QAC239 Final Paper

Dominik Dadak and Yehor Mishchyriak

12.12.2024

Music Genre Classification from Album Covers

Introduction

In the modern age of music consumption, recommendation algorithms are extremely important for marketing new music to new listeners. In order for listeners to be satisfied with their experience, services like Spotify, Apple Music, and others need to get the right music in front of the right audience, and with around 100,000 songs being uploaded to their servers every day[1], manual labeling of each artist, song, and album's genre is an impossible task. Previous work in this field has dealt exclusively with utilizing the audio features of a song in order to classify its genre, with researchers training models reaching accuracies upwards of 90 percent. [2] While the results from past work are impressive, we believe that the results can be further improved by utilizing image classifiers to classify song and album genres from their associated album covers. First, we created a relatively simple model as a baseline for comparison. Due to being constrained by the amount of data, we decided to focus on the 3-class categorisation problem, that is, distinguishing between rap, country, and pop. The baseline model achieved a sub-par accuracy of 50.57 percent. Then, we used ResNet50, a pre-trained convolutional neural network (CNN), and were able to achieve an accuracy of around 74 percent, marking a significant improvement. We believe that this research, used in tandem with existing music classifiers, will provide an extra tool for music platforms to recommend music to their users in a more efficient fashion.

[1] Ingham (2022)
[2] Wijaya (2024)

Data and Methods

The dataset that we used was the MusicOSet, a dataset containing an abundance of metadata on artists, albums, songs, and more.[3] While this dataset contained plenty of metadata about the albums that we wanted to use as training data, we still had to develop a crawler that downloaded the album images from Spotify's servers. Additionally, we ran into some hiccups when trying to label the album images. Firstly, the dataset we used seemed to be entirely based off of the Spotify API. This normally wouldn't be an issue, but this made deciding what label to assign to each album more challenging, as the Spotify API does not provide genre labels for songs or albums; only artists are given this information. Thus, it was necessary to compromise and, for each artist credited to an album, we would assign the album with the genres associated with the artist. The artist data would be merged with the album's index, and the downloaded album cover would be assigned the same index as a name, thus linking the entry in our dataframe of labels with the appropriate image file.

It is also important to note that we encountered a problem with the formatting in one of the cells of the provided dataset. This caused some of the data to become mis-labeled, and thus was ruinous to our model's accuracy. We identified and eliminated the error, and the file that the included scripts download is the fixed version. We also provide a way to fix the error independently in our data collection script.

After gathering and analyzing the labeled data, we noticed that the genres provided by the dataset were too granular to start training a model with. Of the over 1,500 genres, many only appeared a handful of times, which would not be enough data points to train a CNN. Thus, it

[3] O. Silva (2019)

became necessary to create a system of generalizing and simplifying the dataset. This was accomplished by manually creating mappings from niche and specific genres–e.g. “drill” or “classic texas country”–to their more general counterpart, i.e. “rap” and “country” respectively. Clearly, this comes at the cost of the model’s granularity–we would no longer be able to distinguish between “hard rock” and “classic rock”, for example. In the interest of time, however, this was a necessity. Finally, integer values were assigned in place of the genre names, making the dataset ready for training.

To create a baseline model, we opted for a Random Forest classifier due to its ability to handle multi-class classification problems. We then chose several features to train the model on, including color histograms, local binary patterns, and Hu moments. We then created a train/test split, consisting of 80% training data. This resulted in an accuracy score of 50.57%, which we sought to improve upon (Figure 1).

As was mentioned earlier, the foundation for our main model was ResNet50. We opted for a pre-trained model to use generalized visual features learned from large datasets like ImageNet. This allowed us to reduce the computational cost and risk of overfitting given the limited size of our dataset. We unfroze all of the network’s layers for further fine-tuning on our dataset, and replaced the fully connected and output layers. We experimented with hyperparameters as well as different cost functions, optimizers, and various regularisation techniques like dropout, label smoothing, and class-specific data augmentation. These choices produced different results, which we will outline further on. We will also talk about the best combination of hyperparameters and regularisation techniques that we chose, and the performance of the corresponding model on 3,4,5-class categorisation problems.

