Music Genre Classification

Klemen Škrlj
Faculty of Computer and Information Science
Univeristy of Ljubljana
Email: ks5379@student.uni-lj.si

***Abstract*—In this article we present our research on applying different methods for music genre classification on a popular database GTZAN. We show our pipeline for extracting features from raw audio signal and scaling them into usable data. Then we list and describe different traditional methods like k-NN, SVM, Random forest, etc. which use these features. We also review application of deep neural networks in this domain. From our results we can deduce that SVM is the most suitable traditional model with classification accuracy of 78.4%, while CNNs work best when input is sub-sampled and majority voting is introduced at the end. Here our classification accuracy is 93.6%.**

***Keywords*—acoustic features, machine learning, deep learning, STFT, CNN**

I. Introduction

Automatic music classification is a fundamental problem for music indexing, content-based music retrieval, music recommendation and online distribution. Since the shift of music to digital platforms like Spotify, Soundcloud and others, the growth of such databases is exponential. It became clear that a fast and efficient system to keep everything organized would be beneficial to all parties involved.

Music genre is one of the most common factors used when we distinguish different songs between each other. Although human response can be a bit biased, we can agree to a subset of broad, worldwide known genres into which we can classify most of the songs. But for some songs, especially newer ones, this is not easy, since they merge many different styles together. For example, if a person says that they like pop music it could mean that they like Michael Jackson or Justin Bieber. Nevertheless, those pieces still have some characteristics, which might not be easy for humans to hear, but a machine learning model can use them to distinguish between genres.

In our research we focus on testing out different approaches to solving a problem of music genre classification. We extract features from raw audio signal and then use many traditional machine learning models and analyze their results. Additionally, we also test how good deep neural networks work on this domain and present different usage types.

In Section II we present related work that was done already done on this topic. In Section III we first present our dataset. Then we show how the features were extracted and selected. After that a hyper parameter analysis is given for each used traditional method and lastly, we present different approaches of using deep learning methods. In Section IV our evaluation metrics are proposed and detailed analysis of results is shown. Section V concludes the paper.

II. Related work

Music genre classification is a longstanding problem in machine learning. Authors of article [1] extract timbral texture, rhythmic content and pitch content from audio signal and use that in simple statistical based models like Gaussian mixture model and k-NN for classification. They also present GTZAN dataset, which is since then widely use in this problem domain. Article [2] describes a method where zero crossing rate, beat spectrum, MFCC and similar features were used in a SVM model. They reported a higher accuracy then previous methods, but more good features are needed for improvement. Authors of [3] compare two different textural feature extractors for music classification. First they convert signal to spectrogram and then get features with two methods; GLCM (Gray Level Co-Occurrence Matrix) and LBP (Local Binary Patterns). Then an SVM classification model is fitted. Their research shows that LBP method outperforms all others and reaches 80% accuracy on a Latin music dataset [4]. Similar procedure was done in [5] where authors combined visual and musical features in an SVM model and reported 77% accuracy on GTZAN dataset.

Since the recent success of CNN models in other domains, the research in this field is more focused on deep learning methods as well. Authors of [6] show how CNN's improve accuracy. They combine maximum and average pooling layers to provide higher layers with more information and use skip connections in the network. As an input they use spectrograms where clips are split into smaller ones so more data for learning is present. In [7] voting system is presented where the CNN network classifies each sub-sample separately and final classification is the majority class. Authors of [8] replace classification head of the CNN network with traditional models where SVM gave the best results. In [9] authors suggest using multimodal approaches for genre classification. Here they use cover photo of an album and actual sound signal. To merge this information they train two separate CNNs and connect them in the last layer. Final results show improvement but usually only one modality is present in current datasets.

Many articles ([10], [11] and [12]) did comparisons between different traditional approaches and deep learning ones. They report that more robust models like SVM or Random forest do the best in most cases, while CNN models work well when song is split into smaller sub-samples which generates more data for learning.

III. Methodology

In this section we present details of our implementation of different systems used for music genre classification. First we present our dataset (III-A), then we show our feature extraction pipeline (III-B) and lastly describe all traditional (III-C) and deep learning (III-D) approaches that were evaluated.

