# "Music Genre Classification with 2D CNN"

Ceren Yilmaz†, Feridun Cemre Gulten†

***Abstract*—Classifying the genre of music is a necessary task for the purposes of organizing a music library and helping people discover the styles of music they find most enjoyable. While there are many methods of performing this classification process, there are many differences between methods, which lead to different results. Human classification is laborious, subject to bias, and impractical for companies like Spotify that need to classify massive datasets of audio files. As a solution of a this problem, classification with CNN model by using spectrograms is proposed in this paper.**

***Index Terms*—Deep Learning, Convolutional Networks, Music Genre Classification, Spectrogram**

## I. Introduction

The purpose of this project is classifying music genres based on their spectrogram using CNN model. The goal of this is getting higher accuracy compared to other classification models. In the project, the CNN is used as a classification model since the image representation of the audio files are used. The project is consists of multiple data pre-processing steps to get the image representation of the audio files and labeled the data.

Music Genre Classification is one of the important development subjects nowadays due to the music industry and music platforms. Most of the music platforms use music genre based platform to present more efficient and joyful experience to their users such as Spotify. The automate version of classification, also provide a strategy for music recommendation to user. Since for recommendation the music genre prefer needs to be determine. The classification is not efficient when it is done by human. For this reason, there are several researches and developments about this topic.(e.g. [1], [2], [3])

The music genre classification can be done by various machine learning (ML), deep learning methods. The audio type to classified, can be different for each case. The different genres can have different specific features parameters. These parameters can be analyzed based on raw audio or image representation of audio. Researches shows the image representation is more efficient way in most cases. Different spectrogram types can be used to represent audio signals in frequency domain. The deep neural network based methods can applied to music genre classification. There are various deep learning/machine learning libraries to help to build the deep learning/machine learning models. Some of the python libraries are "PyTorch","TensorFlow","Keras". These libraries are generally similar usage but the advantages are different. So, while selecting the built in library, the advantages of each one them needs to be considered.

The dataset used in this project is includes 6400 MP3 file and CSV file which includes each songs some features and labels of the audios. Similar genres' spectrograms includes similar features. So, in the project classification done from spectrogram. These spectrograms used in the CNN which is a deep learning model. Based on the similar researches, the most efficient model for the classification of music genres is CNN. In light of this, in this paper to classified the audios CNN model is proposed. The CNN model is build with special python library which is PyTorch. In this paper, the data preprocessing for determine spectrogram and CNN model is explained with details.

This report (paper) is structured as follows. In Section II we describe the related works, the data models are respectively presented in Sections III. The proposed dataset and preprocessing process is detailed in Section IV and in V the learning algorithm and strategy included. In the VI, the results are shown and finally VII it includes conclusion, future work of the project.

## II. Related Work

CNN, one of the important models of deep learning, has been used in researches that have had an impact in the speech recognition recently. The model implementation to speech recognition create a new research and implementation subject which is music genre and artist classification.(e.g. [1], [2], [3]) Most of these studies, trying to classify music 10 different genres; Blues, Classical, Country, Disco, Hip hop, Jazz, Metal, Pop, Reggae and Rock.

The idea of ensemble three different CNN to one CNN was proposed to improve music genre classification. [1] The three individual features which are classical spectrum based, presence of certain tones and presence of individual tones, were implemented to separate CNN and tested on the test data set to compare with the ensemble version of the CNN model. As a result, in the research ensemble CNN model has higher accuracy rate. However, even this method increased the accuracy compared with individuals, the accuracy is not very high.

In one of the research [3], it compared two different model to found efficient model. The advantages and disadvantages are showed these two model ,which are bias towards optimizing the CNN model and extract feature and developed labeled data set, can be efficient for different purposes or different cases. When compared these methods CNN gave more efficient results even it over fit small amount. However, when memory is critical feature extracting can be more

†Department of Information Engineering, University of Padova, email: {}@unipd.it

Special thanks / acknowledgement go here.

efficient because it is not need to keep audio files.

The another research [2], shows the accuracy of the two methods can improve classification. This research idea based on audio signal which used in the model, is short time Fourier Transform version of original signal. The two model used in this paper is combining max- and average pooling and develop model which skip one or more layers. As a result of these research, the improvement on accuracy of the music genre classification can be observed. However, in the paper indeed, there can be more efficient models since spectro-gram is manual features.