As a cost function, we chose Cross-entropy, as it is ideal for multi-class classification tasks and measures the divergence between the predicted probabilities and true labels. The function also allows adjustments to the weight parameters associated with individual classes to combat dataset imbalances or assign greater importance to frequently misclassified categories.

For an optimizer, we chose AdamW, as it combines the advantages of Adam, such as adaptive learning rates and efficient gradient updates, with weight decay to prevent overfitting, making it well-suited for fine-tuning tasks.

We chose the initial learning rate to be 0.0001, which is small but appropriate for fine-tuning to avoid drastically overwriting the pretrained weights (note, we tested for a learning rate of 0.001, but it led to rapid overfitting). Also, we selected CosineAnnealingLR as a scheduler to allow for finer adjustments as the training progresses by reducing the learning rate, which is especially important for improving convergence in the later epochs.

The actual training process involved a maximum of 50 epochs, with a patience parameter for early stopping being 5. During training, the model is set to training mode to enable dropout and batch normalization updates, with losses computed and backpropagated to update the weights. Validation runs in evaluation mode, disabling gradient computations to improve efficiency, and tracks the model's performance using validation loss and classification metrics. The best-performing model is saved whenever validation loss improves.

We were testing models with different choices of training parameters on the 5-class problem and got the following results:

Note: Each model was given a name for further graphical depiction of performance for comparisons

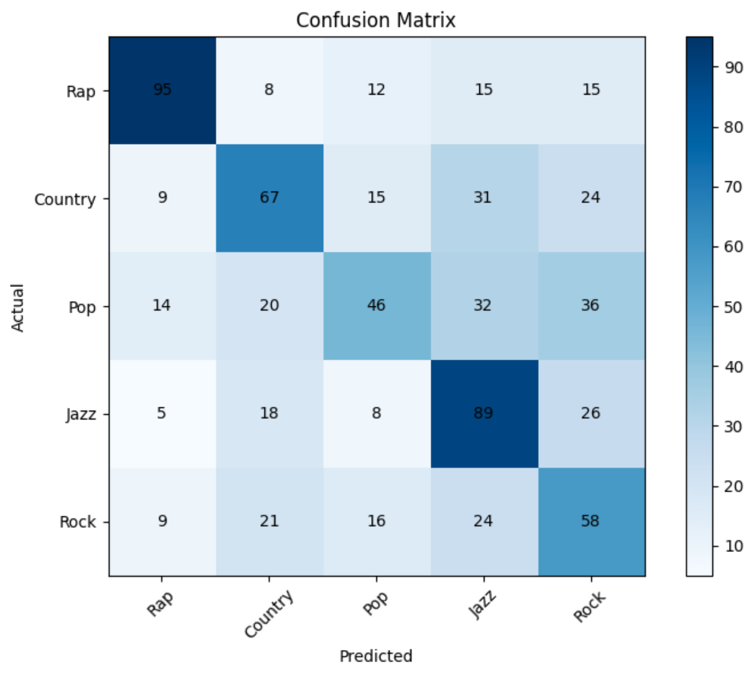
M1:

[No data augmentation, no class weighting, dropout for the fully connected layer = 0.5,

Cross-entropy loss, AdamW optimizer, CosineAnnealingLR scheduler]

Classification Report

	precision	recall	f1-score	support
Rap	0.72	0.66	0.69	145
Country	0.50	0.46	0.48	146
Pop	0.47	0.31	0.38	148
Jazz	0.47	0.61	0.53	146
Rock	0.36	0.45	0.40	128
accuracy			0.50	713
macro avg	0.50	0.50	0.49	713
weighted avg	0.51	0.50	0.50	713