A. Dataset

Similar to other research on this topic, we chose to use GTZAN dataset, which was first proposed by Tzanetakis and Cook in [1]. It consists of 1000 music clips (.wav files), all length of 30 seconds. Every clip was sampled at a sampling frequency 22050 Hz and quantized with 16 bits. The samples from dataset are uniformly distributed between 10 different genres: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock. This makes it really good for fitting classification models since we don't have problems with imbalanced data. We first extracted test set from the data which represents 25% of songs. Since we want for our model to be robust to all available genres, we used stratified sampling. After that we used 15% of the remaining data as validation set (also with stratified sampling) and the rest for training.

B. Feature extraction and selection

Since our initial data consists of raw audio signals we have to perform feature extraction. This is done for two main reasons:

- **Dimensionality reduction**: Whole raw audio clip is usually to large to use efficiently.
- **Meaningful representation**: Although a raw signal contains all the information, we want to help the model with learning and extract meaningful characteristics from this and instead use such vector as an input.

Because we are dealing with signals, we have to use digital signal processing techniques in our feature extraction pipeline. As per [13] we can split manually extracted features into two main categories: time and frequency domain features. To go from time to frequency domain we have to use Fourier transform, more accuratly Short Time Fourier Transform (STFT). For these window and hop size need to be specified. Authors in article [14] tested different configurations on GTZAN [1] and determined that window of size 512 samples and hop length of 256 give the best results so this are constants that we use as well. All of the feature extraction is done using python package names librosa [15].

1) Time domain features:

- **Root Mean Square Energy (RMSE)**: We calculate it using this equation

$$\sqrt{\frac{1}{N}\sum_{n=1}^{N}|x(n)|^2}$$

for every individual frame, where x denotes a signal and N is number of samples in a frame.

- **Zero crossing rate**: We divide a signal into frames and calculate number of signal crossings of x axis for every frame.
- **Tempo**: This feature gives us an estimate for how fast or slow a song is, which would intuitively help us differentiate between genres.

2) Frequency domain features:

- **Spectral Centroid**: Computed frequency around which most of the energy is centered for each window.
- **Spectral Band-with**: Defined as a band-with of an audio signal at one-half the peak maximum.
- **Spectral Roll-off**: Defined as a frequency where 85% of the total signal energy lies.
- **Spectral Contrast**: Each frame is divided into frequency bands and spectral contrast is defined as difference between the maximum and minimum magnitudes.
- **Chroma features**: For each frame we calculate total energy of the signal corresponding to 12 pitch classes (C, C#, D, D#, E, F, F#, G, G#, A, A#, B)
- **Mel-Frequency Cepstral Coefficients (MFCC)**: First a power spectrum from STFT is computed and triangular MEL filter bank (20 different banks) is applied which mimics human perception of sound. Then a discrete cosine transform of the logarithm of all filterbank energies is taken, which gives final MFCC vector.

Since most of the features work by splitting the original signal into smaller frames, we are still left with a vector of values for every feature. We would want a representation which would not take time into an account. If we, for example, take two 30 second clips of two different songs of same genre, for one at the beginning and for the other at the middle of a song. Typically a lot of songs start more slow and build up as time goes on, but the clip from the middle would not have this characteristic so our model would not work as good. This is why we apply statistical measures which try to eliminate and normalize such differences. Based on previous research we knew that mean and standard deviation work well for this problem. By doing this we were left with 2 features from every vector which produces 89 total features for every sound clip.

For our model to work well, we need all the features to be standardized into range with zero mean and unit variance. Without that, all the models that use some kind of distance calculations would be overwhelmed by features with bigger numbers and would prioritize those more. It is also good to scale data for models that use gradient descent since the loss function works better for smaller values. In addition, we also tested, if mapping data to Gaussian distribution would help. We didn't see very large improvement but decided to do it anyway since some models could benefit from it (eg. Logistic regression).

After feature extraction and scaling we performed feature selection based on importance. Sometimes there are some features that don't help the final classification or even hurt the performance. This is why it is a good idea to test for such occurrence. We have a classification problem with

discrete classes and continuous features. This is why we used ReliefF [16] method. We ran it on train data for 1000 iterations with number of neighbours set to 100. Table I shows some of the most and least important features with their scores.

TABLE I
3 MOST AND LEAST IMPORTANT FEATURES

Feature Name	Importance
spectral-contrast7-mean	0.130
spectral-contrast5-mean	0.086
spectral-bandwith-mean	0.081
mfcc20-mean	0.011
chroma2-std	0.011
mfcc18-mean	0.011

Based on those results we employed wrapper method to find the best subset of features. Here we started with an empty set and gradually added features one by one based on their importance score descending. We used every subset to train a RandomForest model and tested performance on validation set. With this approach we found out that the best subset includes every feature (89 features in total).

