**“Arti-GaN”**

FA21 CSCI-P445 Capstone RF 2

Nathanael L. Mann, Mark Bridgewater, Joshua Carrier

Sponsor: Chris Sexton

28 September 2021

**Breakdown of individual contributions**

* Nathanael - Product Design & Documentation
* Mark - Website and Repository Management
* Joshua - Technical Research & Development

**Software Requirements Specifications**

**1 Introduction**

Arti-GaN is an Artificial Intelligence based Image Generator. Using Generative Adversarial Networks and small datasets of images, we are able to create an AI capable of generating similar, but unique images, and through human interaction train those same AI’s to generate more aesthetically pleasing images.

**1.1 Purpose**

We examined a number of pre-existing projects involving artificially generated images, and decided to expand upon that field of research by taking the idea in a slightly new direction. We wanted to challenge ourselves to see if we could push the boundaries of what has been done before. There are a number of software programs that work with AI to create images, but none that we were able to find that use the techniques/systems outlined in this document.

**1.2 Definitions**

**1.2.1 Generative Adversarial Networks (GaN’s):**

Generative Adversarial Networks (GaN’s) are a type of neural network structure that can synthesize new images based on patterns learned from the input dataset. This model contains a generative network, which is usually a deconvolutional neural network, as well as a discriminative network, which is a convolutional neural network. The discriminator is fed the training dataset of images and is trained whether an image is generated or not. The generator then tries to generate a new image that will convince the discriminator that the generated image is part of the dataset. If the discriminator decides the image is not a part of the dataset, the generator will be “punished” and adjust its values accordingly. If the discriminator’s prediction is wrong, it is punished. The end goal for the network will be that the generator will always produce an image that makes the discriminator’s confidence by 50%. This means the generator has reached a point where the discriminator cannot tell the difference between a generated image, and a real one.

**1.3 System overview**

The system will be composed of a backend software program to generate GaN’s and a frontend web page where users can interact with one of five different pre-trained, hand-selected GaN’s and provide them feedback on the aesthetic quality of the produced images.

**1.4 References**

**1.4.1 Harold Cohen created AARON**

AARON is one of the oldest AI systems that is still maintained.

AARON is an AI that can draw and color original art.

**1.4.2 Nvidia GauGAN**

Nvidia’s GauGAN AI allows the user to draw a really basic image of terrain, and the neural network outputs a very realistic-looking image based on the user’s drawing.

**1.4.3 Deep Dream Generator**

The Deep Dream Generator AI takes two separate images and combines them into one. It can do this in multiple fashions such as taking the style from one and applying it to another or to one part of the image.

**1.4.4 Rutgers University's Art and Artificial Intelligence Laboratory**

They made an AI that learns different styles of art and then has to create new art using a different style than the ones it was taught.

**1.4.5 Justin Johnson**

In this application of the paper *A Neural Algorithm of Artistic Style* they made an AI that can take in a style image and another base image, and the AI will take the style from the first image and apply it to the second image.

**2 Overall description**

“Arti-GaN” consists of a two-part training system for it’s Artificial Intelligence Networks. The following sections describe the anticipated format of the overall project.

**2.1 Product perspective**

**2.1.1 System interfaces**

A backend user will be able to upload a directory of “seed images” with which to begin training a network. These images will be automatically formatted to be of equal size for ease of computation. Upon running the program, a GaN will be generated from the seed directory, and can then be uploaded to the webpage where the end-user can interact with it as described in the User Interfaces Section.

**2.1.2 User interfaces**

Upon visiting the website where the project will be hosted, a user will be able to select between 1 of 5 AI “personalities'' with which to interact. A brief description and sample image from that AI will be provided. Once selected the user will be brought to a page where they will be presented with an image generated by the AI that they selected, and be able to rate it as “good” or “bad” with a “thumbs-up” or “thumbs-down” button. Upon rating the image, either way, the algorithm will generate a new image for the user to view, and the process may repeat. At any time a user can go back to the “home page” and re-select which AI they want to interact with.

