To what extent can deep learning be used to classify classical music into its distinct subgenres?

Word Count: 4360

Introduction

Music genre classification has played a vital role in the evolution of the modern music industry. By classifying music into distinct genres radio stations, record companies, and streaming services alike are able to pander to listener preferences by providing the freedom of choice. Nowadays, streaming services have largely become the go-to music service for many listeners due in large part due to the great freedom of choice and 'virtually limitless' music they offer their listeners (Morris et al., 2015). The problem that now arises is labeling that music, so that listener's preferences can still be pandered to while exposing them to unheard music. This is especially important for services like Spotify, as their ability to curate quality, custom playlists for individual users is crucial to keeping their customer base engaged --and keep them earning revenue (Prey, 2020). Spotify now has a market cap of roughly 42 billion dollars, indicating the value they provide users, and giving a general indication of a large market for good automatic musical classification systems (Bloomberg, 2021).

Before widespread internet availability, musical classification was primarily performed by humans, which was widely accepted as the volume of music was relatively low, however, with the rise of the internet and an increasing amount of music being produced, researchers began to lead studies regarding the automatic classification of music using computer systems (Pachet et al., 2000). Since the widespread adoption of the internet, much progress and has been made towards increasing the performance of musical genre classification of the ten broad categories as defined in the GTZAN dataset, a popular dataset compiled by the authors of the 2002 paper *Musical Genre Classification of Audio Signals,* and many systems are now capable of outperforming their human counterparts (Sturm, 2012). Since the original 2002 paper, many new machine learning and deep learning models have been developed and applied to the GTZAN dataset, however, studies involving research into applying these models to sort classical music into its distinct subgenres are either lacking or currently being investigated by very few researchers.

Literature Review

The evolution of automatic music classification begins with the research laid out in the aforementioned 2002 paper authored by George Tzanetakis and Perry Cook, professors at the University of Victoria and Princeton respectively, which notably both established the GTZAN music genre dataset and, proved that automatic music classification using computer systems and advanced algorithms could meet human standards (Tzanetakis et al., 2002). As explained by Bob Sturm, associate professor at the KTH Royal Institute of Technology in Stockholm, in his paper analyzing the GTZAN dataset, the dataset consists of 1000 musical excerpts, 30 seconds in length, belonging to 10 different genres: blues, classical, country, disco, hip hop, jazz, metal, pop, reggae, and rock. Since its creation, the dataset has largely become the gold standard for automatic music genre classification tasks (Sturm, 2012). Besides the creation of the GTZAN dataset, the other important fact established by the paper was that automatic musical classification systems could meet human standards. For clarification, the term ‘automatic’ in ‘automatic musical classification’ is used because these systems were classifying the music based on it’s actual properties and not metadata such as date published, composer, and name. The use of feature extraction by Tzanetakis and Cook, was the primary reason this automatic classification was made feasible.

Isabelle Guyon, a French researcher at the Université Paris-Saclay, defines feature extraction in her 2008 book *Feature Extraction: Foundations and Applications* simply as a procedure used on data that is either too large to be processed and suspected to contain large amounts of redundant data to reduce the number of features in a dataset by creating new features from existing ones (Guyon et al., 2008). Put more simply, feature extraction allows the researcher to either simplify or condense the data they are using to either produce more accurate results or make the data easier to handle and apply to their algorithm or machine learning model. In this case the researcher would be attempting to simplify the data by quantifying it using mathematical functions based on characteristics of each individual song, turning it from complex audio data, to simplified numerical representations. Tzanetakis and Cook chose to extract a variety of features from each 30 second sound-byte including: timbral texture features, spectral rolloff, spectral flux, time domain zero crossings, and Mel-frequency cepstral coefficients (MFCCs). Each one of these simply represents a mathematically derived representation of a certain musical characteristic. Unfortunately, the mathematical derivations of each one of these features is far beyond the scope of this paper, but it will help to think of them as mathematical representations of unique qualities that pertain to each individual 30 second sound-byte.

However, the key takeaway from this discussion on feature extraction was that it is what enabled Tzanetakis and Cook to apply their algorithms to the musical data and enabled their success, which in turn drove a lot of interest into the field of musical genre classification and natural language processing.

