Photo-Realistic Image Synthesis from Text Descriptions

Ryan Cain
Gabriel Kralik
Campbell Munson

Follow this and additional works at: https://scholarcommons.scu.edu/cseng_senior

Part of the Computer Engineering Commons
I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Ryan Cain
Gabriel Kralik
Campbell Munson

ENTITLED

Photo-Realistic Image Synthesis from Text Descriptions

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Thesis Advisor

N. Ling

Department Chair
Photo-Realistic Image Synthesis from Text Descriptions

by

Ryan Cain
Gabriel Kralik
Campbell Munson

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 11, 2020
Deep convolutional generative adversarial networks (DCGANs) have proven capable at generating diverse, photorealistic images of human faces, but it is difficult and often time-consuming to choose what kind of image these generative adversarial networks (GANs) produce. We create a simple, intuitive web application through which users may write a description of a human face in plain text and generate photos that appear to match the given description. In this paper, we show how text can be used to direct the output of a conditional GAN with a DCGAN architecture. While our images did somewhat resemble human faces, they often had artifacts that prevented them from looking like photographs, but they did generally match the input text descriptions. This was likely due to our using a relatively simple DCGAN architecture and trained for a relatively short amount of time. To improve image quality, we recommend experimenting with more advanced GAN architectures trained for a longer amount of time. While a more advanced GAN would likely take longer to train, more advanced models would likely achieve more photorealistic results. To make the model more robust to natural language input, we recommend implementing a text-embedding model to encode the text data that is passed to the GAN.
# Table of Contents

1 Introduction .................................................. 1

2 Background .................................................. 3
   2.1 Introduction to Machine Learning .......................... 3
   2.2 Introduction to Neural Networks .......................... 3
   2.3 Generative Adversarial Networks .......................... 5
   2.4 Deep Convolutional GANs .................................. 5
   2.5 Conditional GANs ......................................... 6

3 Requirements .................................................. 7
   3.1 Functional ............................................... 7
   3.2 Nonfunctional ........................................... 7
   3.3 Constraints ............................................... 7

4 Use Cases ..................................................... 8
   4.1 Write Facial Description .................................... 8
   4.2 Generate Faces ........................................... 8

5 Activity Diagram ............................................. 10

6 Conceptual Model ........................................... 11
   6.1 How The Original Design Changed .......................... 11

7 Technologies Used ............................................ 13

8 Architectural Diagram ....................................... 14
   8.1 Description ............................................... 14

9 Design Rational .............................................. 15
   9.1 User Interface ............................................. 15
   9.2 Technologies Used ......................................... 15

10 Description of System Implemented ......................... 17

11 Test Results ................................................. 18
   11.1 Model Performance ......................................... 18
      11.1.1 Black Hair and Smiling ................................ 18
      11.1.2 Pale and Young ....................................... 19
      11.1.3 Bald and Woman ....................................... 19
   11.2 User Interface ........................................... 19

12 Risk Analysis ................................................ 20
   12.1 Risk Analysis Table ....................................... 20
13 Development Timeline
14 Suggested Changes
15 Lessons Learned
16 Issues and Constraints
  16.1 Web Service
  16.2 Model
17 Ethics
  17.1 Senior Design and the Ethical Justification of the Project
  17.2 Senior Design Project and the Virtues of a Good Engineer
  17.3 Senior Design Project and Safety, Risk, the Public, and Informed Consent
18 Future Work
A Installation Guide
  A.1 Obtaining the code
  A.2 Creating an AWS instance
  A.3 Accessing the instance
  A.4 Setting up the instance to run the website
B User Manual
  B.1 Running the app and accessing the website
  B.2 Using the website
C Maintenance Guide
  C.1 Removing old session folders
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Neuron</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>Neural Network</td>
<td>4</td>
</tr>
<tr>
<td>4.1</td>
<td>Use Case Diagram</td>
<td>8</td>
</tr>
<tr>
<td>5.1</td>
<td>Activity Diagram</td>
<td>10</td>
</tr>
<tr>
<td>6.1</td>
<td>Website conceptual model</td>
<td>11</td>
</tr>
<tr>
<td>6.2</td>
<td>Completed website</td>
<td>12</td>
</tr>
<tr>
<td>8.1</td>
<td>Data-Centric Architecture Diagram</td>
<td>14</td>
</tr>
<tr>
<td>11.1</td>
<td>Black Hair and Smiling</td>
<td>19</td>
</tr>
<tr>
<td>11.2</td>
<td>Young and Pale</td>
<td>19</td>
</tr>
<tr>
<td>11.3</td>
<td>Bald and Woman</td>
<td>19</td>
</tr>
<tr>
<td>13.1</td>
<td>Development Timeline</td>
<td>21</td>
</tr>
<tr>
<td>A.1</td>
<td>Security Group</td>
<td>30</td>
</tr>
<tr>
<td>A.2</td>
<td>PuTTYgen Window</td>
<td>31</td>
</tr>
<tr>
<td>A.3</td>
<td>PuTTY Session Window</td>
<td>32</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In recent years, advances in generative machine learning techniques, in particular with image generation using generative adversarial networks (GANs), have shown impressive results. While exciting, many of these results have little application beyond art or novelty. Similarly, there have been great improvements in computer natural language understanding through the use of deep recurrent neural networks. We plan to take advantage of these breakthroughs in image generation and natural language understanding by combining them to take as input a textual description of a face and output a set of photo-realistic image interpretations of the text.

