Deep Learning project: Audio Scene Classification

Victor Barberteguy*- Presented on 08/12/2023

Abstract

We investigate some possible methods to tackle the DCase21 challenge for audio scene classification. First, we study and test methods based on Convolutional Neural Networks, and show that lighter, specifically-designed networks can achieve close to state-of-the-art performance on such tasks. We explore how some of the parameters have influence on the accuracy or on the training time of the model. Then, following the rising interest in foundation models, we test AudioCLIP that embeds in the same space audio and text features without finetuning. As this foundation models are very complex, we rather focus on data analysis to explain the performance of AudioCLIP.

Keywords: Audio Scene Classification, Acoustic data, Convolutional Neural Network, Foundation Models

1 Introduction

Over all the tasks that Artificial Intelligence aims to solve, audio-related tasks such as pattern recognition, audio generation, are particularly challenging: the data is often noisy, less abundant in open-source, and cumbersome in terms of memory. Therefore, the models designed are often big models that train millions of parameters on huge audio datasets. To address this issue, the **DCase challenge** (Detection and Classification for Acoustic Scene Events) proposed in 2021 a task (**Task A**) that aimed at building low-compklexity architectures for audio scene classification[1].

In this study, we investigate how audio data is processed in 2, and how state-of-the-art audio pattern recognition models like *Resnets* [He et al. 2015] or *Pretrained Audio Neural Netowrks* [Kong et al. 2020] classify it. Then, we focus on a particular solution proposed to the Task A of the DCase challenge in 3.2, namely *SP4SC* [Puy et al. 2021] and compare it to the previous ones. Finally, we came up with the idea that foundation models could also well address this task without further training. This idea is explored in 4. All the codes and resources are available on this page[2].

Note: The neural networks were trained with a Tesla K80 GPU 12GB. All the other code was run on a 2,3 GHz Intel Core i5 quad core. The training times indicated are relative to these specs.

2 Dataset

For this challenge, the provided dataset is the **TAU Urban Acoustic Scenes 2020 Mobile, development dataset**[3]. The latter is quite big (approximately 30Gb). It contains audio recordings of the ambient sound in dfferent places, to which we affect a number between O and 9. The recordings come from different urban places. The original dataset is quite heavy (around $30Gb$). To save computational resources, we randomly pruned some recordings within each class to keep the dataset balanced. Eventually, the studied dataset D_f weighs $10.6Gb$ and is composed of 800 samples of each class as can be seen in Fig. 1.

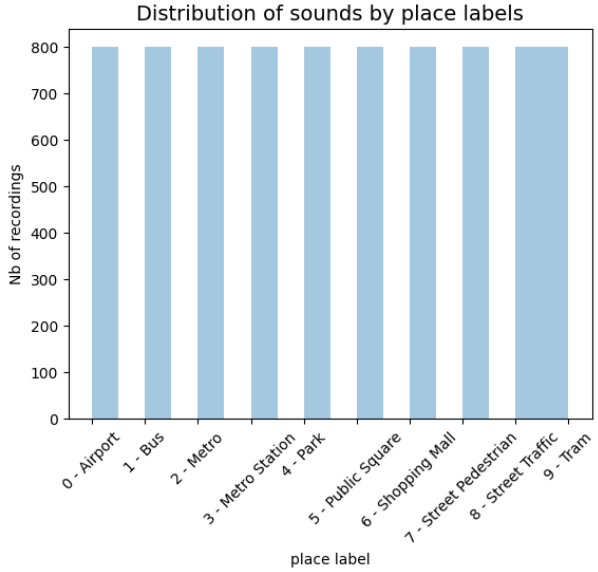

Figure 1: *Distribution plot of audio recordings in the pruned dataset that we study. The distribution is uniform, as there are* 800 *samples for each of the* 10 *classes.*

However, such audio $*.wav$ files cannot be directly fed to Convolutional Neural Networks (CNNs), as the latter expect an grid-like input. To tackle this issue, **LogMel Spectrogram (LMS)** is computed for each audio sample. LMS is a representation of the spectrum of an audio signal, widely used in audio signal processing and speech recognition, as it captures the time and spatial dependencies that are intrinsic to the audio recording. To obtain a LMS, the audio signal is first sampled (in our case, at $44,1KHz$) and then divided into short overlapping frames ($20ms$ with 50% overlap). Each frame undergoes a Fourier transform to convert the signal from the time domain to the frequency domain. The resulting spectrogram is then transformed into Mel-frequency bins, which are perceptually spaced to mimic human hearing. The logarithm of the Mel spectrogram is taken to compress the dynamic range, providing a more compact and human-perceptible representation. Practically, we reused the class written by the authors of [SP4SC], which itself uses the `librosa`[4] library. When transposing the matrix, we end up with matrices of size $(513, 42)$ that we can feed to CNNs, and that look like Fig. 2.

