracy with reference to both the scenes and the recording devices were captured 5.8. By comparison of their mean accuracy and standard deviation across the devices,being the latter an indicator of how well(if low) a model is able to generalize, we confirm that applying IDR improve generalization as wanted.Comparisons can be seen in table 5.3.

### 5.0.4 Pretraining and Knowledge distillation results

Since Large_ViT is already pretrained on the whole set of Audioset, and Small_ViT does not lend itself to this kind of analysis due to its limited size, the attention concerning pretraining is totally posed on Medium_ViT. Given that the available pretrained PaSST models on Audioset account for 85 million parameters, using a subset of Audioset could lead to improvements for a model with an



(a) (b)

Figure 5.8: Accuracy per-scene and device of Large_ViT: a)not using DIR,b)using DIR

order of magnitude fewer parameters.

The construction of the dataset is already discussed in Subsection 4.2.3: following the classic 70%/30% split in training and validation dataset. We pretrain Medium_ViT on this subset for fine tuning purposes, but at the same time we measure its capabilities on only part of the most recognized dataset for assessing performance of models in audio field. On the strength of the achievements concerning Patchout, we did not apply this technique; nevertheless, we seized the opportunity to value the importance of the patches sequence length, once applying overlapping,resulting in the already met configuration of 108 patches, and once without it ,resulting in the more short sequence of 48 patches.The overlapping approach, although it took six times longer to reach convergence compared to the non-overlapping method, proved to be the correct strategy. This outcome aligns with the recommendations from [13]. Table 5.4 shows the results: the validation accuracy metric is referred to the validation on the Audioset' subset, while the test accuracy is related to the fine-tuning on DCASE Task1 dataset. N_epochs refer to the necessary epochs to achieve convergence during pretraining.