We used those features for a PCA projection into 2D space as seen on Figure 1. This gives us some indication to which classes could be harder or easier to differentiate. In this image we can see that classical, metal and hip-hop songs are a bit different, while others don't separate as much.

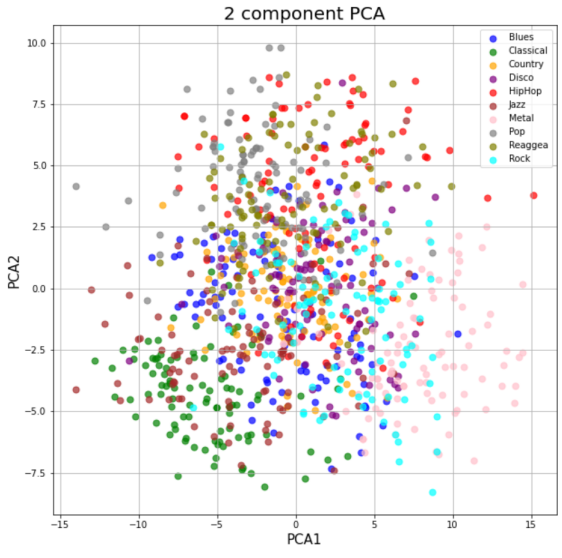

Fig. 1. Feature representation of dataset in 2D

C. Traditional methods

Here we list different traditional learning methods that use hand crafted features, which were previously described. For every model we show how we got the best possible hyperparameters.

1) ***K-Nearest Neighbors****:* k-NN is a distance based supervised learning algorithm. It uses lazy learning which essentially means that a model is not created. When a test example comes, the algorithm finds k most similar examples from train set based on (in our case) Euclidean distance and classifies current example into most common label. Because of this value of parameter k needs to be optimized for every problem separately. We do this by looping over different k values, building a model for everyone and evaluating it on validation set based on accuracy and F1-score. We then choose the best one. As seen on Figure 2, we get best results when k is 10 or 11, so we chose $k = 11$.

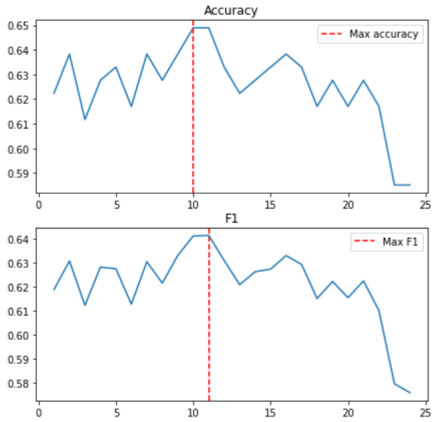

Fig. 2. Accuracy and F1-score for different k values

2) ***Random Forest****:* This is an ensemble learner that combines predictions from n regular decision trees. Each decision tree is trained with a subset of training samples which are chosen by bagging method and only a random subset of size $\sqrt{M}$ (where M is total number of features) is used. To optimize such method, we looped over different number of trees and compared results on validation set. In the final model $n = 20$ was used since it gives the best trade-off between accuracy and model complexity as seen on Figure 3.

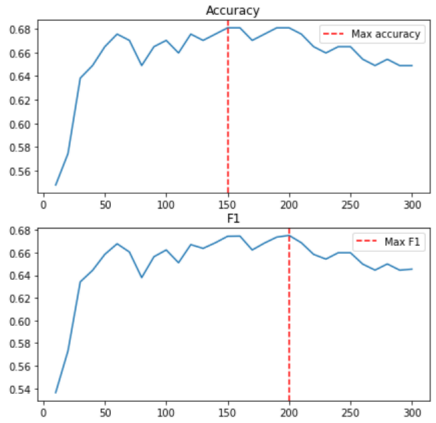

Fig. 3. Accuracy and F1-score for different n values

3) ***Logistical Regression****:* Logistic regression is a classifier typically used for binary classification, but here we use one-vs-rest method. With this approach we build a separate model for each class. When a test example is given, it is ran through each

model and the one that gives the highest probability predicts the final class. When building models, L2 regularization is used to prevent overfitting on train data. Since this method is a statistical one, it benefits the most from mapping all features to Gaussian distribution.