The next big leap forward in musical genre classification came with the reintroduction and development of deep learning systems. While both the computer algorithms that Tzanetakis and Cook used and deep learning fall under the umbrella of machine learning, it's important to note that the two are distinct problem-solving models. According to a paper aptly titled *Deep Learning* by Yann LeCun, chief AI scientist at Facebook, Yoshua Bengio, professor of computer science at the University of Montreal, and Geoffery Hinton, professor of computer science at the University of Toronto, deep learning involves the construction of multilayer processing tools such as neural networks. These models are significantly more complicated than prior algorithmic models, however, the payoff for this increased complexity is that these models are able to produce more accurate and precise results on complex tasks, especially those tasks involving video, speech, and audio. The most popular deep learning model --the neural network-- learns by: first attempting to perform a given task such as classification; then determining its loss, loss in this case is a mathematical measure of how far off the network's prediction was from the desired result, and then having the network slightly tweak itself so that it will perform better on the next task (LeCun et al., 2015). On the other hand, the computational algorithms that make up the other half of the machine learning umbrella, are far simpler, faster to run, and easier to experimentally replicate. For example, Tzanketakis and Cook used a k nearest neighbors algorithm in their experiment, which according to Laszlo Kozma, an assistant professor at the Freie University of Berlin, works to classify something by finding its most similar matches (or nearest neighbors), and classifying it according to which group it is most similar-- no actual learning involved (Kozma, 2008). Even still, because of their abilities to handle more complex, and nuanced problems, many researchers began looking to apply deep learning models to their more difficult research tasks.

Huang et al., a PHD student at Stanford University, along with his colleagues identified automatic music genre classification as a complex problem that would be a good candidate to apply a deep-learning model to. Huang et al. decided to use GTZAN dataset as well, in hopes of providing a direct comparison in model performance to the performance achieved by Tzanetakis and Cook's k nearest neighbors algorithm. Huang et al. also followed a very similar feature extraction process to that performed by Tzanetakis and Cook, however they chose to use a mel-spectrogram instead of its similar cousin, the MFCC, that Tzanetakis and Cook used. The main difference came when Huang et al. chose the models they planned to use. While Tzanetakis and Cook had achieved 61% accuracy with their k-nearest neighbors algorithm and another computational algorithm, Huang et al. decided to test a similar k-nearest neighbors algorithm against a support vector machine (a different computational algorithm from those used by Tzanetakis and Cook), a feed forward neural network, and a convolutional neural network (a similar, but more complicated neural network that follows roughly the same learning process). After training the two deep learning models and applying them to the test data, Huang et al. found that the convolutional neural network outperformed the other machine learning models, with the convolutional neural network achieving an 82% percent prediction accuracy compared to a 60% accuracy for the support vector machine, and a 54% accuracy for both the feed forward neural network and k-nearest neighbors algorithm (Huang et al., 2018). This research indicated that convolutional neural networks were the best in tackling problems relating to automatic musical genre classification, however, the extent to which they were able to categorize different subgenres of music was not yet apparent.

Yandre M.G. Costa, a professor at the State University of Maringa in Brazil, along with their colleagues looked to apply a convolutional neural network to even more specific data. In order to do so, Costa decided to use the Latin American Music Database (LMD), which contained 3,227 full-length music pieces of 501 different artists distributed along 10 musical genres. Costa then followed a similar procedure of feature extraction as performed by Tzanekitas and Cook, and Huang et al., before training his convolutional neural network on the training data. Using this process, Costa was able to achieve 92% prediction accuracy for classifications on the Latin American Music Database (Costa, 2018). This would indicate that convolutional neural networks would be capable of classifying music beyond the basic subgenres as defined in the GTZAN dataset.

Although the aforementioned studies indicated the potential of using a convolutional neural network to classify music into genres and even more specific subgenres, there remains a significant research gap in classifying the different subgenres of classical music. According to Chrisof Weiss of the Fraunhofer Institute for Digital Media Technology, classical music can actually be broken up into five specific distinct time periods: Baroque, Classical, Early Romantic, Romantic, and Modernist, with each style containing its own set of unique features. For example, Romantic music can be characterized by dissonant chords and a fully equal use of all chromatic pitches (Weiss et al., 2015). With the exception of Costa's study on the Latin American Music Database, all of the aforementioned studies and the majority of studies outside the scope of this literature review often use the GTZAN dataset, however, the evidence presented in this literature review shows that convolutional neural networks are capable of classifying music into subgenres more complex GTZAN dataset at a high accuracy.