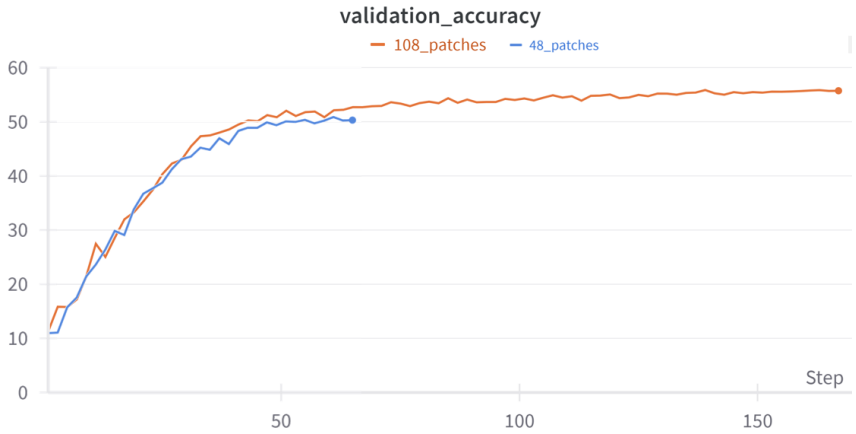


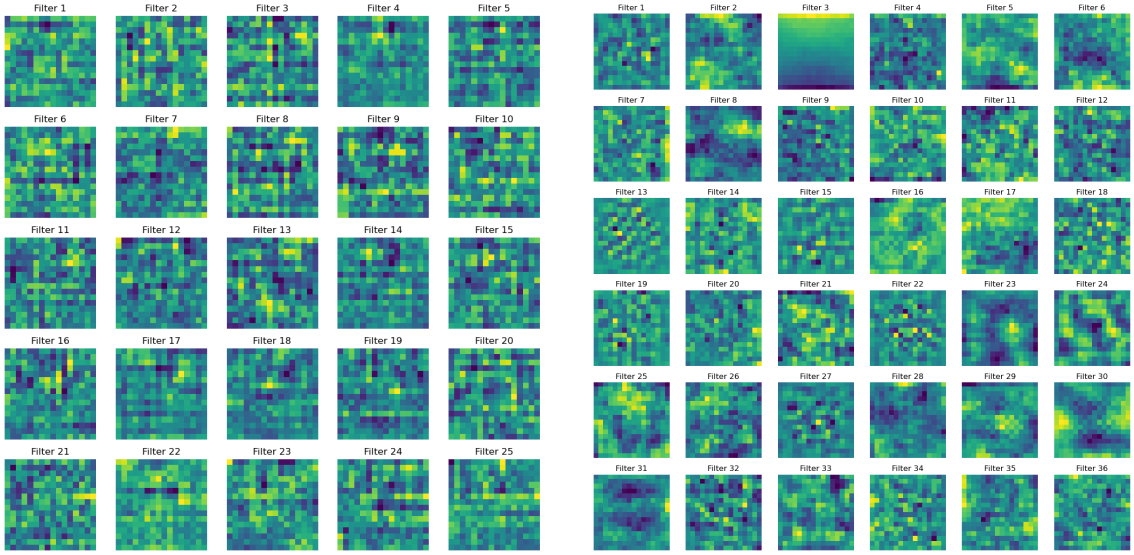

Figure 5.9: Validation accuracy plots on Audioset's subset,using overlapping and not using it.

The effectiveness of training Medium_ViT on a subset of Audioset can be visually checked,

Table 5.3: Comparison of the effects of IDR augmentation in enhancing generalization across devices.

| | **Large_ViT (Data-hungry)** | **Large_ViT (Data-hungry+IDR)** |
|---|---|---|
| **Mean accuracy on seen devices** | 0.524 | 0.567 |
| **Mean accuracy on unseen devices** | 0.521 | 0.552 |
| **Mean accuracy on all devices** | 0.5235 | 0.5638 |
| **Standard deviation for seen devices** | 0.176 | 0.151 |
| **Standard deviation for unseen devices** | 0.200 | 0.171 |
| **Standard deviation for all devices** | 0.1816 | 0.1558 |

looking at the projection filters of the embedding layer,before the fine-tuning stage. Those filters are then compared to the ones belonging to the PaSST deployed to initialize Large_ViT, that instead has been pretrained on the whole set of Audioset. The comparison is not fair because the latter case, with an embedding dimension of 768, has three times more filters compared to Medium_ViT,but the pattern quality presented on the filters show the kind of features the models is extracting.Filters' visualizations are show in figure 5.10.



(a) (b)

Figure 5.10: Projection filters:a) Medium_ViT at the end of its pretraining, b)Large_ViT, pretrained on full Audioset.

#### 5.0.4.1 Knowledge Distillation Results

The knowledge distillation framework was built around Large_ViT and Medium_ViT,treating the former as the teacher, the latter as the student.The techniques applied are the ones described in Subsection 4.3.2:the logit-based, distillation token approach, and the feature based, VIT_KD technique.

The teacher, already pretrained on the whole set of Audioset thanks to the weights taken from ,is

Table 5.4: Training and Performance Comparison of Medium_ViT Models

| Model | Training Set Size | # Patches | N_epochs | Validation Accuracy | Test Accuracy |
|---|---|---|---|---|---|
| Medium_ViT | 2,674,944 | 48 | 40 | 50.27% | 45.05% |
| Medium_ViT | 2,674,944 | 108 | 84 | 55.74% | 47.78% |

then fined tuned on DCASE Task1 dataset for 30 epochs recurring to a structured frequency-wise patchout that halves the patch sequence from 108 to 54 tokens: some of its results are show in 5.2. The plots depicting the evolution of its validation and test accuracy metrics with the associated loss values,and the evoution of the learning rate across the epochs are shown in figure 5.12 and the associa nd the plots of Patchout was also tested with the student.The precautions regarding the application of this regularization technique are due to the possible heavy change it imposes to the spectrograms. Due to this, in order to enable the student and the teacher to actually see the very same images, patchout was synchronized, meaning that the student,during training, would drop the very same patches that the teacher removed for providing its logits/features. Patchout synchronization also came in handy for enabling VIT_KD.
However, as can be seen from table 5.5, Patchout has once again been confirmed to be unsuitable for the Medium_ViT: all the trials performed without it brought better results than the one that deployed it.

