Music genre classification

Emanuele Pocelli, Fabrizio Sordetti

June 25, 2024

Contents

1 Introduction **2**

2 The problem **2**

3 The dataset **3**

4 How we approached the problem **3**

5 Data processing **4**

6 Results **6**

7 Conclusions **7**

1 Introduction

This is the report of the NAML project 2023/2024, consisting in reproducing a paper about music genre classification.

2 The problem

The problem consists in classifying songs according to their genre. The given paper proposes a novel architecture to solve this problem efficiently and with high performance. It is based on convolutional neural networks. As is often the case, when processing audios using neural networks, we work on the spectrogram of the audio and treat It as an image. This is because architectures for images are more advanced, since the interest in image processing is bigger. More in detail, the architecture is composed of two pieces: a unet followed by parallel convolutions. The unet is a recent architecture that features a decoder and an encoder: we get fine grained features in the contracting path, where we downsample and increase the number of filters in each step, then, in the expansive path, we upsample also using the skip features of the contracting path. Such architecture excels in image segmentation. Finally, the parallel convolution is used to analyse the fine grained features. In the paper, multiple music genre classification datasets are used, but we will focus on the GTZAN that is the most famous and probably the most interesting.

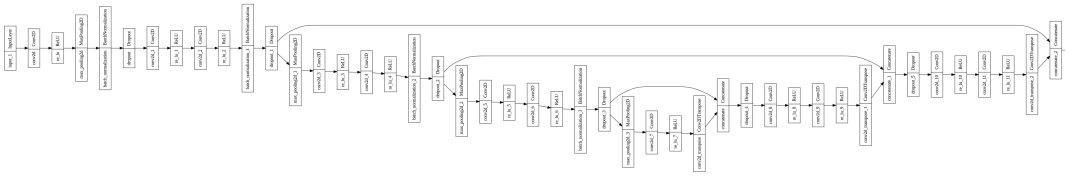

Figure 1: Plot of the unet model. We can see the U-shape, hence the name

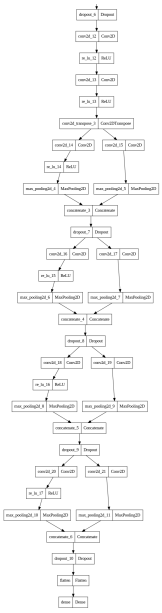

Figure 2: Parallel convolutions

3 The dataset

The dataset used is the GTZAN, which is widely available online. It consists in 1000 songs belonging to 10 different genres. For each song, there is a 30s audio. It is a common dataset often used to test new architectures. It is quite complex, so strong models are needed in order to achieve a decent accuracy.

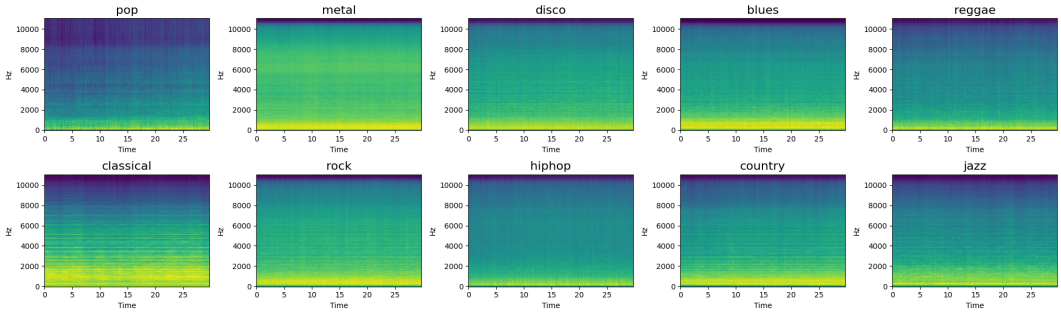

Figure 3: Mean spectrograms for each genre. We can see clear differences for some genres, however for others are much less noticeable