3 Convolutional Neural Networks

Up until now, CNNs have been widely use for Audio Classification, as the LMS computed from audio recordings have strong time and frequency dependence (like images have a spatial dependence). However, such CNNs are very often cumbersome and include several millions or billions of parameters. Thus, one task of the DCase challenge was to design lighter architectures (an exemple is devel-

* victor.barberteguy@polytechnique.edu
Department of Computer Science, Ecole Polytechnique

[1] https://dcase.community/challenge2021/task-acoustic-scene-classification-results-aKim2021b

[2] https://github.com/VictorBbt/Acoustic_Scene_Classification

[3] https://dcase.community/challenge2020/task-acoustic-scene-classification

[4] https://librosa.org

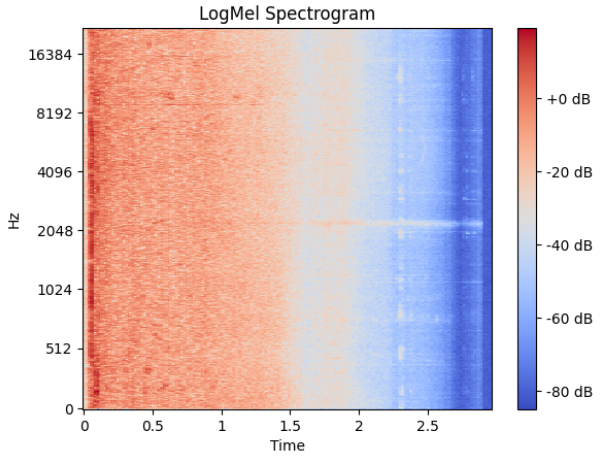

Figure 2: *Distribution plot of audio recordings in the pruned dataset that we study. The distribution is uniform, as there are* 800 *samples for each of the* 10 *classes.*

oped in 3.2) that deal with AC as efficiently as bigger networks such as 3.1

3.1 Pretrained Audio Neural Networks (PANNs)

PANNs [Kong et al. 2020] have been the baseline over the past few years, as such NNs able to transfer to other pattern audio recognition tasks while ensuring competitive performance. Moreover, the architecture is lighter that that of **ResNets** [He et al. 2015], that achieve very good results but are very deep. One well-known trick to backpropagate the gradients without vanishing effect is to add residual connections, that are shortcut connections between the convolutional layers. Hence the gradient is straightforwardly backpropagated to shallower layers allthewhile going through the architecture in a classical way.

Rather than building such a deep architecture, [Kong et al. 2020] came up with two different architectures. The first is quite classical, composed of several convolutional blocks (4 to 6 blocks for CNN6 and CNN14 models) composed of a convolutional layer with kernel size $(5x5)$ along with ReLU and Pooling. In an novel architecture named **Wavegram-LogMel CNN**, the authors want to mitigate the fact that LMS do not capture well the intrinsic frequency dependencies of the sound recordings. They designed a parallel architecture where LMS are treated on one side with 2-dimensional convolutional Blocks on one side, and on the other side 1-dimensional convolutional blocks operate on signals called **Wavegrams**. Wavegrams are a transformed version of **time-domain Waveform** of the signal that tend to capture better the intrinsic frequency relations between close samples from a given audio file. Wavegrams and feature maps extracted from the LMS are then concatenated and fed to a last CNN, which eventually outputs the prediction.

This PANN is pretrained with the *AudioSet* dataset [ref], which contains 1.9 million audio clips and 527 classes. PANNs achieve a state-of-the-art performance of 0.434 accuracy for the Wavegram model and 0.431 for *CNN14*. Moreover, as the dataset is very diverse (and is also augmented and balanced during training), the pretrained network shows good performance on other pattern recognition dataset. That makes it a solid baseline to start from in the DCase challenge. Actually, many solutions proposed for this task of the DCase21 challenge build upon a ResNet or a PANN architecture.

3.2 Separable Convolution and Batch Normalization merging

The challenge A that [Puy et al. 2021] try to address is to make pattern recognition networks less complex. Therefore, they used two tricks to shrink the number of parameters (and the total training time). The base architecture is the lightest PANN model, namely CNN6.