The main goal of the all related works is developed efficient and high accuracy music genre classification. In industry and various area, it started to be important and highly demand topic. When the related works examined, it determined that most of these research analysing spectrogram which is 2D version of audio, is more efficient compared to the raw audio. The efficiency is not just depends on the audio version, it depends on the classification methods and method's parameters too.

In light of these, in this project convolutional neural network (CNN) is used as a model and as a audio 2D version of the audio is used.

## III. Processing Pipeline

In this paper efficient model is proposed to music genre classification. In the project multiple step is followed the get the proper results based on proposed model. The steps are consist of data preprocessing, labelling MP3 files, conversion between MP3 to WAV format and WAV to PNG format and built and train the CNN model. The flow diagram of the process can be shown visually in below figure.

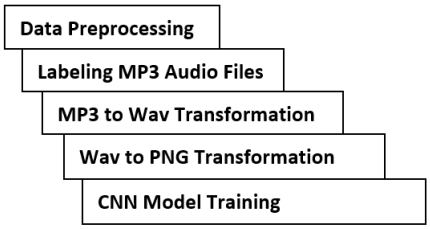


Fig. 1: Flow Chart of Music Genre Classification

FMA dataset which contains 6400 audio files, was used for the development. Each audio file is 30 seconds long and belong the one of the 8 genres. In the development process, the 30 seconds long audios can take space too much and slow the process. For this reason, while transforming the MP3 to WAV first 3 seconds of the audios are saved for classification. In the next steps of the project, these saved sound recordings converted to spectrogram with the Librosa library. Even there are different types of the spectrogram, Mel Spectrogram is the most efficient spectrogram for this project.

Using PyTorch's transforms module data transformation were applied and training/ validation dataset from image files in a directory were constructed. Then, the training dataset randomly splitted into a training and validation set based on a specified split ratio. The train-test split was done as %80 train set and %20 test set. After that, data was loaded using PyTorch considering the batch size(25) and shuffling to prevent the model from memorizing the order of the samples.

The generated image representation of the audio files which is spectrogram, were classified using CNN model which is provided by PyTorch. It takes a 3-channel input, an RGB image of a spectrogram, and passes it through five convolutional layers, each followed by batch normalization, ReLU activation, and max pooling. The output of the last convolutional layer is then flattened and passed through a fully connected layer with output nodes, followed by a softmax activation to produce a probability distribution for output.

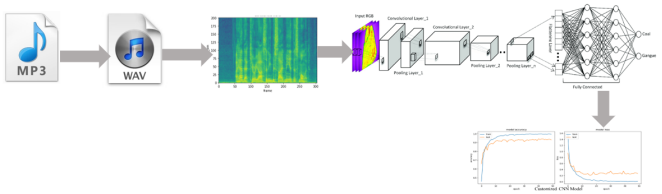


Fig. 2: Classification Process Scheme

## IV. Signals and Features

FMA data is used in this paper. [4] There are three type of FMA dataset small, medium and large. The small version is used in this project because of the capacity and efficiency problems. However, FMA small dataset is enough to train the model and get proper classification. This dataset consist MP3 files and CVS files which are includes various feature and labels of the audios. CSV files includes various features, however in the project most of them are not used. For this reason, the unnecessary features are cleared from the CSV file. Labelling is the most important part of the preparing data. The audio files numbers which are MP3 format, matched with the CSV file and labelling done for each of the audio files. The labelling is based on the genres. There are 8 genres which are Pop, Rock, Instrumental, Folk, Electronic, Experimental, International, Hip-Hop. Each genre includes 800 MP3 files. In the project, 600 MP3 files were used for each genres and just Pop, Electronic and Instrumental genres used as dataset.

MP3 format is a type of digital audio file format that compresses audio data. It uses lossy compression algorithm to reduce the size of the audio data. Since, it is a compressed version of the audio, the quality of the audio, some tones and some features can be sacrificed. In this project, features are important for the classification task. For this reason, the MP3 files transformed to WAV format. While, the MP3 files loaded and transformed to the WAV format, first 3 seconds of the audio was transformed. The reason of this is the 30 seconds

full sound, take too much space and while processing it can give some GPU or capacity errors. So, just the 3 seconds of the audio was saved and transformed. At the same time, this is not causing any big classification problem.

