Album Cover Generation by Genre Using Different CGAN Architectures

John Corio
Nick Kachkovsky
EECS 442 Final Project
University of Michigan

Abstract

Album covers of the same genre are said to share many aesthetic and stylistic characteristics. Through the use of different neural nets, this idea was examined over album covers collected from the Spotify API by studying the images generated relative to the genre label given. The results of three GAN structures, specifically a multi-class, multi-layer Perceptron CGAN, a multi-class convolutional CGAN, and genre-specific DCGANs, were examined with respect to the idea of genres containing their specific aesthetics with an additional final goal of creating stand-alone album covers.

1. Introduction

1.1. Motivation and Problem Description

A commonly held opinion in the music community is that a specific genre has a specific aesthetic to its visual manifestations. Clothing, hair styles, and music videos are included in this discussion, but the most conspicuous element is the album cover.

As such, it is believed that album covers for one genre share characteristics, such as color palette, intensity, complexity of visual effects applied, etc. The issue with this belief is the lack of objectivity in all art forms. By using neural networks to learn and generate albums for specific genres, similarities can be more rigid as it utilizes the objectivity of machine learning to learn and reproduce characteristics fundamental to a genre. After image generation, the results can be studied and compared to their own genre and other genres to find definitive similarities and differences.

Furthermore, this project is also an artistic endeavor. Since programs like Photoshop are costly and time-consuming, being able to utilize a neural net to create interesting album covers from only noise at high speed is another goal of the project.

1.2. Individual Contributions

Data gathering, cleaning, formatting, and preparation were done by John Corio. He queried the Spotify API across a list of 10 genres, searching and collecting 50,000 album covers before storing them on a local machine. Individual formatting and preparation was made before the files were zipped into files and uploaded to Google Colaboratory. He also helped edit and design different network architectures to attempt.

Dataset building, architecture building, training, and visualization were worked on by Nick Kachkovsky. He and John jointly worked on adjusting the dataset classes and initializing the data loaders. Furthermore, he worked on building the architectures for both the generator and discriminator; he was also responsible for setting up the training loop for the network. Finally, he created the visuals for the network output.

2. Related Work

2.1. Album Cover Generation From Genre Tags

Published in October 2017, Hepburn et al. approached the same exact problem. As such, a lot was learned from their approaches, particularly network architecture and hyperparameter settings. The one thing adding to the strength of their project was doing genre tag generation based on pre-trained weights while utilizing a more well-behaved dataset. The dataset, as will be shown, made the biggest difference between their results and ours.

2.2. Unsupervised Representational Learning with DCGAN Networks

Chintala, Radford, and Metz explore unsupervised CNN learning and how it can be applied to GANs. From this paper, we were able to learn the network architecture for DCGANs, and the associated hyperparameter settings (which we adjusted for our purposes). Furthermore, this paper helped us investigate and visualize the internals of the network, which ultimately resulted a much better understanding of the system and correlated to us achieving a valid final result.

3. Method

3.1. Multi-Class Embedding

A technique implemented commonly in works related to multi-class image generation is embedding. Demonstrated by Figure 1, a label is transformed from a number into a matrix filled completely with the label, before being embedded onto the image via its channels. For the embedded models investigated, the genre label was transformed into a dense 3 by 64 by 64 matrix, before being embedded into the RGB image's color channels.

These 6 by 64 by 64 label-embedded images were fed into the discriminator for training or into the generator for direction in generating genre-styled images.

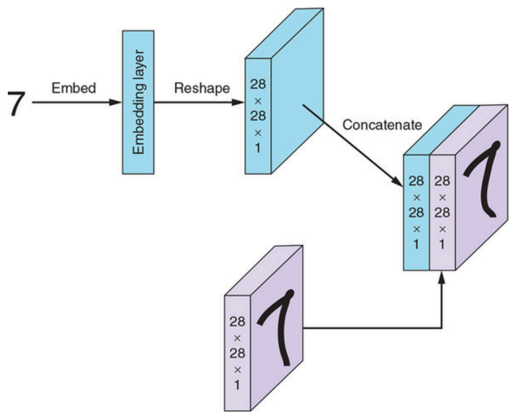

Figure 1: Example of embedding technique
Image credit: Manning Publications

3.2. Multi-Class Embedded Linear CGAN

This model utilized embedded Linear layers for a multi-layer perceptron in an attempt to see if the added computational complexity of convolutional layers was necessary to produce consistent genre-specific features and interesting, believable album covers.