3.2.1 Separable Convolutional Layers

As described in 3.1, in the CNN6 architecture, the output f_{out} is solution of

$$f_{out} = c_{5\times5}(f_{in}), \quad (1)$$

where $c_{5\times5} : \mathbb{R}^{n\times c_{in}} \rightarrow \mathbb{R}^{n\times c_{out}}$ denotes a regular convolution layer with a kernel of size 5×5 (without bias), producing a feature map c_out with spatial dimension n and c_{out} channels. This layer contains $25 \cdot f_{in} \cdot c_{out}$ parameters.

Instead of applying a standard convolutional layer, they apply **depthwise separabale convolution**, which is an idea explored in [Howard et al. 2017]. The idea stems from the fact that a standard convolution composes one operation combining all the input features maps (a kernel is shared by all the channels) and a filtering step that applies the convolution. Rather than doing this in one step, we first **combine** the features with a first convolution $g_{1\times1} : \mathbb{R}^{n\times c_{in}} \rightarrow \mathbb{R}^{n\times c_{out}}$, and then **filter** separately each obtained feature map by applying two convolutions $h_{3\times1} : \mathbb{R}^{n\times c_{out}} \rightarrow \mathbb{R}^{n\times c_{out}}$ and $h_{1\times3} : \mathbb{R}^{n\times c_{out}} \rightarrow \mathbb{R}^{n\times c_{out}}$. The framework of such designed separable convolution blocks (SCB) is shown in Fig. 3 Eventually, the numbers of learnable parameters is eventually $c_{out}(c_{in} + 6)$. The whole architecture is simply made of 4 SCBs followed by a *Global Pooling* and a *MultiLayer Perceptron* (MLP).

3.2.2 Batch Normalization Merging

During training, SP4SC uses classical **batch normalization layers** (BNL) after each SCB. As there is one batch normalization per channel, and as there are 4 parameters within each BNL, $4c_{out}$ parameters are added in each SCB. Yet, at test time, the batch normalization is added within the convolutional layers as a bias, hence adding only c_{out} parameters during test phase. After training, the l^{th} feature map is of the form:

$$\tilde{y}^{(l)} = \gamma^{(l)} \left[\frac{w^{(l)}_{3\times1} * \tilde{x}^{(l)} + w^{(l)}_{1\times3} * \tilde{x}^{(l)} - \mu(l)}{\sigma^{(l} \right] + \beta^{(l)} \quad (1)$$

In this equation, $*$ is the 2D convolution, $\mu^{(l)}$, $\sigma^{(l)}$, $\gamma^{(l)}$, and $\beta^{(l)}$ are the running mean, running standard deviation, scale, and bias parameters of the batch normalization layer on the l^{th} channel. $\tilde{x}^{(l)}$ denotes the l^{th} channel of $g_{1\times1}(X)$, and $w^{(l)}_{1\times3}$, $w^{(l)}_{3\times1}$ represent the kernels of the SCBs. At test time, we compute

$$\tilde{y}^{(l)} = \tilde{w}^{(l)}_{3\times1} * \tilde{x}^{(l)} + \tilde{w}^{(l)}_{1\times3} * \tilde{x}^{(l)} + \beta^{(l)}, \text{with} \quad (2)$$

$$\tilde{w}^{(l)}_{\cdot\times\cdot} = \frac{\gamma^{(l)}}{\sigma^{(l)}} \tilde{w}^{(l)}_{\cdot\times\cdot} \quad (3)$$

$$\tilde{\beta}^{(l)} = \beta^{(l)} - \frac{\gamma^{(l)}}{\sigma^{(l)}\mu^{(l)}} \quad (4)$$

Finally, at test time, c_{out} additional parameters $\tilde{\beta}^{(l)}$ are involved to compute batch normalization.

