GAAAN

For

Project Report

10-May-2020

Prepared by: The Young and The Keras

Team Members: Patrick Caldwell, Joshua Matthews, Michael McDermott

**Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Abstract Art Gan |  | “Model was not robust enough.” | v.1 |
| Abstract Art Gan |  | “Output mode collapsed, discriminator accuracy too low.” | v.2 |
| Anime Variant Gan |  | “Output contained unrecognizable features.” | v.3 |
| Album Art Gan |  | “Current Model.” | v.4 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1.](#_heading=h.gjdgxs) Executive Summary 4

[2.](#_heading=h.30j0zll) Project Overview 5

[2.1](#_heading=h.1fob9te) Problem Statement 5

[2.2](#_heading=h.3znysh7) Scope 5

[2.3](#_heading=h.2et92p0) Project Team 5

[3.](#_heading=h.tyjcwt) Machine Learning Aspects 6

[3.1](#_heading=h.3dy6vkm) Dataset 6

[3.2](#_heading=h.1t3h5sf) Model Creation and Training 6

[3.3](#_heading=h.4d34og8) Inference 6

[3.4](#_heading=h.2s8eyo1) Evaluation of the Model 6

[4.](#_heading=h.17dp8vu) Software 7

[4.1](#_heading=h.3rdcrjn) System as a Software 7

[4.2](#_heading=h.26in1rg) System Architecture 7

[4.3](#_heading=h.lnxbz9) User Interface Overview 7

[5.](#_heading=h.35nkun2) Use Cases 8

[6.](#_heading=h.1ksv4uv) References 9

# Executive Summary

GAAAN is a Generative Album Art Adversarial Network. The purpose of GAAAN was to offer a tool for musicians and artists to have unique album artwork generated quickly at no cost. The tool is a website with a backend running on flask and a Deep Convolutional Generative Adversarial Network using tensorflow-keras. To train the GAAAN, we curated our own dataset using the Spotify API and some custom functions to grab about 54,000 images for training. The output of our GAN was not as detailed as we would have liked it to be, but with future iterations or architecture changes we feel it can be a viable solution to musicians looking to have artwork made for them.

Link to web service:<https://sunlit-mix-274903.wl.r.appspot.com>

Link to repository: <https://bitbucket.org/Goku_Sonic/gan/src/master/>

# Project Overview

## Problem Statement

The purpose of GAAAN is to generate unique album artwork for artists that do not have the time or resources to have it made. It is also a research project to see how well a GAN can make art, given the wide variety of album artwork, even within a specific genre.

## Scope

In its current version, the album art generator will generate a new, unique image each time the webservice is called. The images generated are based on images from genres such as metal, alternative rock, pop, psychedelic, and other sub-genres. Currently the web service does not have the capability of adding text on top of the image, so it is just a background. Future versions may be able to add text, specify the genre, and output at a higher resolution.

## Project Team

Name of the project: GAAAN: Generative Album Art Adversarial Network

|  |  |  |  |
| --- | --- | --- | --- |
| Name of the Team member | Responsibility | Contribution % | Notes |
| Patrick Caldwell | Configuration of Flask server and deployment to Google App Engine and GCP | 33.33 |  |
| Joshua Matthews | Training GAN and tuning hyperparameters | 33.33 |  |
| Michael McDermott | Data sourcing & cleaning, GAN modeling and training | 33.33 |  |

