Machine Learning

KTH | Royal Institute of Technology

Deep Learning in Data Science

Music Genre Classification with CNN

Authors:
Ennio Rampello
Carlo Saccardi
Romain Trost
Isak Gamnes Sneltvedt

E-mail addresses:
ennio@kth.se
saccardi@kth.se
rtrost@kth.se
igsn@kth.se

Abstract [GitHub repo]

The goal of this project is investigating the performance of a Convolutional Neural Network for application in audio classification. CNNs are commonly used in computer vision applications and they deliver outstanding results when dealing with visual data. However, they are not widespread as a model architecture in audio applications and other models are usually more adopted (e.g. RNNs, LSTMS, etc...). Through this work, we will show that a Convolutional Neural Network can be successfully applied to audio classification and it delivers great results. Since it is a model that is specifically designed to extract information from visual features, we have transformed the audio data into 2-dimensional arrays generated by different representations of the sounds, namely the MFCCs and the Mel-spectrograms.
We have performed all our experiments on a popular dataset (GTZAN) that contains one thousand 30-seconds songs belonging to ten different music genres. Since 1000 songs for each genre is not that much, We have further split the songs into 5-seconds audio samples and we have used these samples to compute both the MFCCs and the Mel-spectrograms that we have then used to train our model. In order to improve the performance of our classifier, we have implemented transfer learning using DenseNet121, as proposed by [1], as a base model and we fine-tuned its weights using our dataset. The results that we obtained using transfer learning proved to be much better than the ones obtained with the vanilla CNN.
Furthermore, we have explored the use of Variational Autoencoders for music production and, although we were able to produce some sounds that resemble music, the produced songs were too noisy and it was not possible to clearly recognize the quality of the resulting melody.
We have discovered that VAEs generally produce noisy music which is due to their architecture. Based on other research, the basic model of choice for music production should be a Recurrent Neural Network but we haven't investigated it.

1 Introduction

The purpose of this work was to carry out experiments and evaluate the results in the field of music classification and audio signal processing. However, we decided not to make use of the standard methods that are commonly employed in the field. Instead, we decided to experiment using methods that are generally employed in other application fields. Namely, we have chosen to make use of a powerful model in the area of Computer Vision, which is the CNN. This model delivers outstanding results when dealing with visual data, which is mostly thanks to its ability of extracting relevant features from images. This behaviour is possible because the different convolution layers act as feature extractors, and we the network is made of multiple layers it becomes able to extract very high-level features.
It is not possible to apply a CNN to data that is in the form of audio. So, in order to carry out our experiments we had to extract some information from the audio signals and process this information as an image. Thankfully, it is possible to extract features that can be treated as images from audio signals. Namely, the features that we have used are the MFCCs and the Mel Spectrograms.
For each song these two types of features come in the form of matrices, so it was possible for us to use them as if they were sets of pixels composing an image. The only problem in this case was that we were dealing with matrices, so 2-dimensional arrays. However, in order to feed this information as input to the CNN, we had to transform it to a 3-dimensional array with one or three channels, depending on if we were about to implement the vanilla CNN or transfer learning.
Furthermore, we have decided to test the performance of a Variational Autoencoder when dealing with music generation. So, we have trained the model in such a way that it receives a music genre as input and generates a soundtrack of the chosen duration as output.
Surprisingly, we were able to generate some sounds that resembled the chosen genre. It was clear what genre the produced sounds belonged to. However, as a consequence of using a VAE to produce music, the output sound was really noisy but it was possible to hear actual music in the background.

2 Related work

Our work was mainly inspired by one article [2] and one scientific paper [1]. In [2], the author propose to train a CNN network on 13 MFCCs coefficients extracted from each audio file. Our dataset consists of a collection of 10 folders with their genre name, each containing 100 songs. since 1000 songs is not all that much for training, we had to to generate more training examples. [2] proposed to slicing each song in 10 sub-parts, obtaining 1000 audio files of 3 seconds for each genre. However, we thought that 3 seconds were not enough to extract meaningful features (MFCCs or Mel Spectrograms) as some songs happened to have some silent moments. Thus, we decided to slicing each song in 6 sub-parts, obtaining 600 audio files of 5 seconds for each genre.

Both Mel Spectrograms and MFCCs coefficients are bidimensional arrays, and they are calculated so that they are more discriminative for lower frequencies and less so for higher ones, just like human ears. They are a very good fit for our project since they are easily readable and "interpretable" by a Deep Learning models too. A CNN model requires a 3-dimensional array as an input, where the 3rd dimension represents the channels of the image. However, as already mentioned, the MFCCs only have two dimensions. A way to circumnavigate this problem, as proposed by [2], is to simply add a new axis, and reshaping an array of size (13, 216) into (13, 216, 1). Once this was done, we were all set to train the vanilla CNN model. [2] proposed a network with 3 convolutional layers and two fully connected layers.Each convolutional layer was made of 32 filters and convolution windows size of (3,3), (2,2), and (3,3) respectively. Each convolutional layer was also followed by a MaxPooling layer, and batch-normalization was applied. We also added dropout after the 3rd convolutional layer to make the model more robust.