Table 5.5: Comparison of Different Medium_ViT Models with KD Techniques and Patchout

| Teacher: Large_ViT | Student: Medium_ViT | | | | |
|---|---|---|---|---|---|
| KD Technique | Patchout | Training Set Size | $\lambda$ | $\tau$ | Test Accuracy |
| - | ✗ | 621840 | - | - | 42.76% |
| - | ✗ | 621,840 | - | - | 42.75% |
| Dist_token | ✗ | 586,560 | 0.5 | 1 | 42.49% |
| Dist_token | ✗ | 586,560 | 0.8 | 1 | 43.55% |
| Dist_token | ✗ | 586,560 | 0.8 | 10 | 45.20% |
| Dist_token | ✓ | 586,560 | 0.8 | 10 | 41.03% |
| Dist_token | ✗ | 621,840 | 0.8 | 10 | 46.83% |
| ViT_KD | ✓ | 621,840 | 0.8 | 10 | 41.95% |

The logits-based method with distillation token improved performance by a substantial margin, though it still falls short compared to that of the teacher. The trials also reveals, as already suggested by [17], that higher values of $\tau$ and $\lambda$,which imply greater reliance on the teacher and more nuanced knowledge transfer, lead to better results.
The experiments on VIT_KD ended on a bitter note: it did not lead to improvements . It has to be said, that the configuration for the loss has been modified with respect to the one discussed in section 4.3.4.To the loss computed in equation 4.17, the additional loss based on the distillation token was added. Nevertheless, one can not but notice that the instability and the great loss term

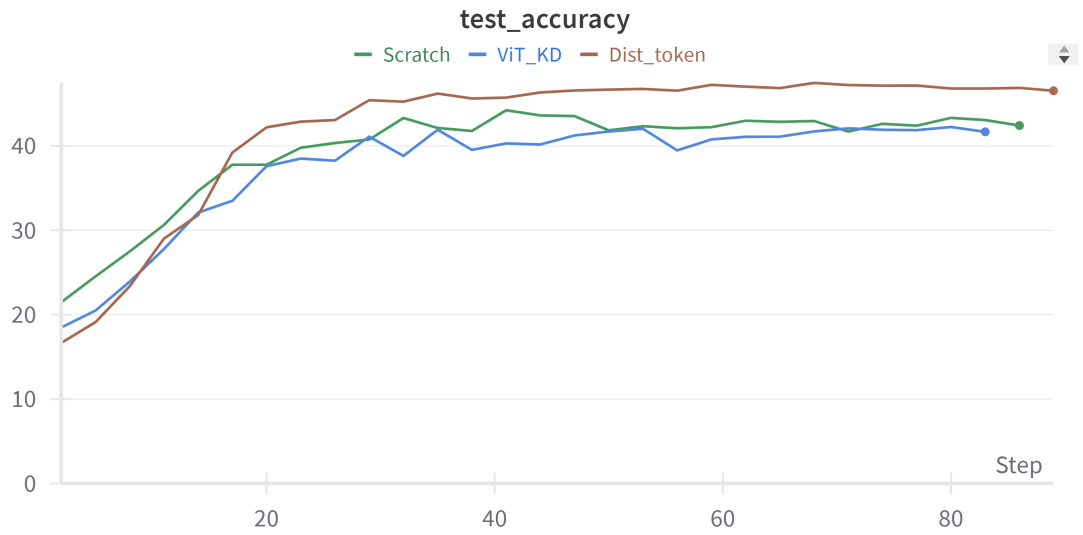



Figure 5.11: Effectiveness of knowledge distillation techniques.

is brought by none other than the generating and mimicking losses, which are the gimmicks of this technique,as can be seen in figure 5.7. The reasons for this phenomenon can be numerous, ranging from the usual problem of not having the right amount of data at disposal, to problem related the learning rate scheduled during training, or the particular nature of spectrograms. Further study is required to fully understand the reasons for this. The test accuracy displayed in tables,are computed averaging the last 5 values that this metrics across the epoch. We stop the training

The learning rate scheduler played an important role concerning the training of the transformers: as in [6],we decided to apply warmup, a technique for which the learning rate starts at a lower value and gradually increases.