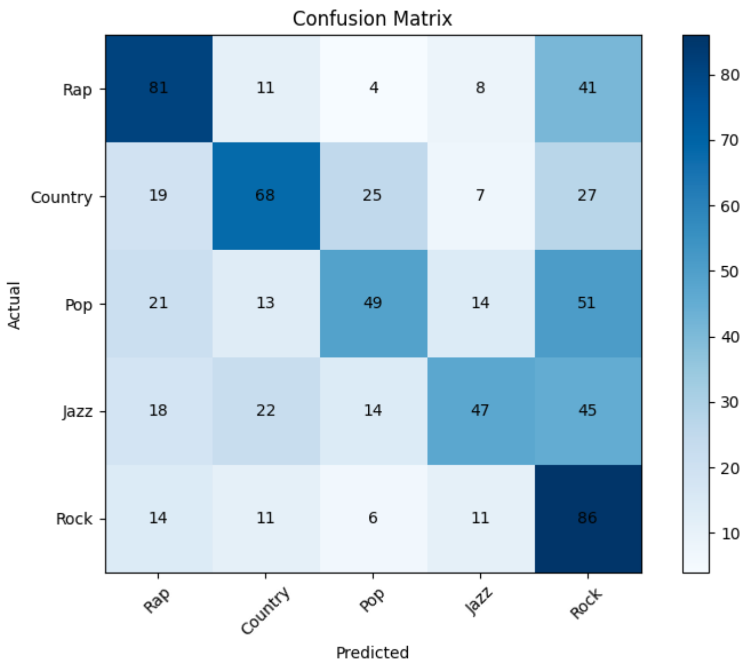


***M2:**

[No data augmentation, no class weighting, dropout for the fully connected layer = 0.5, Cross-entropy loss, SGD optimizer with .9 of momentum and 1e-4 weight decay, CosineAnnealingLR scheduler]

Classification Report

	precision	recall	f1-score	support
Rap	0.53	0.56	0.54	145
Country	0.54	0.47	0.50	146
Pop	0.50	0.33	0.40	148
Jazz	0.54	0.32	0.40	146
Rock	0.34	0.67	0.46	128
accuracy			0.46	713
macro avg	0.49	0.47	0.46	713
weighted avg	0.50	0.46	0.46	713



** M2 configuration turned out to be the least prone to overfitting! This was the only model that finished the entirety of the 50 epochs. We discovered this setup pretty late, so we were not able to train it over more epochs, but this is something we will do very soon and share our results. Hopefully, since the validation loss was dropping very steadily over the whole 50 epochs means that it will generalize better than the other models if run over, say, 1000 epochs. We got our hands on an Nvidia RTX 3090, so we estimate that training will take about 8 hours assuming no early stopping. We will report our results to you as soon as they are available!*[4]

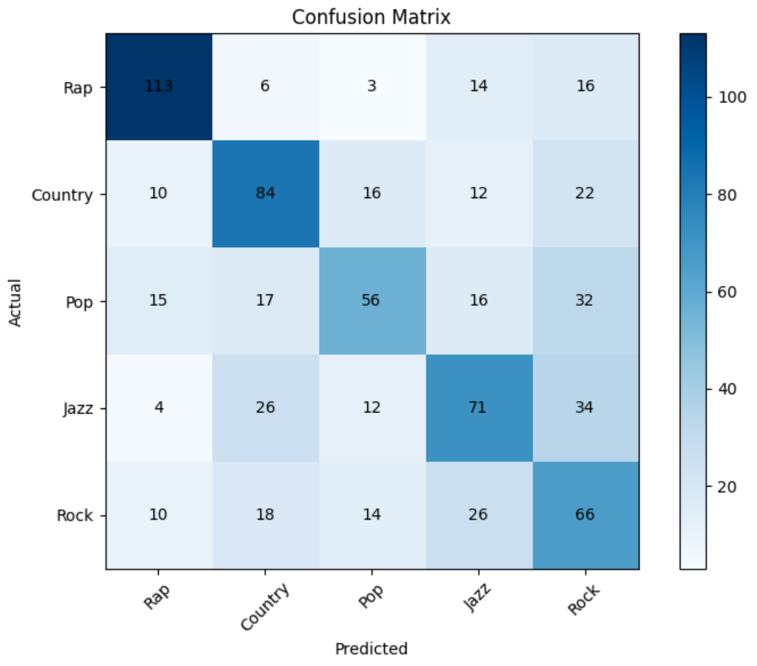
[4] Training lasted for 107 epochs before early stopping was triggered. The accuracy during validation was ~51% and ~48% during testing for the 5-class problem. The issue of rock being frequently confused with other classes got notably mitigated, although did not fully disappear.