*4) **SVM**:* Similar to logistical regression this model as well uses one-vs-rest method for multi-class classification. During learning process, the model tries to find a hyperplane that best divides high dimensional dataset into two classes. Since it uses a kernel to map data into higher dimension, this parameter is important for good results. In our example we use radial basis function kernel. In addition to this we optimized regularization parameter C which is inversely proportional to strength of L2 regularization. Figure 4 shows that $C = 4.4$ gives the best results.

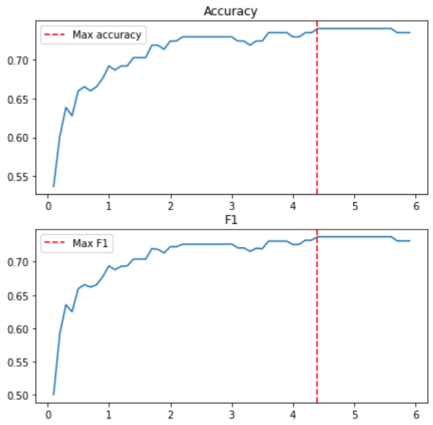

Fig. 4. Accuracy and F1-score for different C values

*5) **Gradient Boosting**:* This is another ensemble classifier where a final predictions is obtained by combining many predictions of weak decision trees. At the beginning, all learning examples are given equal weight. After each iteration the wrongly classified ones are given higher weight while the correctly classified become less important. Here we perform 100 boosting iterations where learning rate is equal to 0.5.

*6) **Multilayer Perceptron**:* Multilayer perceptron is a class of feedforward artificial neural networks. In our case we used a configuration with one hidden layer of 30 fully connected neurons and ReLU activation function since we get the best results. We train a model for 200 iterations with Adam optimizer, 0.0001 L2 penalty and 0.001 learning rate.

D. Deep Learning methods

Short-time Fourier transform of a digital signal reveals the Fourier spectrum on each short segment, that the original clip was separated to. Since this is a 2D matrix, we can make a visual representation called spectrogram, which shows how spectrum of frequencies change over time. Normally we use logarithmic scale on y-axis so the both high and low amplitudes are more clearly visible.

Studies have shown that humans don't perceive frequencies on a linear scale. We are better at detecting difference between lower frequencies than higher ones. So the authors of [17] presented mel scale which fits better with human perception. With this we can generate mel spectrogram of a signal.

The last image representation of an audio signal is done by computing MFCC which we mentioned before. Here 20 filter banks is used to discretized every frame of STFT. Comparisons between before mentioned techniques are seen on Figure 5. For our problem we extracted this images for every song, resized them to 224×224 and normalized them so they can be used with CNN models.

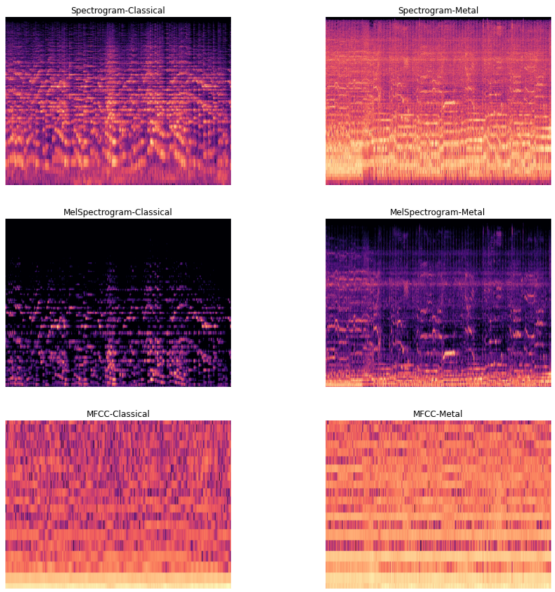

Fig. 5. Spectrogram, mel spectrogram and mfcc for different genres

*1) **Simple CNN:*** We present a CNN architecture as seen in Table II, which was used to train and test models with different types of inputs (spectrograms, mel spectrograms and MFCCs). It consists of 5 convolutional blocks that help extract rich features from inputs. Each one has a 2D convolutional layer, activation layer with ReLU activation function, max pooling layer to condense information to smaller space and lastly a dropout layer to help with regularization. Then the output is flattened and connected to two densely connected layers with dropout layers in between. Last dense layers performs softmax activation and the output of the network are probabilities for each class. During training sparse categorical cross-entropy is used as a loss function and Adam optimizer with 0.001 learning rate.

