Music Genre Classification

Albin Andersson Svensson, Markus Jonsson, Kim Sundelius, Oskar Sundin

Luleå University of Technology, Sweden

***Abstract*—This report will present the results of tackling music genre classification. The method used composed of converting an audio file into an image representation which was then fed into a CNN for image classification. Several pre-processing methods were explored and two different CNN models were tested on two different datasets. Results showed that one of the models VGG19 performed best with a log-frequency preprocessing technique as input to the model. However further experiments, more rigorous testing with more datasets and models must be conducted before being able to accurately determine advantages and disadvantages with the different preprocessing techniques when dealing music genre classification.**

I. Introduction

As of 2020 many have easy widespread access to audio streaming services such as iTunes and Spotify with large music databases. With the help of genres it is possible to label and organize songs based on instruments and rhythm and harmonics [1]. Much of the related work is focused on classifying songs into broad genres such as rock, metal, jazz etc. However there are specific music genres in the hundreds, subsets of the broad genres and songs that may be a combination of several genres [2]. For this reason this study attempted to create a neural network capable of multi-label classification but were unable to produce satisfactory accuracy results. The future work section will describe the main challenges experienced in more detail. Due to these shortcomings the focus of the study shifted towards comparing different preprocessing techniques, to find out the strengths and weaknesses of these techniques when dealing with single-label classification.

In this study, 5 different preprocessing techniques and two datasets fma_small, a subset from the Free Music Archive, and GTZAN are used on two different CNN models to compare performance of the different preprocessing techniques for single-label classification.

Motivation

The automation of genre classification can have applications in playlist creation tailored for specific users and substantial speed increase for classifying large amounts of song data when comparing a humans capabilities of tagging manually versus a neural network.

Contribution

This report could give some guidance when choosing a preprocessing method without delving too deep into the different parameters that contribute to the performance of neural networks and each preprocessing method.

II. Related Work

Music genre classification using GTZAN dataset and a custom dataset [3]. Two models and comparisons between them approach for music genre classification similar to this paper [4]. Genre classification using MFCC and mel-spectrograms as input for a CNN [5]. An attempt to define a “perfect" classifier for the GTZAN dataset is made in [6], achieving an accuracy of 94.5% due to limitations of the dataset itself. In [7], a CRNN model was used for a similar task, acheiving 65.23% accuracy on the fma_small dataset. Regarding state-of-the-art for this task, it proved to be difficult to find a verified result. In [8], it is mentioned that state-of-the-art models performance lies around 69% for fma_small and around 94% for GTZAN.

III. Experimental Setting

A. Task

The goal is to train a neural network to be able to distinguish between features for different genres thus being able to classify songs into specific genres.

B. Model Architecture

The first model that was evaluated with our data for single-label classification was the VGG19 model [9] and the second was a custom made model. The custom made model had the architecture described in the following table.

TABLE I
Architecture of our model.

Layer Number	Layer Type	Number of Filters	Kernel Size	Activation Function	Padding
1	Conv2D	23	(3,3)	relu	same
2	Conv2D	64	(3,3)	relu	same
3	Conv2D	128	(3,3)	relu	same
4	MaxPool2D	-	(2,2)	-	-
5	Conv2D	256	(3,3)	relu	same
6	MaxPool2D	-	(2,2)	-	-
7	Flatten	-	-	-	-
8	Dense	-	-	softmax	-

The input size (image size) is $128 \cdot 128 \cdot 3$ and the output size corresponds to the amount of classes for the dataset.

C. Datasets

Two datasets were used, the popular GTZAN dataset which contains 10 different classes with 100 30 second samples in each class and the FMA-small dataset with 8 main classes with 1000 30 second samples in each class.

The FMA dataset was first intended to be used with our multi label classifier but since it was very imbalanced with the sub-classes we only used the main classes for our single label classifier.

D. Preprocessing

The following images are audio samples that have been converted by their respective preprocessing technique which was used as input for the models. To convert audio samples into these images a Python package for music and audio analysis called LibROSA [10] along with Matplotlib [11] to plot the images. For input size, the shape $128 \times 128 \times 3$ was used, representing width, height and rgb channels respectively. For other parameters, provided default parameters were used, see the Google Colaboratory notebook for full implementation https://colab.research.google.com/drive/1UHi_hZtbZFei1IvZyrKnbzXXQcgoLS_O?usp=sharing.