Generator Architecture

Layer	In & Out Channels	Activation Layer
Input	100, 110	Torch.Embed
Linear	110, 256	Leaky ReLU
Linear	256, 512	Leaky ReLU
Linear	512, 1024	Leaky ReLU
Linear	1024, 2048	Leaky ReLU
Linear	2048, 4096	Leaky ReLU
Linear	4906, 8192	Leaky ReLU
Linear	8192, 12288	Tanh

Discriminator Architecture

Layer	In & Out Channels	Normalizing Layer	Activation Layer
Input	12288, 12298	X	Torch.Embed
Linear	12298, 8192	Dropout	Leaky ReLU
Linear	8192, 4096	Dropout	Leaky ReLU
Linear	4096, 2048	Dropout	Leaky ReLU
Linear	2048, 1024	Dropout	Leaky ReLU
Linear	1024, 512	Dropout	Leaky ReLU
Linear	512, 256	Dropout	Leaky ReLU
Linear	256, 1	X	Sigmoid

3.3. Multi-Class Embedded CGAN

The Multi-Class CGAN followed the same embedding schema as well as roughly the same network structure as that of the Linear CGAN, except replacing Linear layers with fewer Convolutional layers. The aim of this network was to move in a different direction than the Linear CGAN by using convolutions to discern local structures in the dataset in order to produce lifelike forms and variations in color and texture.

Generator Architecture

Layer	In & Out Channels	Normalizing Layer	Activation Layer
Input	100, 100	X	Matmul(Input, Labels)
ConvTranspose2d	100, 512	BatchNorm	ReLU
ConvTranspose2d	512, 256	BatchNorm	ReLU
ConvTranspose2d	256, 128	BatchNorm	ReLU
ConvTranspose2d	128, 64	BatchNorm	ReLU
ConvTranspose2d	64, 3	X	Tanh

Discriminator Architecture

Layer	In & Out Channels	Normalizing Layer	Activation Layer
Input	3, 6	X	Torch.Embed
ConvTranspose2d	6, 64	X	Leaky ReLU
ConvTranspose2d	64, 128	BatchNorm	Leaky ReLU
ConvTranspose2d	128, 256	BatchNorm	Leaky ReLU
ConvTranspose2d	256, 512	BatchNorm	Leaky ReLU
ConvTranspose2d	512, 1	X	Sigmoid

3.4. Single Class DCGANs

Single Class DCGANs ignored the embedding layer and genre labels in favor of simply feeding one genre's dataset into its own DCGAN architecture separately. By doing this, the model was allowed to focus on learning characteristics about one genre at a time, hopefully reducing bias or genre mishaps introduced by the embedding layer or the dataset.

Generator Architecture

Layer	In & Out Channels	Normalizing Layer	Activation Layer
Input	100, X	X	Torch.Randn
ConvTranspose2d	100, 512	BatchNorm	ReLU
ConvTranspose2d	512, 256	BatchNorm	ReLU
ConvTranspose2d	256, 128	BatchNorm	ReLU
ConvTranspose2d	128, 64	BatchNorm	ReLU
ConvTranspose2d	64, 3	X	Tanh

Discriminator Architecture

Layer	In & Out Channels	Normalizing Layer	Activation Layer
Input	3	X	X
ConvTranspose2d	3, 64	X	Leaky ReLU
ConvTranspose2d	64, 128	BatchNorm	Leaky ReLU
ConvTranspose2d	128, 256	BatchNorm	Leaky ReLU
ConvTranspose2d	256, 512	BatchNorm	Leaky ReLU
ConvTranspose2d	512, 1	X	Sigmoid

4. Experiments

4.1 Dataset and Hyperparameter Considerations

The dataset consisted of JPG file genre label pairs collected from the Spotify API by querying unique artists by genre and then selecting their albums. One very important note to make is Spotify's use of genres in describing artists only, and not albums. Since artists had to be assumed to make albums of only one genre due to this format on the API, some unavoidable error was introduced into the dataset, skewing results. This will be discussed later in the Experiments and Conclusions sections.