# Machine Learning Aspects

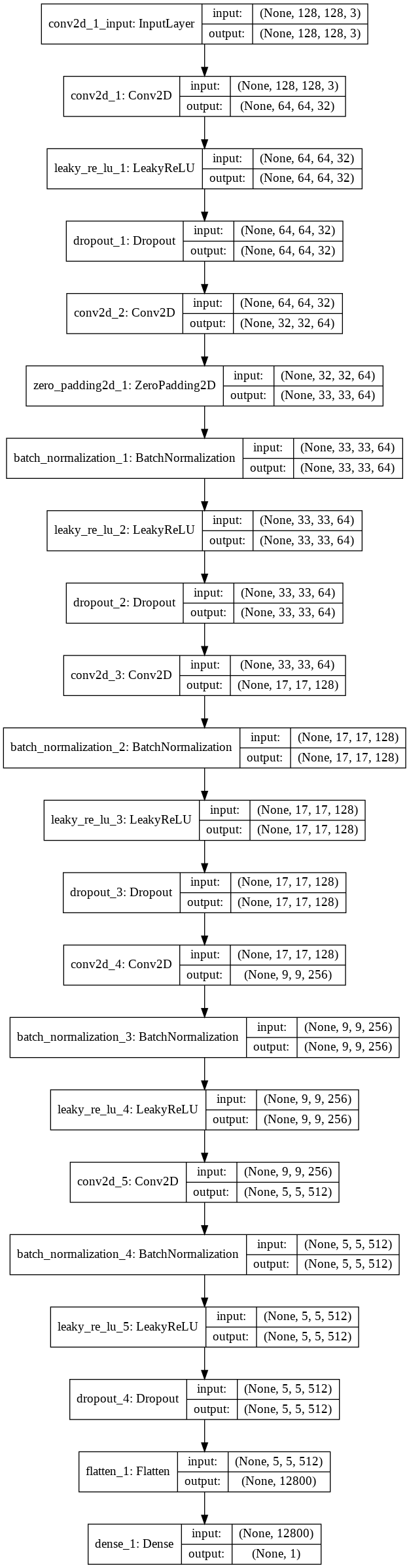
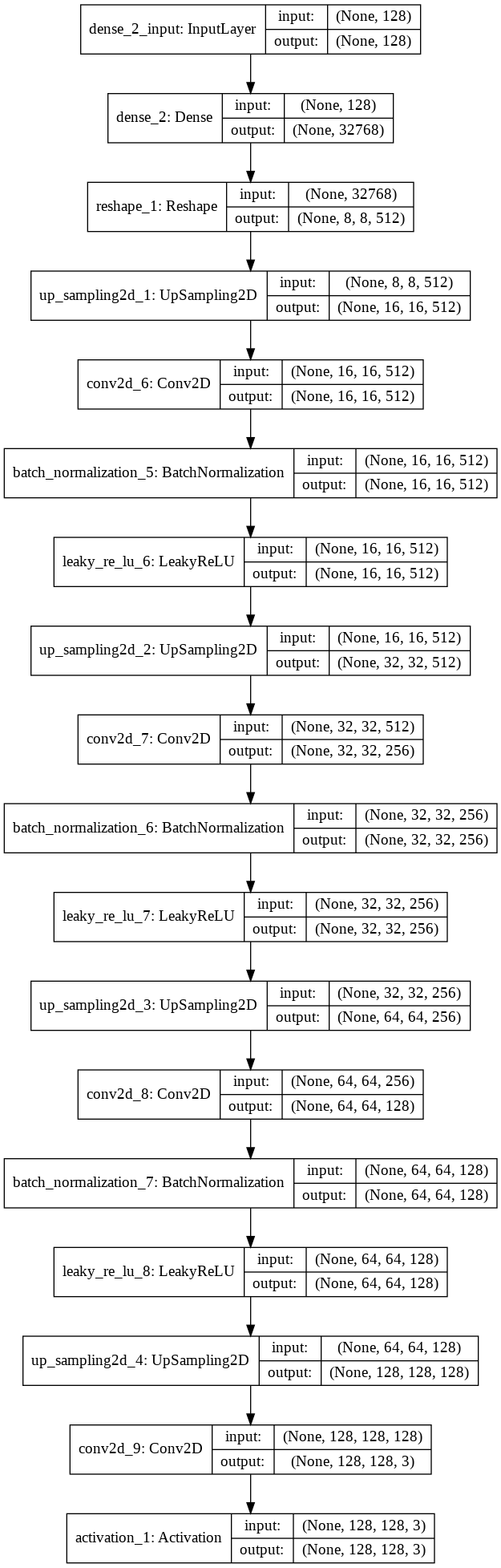
## Dataset

* Originally, we were going to use the one million album artwork dataset available on the internet, but it was very large to download and we felt it was too generic, so we compiled our own dataset of 54,000 using python and the Spotify API.
* In the GAN, all the data from the dataset is used for training the discriminator.
* After the images were sourced, they had to be resized to size 128x128 pixels, and some sub-datasets were created to try training more specific versions of the GAN (eg, specifically on electronic dance music or specifically metal and rock genres).
* The image data had to be normalized from values of 0-255 to -1 & 1
* The dataset was about 20GB if saved and loaded in a pkl file, which used a lot of memory when training, so we ended up dynamically loading the data in batches.

## Model Creation and Training

* Our model is a deep convolutional generative adversarial network. We chose this architecture due to it being more stable than other GAN’s. We considered using StyleGAN, but the custom libraries and setup would have taken more time which would have given us less time to experiment. The architecture consists of a generator model and discriminator model. Both have convolutional layers and batch normalization layers. The discriminator model is trained on both the dataset of sourced images, and the output of the generator. Each epoch, the discriminator is updated on whether it was good at detecting a real vs generated image, and the generator is updated on whether it was able to trick the discriminator or not.