The initial inspiration for our system was that it may be used to help identify suspects in a police investigation. Often a victim or eyewitness description of the suspect will be relayed to a sketch artist who will then synthesize that information to draw a sketch of what the suspect looks like. However, this is time consuming for the eyewitnesses and artists. A software system built around a model trained to generate an array of images based on unstructured text input would enable witnesses to more easily relay their description of the suspect and police to more easily begin their process of searching for or identifying the suspect. In addition to increased speed of suspect facial generation, other benefits of a software system include ameliorated costs to police departments and the possibility to generate an array of interpretations of the description rather than relying on a single interpretation from a single artist. Even with multiple interpretations from multiple artists, a software system may more cheaply (with regard to time and other resources) generate more interpretations for police to analyze. Our model could potentially be more accurate than the sketch. It is also easier to recognize a person based on a digital image than a drawing. One may additionally imagine non-criminal scenarios in which one wishes to search for an individual only by their facial features. In addition to person identification and search, a range of scenarios (artistic and otherwise) photo-realistic face generation based off text input may add value to people’s lives.

We evaluate the success of this project using various key performance indicators: photo-realistic accuracy and clarity of the image and accessibility of the system. Accuracy is based entirely on the system we will build, and is the main motivation behind our project. Clarity will be a secondary objective that measures our improvements of previous
iterations. Accessibility will be based on how we can demonstrate our project by making our research understandable to the public in addition to how easily one may use the system to generate images.
Chapter 2

Background

2.1 Introduction to Machine Learning

Machine learning sits at the intersection of computer science and mathematics, heavily involving probability, statistics, calculus, and linear algebra. Deep learning, while its true definition is a topic of some debate among researchers in the field, can be generally and usefully understood as a subfield of machine learning that uses things known as artificial neural networks (ANNs, neural networks, networks) with many hidden layers. The theory and mathematics behind deep learning have been around for many decades, but have seen incredible success in the past 15 years due to a drastic increase in available data and computer processing power. Because much of the work for our project involves recent advances in deep learning, I will briefly go over some of the basics of machine learning (hopefully enough so that one who is new to the field will be able to follow along from a high-level) but recommend one learn these basics elsewhere as there are many, better resources for doing so.

2.2 Introduction to Neural Networks

What makes neural networks effective is that they can essentially approximate any function. This is known as the universal approximation theorem (2). A neural network is composed of layers; those layers are composed of neurons which take many numeric inputs and yield a single output.

Specifically, they take inputs $x_1 \ldots x_m$, multiply them by a corresponding weight (also referred to in literature as parameter) from $w_1 \ldots w_2$, sum those products and a bias term, then input that result to an activation function. The activation function typically takes an input and squishes it to be between two values (0 and 1, or -1 and 1 are commonly bounds). However, it may just perform a transformation on the input like outputting 0 if the input is less than 0 or the input itself if it is greater than 0; this specific kind of activation function is called a rectified linear activation function and is equal to the function $f(x) = \max(0,x)$. Thus a neuron looks like $y = (i=0\text{maxi } w_i) + b$. Neurons in a network do not need to all use the same activation function, but typically do; this is also true for neurons in a particular layer.