WAV( Waveform Audio File Format) format is a type of digital audio file format like MP3. However, WAV is an uncompressed audio format that stores audio data in a lossless manner. This means it captures and store the original audio features. So, it takes more space compared to the MP3 format. For this reason, it generally used in professional audio production. In light of all this information, even it take more space since original audio is more beneficial, labeled MP3 files transformed to WAV files and save with special python built in functions.

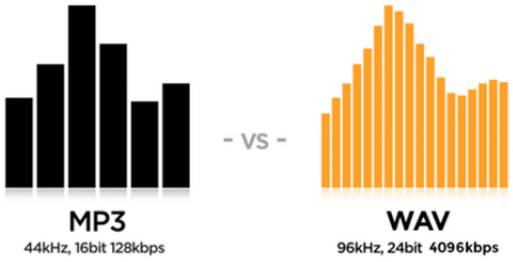


Fig. 3: MP3 vs. WAV Sampling Representation

A spectrogram is a visual representation of the frequency content of an audio signal over time. It is essentially a 2D graph where x-axis represents time, the y-axis represents frequency, and the color or intensity of each point in the graph represent the amplitude of the signal at that point in time and frequency. WAV format is time-domain representation of audio signal. Fourier transform is used to convert time-domain to frequency-domain. It essentially breaks down the audio signal into its individual frequency components allowing to analyze the frequency content of the signal over time. After this transformation, the signal can be used to create a spectrogram by dividing the audio signal into short time windows. While transforming the WAV to spectrogram, there are several methods in python. The different type of spectrograms can be used as different analysis approach. Each spectrogram save as PNG format which is a image representation.

A spectrogram is a visual representation of the frequency content of an audio signal over time. It is essentially a 2D graph where x-axis represents time, the y-axis represents frequency, and the color or intensity of each point in the graph represent the amplitude of the signal at that point in time and frequency. WAV format is time-domain representation of audio signal. Fourier transform is used to convert time-domain to frequency-domain. It essentially breaks down the audio signal into its individual frequency components allowing to analyze the frequency content of the signal over time. After this transformation, the signal can be used to create a spectrogram by dividing the audio signal into short time windows. While transforming the WAV to spectrogram, there are several methods in python. The different type of spectrograms can be used as different analysis approach. In the project, mel-spectrogram is used. Mel-spectrogram is based on mel scale which is a perceptual scale of pitch that is roughly logarithmic. Each spectrogram save as PNG format which is a image representation.

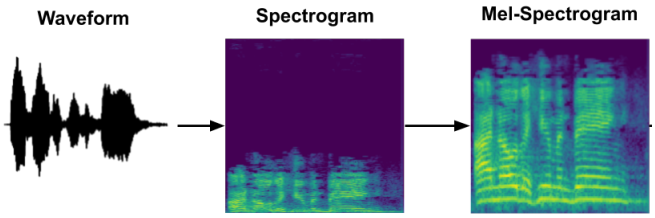


Fig. 4: WAV Format to Spectrogram Format Representation

The images of the spectrograms were taken from the train data folders which have the size as 288x432 . After the images loaded to the tensors the images shape became as ([25, 3, 288, 432]). This is the size of the image while it used as input to model. The feature extraction was done in the model part, the number of the feature is based on batch size.

## V. Learning Framework

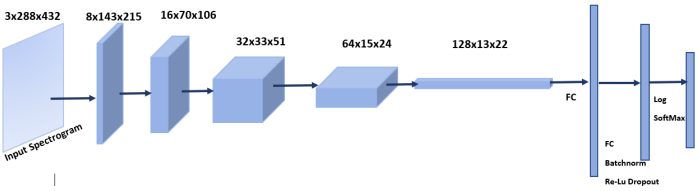


Fig. 5: CNN Model Diagram with Layer

### A. General Explanation

In the project a baseline convolutional neural network (CNN) model designed for music classification. It takes a 3-channel input, an RGB image of a spectrogram, and passes it through five convolutional layers, each followed by batch normalization, Re-LU activation, and max pooling. The output of the last convolutional layer is then flattened and passed through a fully connected layer with 4 output nodes, followed by a Softmax activation to produce a probability distribution for output.