Generator Discriminator



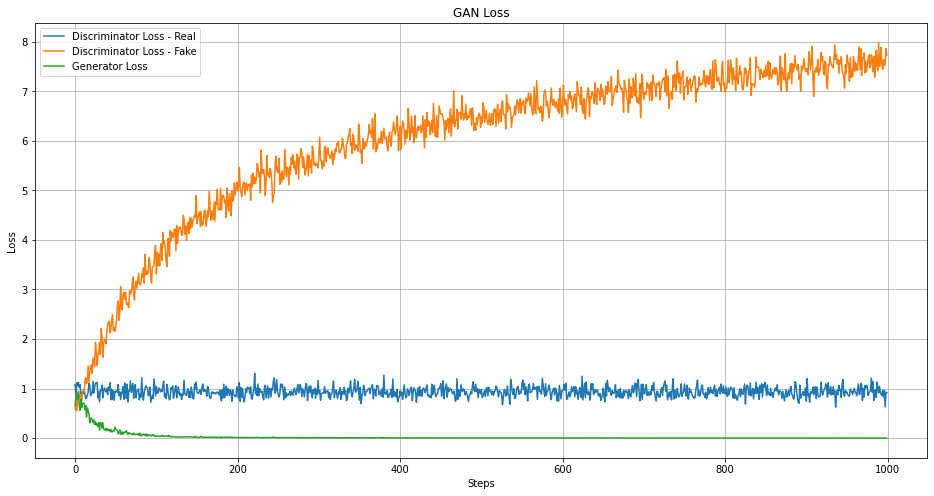
* Description of the training process:
  + We experimented with multiple epoch lengths and sizes of training data.
  + Epoch lengths of 5,000, 10,000, 50,000, 75,000, and 100,000 were used.
  + Training data sizes consisted of 54,000 images for a version that was trained on the whole dataset, 8,000 images for the subset of electronic and psychedelic images, and about 5,000 images for the metal/alt rock subset.
  + The main hyperparameters that we tuned were the learning rate for the discriminator and generator, number of epochs, noise vector size, strides and kernel size in the convolutional layers, and momentum in batch normalization layers. The current versions have a learning rate of .0015 for the discriminator and learning rate of .0015 for the generator. The batch normalization momentum is currently 0.8.
* Link to visual representation of training output:<https://imgur.com/ZyyivO7>

## Inference

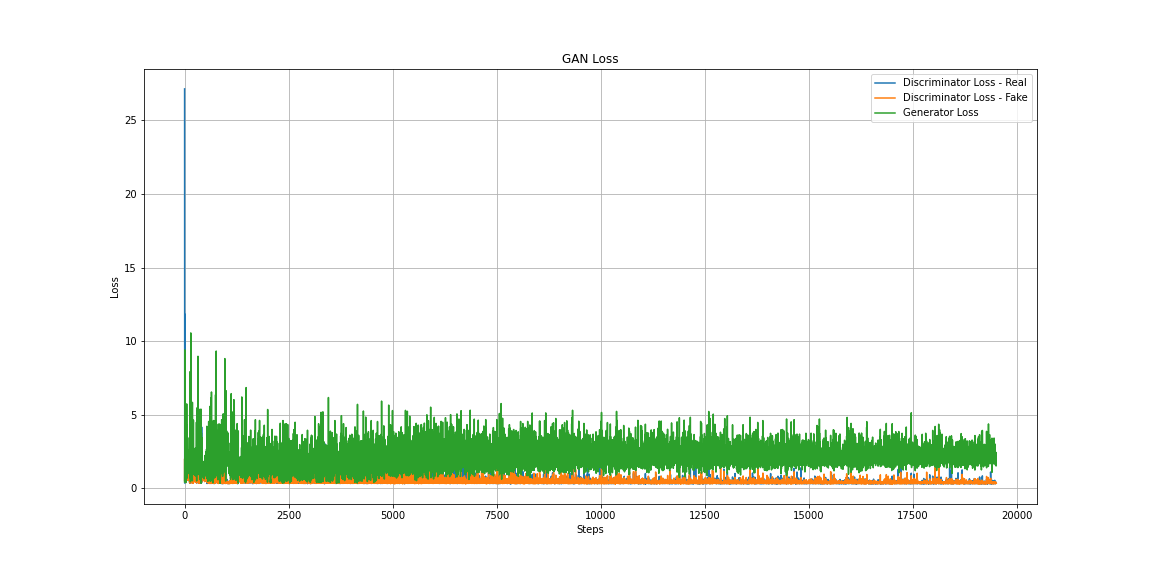
* Our model's “prediction” is the output of an image, given a random noise vector that is generated by the server when it is called.