In my experiment, I hypothesized that a convolutional neural network could be used to classify classical music into its distinct subgenres. This led to the essential question of my research: to what extent can deep learning be used to classify classical music subgenres? To test this hypothesis I designed and implemented a convolutional neural network that was trained to classify the different classical music subgenres on the Maestro database. I was then able to test the model and determine its accuracy for classifying each category, possibly leading to a model that produced a better than chance outcome.

Methodology

Gathering Data

The first step in this experiment was gathering data that could be used to train and test my convolutional neural network (CNN). In this case the data I needed to gather were mp3 raw audio files of classical music pieces as well as the classical music period that they were composed in. Initially I attempted to do this myself by hand, using publicly available recordings and labeling the period myself using the composer name, style, and time period within which they were associated with, however, I ran into many problems with this approach. Firstly, it was hard to maintain instrumental consistency across all of the files I had attempted to source: for example, some files featured only piano, while others had only violin. Secondly, I couldn't be assured that all of the music was at the same decibel level or frequency, which was concerning as it posed to interfere with the feature extraction process. And thirdly, the process of finding, sourcing, and labeling all of this data was just too time-consuming of a process to undertake for the scope of this project, so I looked to see if there were any publicly available datasets.

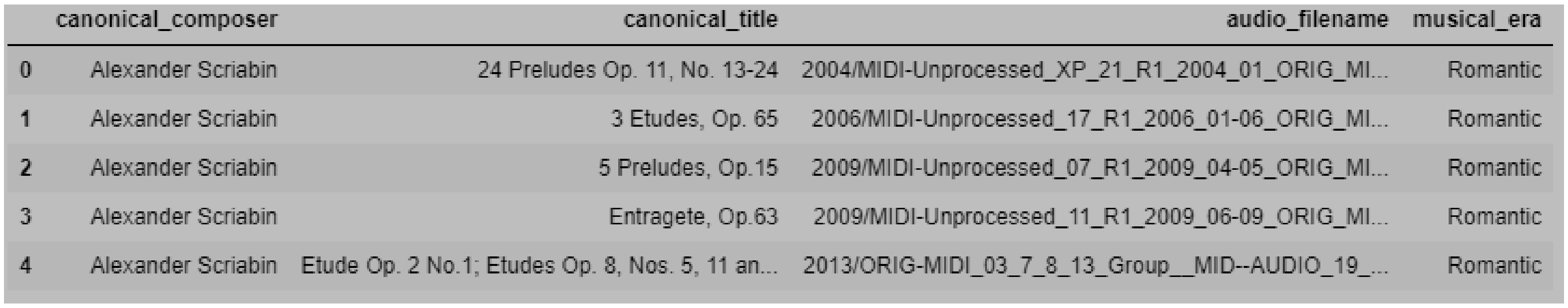
Luckily, the artificial intelligence team at Google that has developed publicly available machine learning tools such as tensorflow before, also run a project called *Magenta*, which focuses on providing resources to help people pursue music related machine learning projects. The Magenta team had developed a high quality classical music dataset called the Maestro dataset, which featured about 200 hours of virtuosic piano performances from the International Piano-e-competition in the form of audio mp3 raw audio files, along with a CSV (comma separated value) file containing information about each file including composer and song name. I decided to download and use this dataset, as I could be assured that each piece would only feature piano, and that all the piano music would be standardized in a way that feature extraction could reliably take place. The added convenience of not having to independently source, download, and label data on my own was also factored into this decision.

Classifying the Data

As mentioned above, while the data came with a column in the CSV file relating each mp4 file to its composer and song name, there was no column or data indicating the classical music period to which the piece belonged. There were two approaches to labeling the data that each had their own pros and cons: assigning the classical music period (Baroque, Classical, Early Romantic, Romantic, Modernist) based on the composer name, or assigning the classical music period based on the song name. For the first approach, the appeal was that it was a much simpler and straightforward process and I was guaranteed to find a time period each composer fit, the drawback was that some composers straddled multiple time periods. For example, during much of his early and mid career Beethoven composed in the classical style, but towards his later career he began to move into a less defined Early Romantic style (Rosen, 1997). On the other hand, labeling pieces by their song name could hopefully avoid the confusion or hard-to-define composers, but was a much more difficult process and would be difficult to automate as there was no solid source that contained musical eras for each of these individual pieces. In the end I used the composer name to assign the musical era. I decided to use Britannica to source the musical era for each individual composer, because it was difficult to find peer-reviewed resources for some of the lesser-known composers.

