Rey-Jouanchicot Jordan

Report :

Advanced Machine Learning : Urban Sound 8K

Contents

1 Introduction and problem description

This a report regarding my project for the Advanced Machine Learning class in Algebra University. For this project I decided to work on a sound dataset, UrbanSound8K, which can be found here : `https://www.kaggle.com/chrisfilo/urbansound8k`, which contains 8732 sounds of less (but mostly around) 4 seconds. There is one CSV file giving info about files where to find them in the dataset,their original file, their location in the original files, and of course their classes in ID but also with the name associated to it, also if the sound to detect is in foreground or background.

There are ten classes of sound, which are : airconditioner, carhorn, childrenplaying, dogbark, drilling, engineidling, gunshot, jackhammer, siren, street music.

2 My approach to solve this problem

My aim is to use a feature like spectrogram which generates a 2D representation of sound and to use this representation with a convolutional neural network to learn from it and be able to classify the sounds.

Spectrogram is a feature used for sound representation, it splits the sound in a certain number of samples called segment of the same size, we can define the number of segment with a variable. For all of this segment, we perform a Fourier transform on each of them independently and make a matrix containing each segment and for each segment for the different frequencies their strength in the segment. To understand more and better spectrogram and this kind of transformation, here is some explication about it : `https://en.wikipedia.org/wiki/Short-time_Fourier_transform`

The sounds are split in 10 folders, it is recommended from the different sources related to this dataset to use cross-folders validation and training through the 10 folders, I followed recommendation. Also, I splitted the dataset in three datasets :

- Training : 60%
- Validation : 20%
- Testing : 20%

I decided to use Google Colab, but in the readme file is explained how to run it directly on your computer with Jupyter notebook. The major issue I encountered using Colab was that the preprocessing can be really slow sometimes. Certainly depending of hardware used for the session but we cannot choose or have information about it.

3 Experiments and evaluation

3.1 Fast representation

One of the first choice I had to make was to decide when to generates the images from the sound at the beginning I did it inside the dataloader, but it was a bit too slow and was really slowing down training. I could have move this to training and validation loops and I would have been able to even use GPU for processing but I decided that I prefered to move this to preprocessing and decided to store the list of images in memory and give it to the Pytorch Datasets objects. I created my own Custom Pytorch Dataset class to make creation of dataset and dataloading clean.

3.2 Representation choice

I studied multiple representation from images, MFCC (which you can learn more about here : `https://en.wikipedia.org/wiki/Mel-frequency_cepstrum`), Spectrogram, and "compressed Spectrogram".

According to multiple reports MFCC is a bit decorrelated with sound classification and the results I got during my first tests seems to validate it on our dataset. So I decided to stop using it. I decided to provide two versions of my classifier :

- Big_NN version ; This version use Transfer learning on wide_resnet_50_2 neural network applied on resized spectrogram.
- NN version : This version use a self created neural network using 4 convolutional layers on what I called "compressed sprectrogram".

For both versions, I decided to not use data transformation, because rotation, and flip does not make senses for these images as they are calculated, they would represent another sound signal really different. Also, changing contrast, saturation and things like this would not be interesting because here colors are representing strength of frequency in the signal, if we change them, it would create a complete different sound. As I am not an expert, I have already enough data, I have not seen notebook doing it and I got really good results without touching this, I decided not to do so.

As I said I made a version using a feature that I called "compressed spectrogram", the idea is to do a spectrogram and for each row (each row representing a time segment and all spectogram having the same number of row) calculating the mean value (so mean frequency). Finally, I get a 1D representation with this of the number of row (in my case 400) and I can reshape it to a 2D image, matrix (in my case square image of 20 by 20 pixels). This representation is really small and with a convolutional networks with only four convolutional and pooling layers and one fully connected layer allowed me to get around 90% of accuracy on validation set and test set.