| **Parameter** | **Value** |
|---|---|
| Training Set | 586560 |
| Number of Parameters | 85,345,546 |
| Pretrained | True |
| Batch Size | 80 |
| Stride | 10 |
| Patch Out Freq | 6 |
| Patch Out Time | 0 |
| Scheduler | cos_cyc |
| Number of Patches | 54 |
| Alpha Mixup | 0.3 |
| Mixup (1) / MixStyle (0) | 0 |
| Learning Rate (LR) | $1 \times 10^{-5}$ |
| Number of Epochs | 32 |
| Training Time (hours) | 34.0 |
| Training Time (minutes) | 52.0 |
| Validation Accuracy (%) | 95.371 |
| Best Validation (%) | 94.572 |
| Test Accuracy (%) | 51.644 |

Table 5.6: Training Parameters and Results of Large_ViT

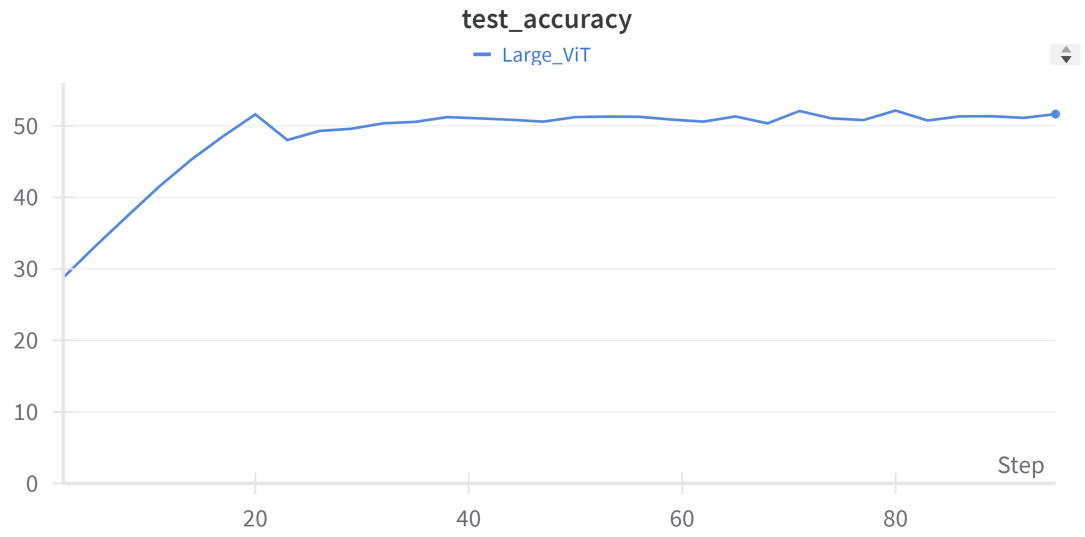
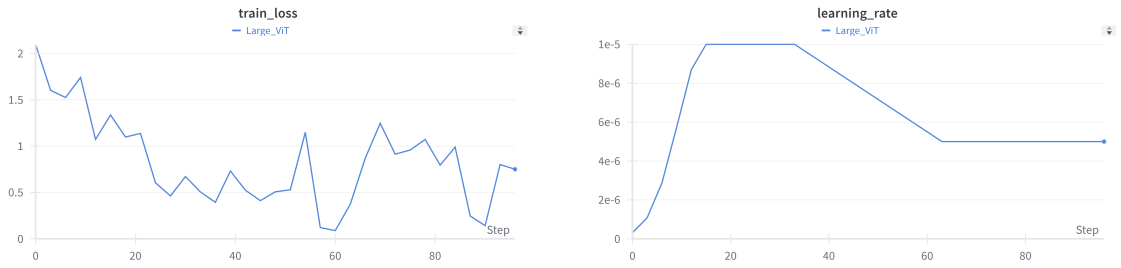



Figure 5.12: Large_ViT test accuracy across the epoch. This model was used as the teacher in the knowledge distillation framework

## 5.1 Research Questions Discussion

[h!] The answers to the proposed research questions that helped us exploring the properties of the Transformer applied to ASC can be resumed following the same section-wise layout with which




(a) (b)