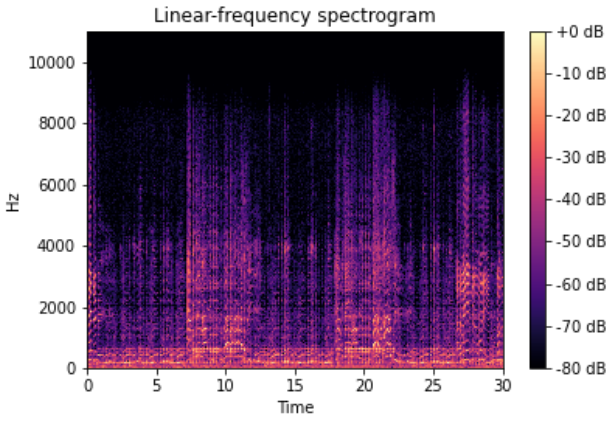

Fig. 1. Converted audio sample “blues.00019.wav" from GTZAN into a Linear-frequency spectrogram.

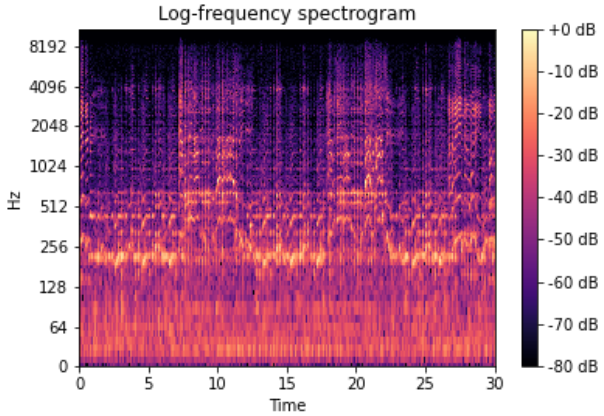

Fig. 2. Converted audio sample “blues.00019.wav" from GTZAN into a Log-frequency spectrogram.

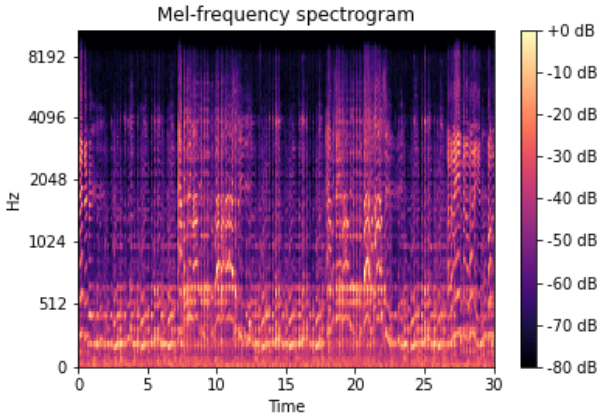

Fig. 3. Converted audio sample “blues.00019.wav" from GTZAN into a Mel-frequency spectrogram.

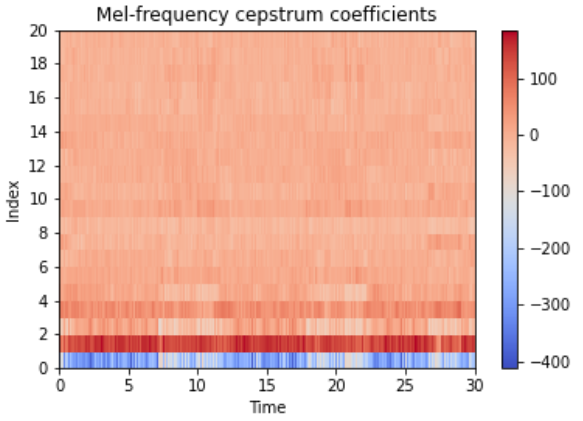

Fig. 4. Converted audio sample “blues.00019.wav" from GTZAN into a Mel-frequency cepstrum coefficients.

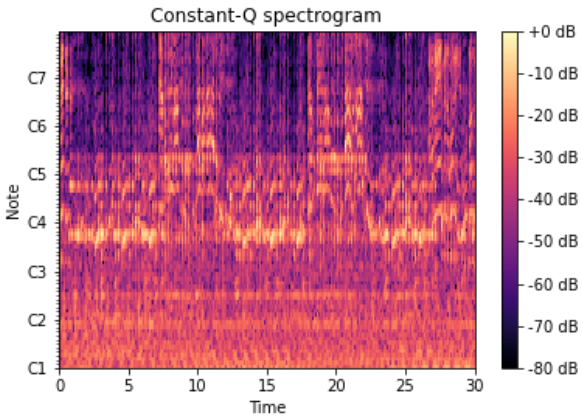

Fig. 5. Converted audio sample “blues.00019.wav" from GTZAN into a Constant-Q spectrogram.