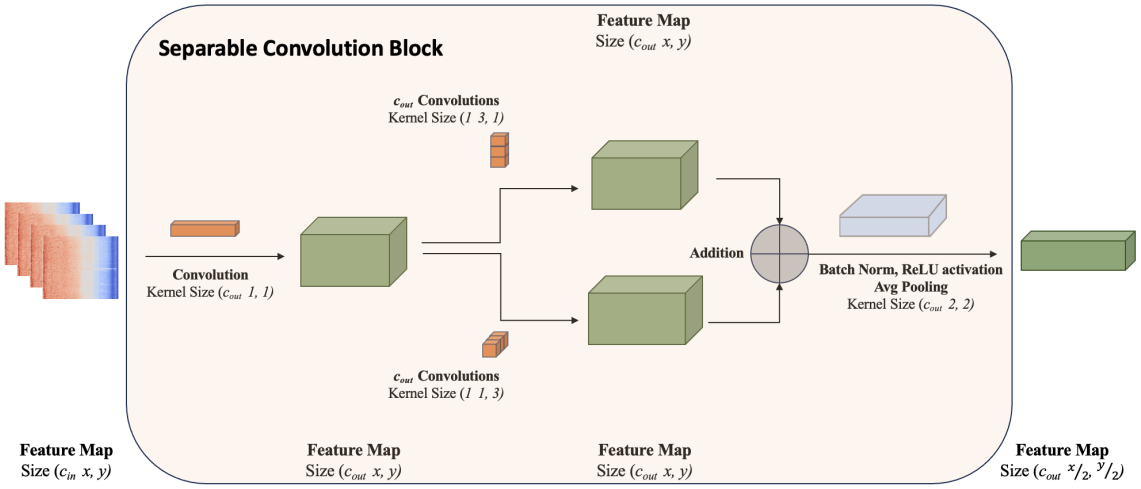

Figure 3: *Separable Convolution Block. In a first step, the input feature maps are combined with a 1×1 kernel. Then, two channel wise convolutions are applied. Once concatenated, the output goes through a Batch Normalization Layer, ReLU Activation layer and a Pooling layer.*

4 AudioCLIP

Rather than training from scratch a CNN architecture, one could think to come up with a strong pretrained model that could achieve good performance off-the-shelf. This is the ambition of foundation models that use huge datasets to tackle complex, multi-modal tasks such as text-to-image or text-to-audio. The latter is particularly suited to our problem, as we can provide textual descriptions of the places we are recording our data.

4.1 Theoretical background

The CLIP (Contrastive Language-Image Pretraining) [Radford et al. 2021] model, developed by OpenAI, is a versatile vision-language model that learns to understand and generate textual descriptions for images. AudioCLIP [Guzhov et al. 2021a] naturally adds the audio modality to this existing model. The key to AudioCLIP's functionality lies in its pretraining process, which involves exposing the model to a large dataset containing images, corresponding natural language descriptions and audio recordings.

During pretraining, the model learns to create a shared embedding space where audio recordings, images, and their textual descriptions are close to each other. This enables CLIP to understand the semantic relationships between images, text and audio. Whereas **transformers** [Raffel et al. 2023][Parmar et al. 2018] are used to embed text and images, the audio head is mainly composed from a CNN-based architecture called **ESResNeXt** [Guzhov et al. 2021b].

4.2 Classification for Audio Data

As the training of such foundation model is complex and extremely computationally costly, we use the pretrained model made available by [Guzhov et al. 2021a] and directly use our dataset. To classify our audio recordings, we manually designed text strings that describe the audio classes (e.g., *"Recording inside a travelling bus"*). Then, we compute the embeddings for the text strings and the audio recordings from our batched dataset. As the embeddings lie in the same space, we can compute the *cosine similarity* between the text embeddings matrix Z_{text} and a given datapoint embedding Z_d: $CS(Z_{text}, Z_d) = Z_d \times Z_{text}^T$. When applying a softmax, we get the probability to belong to each class, thus we naturally choose $argmax(softmax(CS(Z_{text}, Z_d)))$

Model / *Phase*	*ResNet*	*CNN6*	*SP4SC*
Training Time	*21 285 530*	*4 574 287*	*64 335*
Test Time	*21 285 530*	*4 574 287*	*62 474*

Figure 4: *Array of the number parameters at training time and at test time for the three models studied.*

5 Experiments

In this part, we describe our experiments made on the different models explained in 3 and 4. Yet, while the training of CNNs is long but feasible with a standard GPU in *GoogleColab*[5], playing with *AudioCLIP* parameters is unfeasible. In a first part, we compare the training times of different CNN architectures, and the impact of the tricks implemented by SP4SC with an ablation study. In the second part, we focus on the latent space embedding of *AudioCLIP*, exploring how our audio recordings and text tokens are spatially distributed and classified.

5.0.1 CNNs

To recall, the task A of the DCase21 challenge was to build a simple architecture: it must contain less than $65k$ parameters. This has mutitple advantages: the model is cheaper in terms of memory, computation and training time. One of the main disadvantage is - of course - the drop in the prediction accuracy of the model.

We investigated the architectures of 3 different models: ResNet [He et al. 2015], CNN6 [Kong et al. 2020] and SP4SC [Puy et al. 2021]. ResNet is the model with the most parameters, while the two others have less, as summed in Table 4.