Figure 5.13: Large_ViT:a) Training loss, b)Learning rate,with cosine warmup.

we presented them:

1. **What are the optimal feature dimensions for representing acoustic scenes?Are scalograms a viable alternative to spectrograms in the context of DCASE Task1?**

   Concerning the size of the input images,after trials we decide to adhere to the paradigm set forth by the common approach of the Acoustic Spectrogram Transformer. Smaller sizes do not seem to allow the model to effectively capture changes such as online data augmentation.Additionally, from an architectural perspective,the transformer is not significantly destabilized by the changing of image size: the results shown in the previous section can so be retained valid also for other image sizes, as long as they are not too small. The results achieved on scalograms can not alone define the effectiveness of this visual representation: it is not the intention of this thesis to make an overall assessment and comparison of audio to image strategies.That being said, scalograms seems not to be suited for DCASE Task1, given the nature of the audios itself: the time component proved to be less important with respect to the frequency one, as can be seen from the fact that several techniques focused on (e.g FreqMixStyle) the frequency aspect. Moreover, the computational effort of building scalograms from an audio-dataset, in pre Nevertheless, scalograms are another tool for audio processing,indispensable in several fields,

2. **Can Transformer architectures be optimized to meet the memory constraints of DCASE Task1?What is the Behavior of Different Sizes of Transformers?**

   The hyperparameter search on the transformer architecture highlighted that this kind of operation can be useful indeed for space efficiency.However, the most important one in defining the size is the embedding dimension: a search for its optimal value is not recommended.It is instead advisable to adhere to standard token dimensions rather than deviating too far, as this allows leveraging pretrained weights from existing models for example, and more direct comparisons.

As regard the behaviour of Different sizes of Transformers, the transformer designed to fit within the memory constraints of DCASE Task 1 appears to have compromised some of the advantageous properties of the original transformer sizes. On the other hand, the other two implemented models seem to be perfectly functional.The larger one can be implemented with care on a user GPU, while the smaller one among the two mantains the essential properties of transformers. Within the context of DCASE Task 1, they are valuable for fitting into pretraining and knowledge distillation frameworks.

3. **What is the effectiveness of different data augmentation and regularization techniques in improving the robustness and accuracy of Transformer models? What role do training protocols, such as learning rate scheduling, play in enhancing the performance and stability of Transformer model?**

   Some data augmentation techniques stand out for effectiveness compared to the others. Their deployment in general was of fundamentally important: we encompassed several types, from data agnostic technique, to ones more specific to audio processing, making our procedure valid also for other application that do not stem from this field.

4. **What is the necessity of using a larger dataset for pretraining, and how much data is required to achieve optimal results?What are the results and improvement of Transformer Architecture trained with Knowledge distillation?**

   Pretraining seems to be a compulsory step for transformers when a great dataset is not available. As regard quantification to achieve better results compared to training from scratch, the trials showed that the additional dataset intended for pretraining must contain a quantity of unique samples on the order of millions. Knowledge distillation proved to be a very efficient way: it is not a standalone technique, since it required the teacher to be pretrained in order to be an effective guide for the student.

# Chapter 6

# Conclusion

## 6.1 Conclusions

Transformers are now widely integrated in the field of deep learning, being implemented in nearly all the branches of the latter field of science. Its capabilities have been already recognized, and continue to unfold. Capabilities that come at the cost of the non-trivial management of its needs and its complexity. DCASE Task1, but in general the field of ASC, proved to be an effective benchmark for experiencing this challenges. The limited amount of data in comparison with the field of strictly computer vision is evident, and given the nature of sound, the necessity to develop application for edge devices become prominent. Those considerations led this work to rely on several technique aimed at curbing the requests of this model in terms of sheer amount of Data, taking the long way, avoiding the brute approach of feeding data. And so,many Data augmentation and regularization techniques were studied across this work, looking for taking the best from both the worlds: sound and vision. Different transformer sizes, some of them more standardized,some other more rare where deployed, to study the effect of scaling on this model, proving that there are no such things like properties that resize as the model is resized.Nonetheless, the successful implementation of the Knowledge distillation framework confirms the necessity of investing resources in order to implement Vision Transformer for edge devices.The results achieved with the transformer deployed in terms of test accuracy ranking in DCASE Task1,even the bigger ones, only match the average results.
The contributions I felt to have brought with this work are not ideas or experimentation that are going to push the state of the art forward,nor a perfect case study of the implementation of a transformer architecture in ASC, but rather a thorough approach for dealing with a general deep learning model, starting from the processing of the input, studying its architecture trying to understand the intimate meanings of each of its constituents,studying it from different point of view since it becomes once a teacher, once a student.