M3:

[No data augmentation, no class weighting, dropout for the fully connected layer = 0.5, LabelSmoothing loss with smoothing = 0.2, AdamW optimizer, CosineAnnealingLR scheduler]

Classification Report

	precision	recall	f1-score	support
Rap	0.74	0.74	0.74	152
Country	0.56	0.58	0.57	144
Pop	0.55	0.41	0.47	136
Jazz	0.51	0.48	0.50	147
Rock	0.39	0.49	0.43	134
accuracy			0.55	713
macro avg	0.55	0.54	0.54	713
weighted avg	0.55	0.55	0.55	713



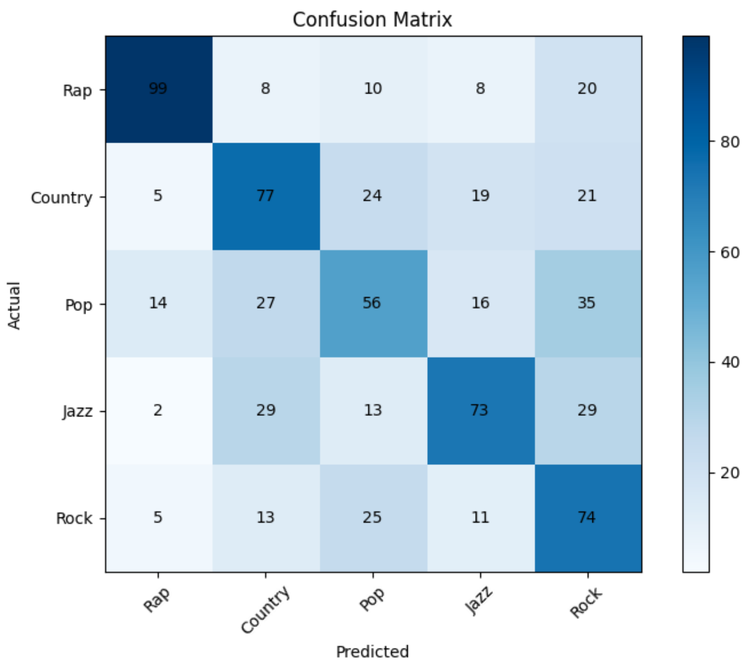
Actual \ Predicted	Rap	Country	Pop	Jazz	Rock
Rap	113	6	3	14	16
Country	10	84	16	12	22
Pop	15	17	56	16	32
Jazz	4	26	12	71	34
Rock	10	18	14	26	66

M4:

[No data augmentation, class weighting inversely proportional to precision, dropout for the fully connected layer = 0.5, LabelSmoothing loss with smoothing = 0.2, AdamW optimizer, CosineAnnealingLR scheduler]

Classification Report

	precision	recall	f1-score	support
Rap	0.79	0.68	0.73	145
Country	0.50	0.53	0.51	146
Pop	0.44	0.38	0.41	148
Jazz	0.57	0.50	0.53	146
Rock	0.41	0.58	0.48	128
accuracy			0.53	713
macro avg	0.54	0.53	0.53	713
weighted avg	0.55	0.53	0.53	713

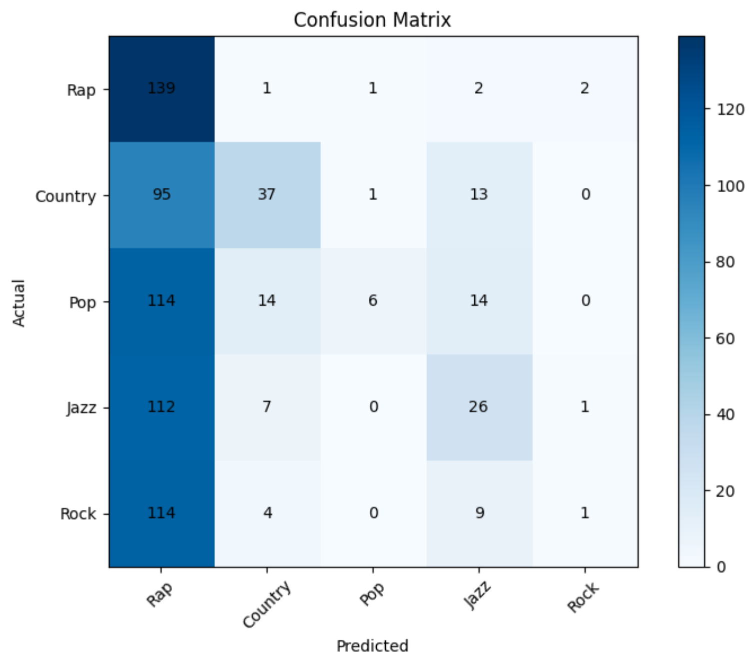


M5:

[Class-specific data augmentation, class weighting inversely proportional to precision, dropout for the fully connected layer = 0.5, LabelSmoothing loss with smoothing = 0.2, AdamW optimizer, CosineAnnealingLR scheduler]*

Classification Report

	precision	recall	f1-score	support
Rap	0.24	0.96	0.39	145
Country	0.59	0.25	0.35	146
Pop	0.75	0.04	0.08	148
Jazz	0.41	0.18	0.25	146
Rock	0.25	0.01	0.02	128
accuracy			0.29	713
macro avg	0.45	0.29	0.22	713
weighted avg	0.45	0.29	0.22	713



*Class-Specific Transforms (for more detail, see the actual code):

Pop: Focused on **vibrancy** and **geometric patterns**.

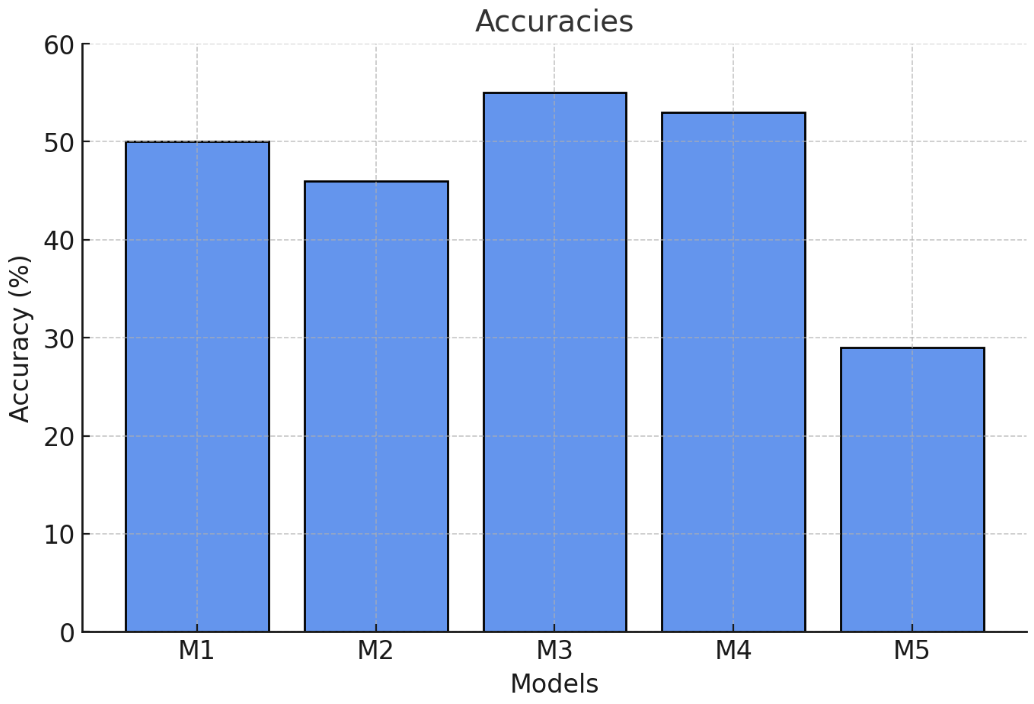
Rock: Highlighted **gritty, high-contrast** aesthetics.