After having initialized the networks, we set common hyperparameters (Batchsize, Num Workers, dropout, Learning Rate, Weight Decay, ...) and train the networks on the dataset described in 2. For the *ResNet* model, we only retrain the first convolutional layer and the output layer of the MLP (the whole training requires too much resources). We see in Fig. 5 that indeed the *ResNet* models outper-

[5]https://colab.research.google.com

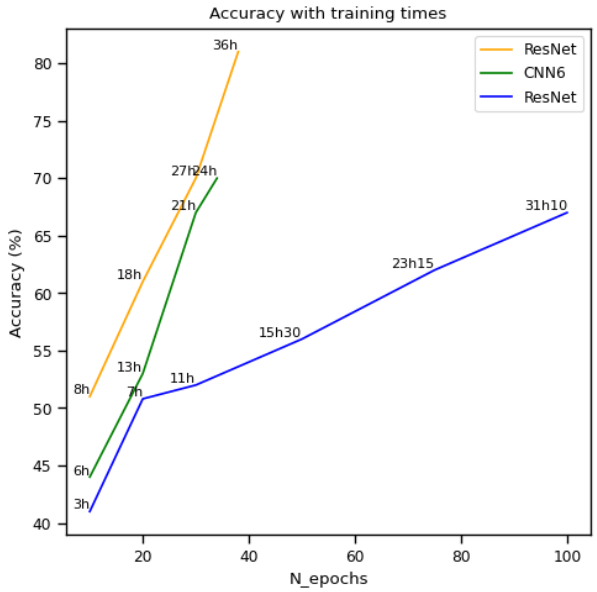

Figure 5: *Plot of the accuracy of the three models at each epoch on test set. Every ten epochs, the training time is indicated directly on the graph.*

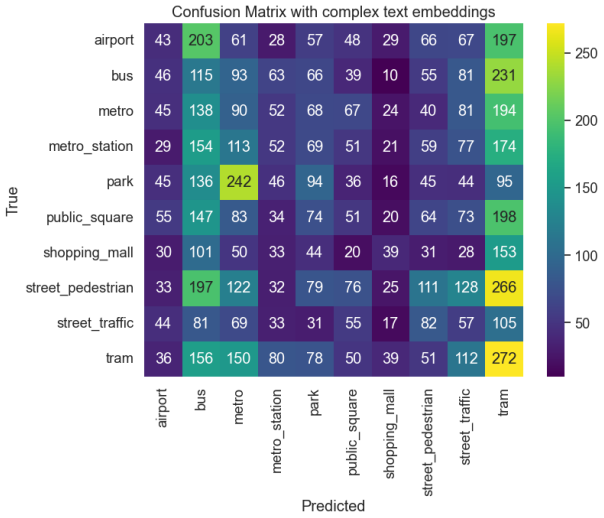

Figure 6: *Confusion Matrix of the classification of audio recordings with complex text tokens, using AudioCLIP.*

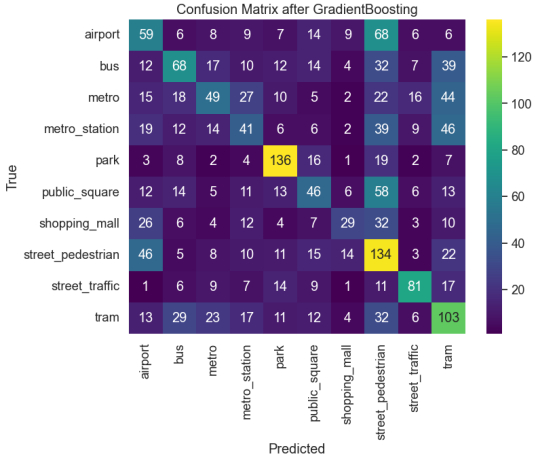

Figure 7: *Confusion Matrix of the classification of audio recordings with complex text tokens using AudioCLIP and Gradient Boosting.*

forms the others in terms of accuracy. However, the training time is a lot higher than that of *CNN6* and *SP4SC*.

Comparing *CNN6* and*SP4SC* during the training phase amounts to study the impact of the channel-wise convolution on the training, as it is the only difference between the two networks (indeed, the batch normalization occurs only when the model is trained). What we can empirically see is that ...

Finally, when loading the network and run the model on the test and validation set, we see that on our studied dataset, the inference is done in $T_{normalBN} = 14min59sec$ to run when doing inference on the network. If we use the batch normalization merging trick, we get $T_{BNMerged} = 14min30sec$. These computation times are similar, as well as the number of parameters (as seen in 4, there are only $2k$ less parameters. However, this trick may well have a more blatant impact when using the full dataset, 3 times bigger than ours.