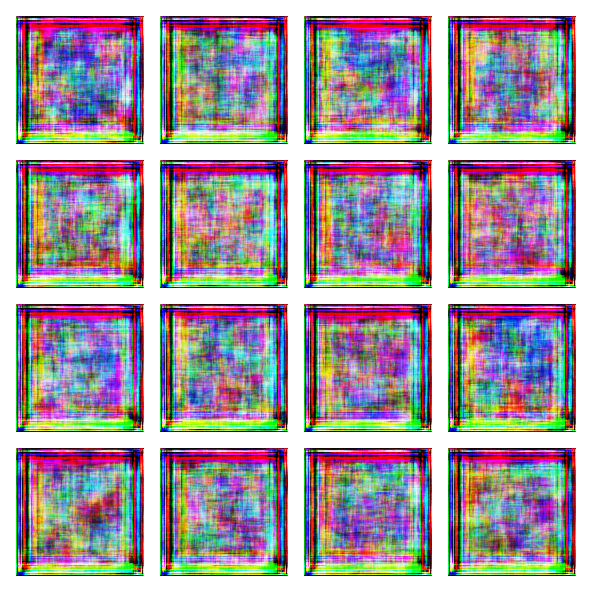
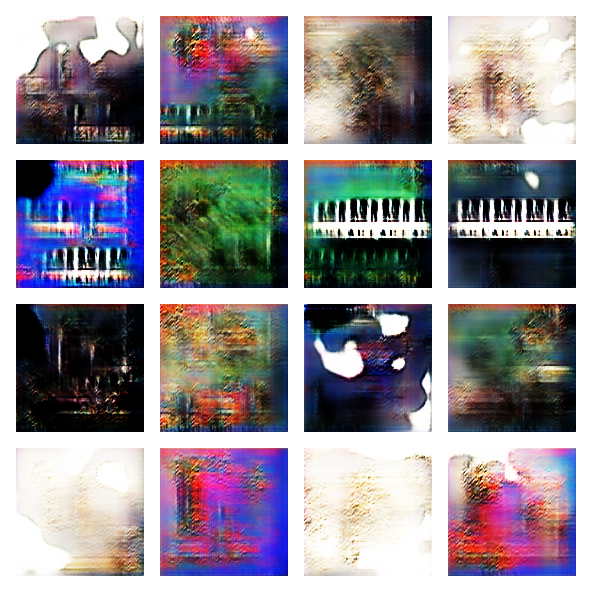
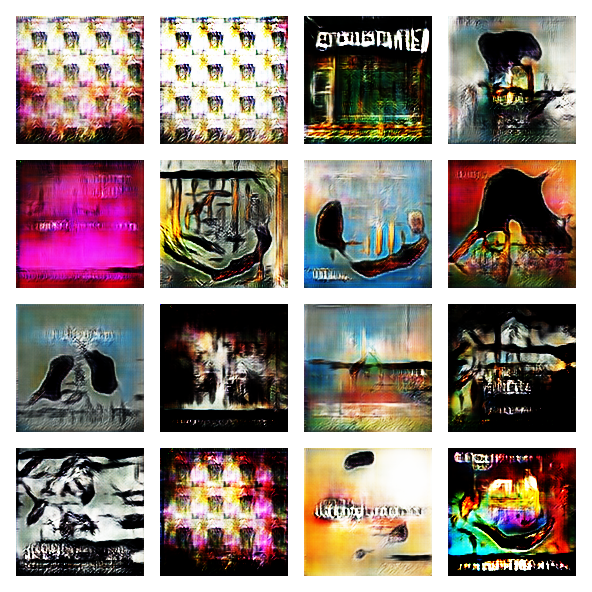
## Evaluation of the Model

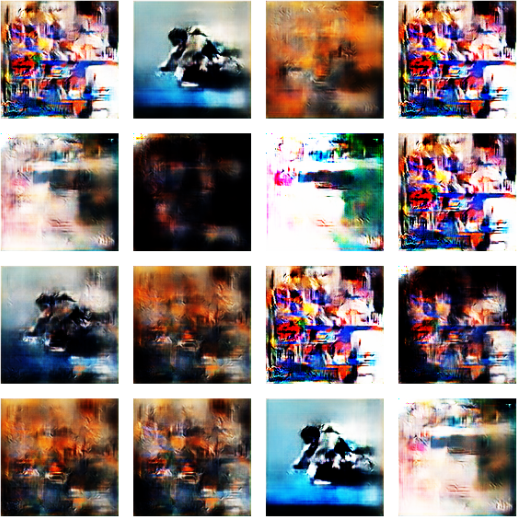
* The model was mainly measured on loss and our own interpretation of whether we thought the output could pass as an album art image or not. The accuracy and loss metrics were mainly helpful if the generator or discriminator rapidly approached zero or mode collapsed. From our research, we have seen that the loss can fluctuate and even if the generator or discriminator have a loss between .5 – 5 the output could still be good or bad. Metrics really just helped us keep an eye on the training to alert us if there were any drastic changes made. Otherwise, every 50 or so epochs we would have the output printed to screen to see what it looked like and judged our model based off of that.
* **Graph of first version loss**



* **Loss Graph after architecture revision**



* **Output of first version trained on full dataset**
* **Output of second version trained on full dataset at 12,000 epochs**
* **Output of final version of rock genre GAN trained only on rock/metal genres (34,000 epochs)**
* **Output of final version of electronic and psychedelic genres**