A layer in a neural network is a group of neurons whose input are either the network’s input if it is the first layer
or the outputs of the previous layer otherwise. The outputs of the neurons in a layer are the outputs of the network if it is the last layer or the inputs to the next layer otherwise.

To make a prediction, one gives an input to the network and propagates forward the results of the neurons layer by layer until the final layer outputs your answer. The weights are updated through what is known as backpropagation (backprop) which involves computing the gradient (or derivative) of the loss function (a function that tells us how close or far from the true value the network’s guess at the answer is) with respect to the weights of the network in a computationally efficient manner. This is the “learning” in machine and deep learning. The inputs to the network are often referred to as features, but features can also refer to the way the final weights of a trained network or the outputs of the weights given input features. The network is thus said to learn the features of a sample. For example, in image classification or object recognition tasks, a learned “feature” of an effective network may be the presence of a human
2.3 Generative Adversarial Networks

Generative adversarial networks (GANs) are relatively recent innovations in the field of deep learning. In short, two neural networks play a minimax game in which one model, the discriminative model, learns to discern whether a sample is from the training data distribution or generated by the second mode, the generative model that learns to generate samples of a distribution (5). The “competition” between the models should lead towards images generated by the generative model that are indistinguishable from samples of the training data. Goodfellow et. al. show many current methods of generating samples of a distribution have limitations in either computational methods or variability in types of inputs (5). The authors argue that the ability to use deep neural networks enables taking advantage of advancements made with deep neural network training in general. For example, generative adversarial networks are very effective at generating images. One of the main motivating factors is that, while deep learning had seen great progress in classification problems, generative problems had seen relatively limited progress. We are choosing to use generative adversarial networks because they are so powerful and there has been much improvement on them since the original paper in 2014.

2.4 Deep Convolutional GANs

Another important and popular improvement is in using convolutional layers in GANs. Briefly, convolutional layers are widely used for tasks involving images because they are more efficient and effective at detecting translation-invariant patterns in images; an eye is an eye no matter where in the image it appears. Radford et. al. show that deep convolutional GANs (DCGANs) can be more effective and more easily trained for image generation (11). DCGANs are also readily modifiable for our problem of conditioned image generation because we can simply input the condition (the encoded text description of the desired image) to the generator and discriminator networks. Additionally, Radford et. al. give a lot of advice on effectively training a DCGAN to generate images; these are very useful as GAN training in general is very sensitive and difficult (11).

Because of their effectiveness, we use a DCGAN architecture for our model; however, we want to pursue a robust model and method of training. Karras et. al. in 2017 use a technique of progressively growing GANs during training to improve image quality (8). In 2017, they achieved state-of-the-art image quality on a dataset of human faces. The authors demonstrate a new training methodology for generative adversarial networks (GANs) in which the generator and the discriminator are trained progressively, that is, they begin at a low resolution and as small models, train those until convergence, and slowly add layers and increase the resolution. They show that this methodology results in increased training speed and much more stabilized training. The images produced also have greatly increased quality.
The main argument behind this method from a theoretical position is that smaller, converged, models make better initial layers for larger models.

Similarly, Karras et. al. in 2019 released a new model that also used progressive training that resulted in even higher quality images than their 2017 model (7). This model and its results went viral in the form of a website https://www.thispersondoesnotexist.com/. Their proposed generator has also proven robust in generalizing to datasets other than human faces. One of the main difficulties in using this model, which is commonly referred to as StyleGAN, is that it is very computationally expensive to train. It is also a rather complex model and modifying it to become a conditional GAN (CGAN) would prove very difficult. However, it would likely be able to produce very high quality images if we managed to get it working.