**2.1.3 Hardware interfaces**

End-users can interact with the website in order to choose the different AI’s and see the images. If the user can access our webpage, then they can use the software. The website will interact with the GaN software to produce the different images.

**2.1.4 Software interfaces**

Our GaN software should run on Windows 10/11. The webpage should be viewable in most up-to-date web browsers in desktop/mobile view. The website will act as the GaN user interface so as long as the server is being run on the Windows 10/11 device, anyone with a compatible web browser should be able to access the website.

**2.1.5 Communication Interfaces**

The website will be viewable on most web browsers and in mobile view as well. The plan is to have the website accessed through HTTPS even though no direct information will be passed back and forth between end-user and client. As described in the user interface section, the end-user will only be able to navigate through the different AI’s and what they generate. However, HTTPS is almost required for most web browsers to access the website and google searches prioritize HTTPS in their algorithm over HTTP. Some variation of JavaScript will be used to communicate between the AI algorithm and website interface.

**2.1.6 Memory Constraints**

When training a neural network with images, you have to be careful not to make your datasets too large. If high-resolution images with color are used, it can take a long time to process even a decent amount of images. In order to combat this, our plan is to keep the initial training set pretty small, both in terms of the number of images and the size of each individual image, and have most of the training done by the users.

**2.1.7 Operations**

**2.1.7.1 Local Operations**

The user will pass a dataset file when running the program. After a period of unobserved training, the GaN will output an image, and the GaN will be considered to be proficient at generating images.

**2.1.7.2 Site Operations**

The user will select an AI to interact with and either like or dislike the image to influence the image produced by the AI. The user can also download the image at any time if they so desire.

**2.1.8 Site Adaptation Requirements**

The software to run the website and GaN algorithm will be installed onto a single machine so it just needs to run on that device whether it be a Windows 10/11 device or Windows Server edition device. As far as the website UI, the website should be dynamic so that desktop and mobile users of any platform/web browser can view and interact with the website.

**2.2 Product functions**

**2.2.1 Local Software**

Allows for the creation og GaN’s that can output an image based on a set of input images, with the goal of producing a similar, yet unique image that is indistinguishable from the seed set.

**2.2.2 Website**

Website allows for play with 5 different AI trained by our datasets and users can either like or dislike images to produce new images based on such. Users then can download those images if they desire.

**2.3 User characteristics**

**2.3.1 Back End Developers**

Backend developers will be able to add to generate GaN’s based on seed images.

**2.3.2 Web Page Visitors**

The web page visitors will be able to choose which model they want to train. Once selected, the user will be able to help train the AI by giving feedback on it’s art.

**2.4 Constraints, assumptions, and dependencies**

**2.4.1 Tensorflow**

A powerful tool that greatly simplifies the process of creating neural networks.

**2.4.2 Matplotlib**

Matplotlib is good for creating graphic representations of statistics of our models.

**2.4.3 Python 3**

Python is a great language for Machine Learning projects. It makes tasks that would be hard in other languages a lot easier.

**2.4.4 Numpy**

Makes making matrices much easier and gives us many options to manipulate them.

**2.4.5 Pandas**

Good for loading datasets in an orderly fashion.

**2.4.6 HTML5**

The last major version of HTML.

**2.4.7 AngularJs**

Great for making dynamic websites that are predictable.

**2.4.8 MIT Licensing**

Our code is essentially open source.

**2.4.9 Image Seed Storage**

In order to minimize necessary storage space, each “seed” file will contain between 50-100 images, formatted to equal sizes.

**2.4.10 Databases**

Image Galleries with which to train the GaN

**2.4.10.1 Abstract Art Gallery**

A dataset containing 2782 images of abstract art.

**2.4.10.2 Bob Ross Paintings**

A dataset containing appx. 500 paintings done by Bob Ross.

**2.4.10.3 Scryfall**

*Magic the Gathering* cards all contain an image (pre formatted in size, with over 20,000 images to choose from).