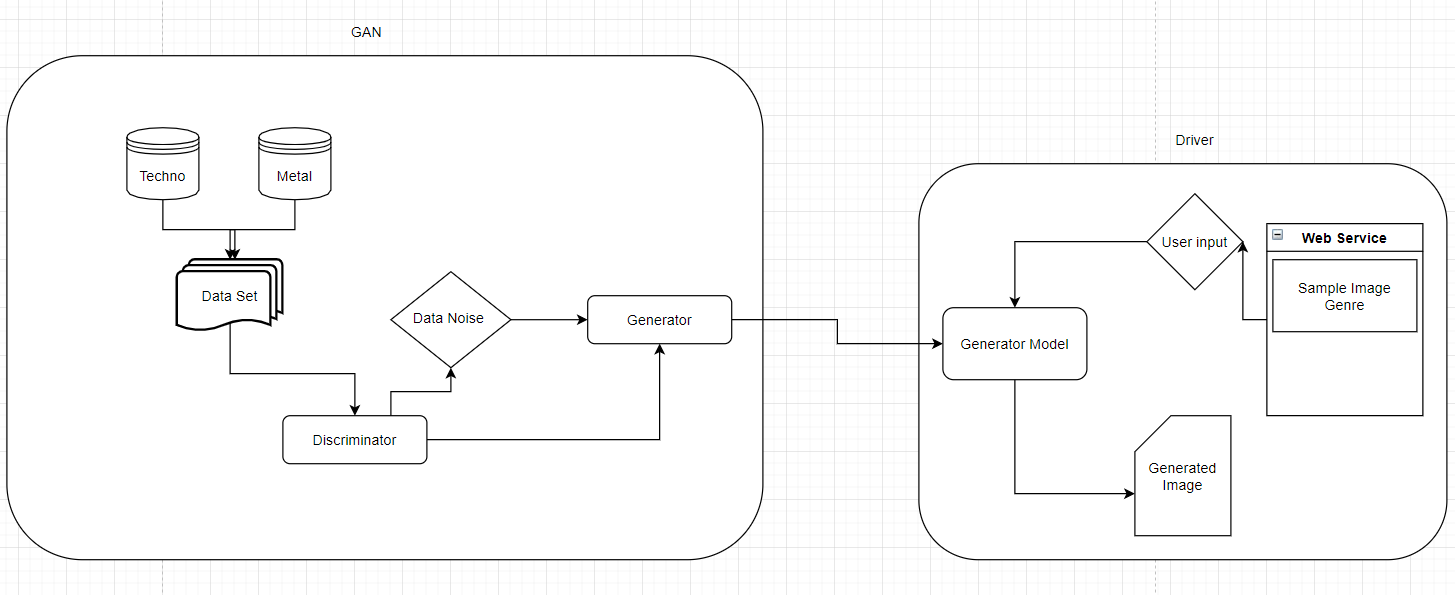
# Software

## System as a Software

* UI
  + Built in python using flask on Google’s app engine.
  + Takes a sample image along with the genre and using them as seed’s takes the trained model and outputs an image that is similar.
  + UI generates an image in matplotlib
* Driver
  + The code on the client side that interacts with the user in order to generate an album cover.
  + Code takes the hdf5 file (trained model) and uses that to output the album art.
* Trainer
  + User never sees this code or interacts with it.
  + Took 80% of the project time in development along with testing.
  + Was designed to take a data set of preexisting album artwork and train the discriminator along with the driver for quick image generation.

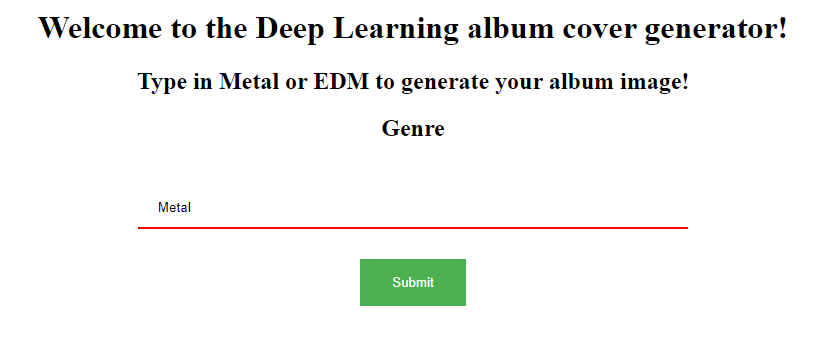
## System Architecture

* The project is delivered as a web service with an already trained model
* We took multiple data sets and trained images using different noise figures in order to get a result that was usable on the web service.