2.5 Conditional GANs

As previously mentioned, our problem is not simply one of generating images of faces, but of generating images based on a text input. This calls for using CGANs. Goodfellow et. al. mention future work involving conditional GANs where the conditioning vector is part of the input to both the generator and the discriminator (5). With advances in natural language understanding (NLU), neural networks have been able to perform better on language tasks than ever before. This means that the features learned by a NLU model may be much more useful than an encoded word vector. Reed et. al. do this in their 2016 paper on text to image synthesis using GANs. They use a pre-trained NLU model to get a feature vector from the text data and then input it to the beginning of the generator and after the image has been mostly processed in the discriminator of a typical DCGAN. They show this is effective on popular bird and flower datasets. Nasir et. al. in their 2019 paper show that this same methodology can be applied to human faces (10). Because the face dataset does not contain raw text, the authors write an algorithm to deterministically turn a set of binary descriptors like (female, has brown hair, chubby face) into a sentence or series of sentences that seem like natural language. Because there simply does not exist a large dataset of images of human faces with corresponding text descriptions, this is the next-best method to try to test CGANs in generating images of human faces. We do something similar in order to properly train our model.
Chapter 3

Requirements

3.1 Functional

- Photo-Realistic Face Generation (Critical)
- Intuitive User Interface (Critical)

These are the main functional requirements because without them, we wouldn’t be meeting the client’s specifications. The GAN needs to be able to generate photo-realistic faces based on user provided descriptions, as this is the main goal of the project. The website will have an intuitive interface that will make it easy for users to interact with our model and generate images at their discretion.

3.2 Nonfunctional

- Recognizably Human Generated Faces (Recommended)
- Faces Appear Derived from Text Description (Critical)

We intend to have our GAN generate faces that look like human faces. The more similar our generated faces are to actual human faces, the more we will have succeeded in this requirement. We want to ensure that the faces our GAN generates are reasonably similar to the description provided by the user.

3.3 Constraints

- Time
- Computing Resources

Time will be a constraint in terms of development of the GAN architecture and training the GAN.
Chapter 4

Use Cases

4.1 Write Facial Description

Users will be able to write a description of the face they want to generate in a text box on the website. This description would include major facial features such as hair color, eye color and skin tone.

4.2 Generate Faces

Once the user has written the description of the face they want to generate, they will click the "Generate Faces" button. This will pass the description to our GAN which will generate faces based on the description, and pass them back to
the website to be displayed to the user.
Chapter 5

Activity Diagram

Figure 5.1: Activity Diagram
Chapter 6
Conceptual Model

6.1 How The Original Design Changed

The above image is the original design we had for our website. However, as the year progressed and we ended up switching from 128x128 images to 64x64 images, we thought it smart to include more images on the page than we originally intended. We also ended up going with a larger text box to make it easier to write and see longer descriptions. Other than those changes, the final version of the website ended up very similar to our conceptual model.
Photorealistic Image Synthesis from Text Descriptions

Enter a description of the face you’d like to generate

Generate Faces

Figure 6.2: Completed website
Chapter 7

Technologies Used

In this section, we will list the technologies we used in developing the project

- Python
- Tensorflow-Keras
- Flask
- HTML and CSS
- JavaScript
- Google Cloud Platform
- GitHub
- AWS
Chapter 8

Architectural Diagram

8.1 Description

Our system uses a data-centric architecture that has the users communicating with a web tier. This web tier passes text descriptions provided by the user to the GAN, which generates faces and sends them back to the web tier to be displayed to the user.
Chapter 9

Design Rational

9.1 User Interface

We approach the interface with ease-of-use and effectiveness in mind. The user will input text that is then passed to the GAN we have developed and trained. The GAN will then process the text input and output group of images. We will then display the images to the web-page for the user to see.

9.2 Technologies Used

- As a team, we have some experience with Python. Python is very flexible because of the frameworks that make the development of websites and applications quick and easy. There are also many deep learning and data-processing frameworks available in Python, making training of the GAN easier.

- Keras is a high-level Python API for deep learning model development, training, and deployment, and we will be using the Tensorflow back-end for it. Keras having a Python API means it easily interacts with our web-based components as well. We also have more experience with Keras compared to other frameworks.

- Flask is a Python oriented web framework. It simplifies many aspects of web development and its integration with WTForms and Jinja2 allows for simpler creation of HTML via templates

- HTML and CSS will help us develop the front-end of our website and make the user interface more aesthetically pleasing and intuitive.