Country: Brought out **rustic, natural tones**.

Jazz: Emphasized **smooth textures** and **abstract designs**.

Comments: M5 failed as the transforms caused most samples to be classified as "rap." Class-specific transformations may still be effective if based on the neural network's extracted features rather than prior knowledge of cover styles. GradCAM would help identify these features—more on this later.

All in all, the best combination of training parameters is M3:

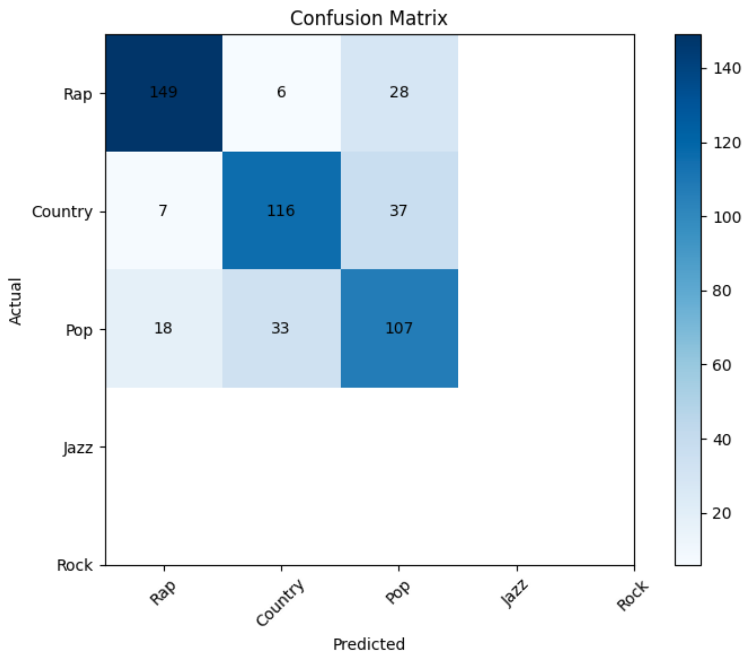


Now, we are going to test the model's performance on 3,4-class problems. An obvious prediction is that it will perform much better due to less room for misclassification owing to less feature overlap between the classes in question.

3-class problem:

Classification Report:

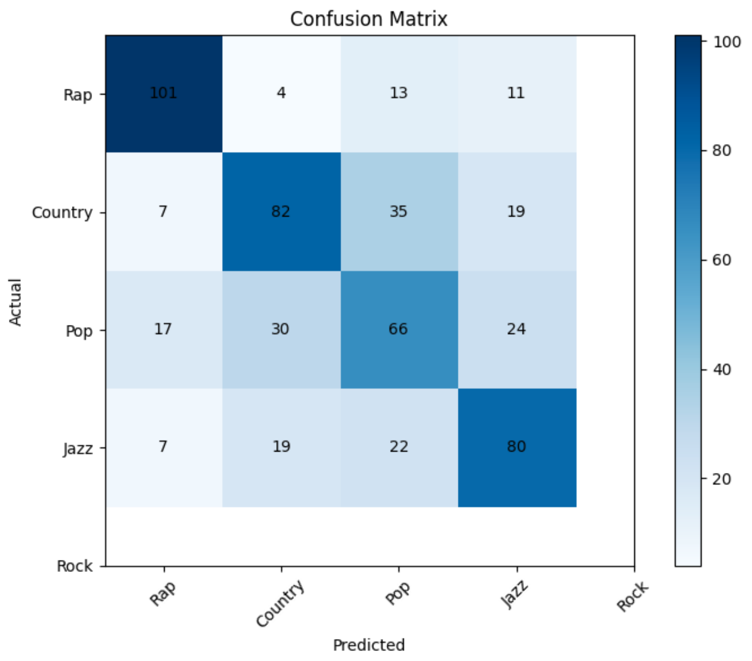
	precision	recall	f1-score	support
Rap	0.86	0.81	0.83	183
Country	0.75	0.72	0.74	160
Pop	0.62	0.68	0.65	158
accuracy			0.74	501
macro avg	0.74	0.74	0.74	501
weighted avg	0.75	0.74	0.74	501