## 6.2 Future Work

The presented work explored other fields of deep learning, without necessarily reporting them here, given the higher priority of performing experimentation on the above discussed arguments. These little-opened doors however could play a major contribute for future works:

- **Domain generalization and adaptation:** In this case, those techniques would be aimed at countering the variations in data distribution caused by the recording devices. It would be interesting to implement them leveraging again on the ViT architecture like in [58].Moreover,due to the historical advantage CNNs have over ViTs, there are several framework built for CNNs like the ones in [11],that could be tailored for ViTs.
- **Data compression techniques:** The next step for getting closer to the implementation on edge devices is quantization: techniques tailored for Vision Transformer are emerging[28]. Other techniques like pruning,however, still lack an implementation for ViTs.
- **Tiny AST:** Taking only some sub-blocks from the ViT architecture, it was possible to bring the "Attention" in DCASE task1[61],not just in the training line, but as a full-fledged submitted model. Maybe the key for the implementation of ViTs in edge devices is the alchemy among CNNs and ViTs.

# References


[1] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Amsterdam, Netherlands, 2020. ILLC, University of Amsterdam.

[2] Ambolt.io. Computer vision – image classification and object detection. Blog post, 2024. Accessed: July 15, 2024.

[3] Lei Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS)*, University of Toronto, Microsoft Research, 2014. Curran Associates, Inc. Draft for NIPS 2014 (not camera ready copy).

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, 2020.

[6] DCASE 2023 Challenge Organizers. Dcase 2023 challenge on low-complexity acoustic scene classification, 2023. Accessed: 2024-07-10.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019. {jacobdevlin, mingweichang, kentonl, kristout}@google.com.

[9] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1422–1430, 2015.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. Equal technical contribution by authors marked with *, equal advising by authors marked with †.

[11] Diogo Samuel Gonçalves Fernandes. Exploring metadata in neural networks for uav maritime surveillance. Master's thesis, Mestrado em Engenharia Informática e Computação, Universidade do Porto, October 9 2023.

[12] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[13] Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer. *MIT Computer Science and Artificial Intelligence Laboratory*, 2021.

[14] Aritra Roy Gosthipaty and Sayak Paul. Investigating vision transformer representations, 2022. Equal contribution. Date created: 2022/04/12. Last modified: 2023/11/20. Description: Looking into the representations learned by different Vision Transformers variants. This example uses Keras 3.

[15] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.

[17] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[18] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[19] Tomas Hrycej, Bernhard Bermeitinger, and Siegfried Handschuh. Number of attention heads vs. number of transformer-encoders in computer vision. 2024.

[20] Sanjiv Kumar Jha. Pretraining and model training in large-scale vision models. 2024. Technology and data science leader at Amazon Web Services (AWS).

[21] Dhruv Kabra. Understanding transformers part 2. Published in Version 1, 2023. Version 1, 5 min read, Jul 7, 2023.

[22] Byeonggeun Kim, Seunghan Yang, Jangho Kim, Hyunsin Park, Juntae Lee, and Simyung Chang. Domain generalization with relaxed instance frequency-wise normalization for multi-device acoustic scene classification. *Journal/Conference Name*, 2024.

[23] Khaled Koutini, Jan Schlüter, Hamid Eghbal-zadeh, and Gerhard Widmer. Efficient training of audio transformers with patchout. In *Proceedings of the International Conference on Machine Learning (ICML)*, Linz, Austria, 2021. Institute of Computational Perception & LIT AI Lab, Johannes Kepler University Linz.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[25] Miroslaw Latka, Ziemowit Was, Andrzej Kozik, and Bruce J. West. Wavelet analysis of epileptic spikes. *Institute of Physics, Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland and Video EEG Lab, Department of Child Neurology,*

*Regional Medical Center, ul. Traugutta 116, 40-529 Wroclaw, Poland and Mathematics Division, Army Research Office, P.O. Box 12211, Research Triangle, NC 27709-2211, USA*, January 2003. Dated: January 17, 2003.

