Deep Learning Project

Killian Susini, Megan Fillion

January 2022

1 Introduction

Like in many tasks, recently developed deep learning architectures showed better results than earlier hand-crafted features based methods. One deep learning architecture that is employed to deal with audio data is a CNN model applied on the spectrogram transformation of the wave data. It is believed that CNNs are well suited to be used on spectrograms because they take advantage of the spatial locality and translation equivariance of the 2d structure. Later, with the success of transformers when applied to images, an analogue technique of staking a self-attention layer on top of the CNN was developed for audio and achieved state-of-the-art results. A recent paper introduced Audio Spectrogram Transformer (AST) [1], an architecture that forgo the CNN entirely, in a manner similar to the ViT architecture [7]. This convolution-free architecture achieved state-of-the-art results in multi-label classification on AudioSet at the time, a large dataset with an ontology of 632 classes. In this project we will solve an audio multi-label classification task (described bellow). We will implement and train an AST model by using using a pretrained ViT model. To test the capabilities of the model we will test it on a large dataset, FSD50K. The paper introducing this dataset trained several classical architectures on the dataset and recorded some metrics, namely their mAP scores. One goal is to train a model that outperform those models. We will also experiment on the impact of several choices that we make to train the network.

1.1 Tasks Description and state-of-the-art

The task that we focus on is a type of multi-label audio classification, called Sound Event Recognition (SER). Given an audio sample, we aim to put appropriate labels on the cause of the sound. There is not many large datasets for SER, especially for multi-label recognition. For mono label recogition, there is ESC-50 and SpeechCommand V2. Both can be automatically labelled very well by the AST Architecture [1] (current SOTA, 95.6% and 98.11% accuracy respectively). For multi-label, there are two large datasets, AudioSet [8] and FSD50K [2]. Multi-labelling is a much more difficult task, and is usually scored using mean Average Precision (mAP), which is the mean of the the average precision of the model on each class. The current state of the art for both datasets are 52.1% and 56.1% (PLSA) [4] mAP respectively. It is interesting to note that PLSA achieved 0.474 on AudioSet, which was beaten by AST soon after (achieving 0.485, itself being beaten later on [6]), both using ImageNet pretraining. Their similar results on AudioSet lead us to believe that a similar score to the state-of-the-art on FSD50K can be achieved with an AST architecture (we won't succeed here, however).

2 Datasets

2.1 FSD50K Dataset

The full dataset we will use is the dataset introduced in [2], refered to as FSD50K. The dataset is composed of 50000 audio files, each labelled with one or more 200 type of sound event. Those classes arranged in an ontological. For example, both *Musical_instrument*, *Plucked_string_instrument* and *Guitar* are classes in the dataset, the last is a child of the second itself a child of the last. It is also possible that a sound event is a child of many sound event, such as *Doorbell*, which is the child of both *Door* and *Alarm*. If the sound of the file was caused by a playing guitar, it will be labelled with *Guitar* and its ancestor, including *Music*. Of course, a file could be labelled with just *Music*. Difficulties encountered in labelling are talked at the end of [], but for our purpose this simple understanding of the concept will do. While some files are labelled with only one labelled, the average number of label is 3.5, and the maximum is 22. Some label have a very low number count (in the 10s). The dataset is split in three part; train (71.9%), validation (8.1%) and test (20%).

2.2 Trimmed Dataset

Because of our limited resources, we will also experiment on a reduced dataset that is as balanced set of 10500 files labelled with one (not multiple) of ['*Music*', '*Human_voice*', '*Animal*', '*Vehicle*', '*Water*']. The classes were selected to have no overlap, and each 2100 sets were created by picking a random subset of the original dataset with those label. The dataset is split into 65% training, 15% validation and 20% test. The task is much simpler but will already allow us to highlight some advantages in the design choices made.

