Report for Music Genre Classification Assignment

Emmanouil Theofanis Chourdakis

February 2020

1 Introduction

In this report we summarize the development and deployment of a model for Music Genre Classification (MGC). The model was developed using the GTZAN[1] [7] dataset for music genre recognition. A Convolutional Neural Network (CNN) is used to extract musically relevant frame features from a song. A tree boosting classifier is then used to estimate probabilities each such frame is associated with a specific genre. A final decision step assigns an overall genre to the song based on these probabilities. We report accuracies slightly higher than what was reported in the literature for the same dataset. Finally, we discuss methods to speed up inference, making the model suitable for deployment in production.

2 Methodology

The dataset we were asked to train our model to is the GTZAN dataset [7]. While it has been used most often in research, it suffers from duplication and mislabelling and it has also been found that its annotations do not agree with human experts [5, 6, 3]. Furthermore the dataset itself is quite small, with only 1000 files of 30 seconds each placing it among the smallest popular datasets for MCG which would normally hinder development of a deep learning model [5, Table 1]. To compensate for the small size of GTZAN we decided to adopt transfer learning, that is, to fine-tune a larger model to this dataset. The approach is the same approach used in [4] and can be visualized in Figure 1. It consists of using an existing CNN called MSD_MUSICNN to extract features from audio frames in our training set, and then use XGBoost[2] to produce taggrams and a final step decides the dominant tag and therefore decides the genre of the song.

The genre decision process is as follows: before the audio is processed by the CNN it is split into 3 second segments with 1 second hop size and each segment is transformed to a 96×1 log-mel spectrogram vector $\mathbf{b_i}$. Each such vector then is passed as input to the CNN which produces a corresponding feature vector of size 743×1. These feature vectors are reduced to their 128

[1] `http://marsyas.info/downloads/datasets.html`

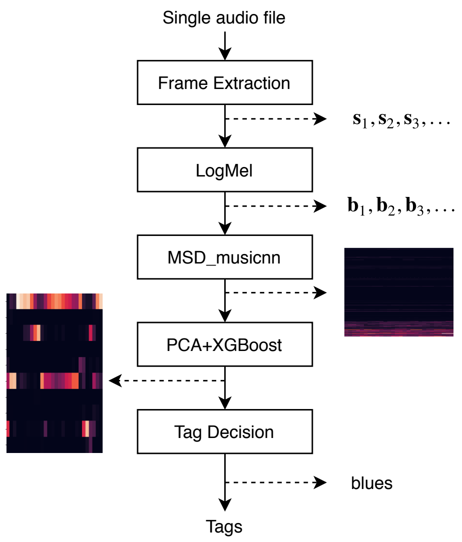
Single audio file
Frame Extraction
$\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \ldots$
LogMel
$\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \ldots$
MSD_musicnn
PCA+XGBoost
Tag Decision
blues
Tags

Figure 1: Audio-tag assignment methodology

principal components using PCA and fed into an XGBoost classifier with the default hyper-parameters. For the CNN we use MSD_MUSICNN from the author's github[2] which has an architecture able to 'learn' features relevant to both the timbral and the temporal characteristics of music. Additionally, the model has been trained using 200K audio files from the Million Songs Dataset[1] making it an excellent choice for a feature extractor. For the classification part we used XGBoost[2]. This classifier is used to assign probabilities for each a tag for each feature vector. We use these probabilities to estimate and plot the temporal evolution of tags (taggrams – Figure 2). Estimating probabilities slows classification by almost 2-fold (190 vs 105 milliseconds for inference on a 30 second song) and we only estimate probabilities to visualize taggrams. For production, we use the classifier's tag decision for each feature vector without estimating probabilities first.

We decide the dominant tag for each song in two ways depending on whether we are provided probabilities for tags or not. We either use probabilities in the taggram or we count tag frequencies. In the case of a taggram we average the probabilities across frames in the taggram and choosing the tag with the highest probability. As we mentioned in the previous section, estimating probabilities however is slow and for practical purposes we use the decisions of the classifier directly and do not estimate probabilities. In that case, the classifier will just return a tag decision for each feature vector and we keep the most popular decision across all vectors in the song, as the genre of the song.

[2]`https://github.com/jordipons/musicnn`

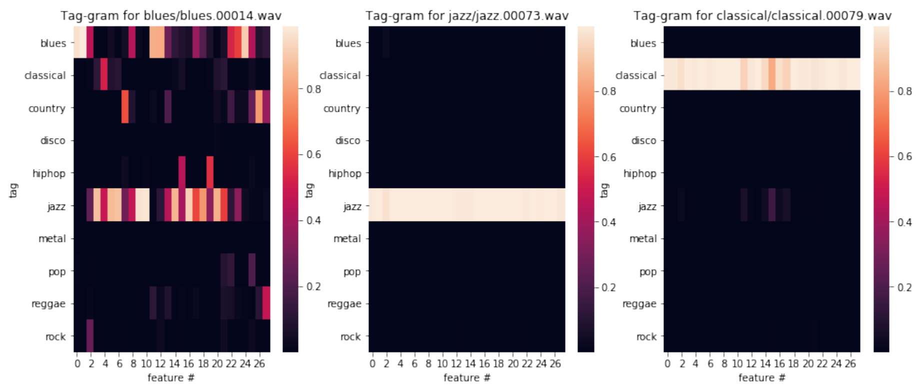

Figure 2: Taggrams for three different songs. Each subsequent feature is extracted from a 3 second segment 1 second after the previous feature. The lighter the 'box" in the taggram the higher the probability that segment is of the specific tag. We observe that contrary to the jazz or classical songs, the blues song is often confused for jazz.