[26] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[27] Liyuan Liu, Jialu Liu, and Jiawei Han. Multi-head or single-head? an empirical comparison for transformer training. *University of Illinois at Urbana-Champaign and Google Research*, 2023.

[28] Zhenhua Liu, Yunhe Wang, Kai Han, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. *arXiv preprint arXiv:YYYY.MMDD*, 2023.

[29] Yu Lu, Shanjuan Xie, and Shiqian Wu. Exploring competitive features using deep convolutional neural network for finger vein recognition. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 10(3):123–135, 2024. This work was supported in part by the National Natural Science Foundation of China under Grants 61775172, 61701153, 61371190, and in part by the Open Project of the Key Laboratory of Electronics and Information Applied Technology of Crime Scene Investigation, Ministry of Public Security of China under grant EISI2016003. Corresponding author: Shiqian Wu (e-mail: shiqian.wu@wust.edu.cn).

[30] Peter Xiangyuan Ma, Steve Croft, Chris Lintott, and Andrew P. V. Siemion. A deep neural network based reverse radio spectrogram search algorithm. arXiv preprint, 2024. arXiv:2302.13854v2 [eess.SP] 19 Jan 2024.

[31] Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:XXXX.XXXXX*, 2023.

[32] R. Channing Moore, Daniel P. W. Ellis, Eduardo Fonseca, Shawn Hershey, Aren Jansen, and Manoj Plakal. Dataset balancing can hurt model performance. *Google Research*, 2020.

[33] Tobias Morocutti, Florian Schmid, Khaled Koutini, and Gerhard Widmer. Device-Robust Acoustic Scene Classification via Impulse Response Augmentation. Technical report, Johannes Kepler University, Linz, Austria, Institute of Computational Perception and LIT Artificial Intelligence Lab, 2023. {tobias.morocutti, florian.schmid}@jku.at.

[34] Hieu Quang Nguyen, Abdul Hasib Rahimyar, and Xiaodi Wang. Stock forecasting using m-band wavelet-based svr and rnn-lstms models. *Department of Mathematics, Western Connecticut State University, Danbury, CT, 06810, U.S.A.*, 2023.

[35] Thong Nguyen, Cong-Duy Nguyen, Xiaobao Wu, See-Kiong Ng, and Anh-Tuan Luu. Vision-and-language pretraining: Methods, applications, and future challenges. *Unpublished Manuscript*, 2022. *Corresponding author(s): anhtuan.luu@ntu.edu.sg; Contributing authors: e0998147@u.nus.edu; NGUYENTR003@e.ntu.edu.sg; xiaobao002@ntu.edu.sg; seekiong@nus.edu.sg.

[36] Christopher Olah. Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015.

[37] Optuna. Optuna documentation. https://optuna.readthedocs.io/, 2024. Accessed: July 15, 2024.

[38] Dang Thoai Phan. Comparison performance of spectrogram and scalogram as input of acoustic recognition task. In *Proceedings of the Conference on Acoustic Recognition*, Berlin, Germany, 2021. Joyson Safety System.

[39] Paul Primus and Gerhard Widmer. On frequency-wise normalizations for better recording device generalization in audio spectrogram transformers. *arXiv preprint arXiv:YYYY.MMNNNN*, 2023.

[40] Paul Primus and Gerhard Widmer. On frequency-wise normalizations for better recording device generalization in audio spectrogram transformers. *arXiv preprint arXiv:YYYY.MMNNNN*, 2023.

[41] Zhongnan Qu. *Enabling Deep Learning on Edge Devices*. Doctoral thesis, ETH Zurich, Zurich, Switzerland, Accepted on the recommendation of Prof. Dr. Lothar Thiele, examiner, and Prof. Dr. Olga Saukh, co-examiner 2022. A thesis submitted to attain the degree of Doctor of Sciences of ETH Zurich (Dr. sc. ETH Zurich).