To illustrate these steps:

1) Add column to CSV titled ‘musical_era’
2) Determine what musical era each individual composer belongs to
3) Write a python function that assigns a musical era label of either Baroque, Classical, Early Romantic, Romantic, or Modernist to each file
4) Run the python function and assign each file the appropriate label
5) Loop and run for each individual file



	canonical_composer	canonical_title	audio_filename	musical_era
0	Alexander Scriabin	24 Preludes Op. 11, No. 13-24	2004/MIDI-Unprocessed_XP_21_R1_2004_01_ORIG_MI...	Romantic
1	Alexander Scriabin	3 Etudes, Op. 65	2006/MIDI-Unprocessed_17_R1_2006_01-06_ORIG_MI...	Romantic
2	Alexander Scriabin	5 Preludes, Op.15	2009/MIDI-Unprocessed_07_R1_2009_04-05_ORIG_MI...	Romantic
3	Alexander Scriabin	Entragete, Op.63	2009/MIDI-Unprocessed_11_R1_2009_06-09_ORIG_MI...	Romantic
4	Alexander Scriabin	Etude Op. 2 No.1; Etudes Op. 8, Nos. 5, 11 an...	2013/ORIG-MIDI_03_7_8_13_Group__MID--AUDIO_19_...	Romantic

The above is a visualization of the first five entries in the final CSV

Feature Extraction with Librosa

I chose to follow a similar procedure for feature extraction as the one Tzanekitis and Cook, and Huang et al. had followed. However, in order to simplify the mathematical derivations of these values, I decided to use a python library called *librosa* (if you have never heard the term 'library' used with programming, you can simply think if it as something that helps me recycle and use code written by other people, making my job much easier). Librosa is a python package for music and audio systems that provides many of the same tools needed to perform feature extraction on the raw audio data that Tzanekiitis and Cook, and Huang et al. used including spectral rolloff, spectral centroid, zero crossing rate, and MFCCs (McFee et al). More specifically, for each audio file I stored in a separate CSV file labeled data.csv the:

- Chromagram
- RMS energy
- Spectral centroid
- Spectral rolloff
- Spectral bandwidth
- Zero crossing rate
- 20 instances of the MFCC
- Musical Era

As calculated by librosa. The calculations and meaning of each of these measurements are beyond the scope of this paper, but they are here for reference. Once again it will help to just think of them as characteristics of the music. Once I had finished the feature extraction, I was ready to move on and use the data to train the actual CNN model once it had been constructed.

Building the Convolutional Neural Network Model

For construction of the actual CNN model, I deferred to Huang et al., and followed their construction parameters/procedure. In order to construct the actual CNN, I used the Keras python library, which is an easy to use python library targeted at users who are interested in creating deep learning models (Keras documentation). Huang et al. used a model with 3 convolution layers, with ReLU activation, softmax output, and cross entropy loss, and I followed their construction parameters and built a similar model in Keras.