4 How we approached the problem

The architecture proposed in the paper has a size of over 300MBs. We had significant limitations on the computational side, both in time and space, so we could not use the very same architecture. Our architecture is based on the one proposed in the paper, but It is revised and is much smaller (15MBs). We tried training bigger models, but a lot of problems came, like strong overfitting or exploding gradients. Very special care was needed in order to avoid these phenomenons, but, as we will see in the next sections, we were hindered by our computational limits and could not overcome these problems because of that. We also used batch normalization, l2 regularization and dropout. The paper did not mention these aspects, but we had a much more stable training with them. It has been a bit tricky to find out "where" to place each of these layers, because CNNs are very sensitive about this.

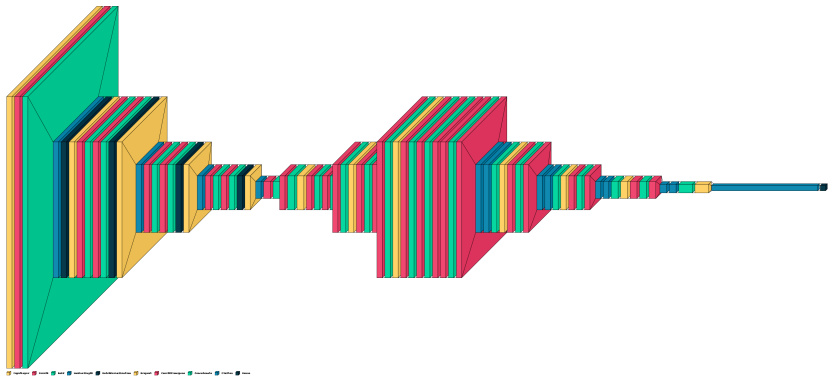

Figure 4: Plot of the model using visualkeras

5 Data processing

When dealing with big neural networks, we need a lot of data. Unfortunately, the GTZAN dataset is relatively small for, only featuring 1000 songs divided in 10 genres. Because of that, the data processing was critical in order to avoid overfitting. Even though the dataset is small for a NN, It was pretty big for the VRAM: we faced a lot of crashes because of memory and we were forced to stream the dataset during training using TensorFlow's data generator class. This, however, made everything slower. First of all, we split each song in 5 different pieces so that each input is 6s long. This is an easy way to have a bigger dataset while not losing too much information: It is still possible to classify the music genre given only a 6s clip. Every clip belonging to the same song all belong either to the training set or the validation set. It is very important to keep the independence between the two datasets, otherwise It would be like cheating. All the data was normalised to be in the range of (0,1). The most important part was the data augmentation. The paper talks very little about the data augmentation: they just state that they implement It using standard librosa functions. Augmentation on a spectrogram is a bit tricky: It is possible to work on the waveform or on the specotrgram Itself. The former provides a stronger effect, but is computationally very expensive, since It is needed to generate the spectrogram of every song in each epoch, the latter is weaker, but since It works on the spectrogram Itself, It is quite cheap to perform. Because of computational constraints, we opted for the second option. This made the training feasible for us at the cost of more overfitting. We used the following techniques: random noise, time shift, time mask, frequency mask.