Some research suggests that using an SVM model on a feature vector that we get from CNN can be beneficial for overall accuracy. For this we first train our simple CNN model on our data. Then we use this model for predictions on our entire train and validation set. This predictions are outputs of the flatten layer, so one one dimensional feature vector for every input. We use this new representation to train an SVM model and then test it on the test dataset.

TABLE II
SIMPLE CNN ARHITECTURE

Layer type	Properties
Conv2D	Number of filters: 16,32,64,128,256 Filter size: (3,3), Stride: (1,1)
Activation	Activation function: ReLU
MaxPooling2D	Pool size: (2,2), Stride:(1,1)
Dropout	Rate: 0.2
Flatten	/
Dropout	Rate: 0.3
Dense	Output size: 128, Activation: ReLU
Dropout	Rate: 0.3
Dense	Output size: 10, Activation: Softmax

*2) **Transfer learning:*** We also test how good transfer learning is on such data. This is done by using an already learned model from which we remove last classification layer. We substitute this with a new classification head with an architecture as seen on Table III.

TABLE III
CLASSIFICATION HEAD FOR TRANSFER LEARNING

Layer type	Properties
Flatten	/
Dropout	Rate: 0.3
Dense	Output size: 50, Activation: ReLU
Dropout	Rate: 0.3
Dense	Output size: 20, Activation: ReLU
Dropout	Rate: 0.3
Dense	Output size: 10, Activation: Softmax

Four different pre-trained architectures are used. First we use VGG-16 [18] which is simillar to our simple CNN model. Then we try out some residual neural networks. Unlike regular ones which have only connections from one layer to the next one, these also have skip connections. With such architecture, the network avoids the problem with vanishing gradients and allows more layers. Here we use ResNetV2 [19] with 50 and 101 layers and DenseNet121 [20] which has even more skip connections. All four models were pretrained on a large image database ImageNet [21]. During our training we freeze these layers and train only the classification head.

Since ImageNet database doesn't include images from spectrograms or something similar, we can expect some poor performances with pre-trained weights. This is why we fine-tune our model as well. For this we chose ResNetV2-101 version. This was done by our intuition because a deep model can usually learn more complex features and maybe improve our results the most. First we train the model as described before and then, after 50 epochs, we unfreeze all weights and additionally train the model at a lower learning rate. We use early stopping for this part which means that the model is trained until validation loss starts to go higher (over 5 consecutive epochs).

IV. RESULTS

In this section we present different metrics that are are used for model evaluation and results of different models and techniques mentioned beforehand.

To evaluate how good each model is we use 4 metrics:

- Classification accuracy: This is calculated by
$$\frac{TP + TN}{TP + TN + FP + FN}$$
- Precision: Calculated by
$$\frac{TP}{TP + FP}$$
and since our test set is balanced, we calculate it per class and average these values for final score
- Recall: Calculated by
$$\frac{TP}{TP + FN}$$
with same process as precision
- F1-score: Calculated by
$$\frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$
with same process as precision and recall

where TP stands for true positive, TN for true negative, FP for false positive and FN for false negative examples.

In Table IV we can see results of traditional methods on GTZAN dataset. For our baseline model we chose a majority classifier, which classified everything into one class. SVM model did the best out of all of them in every metric. After that multilayer perceptron and logistical regression model did similarly. Looks like mapping values into Gaussian distribution helped logistical regression model a lot. The worst was k-NN model, which was expected. Even though values were normalized, Euclidean distance doesn't always work that good in such a multidimensional space.

TABLE IV
RESULTS FOR TRADITIONAL METHODS

Model name	Accuracy	Precision	Recall	F1-score
Baseline	0.1	0.1	0.1	0.0182
k-NN	0.68	0.6932	0.68	0.6789
Random Forest	0.732	0.7331	0.732	0.7314
Logistical Regression	0.756	0.7558	0.756	0.7517
SVM	**0.784**	**0.7892**	**0.784**	**0.7811**
Gradient Boosting	0.74	0.7466	0.74	0.7374
Multilayer Perceptron	0.752	0.7599	0.752	0.7479

Table V shows results of our simple CNN model using different inputs: spectrogram, mel spectrogram and MFCC. All the models were trained on 100 epochs and batch size of 16. Spectrogram representation worked the best with F1-score of 0.73. The worst was MFCC which was expected since it wasn't often used in previous research on this topic. In comparison with traditional methods, the results are worse. One of the reasons for this could be relatively small amount of training data (around 650 images).