5.0.2 AudioCLIP

First, it is worth noting that even when taking the *AudioCLIP* pretrained model off-the-shelf, computing the audio embeddings for our dataset is quite long, as it took around $1h45min$. Then, we compute two sets of text embeddings for our classes: one is called "complex text" (e.g., *'recording of a large urban outdoor public square'*) and the other "simple" (e.g., *'public square'*). The spatial distribution of these 20 text embeddings can be found in Appendix A.

When computing the prediction, we get surprisingly bad results, with results close to random prediction for both sets of text embeddings. Some classes like *bus, metro* and more importantly, *tram*, attract a lot of audio embeddings, as can be seen Fig. 6.

To explore how this misclassification occured, we performed a Principal Component Analysis (PCA) [Maćkiewicz and Ratajczak 1993] and only kept the two principal dimensions to get a glimpse of how the dataset in distributed in the *AudioCLIP* latent space. The results are displayed in Appendix B. We see that the audio recordings superimposed in the 2D space. No matter the class, the audio recordings have the same spatial distribution in the 2D principal latent space. On top of that, the text embeddings are also very close to each other, which makes the classification even more difficult.

To address this problem, we use the **Gradient Boosting Classifier** from the `sklearn` [Pedregosa et al. 2011] library to improve the classification. This approach is quite successful: by taking $n = 50$ estimators, we get an accuracy of 0.37 at the end of the training (that took $20min20sec$. The results are shown in Fig 7. We observe that the *bus* and *metro* classes are better distinguished from others, while *tram* has still a high error rate.

5.0.3 Discussion

The accuracies are summed up in the Table 8. Unsurprisingly, the ResNet model significantly outperforms the others, but is also several orders of magnitude longer to train. It remains a solid baseline, but it cannot be retained as the solution to the Task A of the DCase21 challenge.

Model / Result	ResNet	CNN6	SP4SC	AudioCLIP
Last Epoch	38	34	100	X
Accuracy (%)	81	70	67	0.38

Figure 8: *Accuracies of the models investigated.*

Then authors of SP4SC have proposed a valid solution to the challenge by modifying the CNN6 architecture with **separable convolutions** (SCB) and **batch normalization merging at test time**. Altough the effect of the latter is not significant in terms of accuracy and training time, the implementation of SCBs are responnsible for a notable loss of accuracy (yet sparing some training time). The gain in computation time is easily explainable, as we have less learnable parameters as seen in 4. The loss of accuracy, however, is theoretically unclear, as the operations performed on the feature maps seem similar.

6 Conclusion

Throughout this work, we investigated several models used for Audio Scene classification, focusing on different architectures and methods to reduce the complexity of models. Although big models logically classify with better accuracies, some methods like SP4SC proposed by [Puy et al. 2021] can limit this loss of accuracy while reducing the number of parameters. Furthermore, multimodal foundation models like *AudioCLIP* [Guzhov et al. 2021a] that have gained interest over the last few years are very promising inasmuch as classification problems very often have natural textual descriptions. Taking a step back on this study, we propose two possible directions for future work:

- **Interpretability of convolutions:** understanding how differently classical and separable convolutions affect a image may well help to design lighter, more-refined architectures that will perform better on such audio recognition tasks
- **AudioCLIP embeddings:** we would like to separate audio recordings better and express more insightful text descriptions that are more spatially dispersed in the latent space. Fine-tuning the audio head of *AudioCLIP* on the TAU Urban Acoustic Dataset could also bring better results.

References

GUZHOV, A., RAUE, F., HEES, J., AND DENGEL, A., 2021. AudioCLIP: Extending CLIP to Image, Text and Audio, June. arXiv:2106.13043 [cs, eess].

GUZHOV, A., RAUE, F., HEES, J., AND DENGEL, A., 2021. ESResNe(X)t-fbsp: Learning Robust Time-Frequency Transformation of Audio, Apr. arXiv:2104.11587 [cs, eess].

HE, K., ZHANG, X., REN, S., AND SUN, J., 2015. Deep Residual Learning for Image Recognition, Dec. arXiv:1512.03385 [cs].

HOWARD, A. G., ZHU, M., CHEN, B., KALENICHENKO, D., WANG, W., WEYAND, T., ANDREETTO, M., AND ADAM, H., 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Apr. arXiv:1704.04861 [cs].

KONG, Q., CAO, Y., IQBAL, T., WANG, Y., WANG, W., AND PLUMBLEY, M. D., 2020. PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition, Aug. arXiv:1912.10211 [cs, eess].

MAĆKIEWICZ, A., AND RATAJCZAK, W. 1993. Principal components analysis (pca). *Computers Geosciences 19*, 3, 303–342.