- JavaScript is used as a way of dynamically modifying our HTML code. This is necessary because pictures are not displayed on the website until the user generates faces, so we need to use JavaScript to change the web page once those pictures have been created in order to display them.

- Google Cloud Platform gave us access to cloud computing resources which aided us in working with and training the various models we attempted to use for our project.
• GitHub was used to aid with version control for the project.

• For deploying the website, we used AWS because it provided enough storage and RAM for the model to be run. Originally we intended to use a t2.micro instance, but this proved to have insufficient memory to run the model, so we upgraded to a t2.medium instance.
Chapter 10

Description of System Implemented

For our project, we created a web service using Flask that accepts a description of a human face. The user then clicks the "Generate Faces" button and the Flask app passes the description to a bash script that then calls a Python 2 program that then passes the description to some code we wrote that parses the description for certain attributes. The reason we must parse the description is because the model we use, a cWGAN model developed by Cameron Fabbri, only accepts binary attributes as its conditional input and out of the 40 binary attributes in the dataset, only 9 of them are valid conditions for this model (4). After parsing the description for these attributes and creating vectors that indicate which attributes were in the description, we pass these vectors to the cWGAN model which interpolates over these attribute vectors to create nine images. We call the model to generate pictures three times for each description, so we then end up with 27 images each time the user chooses to generate faces. It is important to note that we use Flask sessions in order to create a session folder each time a user generates faces; this means that multiple users working with the website at once will only see the pictures they generated and not those of the other users. The model saves the generated images into the correct session folder which was also passed through the bash script to the model file and when the web page is reloaded, we use JavaScript to check the session folder for images. If images are present, we display them, but if the user hasn’t generated pictures yet, then we display nothing. The reason we need to call a bash script from our Flask app which then calls a Python 2 file that calls the correct function from the model is that our Flask app is written in Python 3, while all the files from the cWGAN model are written in Python 2. Rewriting all the cWGAN source files in Python 3 would’ve been rather impractical, so we chose this solution instead.
Chapter 11

Test Results

We tested functional and non-functional requirements: that the model generates photo-realistic images of human faces that appear derived from text description input by the user and that the user interface is simple to use.

11.1 Model Performance

Because there is no single loss function for a GAN, evaluating a GAN’s performance often means evaluating the generator’s ability to produce convincing fakes samples of the training distribution. With image generation, this is an incredibly difficult task for a computer to do, but relatively easy for humans to do. This task is even more difficult when it comes to conditional image generation because the pictures not only need to be convincing, but they must also appear to match the condition. This is rather difficult when it comes to our project because the same description may describe different faces, and the same face could appear as the result of different descriptions. Because the problem is in large part phenomenological, that is, it very heavily relies on our own subjective experience in ways that are difficult to quantify, there are no metrics better than manual, human, visual inspection. Because images produced by a GAN tend to be similar in terms of their qualitative appearance, we can rely on inspecting a sample of images during the training process to indicate whether training is successful. However, some combinations of attribute input, or the condition, may produce better results than others. This is likely due to much imbalance in the distribution of features of the dataset. The following subsection shows the prevalence of some selected attribute combinations with accompanying sample images. For reference, our dataset contains 202599 images. Rather than show results enumerating through all the various permutations of attribute inputs our system takes in, we will show a few cases to demonstrate how prevalence in the dataset can affect generated image quality.

11.1.1 Black Hair and Smiling

Images containing a person who is smiling and has black hair make up 23,258 images or 11.48% of our dataset. These images are relatively high in quality.
11.1.2 Pale and Young

Images containing a person who is young and has pale skin make up 7,479 images or 3.69% of our dataset. Consequently these images are lower in quality.

11.1.3 Bald and Woman

Images containing a person who is a woman and is bald make up 17 images or 0.0084% of our dataset. Consequently, these images are much lower in quality and some do not even appear to demonstrate the qualities which were given to the system.