Once we trained the CNN, we implemented transfer learning based on [1] approach. [1], as opposed to [2], suggests to train the models on mel-spectrograms instead of MFCCs. CNN based standard models like Resnet, Densenet, Inception use images having three channels as inputs. Thus, the mel-spectrograms needed to be convert into a three-channel input. To do this, [1] suggests to use different window sizes and hop lengths of (25ms, 10ms), (50ms, 25ms), and (100ms, 50ms) on each of the channels respectively. Different window sizes and hop length ensures that the network has different levels of frequency and time information on each channel [1]. Thus, for each audio we obtain three mel-spectrograms using 128 mel-bins and then we log-scaled them. Since we used different window sizes, the same three mel-spectrograms were reshaped to a common shape of (128, 1500) and then concatenated to have a single tri-dimensional array of size (128, 1500, 3).
Once data pre-processing was done, [1] suggested to initialize ResNet, Inception, and DenseNet models with pretrained weights and fine-tuned on the original GTZAN dataset. We only initialized the DenseNet model, since it was the one that returned the best performances in [1].

3 Data

The dataset that we have decided to use for all our experiments is the famous GTZAN dataset [3]. The reason behind our choice is that this dataset is very easy to use, does not require data cleaning, and contains all the information that we needed to carry out our experiments.
The GTZAN dataset contains a total of 1000 songs. Each song has a duration of 30 seconds and is labeled as belonging to one out of the ten different genres that are present in the dataset. The data is very balanced, in fact there are a total of ten music genres, where for each one of them there are 100 30-seconds songs. The ten genres are: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae and rock.
In order to augment the data, we have decided to split each song into 5-seconds samples, for a total of 6 samples per song. This process resulted in a dataset with a total of 6000 songs. Furthermore, for each sample, we have computed the MFCCs and the Mel spectrogram. The reason for this operation is that we cannot directly use the *.wav* files for the training.

4 Methods

4.1 Data Preprocessing

In order to be able to use the audio samples in a CNN, we had to generate images from the audio files. This was done by using some python libraries, namely *librosa* and *Pickle*. Librosa was used to generate the Mel Spectrograms and the MFCC's from the wav files. Pickle was used to serialize the data and store it as a .pkl file. This was done so that we could store the data in a compact format without having to perform the generating of the images multiple times.

However, for the transfer learning part we had to match the input shape of the pre-trained model. The Mel Spectrograms and MFCC's we got from *librosa* had only one channel, whereas the pre-trained model required 3 channels. This conversion from 1 to 3 channels was done using the package *Pillow*. By iterating through all the generated images and using the *convert()* function provided by *Pillow*, we were able to get the desired number of channels.

4.2 CNN

To begin with we decided to train a CNN network without transfer learning. The CNN model was implemented using the python library tensorflow. We started out by a couple papers that had performed similar experiments earlier and based the model design on that. We also tried to change the parameters in the layers, the regularization and its parameters and the pre-processing of the data. The model that eventually gave us the best result was this structure:

Layer	**Filters**	**Size**	**Strides**	**Activation**	**Nodes**
Convolution	32	3,3	1,1	Relu	-
Max pooling	-	3,3	2,2	-	-
Batch normalization	-	-	-	-	-
Convolution	32	3,3	1,1	Relu	-
Max pooling	-	3,3	2,2	-	-
Batch normalization	-	-	-	-	-
Convolution	32	3,3	1,1	Relu	-
Max pooling	-	3,3	2,2	-	-
Batch normalization	-	-	-	-	-
Drop out	-	-	0.3	-	-
Flatten	-	-	-	-	-
Dense	-	-	-	Relu	64
Dense	-	-	-	Softmax	10

When we trained using the Mel Spectrograms we used an Adam optimizer with a learning rate of 0.001. For the loss we used sparse categorical crossentropy and the metric used for evaluation was accuracy. For the model that was trained using the MFCC's we found that the same loss and evaluation metrics gave the best results, but using an RMSprop optimizer with 0.001 learning rate.

4.3 Transfer Learning

During this project we tested a few different pre-trained networks to use as a core in our model. The one that gave the best results was using MobileNetV2 as a core and the following connected on the output:

Layer	Filters	Size	Strides	Activation	Nodes
Max pooling	-	3,3	2,2	-	-
Batch normalization	-	-	-	-	-
Drop out	-	0.3	-	-	-
Flatten	-	-	-	-	-
Dense	-	-	-	Relu	64
Dense	-	-	-	Softmax	10

For this model, the same optimizers, loss and metric were used for the Mel spectrograms and MFCC's as mentioned in section 4.2.