E. Training Procedure

IV. Results

TABLE II
SUMMARIZED RESULT ACCURACIES WITH FEATURES OBTAINED FROM ALL TESTED PREPROCESSING TECHNIQUES.

Dataset (train, valid, test)	GTZAN (60%, 20%, 20%)		fma_small (80%, 10%, 10%)	
Feature \ Model	Our	VGG19	Our	VGG19
Linear	52.3%	64.3%	42.1%	50.7%
Mel	56.8%	69.5%	46.1%	54.5%
MFCC	53.5%	62.0%	38.8%	46.2%
Log	**58.9%**	**71.0%**	**47.3%**	**55.3%**
CQT	57.2%	67.6%	44.4%	52.7%

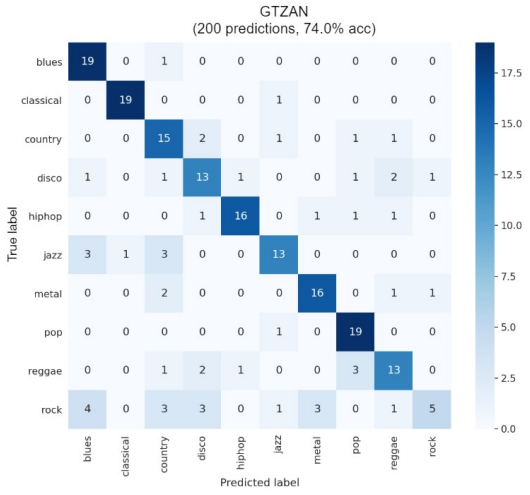

Fig. 6. Confusion matrix on test set for model using our custom architecture.

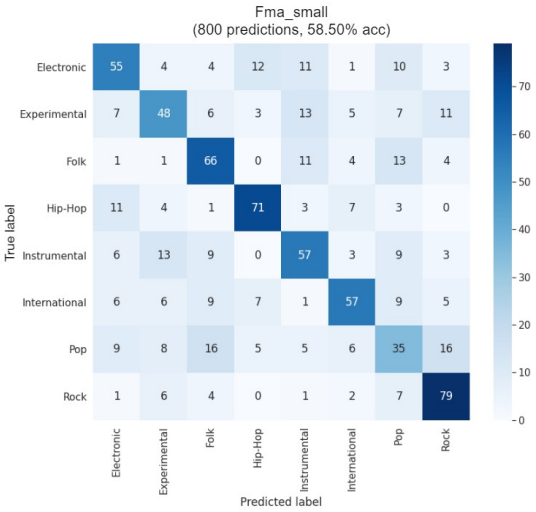

Fig. 7. Confusion matrix on test set for model using the VGG19 architecture.

V. Analysis

From our testing we can see that the best performing preprocessing in terms of accuracy was Log followed by Mel, CQT, Linear and lastly MFCC. We believe that the Log performed best since it captured the frequency spectrum in an appropriate way meaning that it cared less for higher frequencies since it is often very sparse (see fig 1). This seems to be the case since the linear scaling performs the worst out of the spectrograms.

But why is Mel and CQT worse than Log? We believe that since Mel is slightly focused on the “human perceived melody" it focuses on the wrong information and not on vital parts when it comes to identifying genre.

When it comes to CQT, which scales the spectrum equally between the notes C1 to C8, we think a similar situation applies where it doesn’t capture information above C8. CQT should be more tested with a higher threshold for what information we retain. We think that our test results are inconclusive when it comes to deciding whether equal scaling between C notes is preferable for genre identification.

MFCC should in our opinion perform similar to Mel since it focuses on how humans perceive sounds. Similarly to Mel we believe that a focus on what a human hears in a song is not preferable for genre classification.

A more comprehensive reasoning for using spectrograms instead of raw data or MFCC is given in the “Related Work" section of [8].

We also noted later that the GTZAN dataset is not entirely accurate. For instance, there are several mislabelings most commonly in the “Rock" genre and distorted audio clips. See [6] for more in-depth information. This could potentially be a reasoning behind the difficulties when it comes to classifying the “Rock" genre seen in 6.

For the fma_small dataset there seems to be additional difficulties. In our results the two genres to proved most difficult are “Experimental" and “Pop", see 7. A reasoning for the “Experimental" genre being difficult could be because it consists of a relatively large variety of genres as well as tracks with an unspecified genre making it difficult to classify. However it seems more difficult when it comes to reasoning behind why the “Pop" genre would be difficult to classify. In [7] which found similar results, it was mentioned that pop music is often regarded as the softer alternative to rock and that it could lead to similar features as well. Thus making it more difficult to tell apart but this is mostly a theory and needs further investigation.