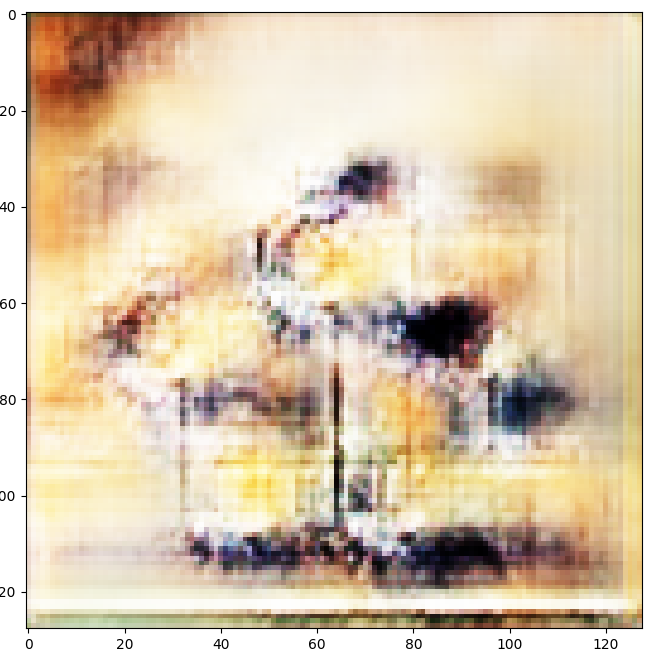
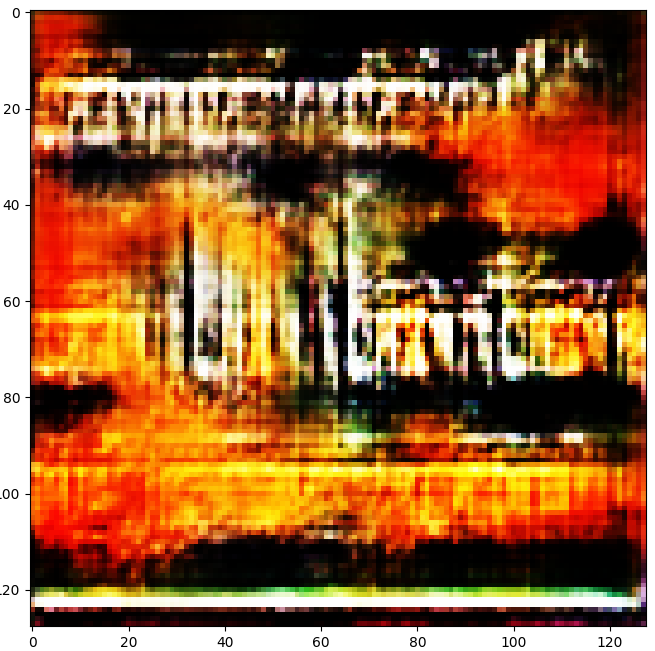
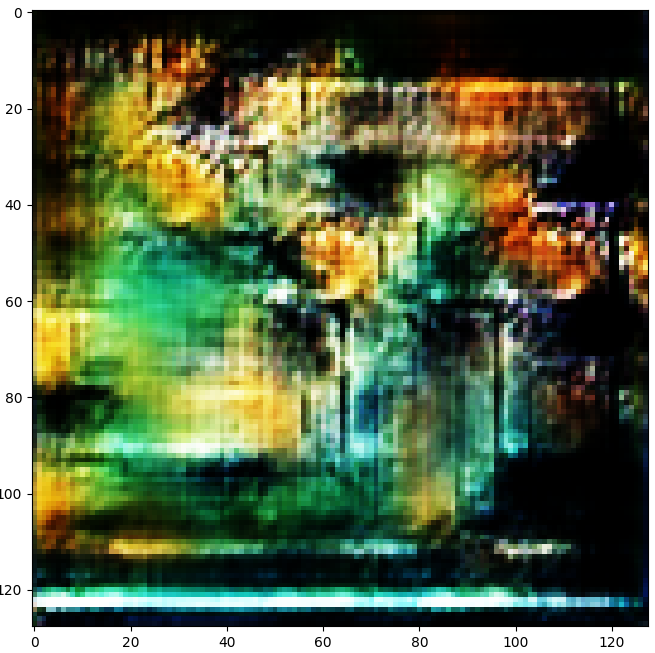
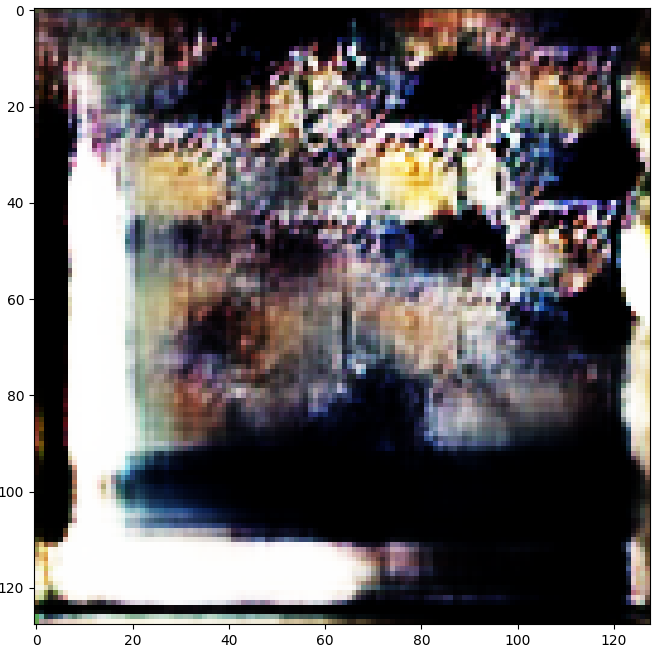
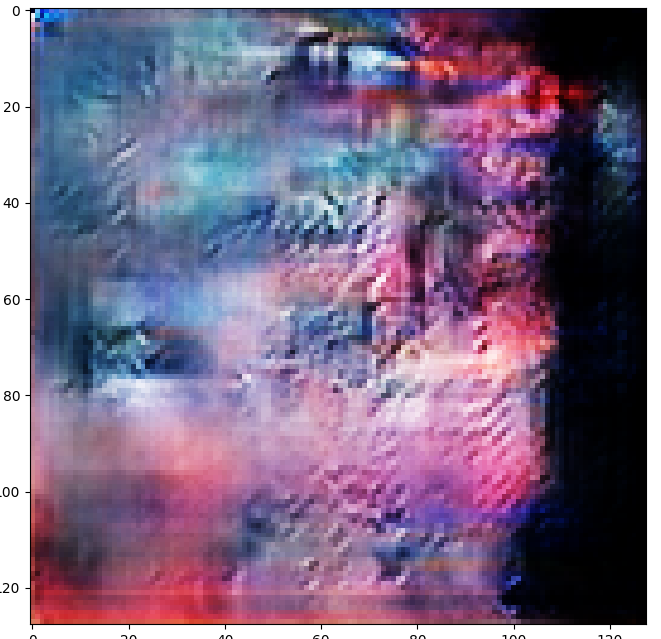