Architectures from papers were built and tested on well-documented datasets, specifically the MNIST numbers and clothing datasets and an outside faces dataset, to ensure they were working suitably. Once concluded, hyperparameters were varied for evaluation. Their starting values were that of the default in Hepburn et al. [1]: learning rate of 0.0002 with batch size 128, network depth of 3 to 4 layers for their album cover discriminator and generator, weight initialization using a normal distribution with mean 0 and standard deviation of 0.02. Hyperparameters were varied until the output of each network on a specific class created "lifelike" forms and colors, and, if present, a noticeable pattern with respect to in-class results.

Using the Adam optimizer and BCELoss, training was performed in the typical GAN scheme. A noise tensor was sent into the Generator and assigned a loss based on its discriminator output, before the Discriminator was assigned a loss based on its ability to determine the validity of batches of target images.

4.2 Embedded Linear CGAN

What Figure 2 shows provides evidence that convolutions are necessary in order to produce believable album covers. The linear layers fail to pick up on local shapes and forms and find no common traits to a genre either. Visually, it creates images interesting enough to *be* an album cover, but none that are actually indicative of a genre. There's too much random noise left in the image and too little form and variation. As such, this network is not useful in any of the analytical principles for the problem at hand, but is still OK artistically.

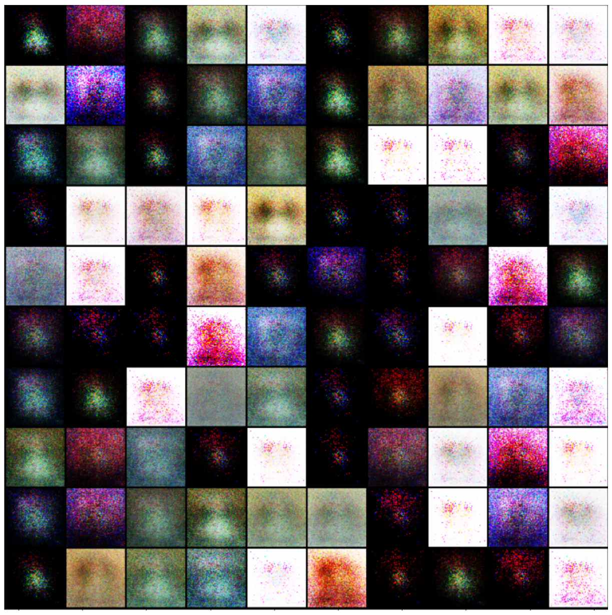

Figure 2: Linear CGAN after 20 epochs

4.3 Embedded Convolutional CGAN

This model also fails to produce analytically meaningful results but does show improvement over the Linear CGAN, as well as over continued training epochs. As is visible in Figures 2 and 3, it produces slight variations in shapes and color, albeit with a lot of random noise. Based on this knowledge and results given by a pre-trained network [1], it's possible that this did not have enough training time to produce as incredible results as Hepburn et al.

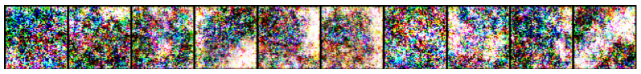

Figure 2: Results after 10 epochs

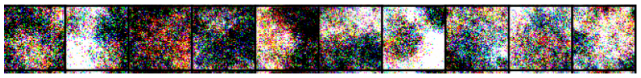

Figure 3: Results after 30 epochs

However, the most likely issue to be affecting the convolution CGAN results is the error inherent in the dataset. As was discovered in pretests using the Fashion MNIST, Numbers MNIST, and faces dataset, the convolutional CGAN is a very powerful and sensitive network. Since the dataset is quite noisy due to the assumption made in order to produce a sizable dataset, it could be having a hard time fitting genre trends that make any reasonable sense, as the embedding forces the model to differentiate between 10 noisy class labels. Artistically, these images are satisfactory, but like the Linear model fail to really create anything of any complexity or specificity.