VI. Conclusion

From the results it seems that log-frequency spectrograms performs the best with the models. We can also see that our model performs worse compared to the VGG19 model.

Our test results gives some intuition about what is important for genre classification but more tests when it comes to parameters when generating the images are needed to make stronger conclusions.

VII. Future work

The main challenges for us with multi-label classification are choosing suitable evaluation metrics and address the imbalance of the data-set appropriately.

The two main ways of dealing with an imbalanced data-set is oversampling or undersampling. Generally this means to either generate synthetic data sampled from the original data or removing some of the original samples respectively. We were looking into using either SMOTE [12] or IRUS [13] for our classifer.

A. IRUS

IRUS stands for Inverse Random Undersampling and takes a lot of inspiration from "bagging" which is basically the idea of having multiple binary classifiers vote on which accuracy we can say that the sample belongs to the according label. The amount of classifiers needed for each label depends on the ratio of the minority (the corresponding label) and the majority (the rest of the data-set).

For each classifier IRUS tries to keep a balance between the positive class and the negative class through randomly picking, with no replacement, samples from the majority class for training and evaluation. The individual classifiers will then learn to separate the positive class from the selected negative samples. The classifiers will then vote (commonly through calculating the mean) on how certain the network is that the sample belongs to that label.

This will require many classifiers depending on how imbalanced the data-set is. Keeping the classifiers simple and maybe use pre-trained networks for finding features in an image might help reduce the cost.

If we would implement this with the FMA-small data-set we would probably simplify the data-set through removing the labels for which we only had very few samples (many labels had even fewer than 10 samples) to keep the ratio small and thus requiring fewer classifiers.

B. General testing

More rigorous testing with more datasets and different models should be done to determine the advantages and disadvantages of the different preprocessing techinques.

More thought into selecting an appropriate input size could improve not only the performance but also the training time. Such as using another aspect ratio for width and height or reducing the the number of channels to one since all of the three rgb channels were used for the purpose of describing one single scalar value. In this project, the image size was not thoroughly inspected and was mainly determined to find some kind of balance between reducing training time without losing too much performance. However, a better balance could most likely be found. Regarding the rgb channels, the models used in this project (with some small modifications) both performed worse when lowering the number of channels. This test was definitely not extensive enough, which is why it is not presented in this report. Creating a single channel from three could be done in many different ways and the models themselves were created and tested in using the three channels.

Another suggestion would be to further investigate using 1D CNNs instead of 2D CNNs. As mentioned in [8], 2D convolutions over frequency dimension is uninterruptible and using a 1D CNN would also provide more computationally efficient model in comparison to a model using a 2D CNN.

References

[1] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, pp. 293 – 302, 08 2002.

[2] F. Pachet and D. Cazaly, "A taxonomy of musical genres," in *Content-Based Multimedia Information Access - Volume 2*, ser. RIAO '00. Paris, FRA: LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 2000, p. 1238–1245.

[3] F. J. Albert Jiménez, "Music Genre Recognition with Deep Neural Networks," Barcelona, Spain, 2017.

[4] H. Bahuleyan, "Music genre classification using machine learning techniques," 2018.

[5] S. Sugianto and S. Suyanto, "Voting-based music genre classification using melspectogram and convolutional neural network," *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pp. 330–333, 2019.

[6] B. L. Sturm, "The state of the art ten years after a state of the art: Future research in music information retrieval," *Journal of New Music Research*, vol. 43, no. 2, p. 147–172, Apr 2014. [Online]. Available: http://dx.doi.org/10.1080/09298215.2014.894533

[7] C. C. Zhang C., Zhang Y., "Songnet: Real-time music classification," 2018.

[8] W. Bian, J. Wang, B. Zhuang, J. Yang, S. Wang, and J. Xiao, "Audio-based music classification with densenet and data augmentation," 2019.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[10] Librosa, accessed 2019-05-21. [Online]. Available: https://librosa.github.io/librosa/

[11] Matplotlib, accessed 2019-05-21. [Online]. Available: https://matplotlib.org/

[12] R.-M. J. C.-D. C. Giraldo-Forero A.F., Jaramillo-Garzón J.A., "Managing imbalanced data sets in multi-label problems: A case study with the smote algorithm," vol. 8258, 11 2013, pp. 334–342.

[13] d. J. M.-H. F. Charte F., Rivera A., "Inverse random under sampling for class imbalance problem and its application to multi-label classification," vol. 45, no. 10, 2012, pp. 3738 – 3750.