3 Architecture

In our dataset loader, we read in the raw audio file and shape it into a filter bank of size 128x1024. What this means is that our spectrogram has a length of 1024, which is our time dimension, and a height of 128 which represents frequency on a logarithmic scale. In general, spectrograms are formed using this logarithmic scale, also called Mel scale, in order to take into account how humans perceive sound. Amplitude is represented as the intensity of each pixel at a certain x, y coordinate using the decibel scale. For our training set, we augment our data by applying time and frequency masks using the torch audio transforms library. This procedure entails covering rectangular sections from the time and frequency axes in order to help the model to generalize better during training. Then for all input, we normalize the spectrogram using the mean and standard deviation, which was precomputed on the whole dataset using the function in utils.ipynb.

In order to correctly shape the ground truth, we take the class associated with the input, which is represented as a string, and transform it into a one hot vector where the index of the positive element represents the class. We then return the normalized and augmented data along with the vectorized class representation.

We will use a pretrained vision transformer as the base of our Audio Spectrogram Transformer in order for our model to learn faster and generalize more accurately.

The Vision Transformer is a model offered by google-research which is pretrained on ImageNet. As stated in the Audio Spectrogram Transformer paper, using cross-media weights for audio classification leads to higher performances. In a nutshell, ViT splits the input

image into subsections, which we will refer to as 'patches', and feeds each patch along with a class and distillation token into the transformer encoder.

In order to make our input fit into the pretrained visual transformer, we split the spectrogram into 16x16 patches, with a frequency and time stride of 10 in order to have 6x16 time and frequency overlap in each patch, and pass it through a convolutional projection layer. The projection layer has the same output size as for the ViT but seeing that our data has only one channel (as opposed to 3 for an image), we need to average the weights of the Vision parch_embedding layer in order to fit our input.

Along with the patch embedding, we concatenate a positional embedding token to each patch in order to take into account the location of each patch in the spectrogram. This step is necessary in the Audio Spectrogram Transformer model and not in the Vision because we need to take into account temporality when dealing with audio input, which isn't pertinent to images. To the pretrained model, we append a linear layer of size(in:embedding_dim, out:number_of_classes) in order to predict the associated class of the input.

In the forward pass, we load in a batch, pass the data into the patch_embedding layer which will split the input spectrograms into 16x16 patches and create 1D patch embeddings of size 768. Then we initiate the class and distillation tokens for each entry in the batch and prepend these tokens to the patch embeddings. To this data, we then append the positional embeddings for each patch. Subsequently, we pass these embeddings through the layers of the ViT and then predict the class using the last linear layer.

We should ask ourselves why do the (ViT) AST architecture works better than some CNN based models. The key is presented in both the ViT and AST papers [1], and it is the large receptive field that is accomplished in the very firsts layers of the model. Contrary to a series of CNN, two pixels on an image will already be able to communicates some informations to each other regardless of their distances in the image (spectrogram). With CNN, you would need to add strides and/or dilatations to allow two pixels informations to meet in a sub linear number of layers with respect to their distance in the image. This allow the model to mix the information of the whole image very soon in the model, so should allow an easier training.

4 Experiments

In this section, we start by evaluating some design choices, and then procede to train the model on the FSD50K dataset.

4.1 Small vs. Tiny Weights

The google-research team offers us multiple pre-trained vision models with 16 by 16 patches, including 'vit_deit_tiny_distilled_patch16_224', 'vit_deit_small_distilled_patch16_224', and 'vit_deit_base_distilled_patch16_224'. The difference between tiny, small and base is the size of the model used to train the transformer on ImageNet input. Therefore the, the larger the model, the more complex the architecture is, and the heavier the model is to load. Seeing that we did most of our training using the tiny model, we decided to try the base model to compare. But the base level was too heavy to load into our GPU on google colab so we decided to train our AST using the small model. On 100 epochs using the slim dataset, we saw virtually no difference in the map and accuracy scores during training (Plot 3,4) but a significant difference in train time. This comes to show that sometimes more complex

architectures are not necessarily better. For a harder problem however, the Paper did note a boost in performance. We cannot over spend the time training the larger model, and so we will stick with the tiny version.