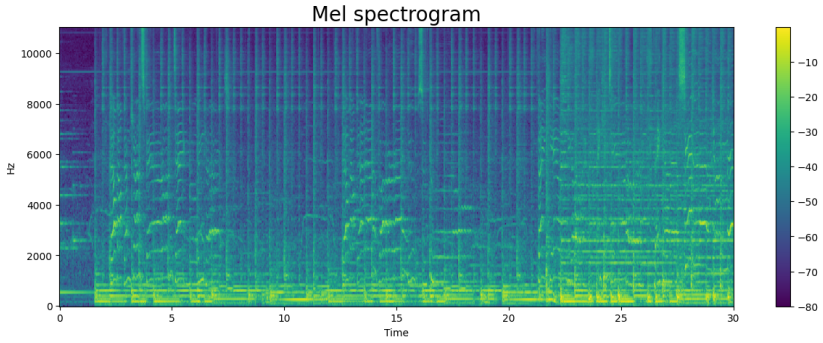

Figure 5: Example of a spectrogram

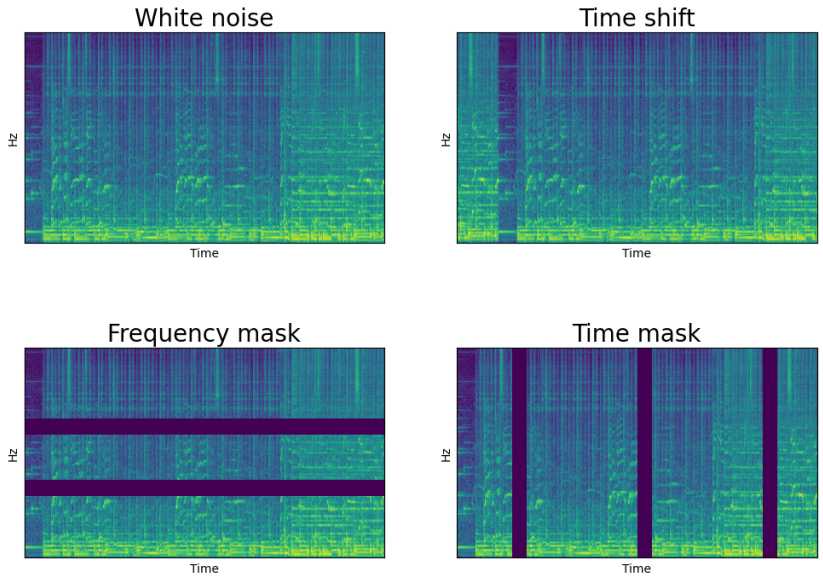

Figure 6: The different techniques we used

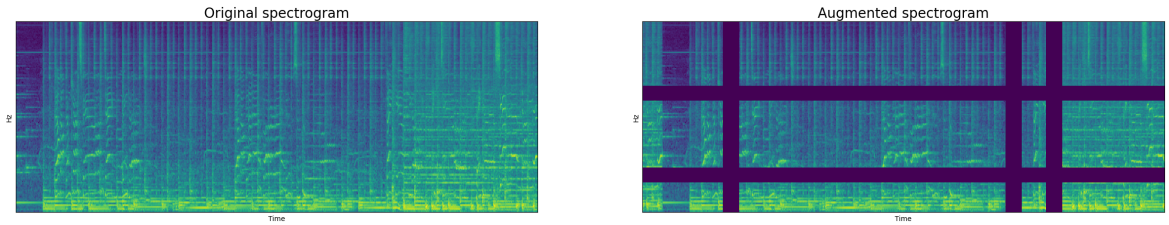

Figure 7: All the techniques combined

6 Results

We were pretty happy with the results we got. There is a bit of overfitting, but overall It is decent considering the size of our model and the complexity of the problem.

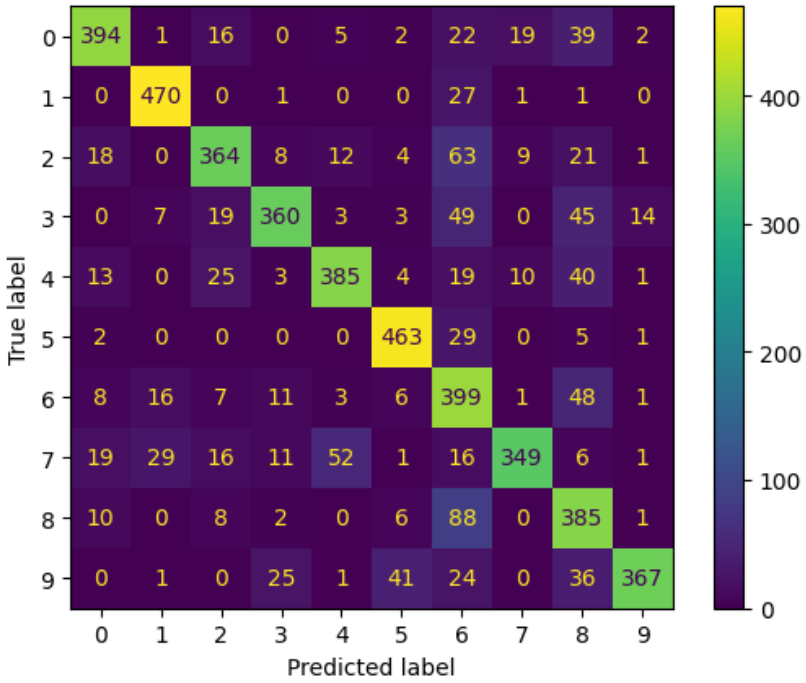

Figure 8: Confusion matrix

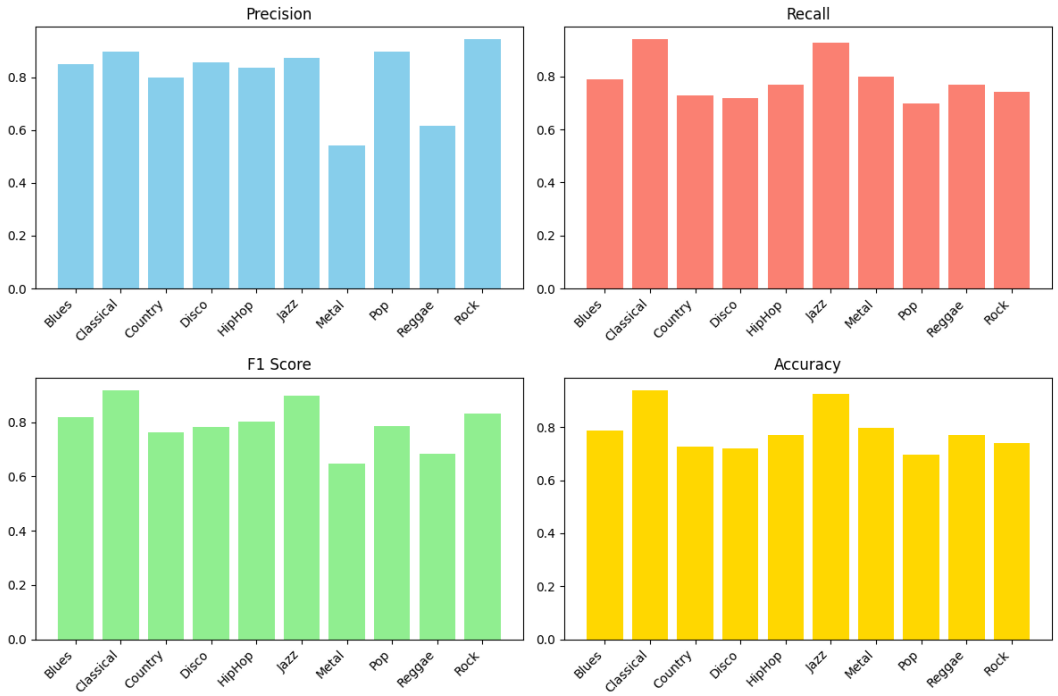

Figure 9: Statistical results for each genre

7 Conclusions

Computational limitations unfortunately hindered our model. Using a stronger augmentation will certainly decrease the overfitting and would allow to increase the model size. We were not able to replicate the results of the paper, but they seem very unlikely, given they achieved 99.9% accuracy. It could also be a symptom of overfitting. They did not explain in detail neither the model architecture nor the data processing scheme.