The architecture of this model follows a common pattern for image classification tasks, where the initial convolutional layers extract local features from the input image, while the subsequent layers combine these features to form

increasingly abstract and high-level representations. The batch normalization layers help to reduce internal co-variate shift, while the dropout layer helps to regularize the model and prevent over fitting.

The choice of kernel size, stride, and padding in each convolutional layer determines the receptive field of the filters and the degree of down sampling. In this model, the kernel size is fixed at 3x3, the stride is fixed at 1, and no padding is used. This leads to a gradual reduction in the spatial dimensions of the feature maps through the pooling layers, resulting in a more compact representation that can be more easily processed by the fully connected layer.

### *B. Layer's Channels Explanation*

in˙channels and out˙channels are parameters in the convolutional layers that define the number of input and output channels, respectively. In a CNN, each convolutional layer learns a set of filters, where each filter is a small matrix of weights that is convolved with the input feature maps to produce a set of output feature maps.

The in˙channels parameter specifies the number of input feature maps to the convolutional layer. For example, in the first convolutional layer in this model, in˙channels is set to 3, since the input is a 3-channel image (likely an RGB spectrogram).

The out˙channels parameter specifies the number of filters to be learned by the convolutional layer. Each filter produces one output feature map, so the total number of output feature maps is equal to out˙channels. In this model, the number of output channels increases from 8 to 128 in each successive convolutional layer, indicating that the model is learning increasingly complex and abstract features as it progresses through the layers.

### *C. Batch Normalization*

In this model, each batch normalization layer is applied after its corresponding convolutional layer. Since the number of output channels from the convolutional layers increases from 8 to 128, the num˙features parameter is set accordingly for each batch normalization layer to match the number of output channels from the corresponding convolutional layer.

For example, in the first batch normalization layer (self.batchnorm1 = nn.BatchNorm2d(num˙features=8)), num˙features is set to 8, which matches the number of output channels from the first convolutional layer (out˙channels=8). This ensures that the statistics (mean and variance) used for normalization are calculated separately for each channel, and that each channel is normalized independently of the others.

### *D. Numerical Explanation*

- The input image size is 288x432.
- After passing through the first convolutional layer with a kernel size of 3 and stride of 1, the output size will be (288-3+2*0)/1 + 1 = 286x430.
- After passing through the first max pooling layer with a kernel size of 2, the output size will be 143x215.
- After passing through the second convolutional layer with a kernel size of 3 and stride of 1, the output size will be (143-3+2*0)/1 + 1 = 141x213.
- After passing through the second max pooling layer with a kernel size of 2, the output size will be 70x106.
- After passing through the third convolutional layer with a kernel size of 3 and stride of 1, the output size will be (70-3+2*0)/1 + 1 = 68x104.
- After passing through the third max pooling layer with a kernel size of 2, the output size will be 34x52.
- After passing through the fourth convolutional layer with a kernel size of 3 and stride of 1, the output size will be (34-3+2*0)/1 + 1 = 32x50.
- After passing through the fourth max pooling layer with a kernel size of 2, the output size will be 16x25.
- After passing through the fifth convolutional layer with a kernel size of 3 and stride of 1, the output size will be (16-3+2*0)/1 + 1 = 14x23.
- After passing through the fifth max pooling layer with a kernel size of 2, the output size will be 7x11.

Finally, the output of the last convolutional layer is flattened and passed through a fully connected linear layer, which has 8 output features corresponding to the 8 classes in the classification task. Therefore, the number of in˙features for the model is 7 x 11 x 128 = 9856

### *E. Forwarding The Model*

Given an input tensor x, the forward method applies a sequence of convolutional, batch normalization, activation, and max pooling layers to extract features from the input.

1) Convolutional layer 1: x = self.conv1(x) applies the first convolutional layer to the input tensor. The output tensor x has 8 channels.
2) Batch normalization layer 1: x = self.batchnorm1(x) applies batch normalization to the output of the first convolutional layer. This normalizes the output to have zero mean and unit variance.
3) Re-LU activation layer 1: x = F.relu(x) applies the rectified linear unit (ReLU) activation function to the output of the first batch normalization layer. This introduces non linearity to the model and helps to prevent the vanishing gradient problem.
4) Max pooling layer 1: x = F.max_pool2d(x, kernel_size=2) applies max pooling with a 2x2 kernel to the output of the first ReLU activation layer. This reduces the spatial size of the output tensor by a factor of 2.
5) Repeat steps 1-4 for convolutional layers 2-5.
6) Flatten: x = torch.flatten(x, 1) flattens the output tensor into a 1D vector for input to the fully connected layer.
7) Dropout: x = self.dropout(x) applies dropout regularization to the flattened output tensor. This randomly drops out some units to prevent over fitting.