The transfer learning addition was mostly based on the paper "Rethinking CNN Models for Audio Classification" [1]. The paper gave a good basis on how we could go forward with designing and training the network and get a good result. After discovering the paper published by Kamalesh Palanisamy et. al. , we wanted to the same core that they had used, DenseNet121. The network that we laid on top of the DenseNet had the following structure:

Layer	Filters	Size	Strides	Activation	Nodes
Max pooling	-	3,3	2,2	-	-
Batch normalization	-	-	-	-	-
Drop out	-	0.3	-	-	-
Flatten	-	-	-	-	-
Dense	-	-	-	Relu	64
Batch normalization	-	-	-	-	-
Drop out	-	0.3	-	-	-
Dense	-	-	-	Softmax	10

The model uses an Adam optimizer with a learning rate of 0.0001. The loss is sparse categorical crossentropy, and the metric is accuracy.

4.4 Variational Autoencoders

To increase the amount of data used for the training of both our CNN and transfer learning models, variational autoencoders were used to generate new audio samples. The network's encoder has 4 convolutional layers, each followed by an instance normalization layer that normalizes all features of one channel. The network's decoder is composed of 5 convolutional layers, each of which is followed by a batch normalization layer. The model was trained for 25 epochs.

5 Experiments

The most common way of training Convolutional Neural Networks using audio data is to represent the latter in the form of 2-D spectograms [1], notably either through *mel spectograms* or *mel frequency cepstral coefficients (MFCCs)*. A mel spectogram describes the different frequencies present in an audio signal converted the mel (logarithmic) scale. An MFCC is obtained by taking the logarithmic magnitude of a mel spectogram and then applying discrete cosine transformation to it. To evaluate which data representation would yield the best results, both were used for training a vanilla CNN, for which their respective performances are displayed in Table 1. As shown, the vanilla CNN achieves a test accuracy of 79% when using MFCCs, performing considerably better than the 53.5% validation accuracy achieved when using mel spectograms. For these two cases, we converted the MFCCs and Mel-spectrograms into 3d arrays, with the third dimension being equal to one. In other words, we only considerd one channel.

One thing to note when considering transfer learning, is that CNN based standard models like Resnet, Densenet, or Inception use images having three channels as inputs. Our initial approach to get the MFCCs in this format was to plot them with librosa, and then download them as .png images. Then, using the pillow library to load the images gave us the desired three-channel input. However, this yielded a validation accuracy of only 63.3% when training our transfer learning CNN with these inputs (see Table 1). Therefore, we applied the data pre-processing described in [1] to our mel spectograms, instead of using the MFCCs, to get the desired inputs. Using these to train both our vanilla and transfer learning CNN models yielded a much improved overall test accuracy of 60% and 92%, respectively (see Table 1).

Model\input data	Mel Spectrograms	MFCCs	Mel Spectrograms (3-channel)	MFCCs (3-channel)
Vanilla CNN	53.5%	79%	60%	\
TL MobileNetV2	\	\	\	66.33%
TL DenseNet121	\	\	92%	\

Table 1: Test accuracy achieved by the different models using mel spectograms and MFCCs.

The audio files generated from the variational autoencoders, although did resemble audio of their respective genre, did contain a substantial amount of noise. Low-pass and high-pass filters were applied to the generated audio files in the hopes of removing said noise, however due to the noise's large frequency range, this method didn't prove to be effective.

6 Conclusion

Throughout this work, we were able to show how powerful a method such as a CNN actually is. We did so by applying to something that it wasn't designed for. That said, we are proud of the results that we have obtained, especially the 79% classification accuracy when using a vanilla CNN on the MFCCs and even more so for the 92% accuracy that we have obtained by implementing transfer learning and applying the model to the Mel Spectrograms. In particular, we have been able to show the degree of improvement obtained when fine-tuning a pre-trained network.
On the other side, we have experimented with the VAE, which did not bring results that can be of valid use in the field of music production. However, we have learned how to best implement a VAE, which is something that we haven't treated in any of the courses, and we are happy to at least having gained some knowledge. Furthermore, we have delved briefly into music production through generative models and we have acquired an initial understanding of the best practices and best models to use when dealing with this kind of problem. In the future we could use the sounds produced by the VAE to augment the data used for the training of the CNN. In this way, although the sounds are not great to listen to, they might be useful to reach even higher classification accuracies when training the CNN.
A very interesting future experiment that we could carry out would be to implement a music generation model based on a RNN. This type of model have proven to deliver outstanding results, meaning that the music that it is able to produce when properly trained is comparable to the human-produced one.
Overall, we have gained a lot of knowledge from working on this project and we hope that this would be just the beginning of a journey throughout the topics of music classification and music production.

References

[1] Kamalesh Palanisamy, Dipika Singhania, and Angela Yao. Rethinking cnn models for audio classification, 2020.

[2] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2019.

[3] G. Tzanetakis and P. Cook. Musical genre classification of audio signals, 2002.