PARMAR, N., VASWANI, A., USZKOREIT, J., KAISER, , SHAZEER, N., KU, A., AND TRAN, D., 2018. Image Transformer, June. arXiv:1802.05751 [cs].

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12*, 2825–2830.

PUY, G., JAIN, H., AND BURSUC, A. 2021. Separable convolutions and test-time augmentations for low-complexity and calibrated acoustic scene classification. Tech. rep., DCASE2021 Challenge.

RADFORD, A., KIM, J. W., HALLACY, C., RAMESH, A., GOH, G., AGARWAL, S., SASTRY, G., ASKELL, A., MISHKIN, P., CLARK, J., KRUEGER, G., AND SUTSKEVER, I., 2021. Learning Transferable Visual Models From Natural Language Supervision, Feb. arXiv:2103.00020 [cs].

RAFFEL, C., SHAZEER, N., ROBERTS, A., LEE, K., NARANG, S., MATENA, M., ZHOU, Y., LI, W., AND LIU, P. J., 2023. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, Sept. arXiv:1910.10683 [cs, stat].

Appendix A

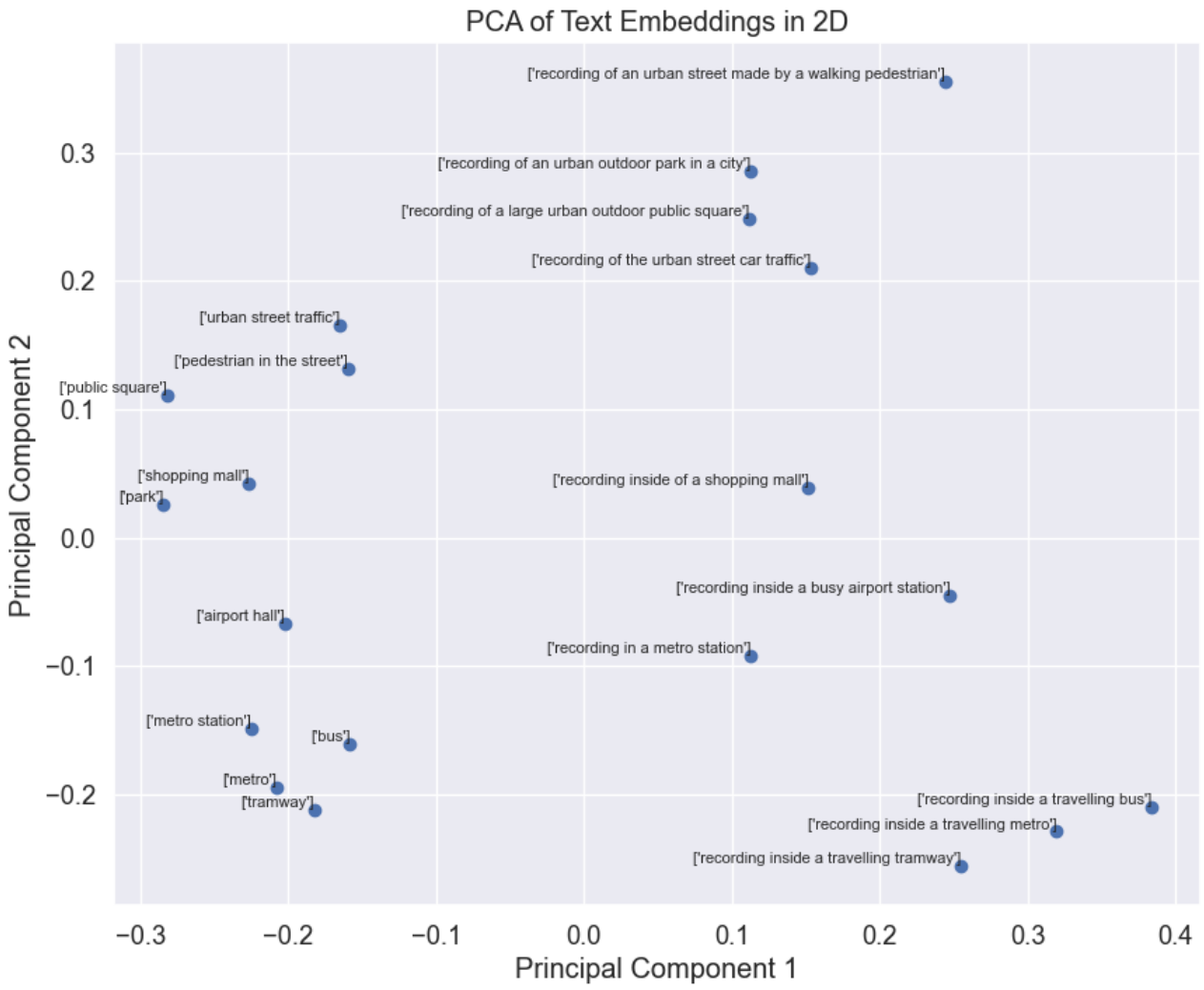

Figure 9: *Visualization in 2D of the text embeddings of AudioCLIP.*

Appendix B

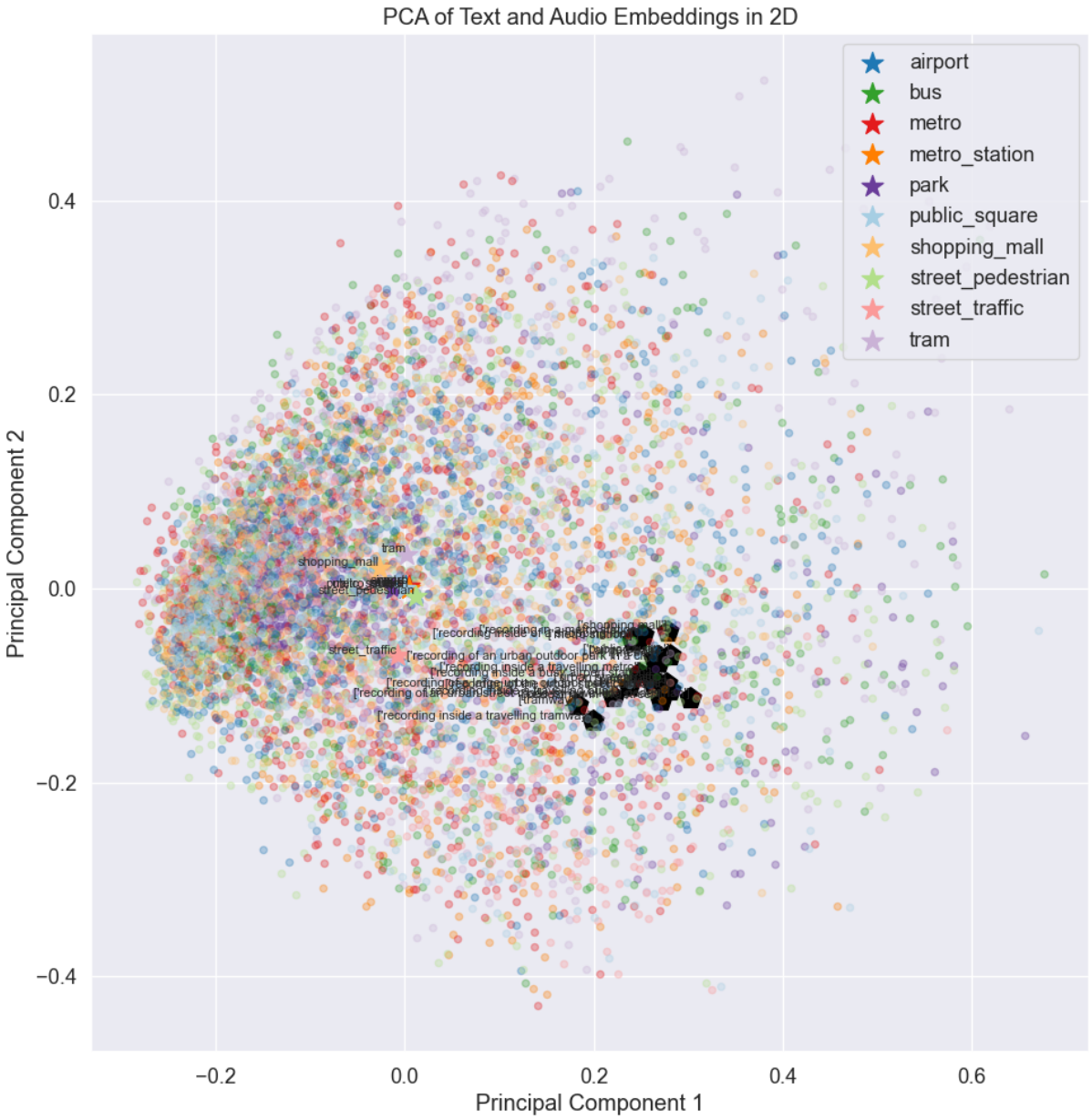

Figure 10: *Visualization in 2D of the audio and text embeddings of AudioCLIP.*