' **Model**	**Acc (avg.)**	**PR-AUC**	**ROC-AUC**	**Train**	**Inf.**
SVM[4]	0.7724	–	–	—	—
SVM	0.7860	0.8822	0.9740	4m40s	45.9s
XGBoost	0.7826	0.8861	0.9707	26.2s	569ms

Table 1: Results comparing different classifiers. The topmost results are the ones reported in [4]. We observe that the SVM classifier trained with the same hyper-parameters scores more than 1% higher than the one previously reported. We also observe that the XGBoost classifier, while scoring a little lower than our SVM model, is much faster in both training and inference and thus preferable.

3 Results

Given the peculiar nature of the GTZAN dataset and its small size we decided to use the split used in [4][3] which the authors claim has been fault filtered. The split is 443, 197, and 290 songs for the train, validation, and test set respectively (keeping only 930 songs from the original 1000). By the same split we can also compare our models' performance against theirs. We report a best average accuracy of 78.60% in our test-set (compared to 77.24% reported in [4]). We also report an average precision of 0.88 and ROC-AUC of 0.97. For completeness, we also trained a classifier using the same configuration as theirs (SVM with a Radial Basis Function (RBF) kernel and $C = 1$) which surprisingly scores more than 1% higher than in their case. Results can be seen in Table 1. The class

[3]`https://github.com/jordipons/sklearn-audio-transfer-learning`

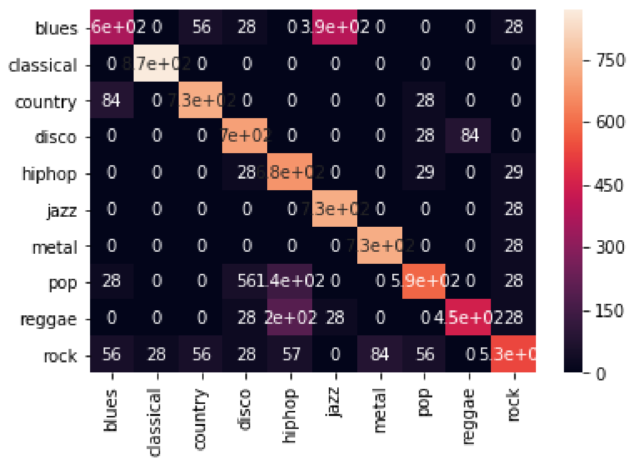

Figure 3: The confusion matrix for our test set. We can see that ‘blues’ is most often misclassified as ‘jazz’ which is consistent with what we show for one song in Figure 2. Additionally we observe pop and reggae are also quite often misclassified as ‘hiphop’. On the other hand, our model scores perfect accuracy for ‘classical’.

confusion matrix can be see in Figure 3.

4 Preparing for deployment

We included a Docker-Compose file which deploys a Flask[4] microservice that can accept files using the `HTTP POST` method and return a `json` reply with their respective musical genre predicted by the classifier. Such services can be deployed to cloud providers such as Microsoft Azure and Amazon EC2 and can be easily integrated with e.g. web or other services. An example of a request to such service can be given from the command line:

```
curl -F ’audio=@classical.00067.wav’ http://localhost:5000
```

where `classical.00067.wav` is an audio file in the local directory. The result is of the form:

```
{"duration":1.94...,"message":"classical","status":"success"}
```

We observe that classification takes 1.9 seconds to happen for a 30 second song. We can reduce this time by taking only a 7 second window around the middle of the song instead of the full song. This reduces time taken to classify to around 0.94 seconds and also guarantees equal tagging time regardless of song duration. While we have not measured model performance using this approach we do not

[4]`https://flask.palletsprojects.com/en/1.1.x/`

expect performance to drop dramatically since we expect the majority of songs to recognizable in their middle (in contrast e.g. to an intro or outro that might confuse genre recognizers).

5 Conclusion

This report summarized model training and deployment for a simple Music Genre Classifier for the GTZAN dataset. We presented an architecture that leverages transfer learning from a much better model trained on a different dataset to account for GTZAN's shortcomings. We reported slightly better performance than a similar previous work. Future work can include exploring whether those differences are due to the different genre decision step or the classification strategy. Future work can also include measuring classification vs computation performance when choosing part of the song instead of the whole song to classify which is important in production. Further study can also be done to test our assumption made in the previous section that the genre of a song can be reliably determined from a 7 second segment segment in its middle.

References

[1] Thierry Bertin-M. et al. The million song dataset. In *12th International Conference on Music Information Retrieval*, October 2011.

[2] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2016.

[3] H. Pálmason et al. Music genre classification revisited: An in-depth examination guided by music experts. In *International Symposium on Computer Music Multidisciplinary Research*, September 2017.

[4] Jordi Pons and Xavier Serra. musicnn: pre-trained convolutional neural networks for music audio tagging. In *Late-breaking/demo session in 20th International Society for Music Information Retrieval Conference (LBD-ISMIR2019)*, 2019.

[5] J. Ramírez and M J. Flores. Machine learning for music genre: multifaceted review and experimentation with audioset. *Intelligent Information Systems*, pages 1–31, 2019.

[6] B. L. Sturm. An analysis of the gtzan music genre dataset. In *2nd international ACM workshop on Music information retrieval with user-centered and multimodal strategies*, November 2012.

[7] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.