I made a version, called Big_NN, taking spectrogram in input, however as the spectogram size changes a lot depending of sounds I decided to just resize it to an image of 100 by 100 whatever the input size is, another reason is to avoid memory issues because this images are stored in-memory. Then I used a pretrained

model from Torchvision, I tested mainly on wide_resnet_50_2, which seems good choices thanks to residual connection allowing faster evolution of the weights to the new task, I choose the 50 versions because I believe it is large enough to be able to learn correcly for my image size. I replaced the last layer and did not freeze training of others layers, because it is pretrained on Imagenet which images are really far from ours, so with freezing unfortunately the model was not able to have so good results but using pretrained weight allowed us to have better initialization than random one, so allowed faster convergence of the weights.

3.3 Some optimisations

I spent some times reading about optimizers to decide which one using, to try to optimize results of the networks. I decided to use AdamW which is a modificated version of Adam which should be able to generalize better tham Adam but also include weight decay and converges faster than optimizer like Stochastic Gradient Descent.

I also tested Amsgrad variant of AdamW, it seems to improve the loss, so I decided to keep it used for the final version.

3.4 Results

During this project I had the opportunity to compare multiple representation of sounds, I was really surprised to see how good was able to be the small convolutional neural network on the "compressed spectrogram". Also, the small number of epochs required to get fine-tuned wide_resnet_50_2, is also really impressive and I got a validation loss of less than 0.240, showing a really good confidence and good prediction of the model and a validation accuracy of around 0.93. We can see with confusion matrices that results seems pretty balanced, the class with the "lowest" good prediction rate seems to be car horn with biggest confusion with street music.
Following, two confusion matrix, first one on wide_resnet_50_2 trained, second one on smaller neural network on "compressed visualization".
We can see on the confusion matrix of compressed visualization that two of the more confused class are the class 1 and 6, they are the class with less values than the others according to our data analysis, so this can be linked. We do not see the same pattern on the larger version using wide_resnet_50_2.

Figure 1: wide_resnet_50_2 trained confusion matrix

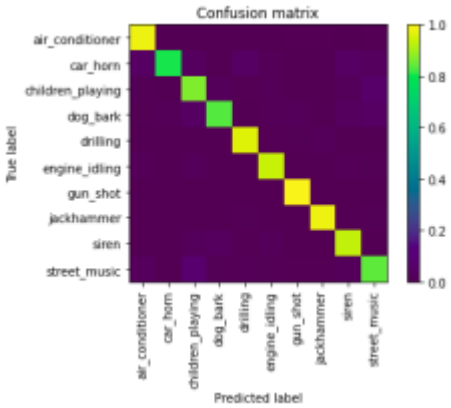
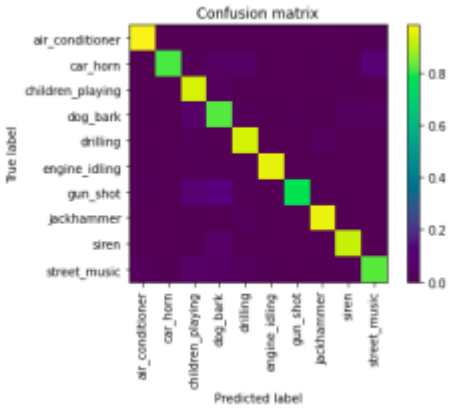


Figure 2: Smaller neural network on compressed representation confusion matrix



4 Conclusion

During this project I learned a lot about sound processing and gained experience with Pytorch, it was really interesting. I got really good results. I also liked that we could apply something like this in real-life, for example to be able to detect gun shot automatically, and for example inform the police, using microphones from cities.

5 Sources

For this project, I used multiple references to compare my results but also to get some ideas of how processing this ressources :

https://arxiv.org/pdf/2007.11154.pdf

https://www.kaggle.com/prabhavsingh/urbansound8k-classification/notebook

https://pytorch.org/

https://www.fast.ai/2018/07/02/adam-weight-decay/