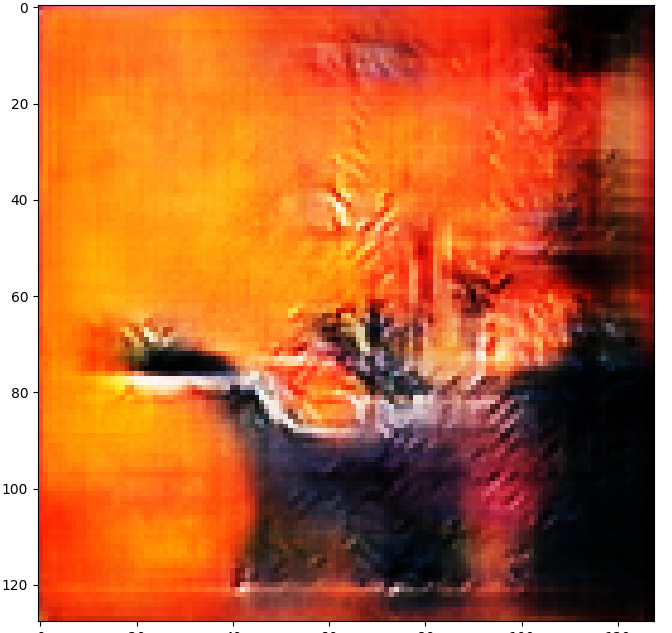
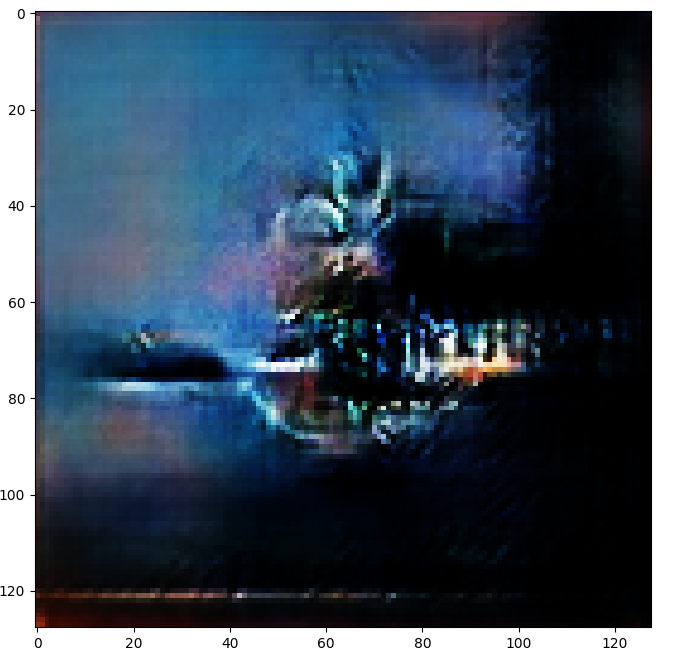
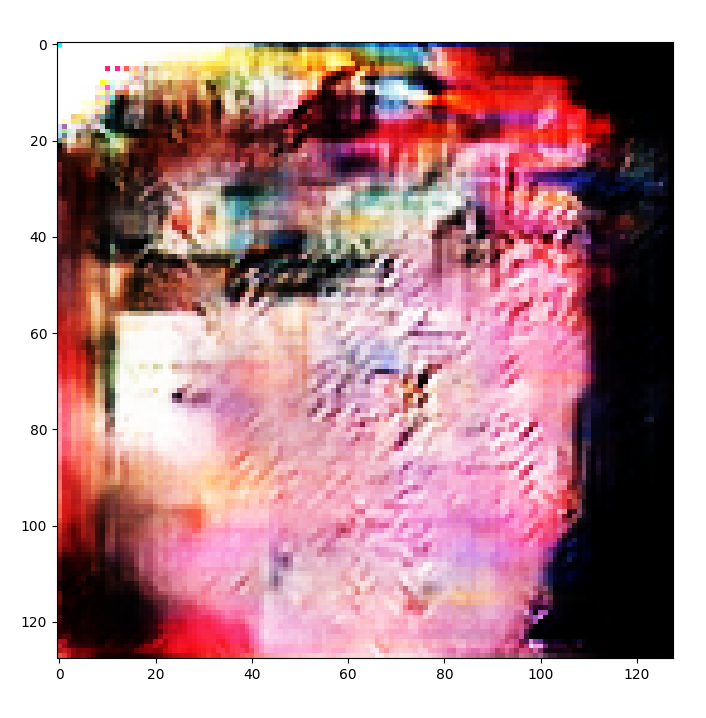
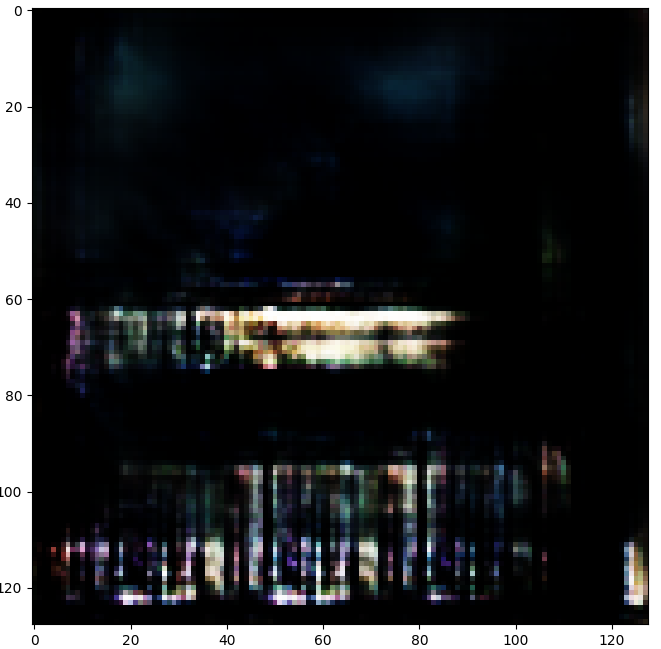
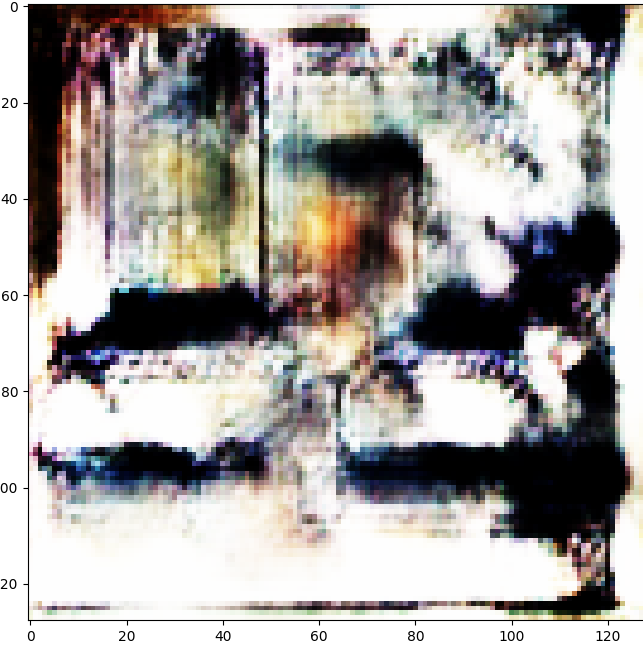