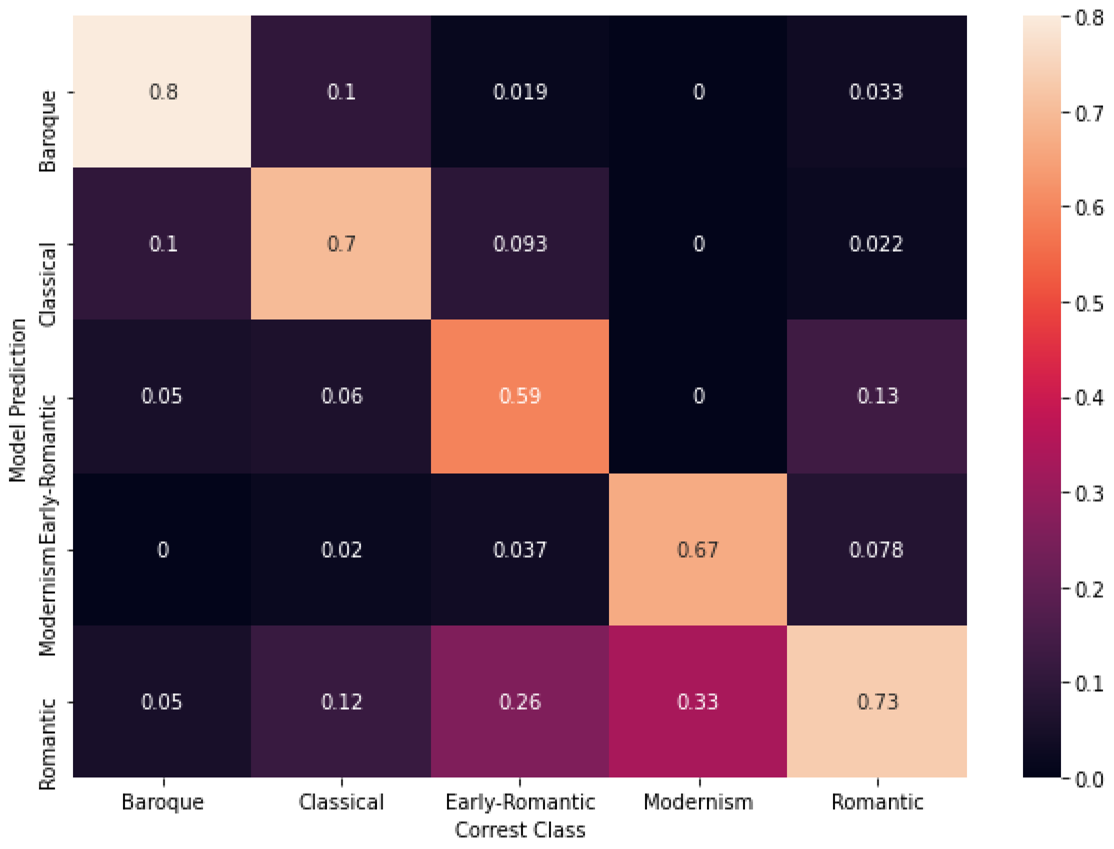
After building the model, I still had to train the model on the feature extraction data that I had collected and stored in my data.csv file from the feature extraction step. However, before I could start training my model, I had to split the data into my training set and my test set. According to Aurelien Geron, author of best-selling book *Hands-On-Machine-Learning with Sci-kit learn and Tensorflow*, this is an important step, because if we allow the model to train on the data that we later test it on, it will perform better than it would on the general population and the results would be an overestimation of the model's abilities (Geron, 2019). To prevent this we split our data into a training set and a test set, it is standard practice according to Geron to use an 80/20 split for this process, and to do so randomly, so as to prevent any bias from the researcher. In my experiment I did the 80/20 split and set aside 20% of the data to be tested on at a later step. With the remaining 80% I split the data into 20 batches, so that the constraints on my computer's hardware would be minimized and trained the data on each individual batch.

When the model had finished training, I used Keras' built-in 'evaluate' function that takes the trained model and the unseen test data, and returns the model's accuracy on the task of classifying the test data.