4.2 Impact of pretraining

We analyse the effect of pretraining on our problem. Using the knowledge (model) that was learned on one task to train a new model one another task is called transfer learning. Since ViT has a similar attention-only based model for which we have pretrained model, we will use that for our problem.

It is well known that in transfer learning is most efficient when transferring the knowledge to a related task. However, available pretrained ViT (Vision Transformers) architectures were trained on ImageNet (So for a vision classification task), not at all on an audio dataset. While this model also input 2d "images", those are gray scale spectrograms, not 3-colored images, where one dimension is time and the other is frequency instead of the spatial nature of an image dimension. The difference between the two tasks is very high. So it is interesting to see how much the pretraining can help us, and how much letting the pretrained layers adapt to our task will help.

The accuracy results on the test set are found in table 1. It is clear the pretraining on ImageNet already makes our task much easier, even if we do not adapts the weights of the pretrained ViT Architecture. Without pretraining, there is a slow but steady learning being done. However, even after 100 epochs, it still fall short of one epoch of learning on the pretrained model. In fact, even if we freeze the weights on the ViT layers (so we only update the last fully connected layer that maps the ViT output to our 5 classes), it still performs much better within the time frame. Finally, letting the pretrained weights adapt to the new task shows a very high performance boost, helping to achieve the best score on the slim dataset. Freezing the weights however did make the training faster, but the time boost is not worth the loss in performance. As noted last section, whether the size of the ViT model we use had little impact on accuracy but significantly stretch the training time. For the FSD50K dataset, we will use pretraining and we won't freeze the weights.

4.3 Data Augmentation and overfitting on multi-label audio recognition

It is usual to augment a dataset for training to avoid the problem of overfitting and in general augment the performance of a model cheaply. Introduced in [3], a simple and effective data augmentation consist in randomly hiding a chunk of the time (x-axis) and a chunk of the frequencies (y-axis).

When we first train the model we forgot to activate this feature during training and ended up with an interesting result. During training, overfitting is usually easily detectable by a divergence between the training loss and the validation loss, where the training loss continue to decrease while the validation loss begins to increase. As shown in Plot 5, this did not happen, and therefore we could have concluded that we were not overfitting. However, in Plot 1, the mAP on the validation set shows a slow and steady decline starting around the 10nth epoch mark. Our hypothesis is that this is an example of overfitting that was mask by the multi-binary cross entropy loss that we used. The loss compute one binary cross entropy loss between the predicted probability (after a sigmoid on each output) and the actual label per label, and then sum them up (all with equal weight) to produce the

total loss. Because some labels have so few associated data points, it is easy to imagine overfitting on them while doing better and better on the other labels. So the idea is that the improvement on the easier labels overcome the loss on the harder ones, while the mAP will shows the slow loss in performance. A good idea would be to do a weighted sum of the cross entropy losses (weighted by number of labels) to not penalise the model for failing to fit the few data points it has on some of the labels (we did not try this, however). Activating the data augmentation allowed to stop the overfitting in addition to a general improvement of the results (as expected). Plot 6 and 5 showed the training and validation loss throughout the training

4.4 Overlap

As explained in the architecture part, the model uses 16 by 16 patches as "words" in the transformers. The patches can overlap or not, we can controle this by setting the frequency stride and the time stride of the first Convolutional layer of the model (this still count as a convolution free model, it is just taking advantage of the already implemented function). The trade off between overlaping patches and not is that more overlap means more "words" in the transformers to describe the input which increases the computation and training time, but could also help achieve a better score. In [1], the difference in performance was analysed on the AudioSet dataset, and the overall result was that more overlap lead to better results. We also did an experiment on this factor, and because our computing resources are limited we were also interested in how much slower it made the training of the network. The results on the test set are in table 1, and plot 2 shows the accuracy on the validation set throughout the training. whether or not we retrained the ViT layers, the overlaping patch produce little differences on the final accuracies. (We lost some results data, so we cannot show them, but at least on the small dataset and within 100 epochs there was no effect on the use of patch overlap.)