8) Fully connected layer 1: x = self.fc1(x) applies the fully connected layer to the output of the dropout layer. The output tensor has 4 units, which correspond to the 4 possible classes.
9) Softmax activation: x = F.softmax(x) applies the softmax activation function to the output of the fully connected layer. This converts the output into class probabilities.

### *F. Training The Model*

A training method was constructed that takes the model, a device (GPU or CPU), training data loader, validation data loader, and number of epochs as inputs.We initialized the cross-entropy loss function and Adam optimizer with a learning rate of 0.0005. In each epoch, the model is set to training mode using model.train(). The loop then iterates over each batch of data in the training data loader. For each batch, the data and target labels are sent to the specified device. The model is called with the data batch to get the output. The optimizer's gradients are zeroed out with optimizer.zero˙grad(). The loss is computed using the cross-entropy loss function. The gradients are then computed using loss.backward(), and the weights are updated using optimizer.step(). The running loss and accuracy for this batch are recorded and printed.

The model is then set to evaluation mode using model.eval(). The loop then iterates over each batch of data in the validation data loader. For each batch, the data and target labels are sent to the specified device. The model is called with the data batch to get the output. The loss is computed using the cross-entropy loss function. The running loss and accuracy for this batch are recorded and printed.

## VI. Results

As a input of the CNN model, spectrogram of the audio files are used based on their genres. In the below Figure, it can be observed three different genres spectrogram. It can be hard to observed the difference between them for human eyes in some cases, but it is not hard for the deep learning model. Even in some points, the difference between the spectrograms were obvious.

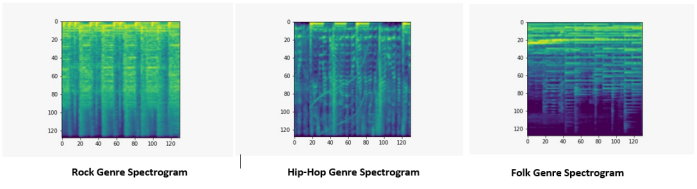


Fig. 6: Rock, Hip-Hop, Folk Spectrogram

The loss and the accuracy values were plotted as shown in the Figure 7 for the genre group [pop,instrumental,electronic]. From the plot it can be observed that the loss is decreasing while the accuracy is increasing through the epochs time. From the results it can be said that over fitting or under fitting issues are not observed. Training accuracy is observed as higher than the validation accuracy which expected. At the and the the validation accuracy is found as %60. Finally, the confusion matrix were plotted and the relation between true and predicted values were observed.

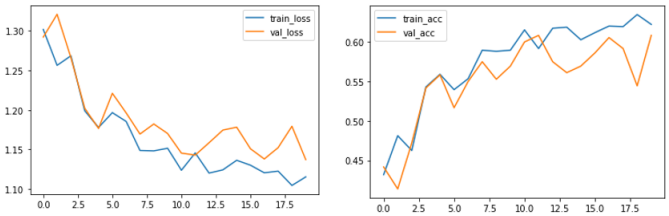


Fig. 7: Accuracy and Loss Graph of Model of [pop,instrumental,electronic] Training Set


Fig. 8: Confusion Matrix of [pop,instrumental,electronic]

The loss and the accuracy values were plotted as shown in the Figures 9 for the genre group [rock, hip hop, folk]. From the plot it can be observed that the loss is decreasing while the accuracy is increasing through the epochs time. From the results it can be said that over-fitting or under-fitting issues are not observed. Training accuracy is observed as higher than the validation accuracy which expected. At the and the validation accuracy is found as %75. Finally, the confusion matrix was plotted and the relation between true and predicted values were observed.

