**“Arti-GaN”**

FA21 CSCI-P445 Capstone RF 4

Nathanael L. Mann, Mark Bridgewater, Joshua Carrier

Sponsor: Chris Sexton

26 October 2021

**Software Detailed Design**

**1 Data Design**

**1.1 Images (In Program)**

Images are interpreted as a three-dimensional array. Each entry in the array corresponds to the X (vertical) and Y (horizontal) coordinates (starting from the top-left corner of the image) of the pixel and contains a sub-array containing the integer values of the pixel’s RGB values. This results in an image being represented as a numpy.array[y][x][3].

**1.2 Training Sets of Images**

All images in any given training set must be formatted to the same size as each other images in the directory. The specific size of the images in a training set is irrelevant so long as they are all the same. Training sets need to be stored in directories on the same level as the main program and can be passed in on the command line to generate a GaN and output image. Images should be in png format.

**1.3 Output Images**

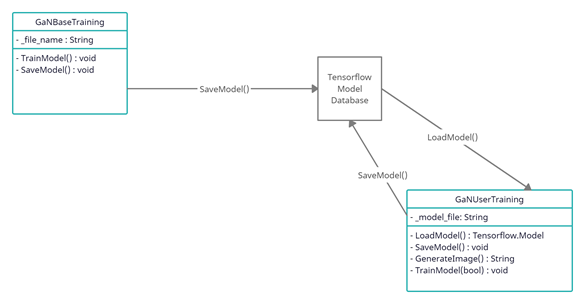
An output image will be created and stored at the same level as the main program and will be named “(training\_directory\_name)” + “\_output” Output images are stored as .pngs.

**1.4 Data Models**

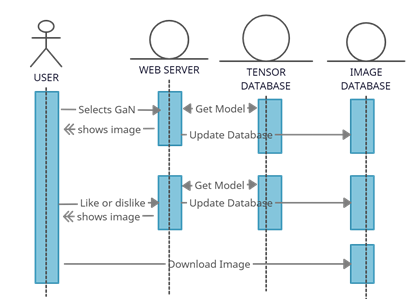
The models are saved to a file using TensorFlow.model.save() and will be reloaded using TensorFlow.model.load().

**2 Architecture Design**

**2.1 Backend Training Model**



**2.2 Website model**



**3 Interface Design**

The program uses a website as the external interface. The navigation bar will contain options to navigate to the front page, the about page, and a gallery of images generated by the various GaN’s, and to each individual GaN’s page.

**3.1 Front Page**

The front page includes all data models for the GaNs (and their name) as well as a navigation button that will prompt the website to make a call to the backend with the selected GaN to generate a new image and display it on the webpage.

**3.2 GaN’s Personal Pages**

The GaN’s most recent output image will be displayed at the center of the webpage. Below that, there is a download button for users to save images, and in addition, a ‘like/dislike’ button which will trigger the backend data model to update the output image based on the user’s input.

**3.2.1 GaN Galleries**

On each GaN’s respective page will be a link to view an image gallery of some of the GaN’s previously generated images.

**3.3 About Page**

The about page will contain information about the team and the project.

**4 Procedural Design**

**4.1 GaN Generation/Training**

* Load the dataset into a NumPy array
* Format the images for entry into our neural network
* Create a GaN:
  + Create a discriminative neural network to check if an image is what we’re looking for
  + Create a generative neural network to generate images to try and fool the discriminator
* Train model with a 70/30 split between training and testing data
* Save model in a file and end program

**4.2 Voting on Images**

* Load the model
* Have generator load an image and send it to the user for feedback
* Once feedback has been received, adjust the model accordingly and generate another image to repeat this process
* Once it’s told to stop generating images, save the model and end the program

**Breakdown of individual contributions**

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

**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