4-class problem:

Classification Report:

	precision	recall	f1-score	support
Rap	0.77	0.78	0.77	129
Country	0.61	0.57	0.59	143
Pop	0.49	0.48	0.48	137
Jazz	0.60	0.62	0.61	128
accuracy			0.61	537
macro avg	0.61	0.62	0.61	537
weighted avg	0.61	0.61	0.61	537



Results

The M3 model demonstrates

74% accuracy for 3 categories,

61% accuracy for 4 categories, and

55% accuracy for 5 categories.

Analyzing the album covers visually as well as based on the confusion matrices, we can state the following:

- Feature Overlaps:
    - Pop, Country, and Jazz:
        - These genres overlap significantly due to shared visual styles like warm tones, abstract designs, and natural aesthetics.
        - This results in confusion across these classes.
    - Jazz and Rock:
        - Both genres share abstract and dark design elements, leading to frequent misclassification.
- Distinct Features:
    - Rap stands out with fewer misclassifications, indicating more distinct visual features like bold typography, urban imagery, and high-contrast designs.
    - Pop struggles with feature distinction, as its vibrant and colorful designs are not uniquely distinguishable from other genres.

We can see the accuracy declining as we increase the number of classes. Why is that? What is it that we are lacking? The answer is: data. As stated above, certain classes possess substantial feature overlaps. That said, classifying them is not impossible, however we need significantly more data for that. Additionally, despite our unsuccessful attempt of employing class-specific data augmentation, it has a great potential to improve the accuracy of predictions. We believe the approach should be as follows: apply feature extraction and Grad-CAM[5] for their visualisation to see what features the model relies most on in classification. Then, come up with a set of class-specific transformations to eliminate the feature overlaps present across the classes. In addition, a multi-class categorisation problem (especially with numerous classes) can greatly benefit from ensemble learning. One may train several “narrow” models and then take a weighted average of their outputs for making a prediction.

Conclusion

This research project set out to show the practical application of using album covers in classifying music genres. We accomplished this by training a convolutional neural network using transfer learning. We also drew conclusions about similarities and differences in album covers’ styles across the presented genres. Overall, given tight constraints in time and resources, we achieved good results. Those, of course, can be greatly improved following the steps outlined in the preceding section. After these steps are implemented, incorporating our model in tandem with a model based off of audio features would be an interesting path to pursue. For example, combining a model using Mel-frequency cepstral coefficients with our own could potentially increase the accuracy of both models[6].

[5] https://github.com/jacobgil/pytorch-grad-cam
[6] Wijaya (2024)

Please, find the code on GitHub following this link

LINK

Appendix

- Ingham, T. (2022, October 7). It's happened: 100,000 tracks are now being uploaded to streaming services like Spotify each day. *Music Business Worldwide*. https://www.musicbusinessworldwide.com/its-happened-100000-tracks-are-now-being-uploaded/. Accessed Dec. 12, 2024.
- O. Silva, M., M. Rocha, L., M. Moro, M., & Universidade Federal de Minas Gerais. (2019). *MusicOSet* [Dataset]. https://marianaossilva.github.io/DSW2019/
- Wijaya, N. N., Setiadi, D. R. I. M., & Muslikh, A. R. (2024). Music-Genre Classification using Bidirectional Long Short-Term Memory and Mel-Frequency Cepstral Coefficients. *Journal of Computing Theories and Applications*, *1*(3), 243–256. https://doi.org/10.62411/jcta.9655