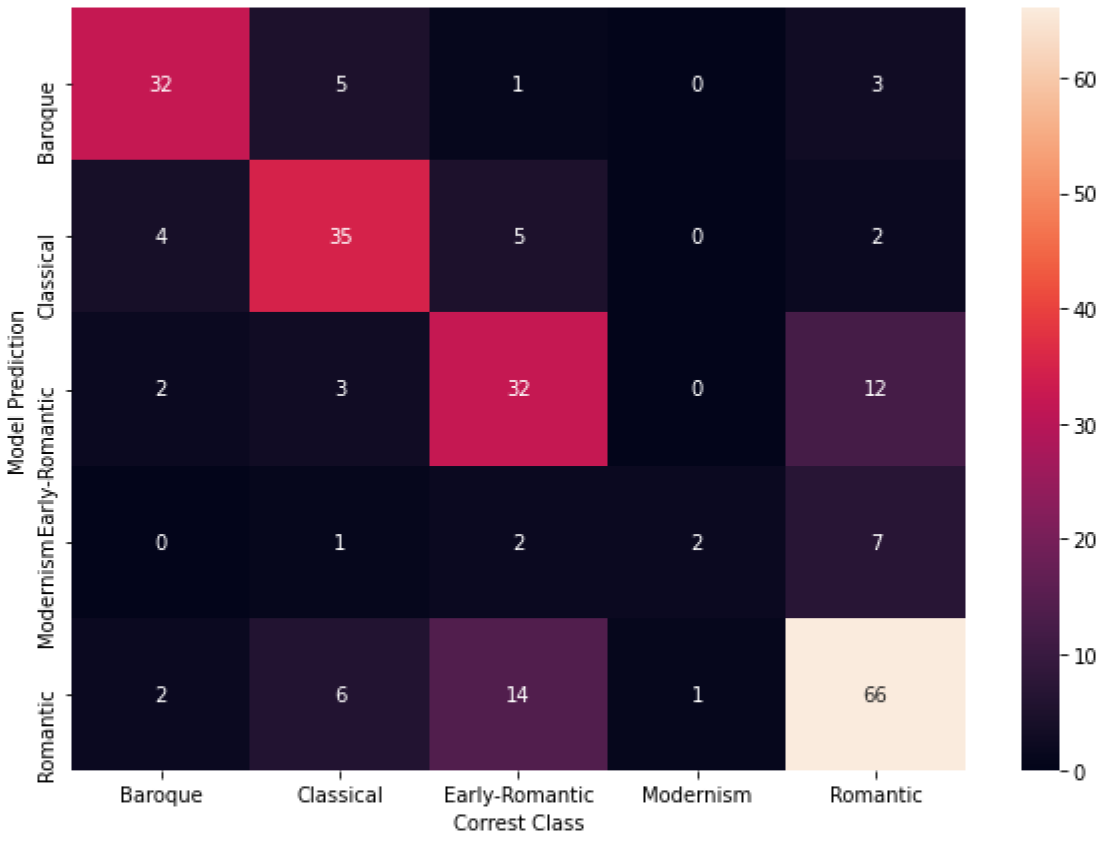
Results

A preliminary examination of the results show that the model's prediction accuracy was daily solid at 70.46% correct predictions on the test data. That said, the model had an accuracy of 91.09% on the last sample of the training data, indicating that the model is overfitting. According to Tom Dietterich, a professor of computer science at Oregon State University, overfitting is a phenomenon where the model attempts to try to fit the training data too well. Overfitting is indicated when the model performs better on the training data than on the test data, and is usually a bad thing as it means the model will not generalize as well to the general population (Dietterich, 1995).

Even still, given that there were five categories, and a random guess would have a 20% chance of being right, 70.46% appears to still be a solid accuracy, however, this single number accuracy score does not tell the full story of the data.



A probability confusion matrix for the data is obtained using the CNN's predictions on the test data. It displays the correct class on the bottom axis and the class predicted by the CNN model on the left-hand side. The values within the confusion matrix correspond with the probability with which they occurred during the testing process. Values along the central diagonal represent the accuracy with which the CNN identified music from that specific genre. From this confusion matrix, we can see that the CNN was the best at classifying music from the Baroque and Romantic time periods with .80 and .73 accuracy respectively. It's also noteworthy that the CNN seemed to struggle most with pieces from the romantic era, as it only had a 0.59 accuracy when dealing with instances from this period. This was anticipated earlier in the paper, as many musical pieces from the Early Romantic period heavily influenced pieces in the Romantic era, so they share many traits and it would be expected that they would be the most difficult to tell apart from each other. However, the reason why the CNN is more accurate at predicting instances of Romantic music than Early Romantic music (0.73 vs. 0.59) is likely due to an imbalance in the number of testing/training/data instances between each class.



A frequency confusion matrix helps display the imbalances in the training/testing/data instances. The CNN shows a bias towards the Romantic era over the Early Romantic era simply because the romantic era had more training data instances. For example: if a dataset has 95% pictures of dogs and 5% pictures of cats and you ask it to predict whether or not a certain animal is a dog, it may always say dog when presented with an image, because it assumes that if it always says dog it should have roughly a 95% chance of being correct. Obviously this model would generalize terribly in real life, and is a great exaggeration of the current situation with my CNN, but it does highlight a concern with the frequency imbalances. We can also note that the model tends to perform better on the classes that have a higher frequency in the dataset. The fact that the modernist only has roughly 30 songs in both the training and test data, also indicates that it could potentially be overwritten by this bias.

Discussion

Implications

The results indicate that CNNs hold the potential to perform automatic music genre classification on classical music using the Maestro dataset, however, all this is with the caveat that further testing be done, and the Maestro dataset be modified to include more even class representation. Intuitively, the results make sense as well, as the most misidentified instances (predicted romantic when it was actually early romantic and predicted early romantic when it was actually romantic) as indicated tend to also be the most similar, as influences from the early romantic are heavily adopted and extended by romantic music and the general tone and attitude of many romantic era pieces.

The results also indicate that large companies such as Spotify should also be able to use CNNs to perform automatic music genre classification of subgenres given their access to even larger music repositories than the one used in this experiment, and their access to even better hardware and more computationally complex software.

Limitations:

Apart from the slightly ambiguous results produced by my experiment, due to the large imbalance between the number of items belonging to each class in the dataset, there are a few limitations worth mentioning.

One limitation of this experiment was that it was trained on the Maestro dataset, which contained only piano notes in a relatively constrained and controlled environment. This means it might not generalize well to either types of classical music that involve multiple instruments or different instruments, such as those performed by quartets or full symphonies.

Another limitation was that the composer name and period to which a certain piece belonged might not have had a perfect or strong correlation for some of the composers. This could result in mislabeling of the training and test data, potentially causing the machine to learn the wrong information, or generating incorrect results. However, because some pieces with obscure song names might not have easily accessed information about their period, I still support the decision to use the composer name.

Another limitation was the imbalance between the number of different instances belonging to each class. This imbalance could result in certain classes essentially being 'weighted' heavier than others, as they make up such a relatively small proportion of the training data that the CNN never sufficiently learns how to classify them. This is probably the greatest limitation in the dataset, as the "Modernist" class only had around 30 total audio files, when the dataset as a whole had over 1,100. Because of this, our results may be an overestimation of what the CNN would actually be capable of in an evenly weighted experiment.

Even with its many drawbacks, I still support my use of the Maestro database, as it greatly simplified this experiment and shows that automatic music genre classification of classical music using a CNN merits further research.

Future studies

In conclusion, automatic music genre classification of classical music using deep learning techniques has shown promise in this experiment and therefore merits further research. It may help to use a slightly less complex deep learning technique in the future to prevent the type of overfitting that occured in this experiment. Alternatively, an analysis of the parameters should also take place, to see if removing certain parameters will also take away some of the 'statistical noise' and improve the model's performance. Most importantly however, the dataset should likely be modified, so that like the popular GTZAN dataset before it, it contains an equal number of songs for each class. Further addition of other types of instruments playing in the same periods, such as baroque violin pieces, should also be added to the dataset to expand its generalizability and real world potential.

I plan to continue this experiment by trying to add more songs to the dataset, in an effort to make the proportion of each class more even. Secondly, I will try using a slightly similar CNN this time, to hopefully avoid overfitting, so that my model will generalize better to the test data. By refining this experiment, we will be one step closer to finding the extent to which a deep learning system can be used to classify classical music into its distinct subgenres.

References

Bloomberg. (2021, May 17). *NYSE: Spotify Technology*. Bloomberg. Retrieved May 17, 2021, from https://www.bloomberg.com/quote/SPOT:US

Costa, Y. M., Oliveira, L. S., & Silla Jr, C. N. (2017). An evaluation of convolutional neural networks for music classification using spectrograms. *Applied soft computing*, *52*, 28-38.

Dietterich, T. (1995). Overfitting and under computing in machine learning. *ACM computing surveys (CSUR)*, *27*(3), 326-327.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

Guyon, I., & Elisseeff, A. (2006). An introduction to feature extraction. In *Feature extraction* (pp. 1-25). Springer, Berlin, Heidelberg.

Huang, D., Pugh, E., & Arianna, S. (2016). Music genre classification. *Int. J. Eng. Comput. Sci*, 1-6.

Keras API Reference. (n.d.). keras.io. Retrieved May 17, 2021, from https://keras.io/api/

Kozma, L. (2008). k Nearest Neighbors algorithm (kNN). *Helsinki University of Technology*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015, July). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference* (Vol. 8, pp. 18-25).

Morris, J. W., & Powers, D. (2015). Control, curation and musical experience in streaming music services. *Creative Industries Journal*, *8*(2), 106-122.

Pachet, F., & Cazaly, D. (2000, April). A taxonomy of musical genres. In *RIAO* (pp. 1238-1245).

Prey, R. (2020). Locating power in platformization: Music streaming playlists and curatorial power. *Social Media On Society*, *6*(2), 2056305120933291.

Rosen, C. (1997). *The Classical Style: Haydn, Mozart, Beethoven* (No. 653). WW Norton & Company.

Sturm, B. L. (2012, November). An analysis of the GTZAN music genre dataset. In *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies* (pp. 7-12).

Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, *10*(5), 293-302.

Weiss, C., & Müller, M. (2015, April). Tonal complexity features for style classification of classical music. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 688-692). IEEE.