TABLE V
RESULTS FOR SIMPLE CNN MODEL

Model name	Accuracy	Precision	Recall	F1-score
CNN-spectrogram	**0.724**	**0.7528**	**0.724**	**0.7267**
CNN-melSpectrogram	0.608	0.6374	0.608	0.6123
CNN-MFCC	0.544	0.5418	0.544	0.5383

To test this hypothesis we split clips into 3 second intervals and converted them to spectrograms and mel spectrograms. We got 10 times the amount of data and results presented in Table VI show improvement. Models were now trained for only 50 epochs. Additionally we tested how classification improves when predicting final class for whole song by classifying each 3 second clip individually and using majority voting for final classification. Results are also seen in Table VI where model name ends with "MV". Compared to initial results on whole songs we can see that the accuracy improves by 29.3%

TABLE VI
RESULTS WHEN USING MORE DATA (3 SECOND CLIPS)

Model name	Accuracy	Precision	Recall	F1-score
Spectrogram-3s	0.841	0.8688	0.841	0.8464
MelSpectrogram-3s	**0.8698**	**0.8701**	**0.8699**	**0.8691**
Spectrogram-3s-MV	**0.936**	**0.9463**	**0.936**	**0.9377**

We also tested the effect of having an SVM model as final classifier on features from CNN. The results are presented in Table VII. We used CNN-spectrogram model for this. As we can see this method did not work for our example since the results are comparable to the base model.

TABLE VII
RESULTS FOR CNN+SVM MODEL COMPARED TO ONLY CNN MODEL

Model name	Accuracy	Precision	Recall	F1-score
CNN+SVM	**0.732**	**0.7333**	**0.732**	**0.7276**
CNN	0.724	0.7528	0.724	0.7267

Table VIII shows results of transfer learning models using both spectrograms ("S" and the end of model name) and mel spectrograms ("MS" and the end of model name) inputs. All of them were trained only on 50 epochs because of higher computational complexity and thus slower training process. We can see that the results are not as good. The main reason for this are pre-trained weights. When only training the classification head, the model can't learn as much because the features that are extracted from convolutional layers are't good for our problem domain.

As mentioned before, we then tested how classification improves when using fine-tuning. The results are shown in Table IX. Here we used already trained ResNet101V2 architecture on 50 epochs with spectrogram images as input. After that fine tuned the model for 25 epochs, until validation loss stopped going down. As we can see both accuracy and F1 score increase for 10% and 16% respectively. But even with that, the results are still a bit lower than the ones we get from our CNN model.

TABLE VIII
RESULTS FOR TRANSFER LEARNING

Model name	Accuracy	Precision	Recall	F1-score
VGG16-S	0.6	0.5754	0.6	0.5689
ResNet50v2-S	0.58	0.6002	0.58	0.5703
ResNet101V2-S	**0.628**	**0.6283**	**0.628**	**0.6098**
DenseNet121-S	0.552	0.5502	0.552	0.5173
VGG16-MS	0.588	0.5931	0.588	0.5686
ResNet50v2-MS	0.532	0.541	0.532	0.5107
ResNet101V2-MS	0.516	0.5159	0.516	0.504
DenseNet121-MS	**0.592**	**0.6297**	**0.592**	**0.6001**

TABLE IX
RESULTS AFTER FINE-TUNING

Model name	Accuracy	Precision	Recall	F1-score
ResNet101V2	0.628	0.6283	0.628	0.6098
ResNet101V2-FineTune	**0.692**	**0.7514**	**0.692**	**0.7068**

On Figure 6 we can see confusion matrices of best model from traditional methods, SVM, and from best CNN best model, CNN-spectrogram. Here we excluded our overall best model, which took 3 seconds clips and did majority voting, since it can't be directly compared to SVM one. This analysis somewhat confirms our initial idea from 2D PCA projections[1], where we said that classical music might not be so hard to distinguish from other ones. Here both models did the least mistakes. For SVM rock and disco were the most problematic, while CNN model had the most trouble with blues and country.