**2.4.11 Training Time**

Anticipated goal of > 1 hour to reach equilibrium in the GaN’s discriminator vs generator

**2.4.12 User Interaction Time Frame**

Anticipated launch in late January 2021, and data collection for appx. 3 Months throughout the duration of the project.

**2.4.13 Malicious Users**

End users on the website may intentionally answer incorrectly on the aesthetic quality of the art. We choose to recognize this as subjective preference, which aligns with the goal of the project, so the concern is minimal.

**2.4.14 Programming Practices**

When coding, we will follow the Google Style guide.

**3 Specific requirements**

**3.1 External interfaces requirements**

The website homepage should include 5 buttons to pick between the 5 different AI’s. Each AI page should have the image they produce, a like and dislike button, and a download button.

**3.2 Functional Requirements**

**Functional Requirement 3.2.1**

TITLE:Button - Homepage

DESC:Selects AI 1

**Functional Requirement 3.2.2**

TITLE:Button - Homepage

DESC:Selects AI 2

**Functional Requirement 3.2.3**

TITLE:Button - Homepage

DESC:Selects AI 3

**Functional Requirement 3.2.4**

TITLE:Button - Homepage

DESC:Selects AI 4

**Functional Requirement 3.2.5**

TITLE:Button - Homepage

DESC:Selects AI 5

**Functional Requirement 3.2.6**

TITLE:Website chooses which AI to provoke.

DESC:Depending on what button was hit for either AI 1, 2, 3, 4, or 5, the website will provoke the AI to create an image.

**Functional Requirement 3.2.7**

TITLE:Image - AI selected

DESC:Shows an image produced by AI 1, 2, 3, 4, or 5, depending on which selected

**Functional Requirement 3.2.8**

TITLE:Button - AI Image

DESC:This button will like the image produced and influence the AI to produce another one

**Functional Requirement 3.2.9**

TITLE:Button - AI Image

DESC:This button will dislike the image produced and influence the AI to produce another one

**Functional Requirement 3.2.10**

TITLE:Website provokes AI to create another image

DESC:The website will provoke the AI to create another image based on whether the user hit the like or dislike button.

**Functional Requirement 3.2.11**

TITLE:Button - Download image

DESC:This button will allow the user to download the image on the screen.

**3.3 Performance Requirements**

On the backend, we want to be able to load a dataset, create a new model, and then save that model for later use with the front end. The goal is for a new model to be created and trained in under thirty minutes, though it may take longer. One of the main parts of this project is to have users help train the network. This leads us to believe that we should make it a high priority to generate new images very quickly. If it takes more than a few seconds to synthesize a new image, the user is more likely to get distracted between voting on each image. If the users are getting distracted too often, training our model will be a painfully slow process. We want to try and stop this by generating images as fast as possible.

**3.4 Logical Database Requirement**

We may be using images pulled from a database to train our models.

**3.5 Software System Attributes**

**3.5.1 Reliability**

The program should be able to work reliably with no issues of crashes from memory overload or storage issues for the time the website will be running.

**3.5.2 Availability**

All documentation and source code can be found on the github repository for anyone to download. The website will be running from January to April for people to interact with.

**3.5.3 Security**

There will be no user login or input from end-users other than their influence on the production of new images by the AI so there shouldn’t be any malicious intent there. As stated before, end users on the website may intentionally answer incorrectly on the aesthetic quality of the art. We choose to recognize this as subjective preference, which aligns with the goal of the project, so the concern is minimal.

**3.5.4 Maintainability**

Any person can download the repository and/or pull/push to this repository so anyone can update or keep the code up-to-date.

**3.5.5 Portability**

The program should be able to be installed on any Windows 10/11 or Windows Server device and run as a website by pulling this repository. Most browsers whether desktop or mobile should be able to visit the website.

**Key Personnel information**

Nathanael L. Mann (Project Manager)

Email: mannnl@iu.edu

Git: IAMNameless9t9

Mark Bridgewater (Primary Front End Developer)

Email: mobridge@iu.edu

Git: MarkoBridgewater

Joshua Carrier (Primary Back End Developer)

Email: carriejr@iu.edu

Git: jcarrier0026