## User Interface Overview

* Description:
  + The software is server based. No installation needed.
  + We train the models in Google Colab then upload the trained model to the server which uses the driver code in order to generate an image that is similar to that genre.
  + In the code repository is a directory that contains the python file that was made to source the data and a Google Colab notebook that was used to train the models.
  + Screen Capture



# Use Cases

* Metal Use case
  + The driver will recognize that Metal has been chosen
  + The Metal model with a noise batch size of 32 and noise of 100 will be loaded within the driver
  + 
* EDM use case
  + The driver will recognize that EDM has been chosen
  + The EDM model with a noise batch size of 32 and noise of 100 will be loaded within the driver
  + 
* Default use case
  + The driver will recognize that Metal has been chosen
  + The General model with a noise batch size of 32 and noise of 128 will be loaded within the driver
  + Include screen capture images captured doing these test use cases
  + 

# References

* + Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv:1511.06434* (2015).
    - <https://arxiv.org/pdf/1511.06434.pdf%C3>
  + Shrestha, Anish. “Generating Modern Art Using Generative Adversarial Network(GAN) on Spell.” *Medium*, Towards Data Science, 3 Feb. 2020, towardsdatascience.com/generating-modern-arts-using-generative-adversarial-network-gan-on-spell-39f67f83c7b4.
  + Wibowo, Haryo Akbarianto. “Generate Anime Style Face Using DCGAN and Explore Its Latent Feature Representation.” *Medium*, Towards Data Science, 13 Apr. 2019
    - towardsdatascience.com/generate-anime-style-face-using-dcgan-and-explore-its-latent-feature-representation-ae0e905f3974.
  + <https://spotipy.readthedocs.io/en/2.12.0/>
  + <https://archive.org/details/audio-covers>