4.4 One Class DCGANs

DCGANs produced the most promising results. These covers provide something to visually analyze as well as *looking* like album covers, albeit ones that would be considered more abstract. More believable structures and color patterns were created by the generator, showing up in a variety of shapes and textures. It even seems to output symbols similar to writing in one generated image, which is an interesting development. These could be believable album covers with a little editing or as is.

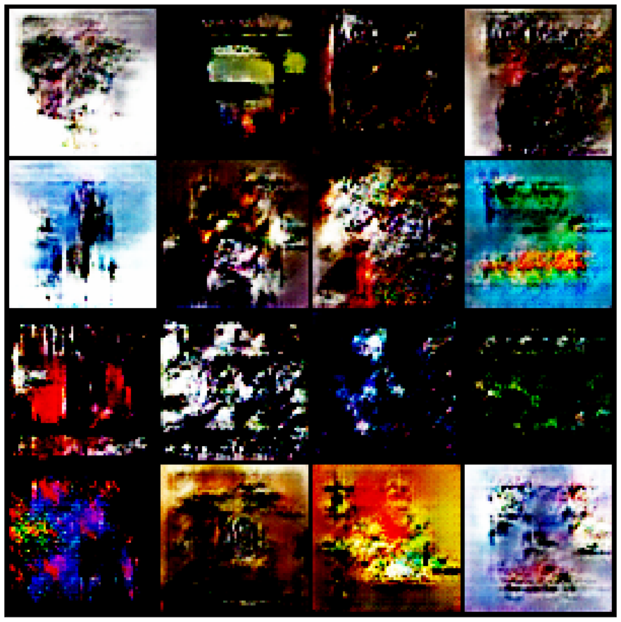

Figure 4: Separated DCGAN, 40 epochs

One can now analyze network output for the Shoegaze genre (see Figure 4) for trends. It appears to have learned some characteristics similar to most album covers: a semi color-static background defining a limited color scheme and some sort of main figure or form dominating the cover. There is a maximalism seen in these covers, specifically in texture and color intensity. The generator produced a lot of darker colors but sometimes utilized a white background centered around a dark structure. As such, this network is picking up on the fact that Shoegaze album covers typically exhibit aggressive, visually intense cover art, staying in theme with the music's heavy emphasis on Wall of Sound, heavy distortion or sound effects, and audio manipulation.

5. Conclusions

The most important thing to note about this project's goal to learn a set of definitive, comparable features for each genre is that it can be done with little more than an adequate dataset.

In order to properly identify and learn features, the dataset needs to have correctly labelled albums. The Spotify API used in this project simply did not allow for this, creating poor results in our CGAN architectures. A consideration for strengthening the dataset is possibly zooming out and considering more broad genres rather than niche ones (Slowcore for example).

When examining the architectures, their strength is known. These models follow architectures from published papers and produced satisfactory image generations from noise tensors on data like clothing, numbers, and faces. This makes it fairly obvious the dataset for this project unfortunately had too much noise for the networks to overcome. Extracting pre-trained features from other architectures, as seen in Hepburn et al. [1], is something that could push this model further, in both the cases of noisy data and sufficient data.

Overall, the artistic side of this project is nearly fulfilled. Given an input image of random noise, a legitimately interesting album cover can be produced from the DCGAN model, possibly even the convolutional CGAN as well. However, in order to make it more meaningful or emotionally resonant the network could use some extra information. An idea like feeding color and shape hints or providing images other random noise to generate images from could produce more visually pleasant color schemes and more lifelike structures and forms. Without going too much into detail, the possibilities to produce a more cohesive album cover for artistic purposes are exciting.

References

[1] Hepburn, A., McConville, R. and Santos-Rodriguez, R., 2017. Album Cover Generation From Genre Tags. [online] Ryanmcconville.com. Available at: <https://ryanmcconville.com/publications/AlbumCoverGenerationFromGenreTags.pdf> [Accessed 16 December 2020].

[2] Radford, A., Metz, L. & Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks