Data Input Approaches for CNNs towards Music Genre Recognition

Guillermo Benito Calvino[†], Sofia Pacheco Garcia[†]

***Abstract*—Convolutional Neural Networks (CNN) are a well-studied tool proven to be very efficient for classification tasks. Within this problems the music genre classification of a song is included. The efficiency of a CNN is (obviously) highly dependent on its architecture. That is why some papers, like [1], study which is the best one for their specific problem. Conversely, willing to study the relevance of the input of a network, we have taken the best model from [1] and we compare its performance when being fed by the same data with different degrees of preprocessing: raw data, its spectrogram and the combination of both. We observe that the spectrogram yields better results and that the combination does not provide any advantages, meaning that the raw data's information is redundant to that of the spectrogram. We also infer that segmenting data into smaller pieces makes the network perform better.**

***Index Terms*—Supervised Deep Learning, Convolutional Neural Networks, Music Genre Recognition, Preprocessing.**

I. Introduction

Deep learning, specially CNNs, are widely used for classification tasks. In this report we explore the problem of music genre classification for song audio samples.

When thinking about this type of matters, one usually focuses on the network architecture, but another important aspect is the input data. The possibility of augmenting the relevant information inputted to a network can provide big improvements in its performance. Following this last idea, we have decided to use the neural network described in [1] with different possible inputs in order to compare their performance. We have used a 1D input (raw signal), a more preprocessed 2D input (spectrogram) and a combination of both. This comparisson, but with a different network architecture, is also made in [2] but lacking the mixture kind.

All in all, the main questions we are trying to address in this project are the following. Is it really more useful, as commonly believed, to use more preprocessed data? Is it worth using the mixture of the two data kinds for this network? Do their contributions complement each other or are they just redundant? In order to answer them we have rebuilt the best performant network in [1], which consists of a residual convolutional block of three layers, a max and average pooling and three dense layers. We then have trained it with the possible inputs and measured their effectiveness with their accuracy on a test set. Another implementation we have made in order to improve the input data is to segment each data sample into different overlapping pieces, also showed to be useful in [1]. If the combination of 1D and 2D data works better as an input, it could improve many other classification networks.

This report is structured as follows. In Section II we talk about the papers in which we have mainly based our work. Section III is a general overview of the network model while in Section V you will find a more exhaustive description of the learning algorithm. The different data inputs that are used are described in Section IV. Finally, the accuracy results are shown in Section VI.

II. Related Work

In [1] two similar networks are built in order to solve the music genre classification problem. Their input is the audio spectrogram. They both rely on a CNN architecture, followed by pooling and some dense layers for classification. The main difference between them lies in the convolutional block: the first network uses a classical CNN while the second one has a residual block. They show that the second one has a better performance. Following this line of work we have tried to recreate the second network and to compare its performance for different inputs other than the spectrogram. This last idea comes from [2], where they show that just by adding one convolutional layer, the same network can be used for inputs with different dimensions. Instead of adding a layer, we have decided to change the first convolutional layer of the network so that the comparison is more fair (all the networks having the same number of layers).

III. Methodology

In this section the architecture of our networks will be described. This is of great relevance in Deep Learning problems, which already handle the task that often requires the most human effort in the rest of the Machine Learning approaches: feature extraction. For kinds of problems like the one discussed in this paper, the design of the network can have a high impact in the performance of the model. That is precisely what will be explained bellow.

[†]Master Physics of Data, Department of Physics and Astronomy, University of Padova,
email: {name.surname}@unipd.it

We have built three different networks, the three of them sharing their general compounds: a Residual CNN block, to which the input is fed, that works as a feature extractor; a max and average pooling that reduces the size of the feature maps; and the classifier, a fully connected block formed by dense layers. The diagram in Figure 1 shows this general outline. This structure has been chosen to match the one used in [1].

The only difference among the three networks lies in the first convolutional layer of the CNN module due to the dimension of inputs they are fed with. The first network's input is the raw audio signal, the second's is its spectrogram and, finally, the third network's input is the combination of the two previous ones. We do this with the aim of checking out the contribution to the classification of the two inputs. If the performance resulted better for this last case, that would mean that the raw audio signal and the spectrogram could contribute with different information.

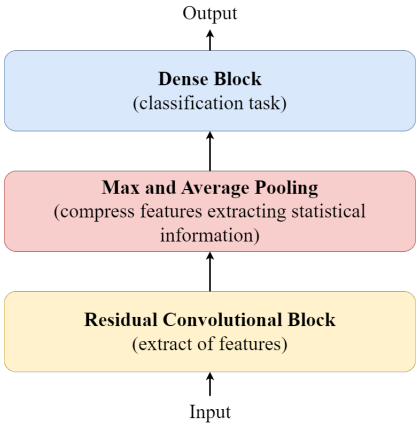

Fig. 1: General diagram of the different blocks that build the network.

IV. Data: signals and preprocessing

The dataset is called FMA and it is described in [3]. We have used the small subset which is composed of 8000 song samples of about 30 seconds each. The sampling rate for all of them is fixed to 44100 Hz. Since we wanted samples that were exactly equal in length, we have cut all of them to the same size.

The two different signals we have worked with are the raw audio and its spectrogram.

- **Raw audio signals:** one-dimensional arrays with the amplitude information of the audio. The dimension for each sample is of $(1, 10^6)$.
- **Spectrogram:** it is the Short Time Fourier Transform (STFT) magnitude spectrum of the samples, typically used when dealing with audio data. It is a two-dimensional array containing the time-localized frequency and amplitude information of the raw signal's Fourier Transform. The chosen value for the length of the STFT window is the typical one for music signals, 4048, and we have set a hop length such that there is a 50% overlap. Its dimension for each sample is of $(1025, 977)$, the first dimension corresponding to the frequency and the second to the time.

Moreover, we have decided to input the data in two different ways: without further preprocessing and segmenting the samples into smaller parts (of approximately 3 seconds) with 50% overlap as done in [1].

Finally, the data has been split in 80% training set, 10% validation set and 10% test set, just as done by [3].

V. Learning Framework

As it has been explained in Section III, the network consists of three parts. The first layer of the convolutional block is different depending on the input signal type.

- **Residual CNN Block:** it is made of three convolutional layers. The output of the first layer is added up to the output of the third layer, yielding the result of the block.
  - **First Convolutional Layer:** as we mentioned before, this part is different for each network. This is because even though the inputs have different dimensions, we want that after this layer all of them have the same shape to be able to use the exact same network from now on. Thus, the main aim of this convolutional layer is to extract features while compressing this information into matrices of dimension $[1, 974/s]$, being s the number of data segments. In order to achieve this we have used the different convolutional layers described in Figure 2. Note that for the case of input data 1D + 2D, the number of feature maps are half for each kind of input data (128 for 1D data + 128 for 2D data).
  - **Rest of the Convolutional Layers:** the remaining are two convolutional layers designed to extract more features.
- **Max and Average Pooling:** we then perform a max and average pooling separately. The size of the used kernel is exactly the same as that of each feature map, transforming these last into scalars. The size of the output of each of the poolings is 256. They are then concatenated in a single vector. The purpose of this is to reduce the dimensionality of the next layer (because we want it to be linear) while providing meaningfull statistical information.
- **Fully Connected Block:** it is composed of three linear fully connected layers. The purpose of this module is to perform the classification task. The dimensionality of the layers is reduced along the block so that the last one has

as many neurons as possible classes (the eight possible genres in our case).

After each of the mentioned layers, we place as activation function an Exponential Linear Unit. We chose the ELU because it preserves the advantages of the ReLU (used in [1]): it mitigates vanishing gradients and accelerates the convergence of the optimizer; while solving the zero-dying problem and being more robust to noise than LReLU.

Moreover, between each layer and its activation function we have implemented batch normalization. Lastly, we have tried to use 0.2 dropout after each layer except the last dense one to try helping preventing overfitting.

The optimizer we use is Adam with a learning rate of $5 \cdot 10^{-4}$. The learning rate is smaller than the default one since it seemed that the optimizer was escaping from the minima. As usually in multiclass classification, we use categorical cross-entropy as a loss function. Note that the cross-entropy is just a combination of a softmax and a log-likelihood loss function. The job of the softmax is to transform a vector of K real numbers into a probability distribution of K possible outcomes. The size of the mini-batches is 50 and the number of epochs is around 15.

For the cases in which we segment the input samples, the final prediction probabilities of an audio sample are computed as the sum of the predictions of each of its segments. Note that during the training the loss is computed for the prediction of each segment instead of for the final prediction (the whole audio sample). Instead, for the validation, the loss is computed once the prediction of the whole song is calculated.

VI. Results

As it has been previously stated, our main goal is to compare the network performance for the different inputs. In total we have three kinds of inputs (1D, 2D and 1D+2D) and, for all of them, we have tested the network with and without segmentation. The performance is measured through the accuracy of the model in the test dataset. The results can be seen in Table 1. As it can be appreciated, a highest test accuracy is yield by the models trained with segmented data. The 1D input gives a lower accuracy in any case. This is probably due to it being the less preprocessed one. The results gotten for the model trained with 2D input are similar to those obtained for the model fed with both kinds of data. This could mean that the raw data does not add any new relevant information to the machine.

	1D input	2D input	1D+2D input
Not Segmented	0.264	0.395	0.376
Segmented	0.314	0.429	0.428

Table 1: Accuracy of the test dataset for the different inputs.

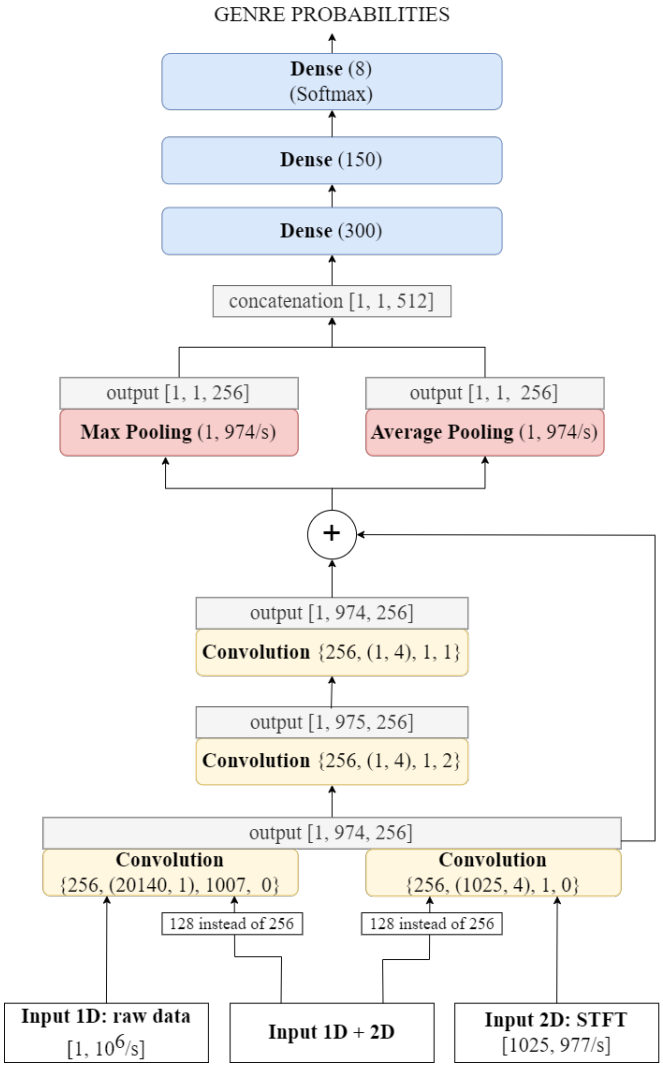

Fig. 2: Detailed diagram of the three different networks depending on the input. The convolutional layers are described as {features, (kernel shape), stride, padding}.

Once the best inputs have been found (spectrogram and mixture inputs with divisions), we have tried to maximize their performance by adding dropout. The dropout results are shown in Table 2. Comparing them with the ones obtained without dropout we realize that it deteriorates them. It should not be used in our case.

2D input	1D+2D input
0.378	0.369

Table 2: Accuracy of the test dataset for the model trained with 2D and 1D+2D inputs with segmentations, with 0.2 dropout.

However, all the accuracies are pretty low, even the highest ones. It is probably due to the small size of the used dataset or to a possible implementation error from our side.

	2D input	1D+2D input
Without Dropout	0.469	0.00
With 0.2 Dropout	0.423	0.00

Table 3: Accuracy of the test dataset for the model trained with 2D and 1D+2D inputs with segmentations, without and with 0.2 dropout.

VII. Concluding Remarks

During this project, we have built a similar residual convolutional network to that of [1], which aims to classify music samples into different genres. We have compared it for different inputs. The results make us able to answer our initial questions: a more preprocessed data seems to make the network perform better and the information provided by the raw data is redundant to that of the spectrogram.

We have faced several difficulties when developing the project that could explain our low accuracies, all of them related to the low computational power we have disposed of. We have been working with only one computer with Nvidia GPU and with the Google Colab's time-limited GPU for the other computer. This has made us train the networks with less amount of data than we would have liked to. We suspect this might have been the main reason why our results are not even close to those of [1], even if the approach was, to a large extent, the same.

Nevertheless, apart from training with a larger dataset, there are several things we have thought of that could be further implemented in order to improve the performance of the model. For example, we could try to introduce more convolutional layers, extracting more useful features for classification. We could have also tried different kernel sizes and strides specially for the first convolutional layer of the 1D input.

At a technical level, we have specially learned how to manage data sets where the samples are big (order of Mbyte) and to work carefully with the dimensionality of the convolutional layers. At a more abstract level we have acknowledged the importance of making an educated guess over the parameters of the network since it is too computational and time demanding carrying out many tests.

References

[1] W. Zhang, W. Lei, X. Xu, and X. Xing, "Improved music genre classification with convolutional neural networks," in *Interspeech*, 2016.

[2] J. Pons, O. Nieto, M. Prockup, E. Schmidt, A. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," 2018.

[3] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "Fma: A dataset for music analysis," 2017.