4.5 Results on FSD50K

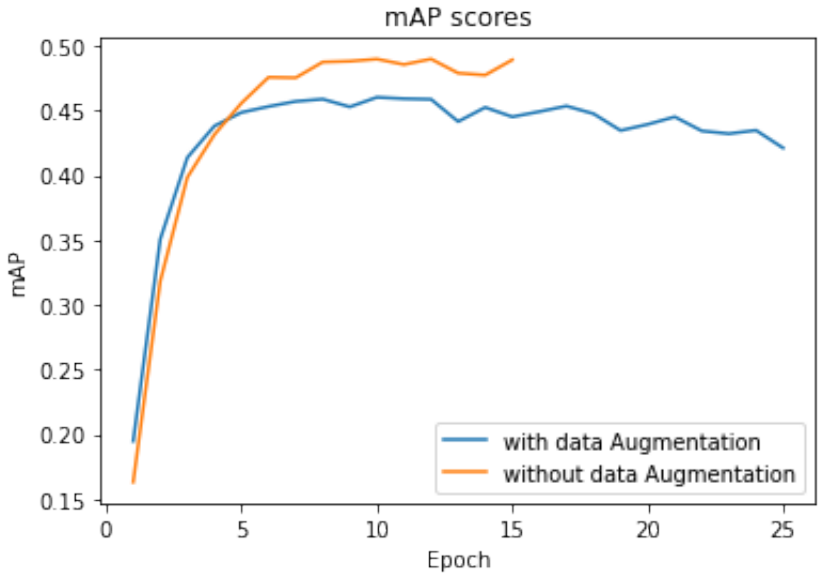
Here we present our result on FSD50K. We manage to get a final mAP score on the test set of 0.46 after 15 epochs (see table 2). To achieve this result, we had to use a batch size of 16, no overlap between the 16 by 16 patches and the tiny version of a pretrained architecture. The main reason for those choices were to make the training time on the whole dataset manageable. On validation, the mAP approaches 0.50 (See Plot 1), which can indicate that achieving a better score should be feasible with more time and tuning of our implementation.

5 Conclusion and possible exploration

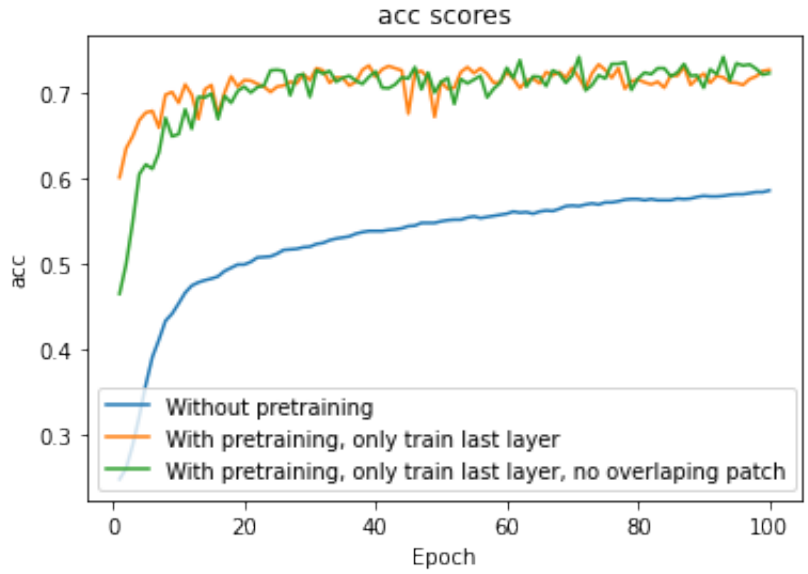
We successfully train a performant model on the hard task of Sound Event Recognition with multiple labels. We saw that many choices in the architectures impact heavily both the training time and the score we can achieve. Possible explorations would be to look at which of the labels our model had an easier or harder time to predict in FSD50K, in a manner similar to what was done in the paper introducing the dataset [2]. We also could try another architecture on the dataset which aim to the same kind of spatial mixing (the large receptive field we were referencing) as AST/ViT, but only through Convolutions [5]. Such an architecture may be simpler to train than a Transformers based one.