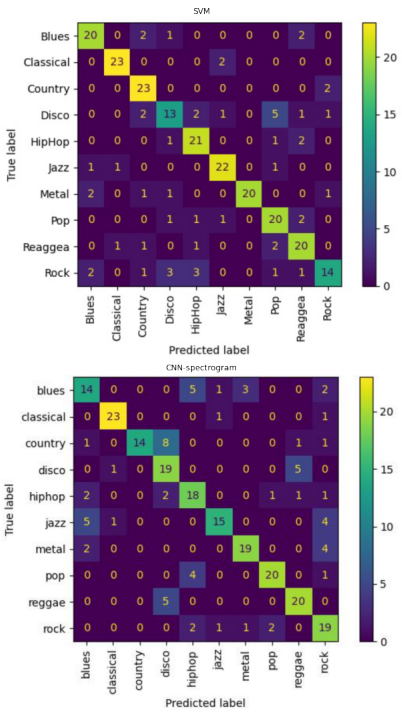

Fig. 6. Confusion matrices of two best models

V. Conclusion

In this work we have proposed many different approaches for music genre classification. We used GTZAN dataset to train and test our models. We presented features that can be extracted from audio signal and a way to represent them in a useful way for our models. From our results we see that an SVM model works best with this features and gets classification accuracy of 78.4%. We tested different inputs for our CNN models and found out that spectrograms work the best. Transfer learning methods by themselves didn't do as well, but we saw improvement when we did fine-tunning of the weights. The best CNN (and overall) model was the one that uses 3s clips during training and predicts final song class by majority voting. This approach gives 93.6% classification accuracy.

In the future we should focus on usage of deep learning models since they show a lot of promise. For this we need larger amount of data. We could also perform data augmentation steps to make models more robust. We should test the effects of random changes of songs volume, pitch shifts or addition of noise in the spectrogram images.

References

[1] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[2] C. Xu, N. Maddage, X. Shao, F. Cao, and Q. Tian, "Musical genre classification using support vector machines," vol. 5, 05 2003, pp. V – 429.

[3] Y. M. G. Costa, L. S. Oliveira, A. L. Koerich, and F. Gouyon, "Comparing textural features for music genre classification," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 1–6.

[4] C. Silla, A. Koerich, and C. Kaestner, "The latin music database." 01 2008, pp. 451–456.

[5] L. Nanni, Y. Costa, A. Lumini, M. Kim, and S. Baek, "Combining visual and acoustic features for music genre classification," *Expert Systems with Applications*, vol. 45, 10 2015.

[6] W. Zhang, W. Lei, X. Xu, and X. Xing, "Improved music genre classification with convolutional neural networks." in *Interspeech*, 2016, pp. 3304–3308.

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

[8] A. Elbir and N. Aydin, "Music genre classification and music recommendation by using deep learning," *Electronics Letters*, vol. 56, no. 12, pp. 627–629, 2020. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/el.2019.4202

[9] S. Oramas, F. Barbieri, O. Nieto Caballero, and X. Serra, "Multimodal deep learning for music genre classification," *Transactions of the International Society for Music Information Retrieval. 2018; 1 (1): 4-21.*, 2018.

[10] N. Ndou, R. Ajoodha, and A. Jadhav, "Music genre classification: A review of deep-learning and traditional machine-learning approaches," in *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, 2021, pp. 1–6.

[11] A. Elbir, H. Bilal Çam, M. Emre Iyican, B. Öztürk, and N. Aydin, "Music genre classification and recommendation by using machine learning techniques," in *2018 Innovations in Intelligent Systems and Applications Conference (ASYU)*, 2018, pp. 1–5.

[12] D. S. Lau and R. Ajoodha, "Music genre classification: A comparative study between deep learning and traditional machine learning approaches," in *Proceedings of Sixth International Congress on Information and Communication Technology*, X.-S. Yang, S. Sherratt, N. Dey, and A. Joshi, Eds. Singapore: Springer Singapore, 2022, pp. 239–247.

[13] H. Bahuleyan, "Music genre classification using machine learning techniques," *arXiv preprint arXiv:1804.01149*, 2018.

[14] A. Elbir, H. O. Ilhan, G. Serbes, and N. Aydin, "Short time fourier transform based music genre classification," in *2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT)*. IEEE, Apr. 2018. [Online]. Available: https://doi.org/10.1109/ebbt.2018.8391437

[15] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[16] I. Kononenko, M. Robnik-Sikonja, and U. Pompe, "Relieff for estimation and discretization of attributes in classification, regression, and ilp problems," *Artificial intelligence: methodology, systems, applications*, pp. 31–40, 1996.

[17] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, Jan. 1937. [Online]. Available: https://doi.org/10.1121/1.1915893

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," 2016.

[20] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2018.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.