[42] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[43] Florian Schmid, Shahed Masoudian, Khaled Koutini, and Gerhard Widmer. CP-JKU submission to DCASE22: Distilling knowledge for low-complexity convolutional neural networks from a patchout audio transformer. Technical report, Institute of Computational Perception (CP-JKU), LIT Artificial Intelligence Lab, Johannes Kepler University Linz, Linz, Austria, 2022.

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2015.

[45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[46] Andreas Peter Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2205.12607*, May 2023. Accepted by TMLREveryoneRevisions.

[47] Igor Szöke, Milan Skácel, Lukas Mosner, Jan Paliesek, and Jan H. Černocký. Building and evaluation of a real room impulse response dataset. *IEEE Journal of Selected Topics in Signal Processing*, 2019.

[48] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2021.

[49] Magda Alexandra Trujillo-Jiménez. Summarization of video from feature extraction method using image processing and artificial intelligence, January 2018.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[51] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

[52] Voxel51, Inc. Voxel51 documentation. https://docs.voxel51.com/, 2024. Accessed: July 15, 2024.

[53] Tanishq Vyas. A glimpse of the sound. 3 min read, Jul 28, 2022, 2022.

[54] Wikipedia contributors. Algoritmo di canny. https://it.wikipedia.org/wiki/Algoritmo_di_Canny, 7 2024. Accessed: July 11, 2024.

[55] Wikipedia contributors. Mel scale. https://en.wikipedia.org/wiki/Mel_scale, 7 2024. Accessed: July 8, 2024.

[56] Wikipedia contributors. Morlet wavelet. *Wikipedia, The Free Encyclopedia*, 2024. https://en.wikipedia.org/wiki/Morlet_wavelet.

[57] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *International Conference on Machine Learning (ICML)*, 2015.

[58] Jinyu Yang, Jingjing Liu, and Ning Xu. Tvt: Transferable vision transformer for unsupervised domain adaptation. *arXiv preprint arXiv:YYYY.MMDD*, 2023.

[59] Zhendong Yang, Zhe Li, Ailing Zeng, Zexian Li, Chun Yuan, and Yu Li. VITKD: Practical guidelines for ViT feature knowledge distillation. *Tsinghua Shenzhen International Graduate School, International Digital Economy Academy (IDEA), Beihang University*, Year.

[60] S. Y. Yeung. Introduction to image formation and filtering. https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html, 2024. Accessed: July 15, 2024.

[61] Jinyang Yu, Zikai Song, Jiahao Ji, Lixian Zhu, Kele Xu, Kun Qian, Yong Dou, and Bin Hu. TINY AUDIO SPECTROGRAM TRANSFORMER: MOBILEVIT FOR LOW-COMPLEXITY ACOUSTIC SCENE CLASSIFICATION WITH DECOUPLED KNOWLEDGE DISTILLATION. Technical report, National University of Defense Technology, Computer Dept., Changsha, P.R. China and Key Laboratory of Brain Health Intelligent Evaluation and Intervention, Ministry of Education (Beijing Institute of Technology), P. R. China and School of Medical Technology, Beijing Institute of Technology, P. R. China, Changsha, P.R. China; Beijing, P.R. China, 2023. * Kele Xu and Yong Dou are corresponding authors. † Kun Qian and Bin Hu are co-corresponding authors.

[62] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis E.H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *National University of Singapore*, 2021. National University of Singapore, YITU Technology, Institute of Data Science, National University of Singapore.

[63] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. Work done while Hongyi Zhang was affiliated with MIT and David Lopez-Paz was affiliated with Facebook AI Research (FAIR).

[64] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. *arXiv preprint arXiv:2104.02008*, 2021.

[65] Haoran Zhu, Boyuan Chen, and Carter Yang. Understanding why vit trains badly on small datasets: An intuitive perspective. *arXiv preprint arXiv:2301.xxxxx*, January 2023.

[66] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2018. Barret Zoph: barretzoph@google.com, Vijay Vasudevan: vrv@google.com, Jonathon Shlens: shlens@google.com, Quoc V. Le: qvl@google.com.