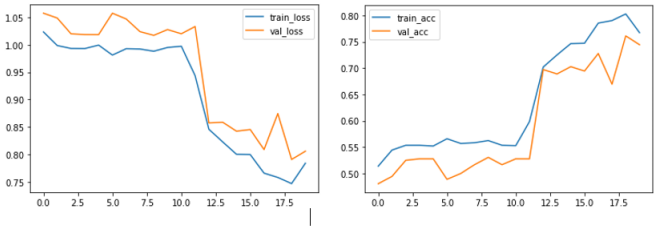


Fig. 9: Accuracy an Loss Graph of Model of [rock, hip hop, folk] Training Set

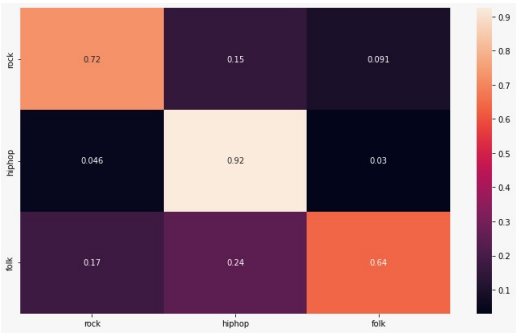


Fig. 10: Confusion Matrix for [rock, hip hop, folk]

Different accuracies were observed for different groups. Even the accuracies is not higher than %90 , appropriate predictions and performance can be obtained using the model. The accuracy for second group [rock,hiphop,folk] was found higher than the first group [pop,instrumental,electronic] . The reason may be attributed to the similarity between the spectrograms. Additionally, because just 3 seconds from the beginning of the mp3 files converted to the wav ,some patterns may not be sufficient for the model. For example taking the song from such as 0:15 to 0:18 or taking 5 second of the song may result more accurately.

In the first step of the project 800 .wav files were recorded and 800 spectrogram images were saved to the folders for each genres. However , because the limitation of the RAM capacity and the GPU issues the model could not be trained using all the data for each genres. Thus, 600 images were considered for each genre and the they were tried to classified by separating different groups. For example [pop,instrumental, electronic] and [hiphop, rock, folk]. When we tried to get more than 3 genres the colab environment tends to collapse. Also, we tried to use more than 3 genres or all the genres with less amount of images (such as 200 or 300 for each), however the accuracy could not obtained properly with less amount of data. Thus, 600 images were used for each genre by grouping them.

Epochs time set as 20 while training the model. More epoch time needs more RAM or GPU support ,so even a fewer amount of epochs time , the accuracy was obtained properly. As a note ,Increasing epochs time may result with more accurated results.

## VII. Concluding Remarks

Although all genres couldn't used during the process, the accuracy was enough for the classification for the usable ones. When compared the other methods rather than the deep learning, CNN model is gave efficient result for the music genre classification. There were some important points of the project which are GPU and capacity problems. Solving these problems can give higher accuracy with more genre and dataset.

In the end, the music genre classification is successfully done by CNN model with PyTorch library. For the future improvements, more than one model can be combined, even more than one spectrogram type can be used. These combination of different approaches can give higher accuracy. However, it can be caused more complicated problems too.

In the project deep learning models are deeply investigated. Even in the project one model was used, while searching the proper model various model was studied. The different Python libraries were deeply investigated to use them by efficiently. Understand the general problem and concept is a advantage of the learning outcome. The another learning point is analysing and preprocessing the data. This point is one of the hardest part but learning outcome is high too. Since, the dataset is too big managing the GPU, capacity is a hard part of the project. At the same time, reading the documentation and the understanding the dataset was little bit complicated. One of the main difficulties was the capacity and GPU. Since there is no Google Collab Pro, the GPU was not usable in sometimes. This caused the process to slow down. For example, changing the epochs parameter can effect the learning accuracy. However, when its increased system can crushed because of the overload. However, generally this problem was solved with different approaches such as using less epochs, not get the all the genres.

## References

[1] T. Lahovnik and V. Podgorelec, "Music genre classification based on spectrograms of the sound recording using an ensemble of CNNs," in *Proceedings of Student Computing Research Symposium (SCORESâ€™22), ACM*, (New York, NY, USA), Oct. 2022.

[2] W. Zhang, W. Lei, X. Xu, and X. Xing, "Improved Music Genre Classification with Convolutional Neural Networks," in *Proc. Interspeech 2016*, pp. 3304–3308, 2016.

[3] E. Zacharia, "Predicting the genre of music samples without extracting audio features using a convolutional neural network," Master's thesis, Specialty in Data Analytics, The University of Chicago, 2021.

[4] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.