6 Plots and Tables

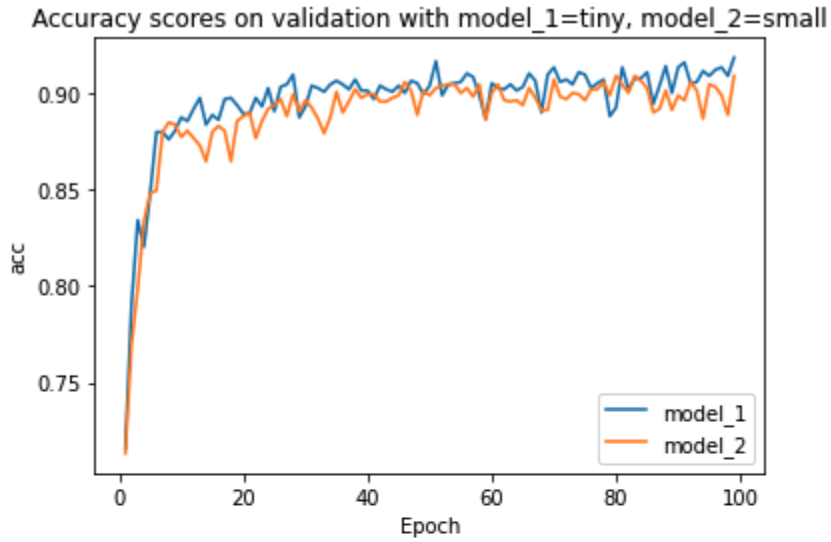
Plot 1

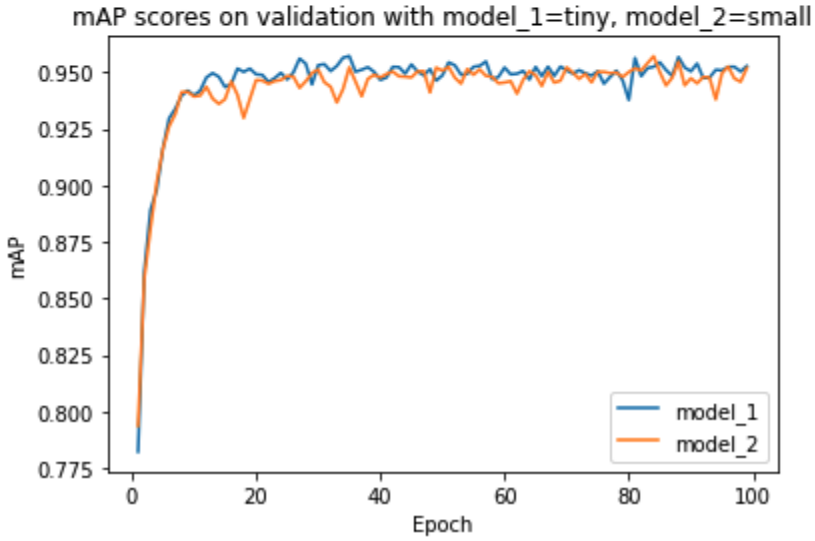


Plot 2

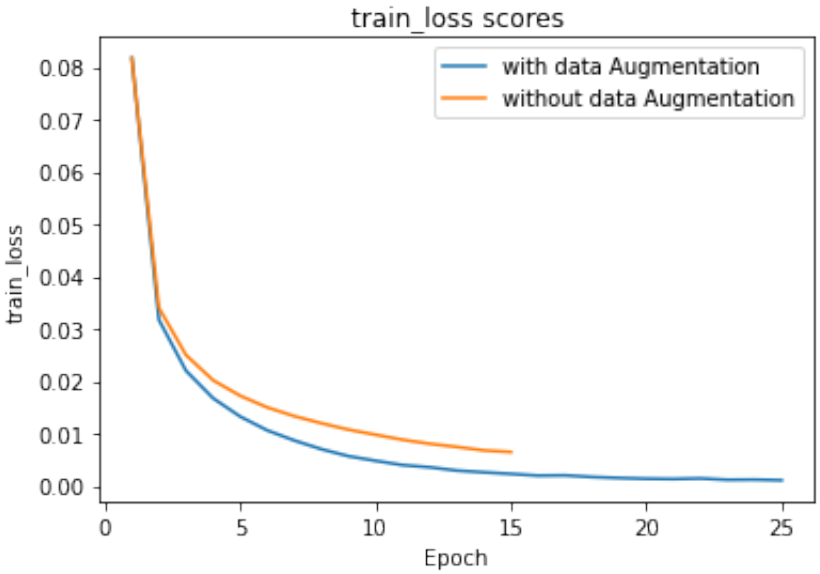


Plot 3 (accuracy) & 4 (mAP)

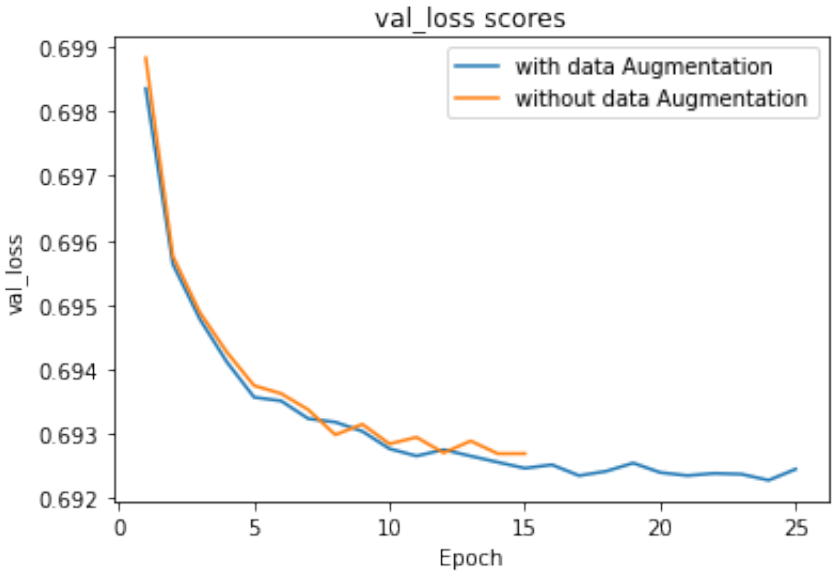




Plot 5 (colors are inverted !!!)



Plot 6 (colors are inverted !!!)



Slim Dataset, 100 Epochs, 8 batch size	accuracy (Test)
Small ViT pretrained	0.908
Tiny ViT pretrained	**0.909**
Tiny ViT pretrained (frozen weights)	0.719
Tiny ViT pretrained (frozen weights, no patch overlap)	0.75
Tiny ViT trained from scratch	0.584

Table 1

FSD50K 16 batch size, no patch overlap	mAP
Tiny ViT pretrained (no data augmentation)	0.406
Tiny ViT pretrained	**0.464**

Table 2

7 Citations

1 Yuan Gong, Yu-An Chung, James Glass "AST: Audio Spectrogram Transformer"

2 Eduardo Fonseca, Xavier Favory, Jordi Pons "FSD50K Dataset"

3 SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition

4 PSLA: Improving Audio Tagging with Pretraining, Sampling, Labeling, and Aggregation

5 Patches Are All You Need?

6 Attention Bottlenecks for Multimodal Fusion

7 Visual Transformers: Token-based Image Representation and Processing for Computer Vision

8 Audio Set: An ontology and human-labeled dataset for audio events