11.2 User Interface

The user interface does not need do much, so we can focus on making it visually pleasing and intuitive. We tested this by allowing users unfamiliar with the project navigate the website and use it, and they found it easy to use.
# Chapter 12

## Risk Analysis

### 12.1 Risk Analysis Table

<table>
<thead>
<tr>
<th>Risk</th>
<th>Consequences</th>
<th>Probability</th>
<th>Severity</th>
<th>Impact</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lack of Knowledge</td>
<td>System incomplete, delayed development</td>
<td>0.75</td>
<td>8</td>
<td>6</td>
<td>Do sufficient research, so we understand exactly what to do</td>
</tr>
<tr>
<td>Project Scope</td>
<td>Project not complete on time, or missing features</td>
<td>0.5</td>
<td>10</td>
<td>5</td>
<td>Focus on critical requirements first</td>
</tr>
<tr>
<td>Time</td>
<td>Project not complete on time</td>
<td>0.4</td>
<td>10</td>
<td>4</td>
<td>Follow our development timeline and stay focused</td>
</tr>
<tr>
<td>Bugs</td>
<td>Delayed development process</td>
<td>0.99</td>
<td>4</td>
<td>3.96</td>
<td>Read documentation, review code</td>
</tr>
<tr>
<td>Group Members Getting Sick</td>
<td>Delayed development</td>
<td>0.1</td>
<td>3.5</td>
<td>0.35</td>
<td>Try to stay healthy, work while sick</td>
</tr>
</tbody>
</table>
Chapter 13

Development Timeline

Figure 13.1: Development Timeline
Chapter 14

Suggested Changes

• One beneficial change for our project would be changing the way we generate sentences for the model to train on. The CelebA dataset we used has around 200k faces, each with 40 binary attributes. The original plan for the project was to use text embeddings for each image to train the model so that users could type descriptions in a sentence format on the website. Currently, the model we use only accepts binary attributes, however one of the ideas we have for future work on the project is changing the model so that it accepts text embeddings instead. Given that this is a feature we initially planned to implement, we wrote a program that parsed each vector of binary attributes and converted them to sets of sentences describing each face, so that the model could train on those sentences. Currently, the way we convert the binary attributes to sentences results in each sentence following a similar template. It would be better if there was some randomness introduced into the sentence generation so that the sentences the model is training on would be more varied from a stylistic perspective, which would be more similar to the real sentences users would be typing.

• Another positive change would be periodically deleting session folders. The best way to do this would likely be creating a script that runs in the background on the instance and deletes session folders periodically so that the instance isn’t wasting storage space unnecessarily.

• The website could also benefit from changing how session folders are generated. Currently, anytime a user generates pictures, a new session folder is created to contain them. Ideally a session folder would only be created each time a user goes onto the website, and each new set of pictures they generated while using the site would replace the last set in the same folder.

• It would also be helpful if the model we used was written in Python 3 instead of Python 2. This would make it easier to call functions in the model.
Chapter 15

Lessons Learned

Because of the great breadth in machine learning, there is a great deal of research and experimentation being done constantly. This means that it is sometimes difficult to know what work has been done already. In our research for the project, we found many examples of generating images based on text descriptions but with datasets that we were not using. This led us to testing many different architectures and implementations without much success. Only after various failures did we find an implementation that worked for the dataset we were using. Despite the difficulty, we still should have done more research before committing time to time-consuming experimentation whose success we did not have great reason to believe in.
Chapter 16

Issues and Constraints

When working on our project, we came across a wide variety of constraints and issues that we had to overcome. There are two categories for these issues; website related issues and model based issues.

16.1 Web Service

One constraint we have is that our flask server only supports having a single instance running at a time. This means that our web application itself is not very scalable, and any attempts to make it more so would require heavy modification of our code. Another constraint is that there is no way to save the generated images that one receives from the website, and once they are overwritten they are lost from the user’s perspective. Another constraint is that the web service manually parses user input for targeted attributes instead of sending the user’s description to the model directly. This constraint will be expanded on more in the following section.

16.2 Model

The first and biggest constraint is that the model only identifies and uses 9 separate attributes that the model identifies. These attributes are as follows; bald, bangs, black hair, blond hair, eyeglasses, heavy makeup, male, pale skin and smiling. For reference, the CelebA dataset contains 40 different attributes, as well as information on how many photos contain these attributes (9). This hampers what the model can conditionally generate, and is not ideal for our website. The limited amount of attributes also forces our website to manually parse the user input instead of feeding it directly into the model. Another large constraint is the poor quality of the weights in the model, which leads to poor quality images. These images often contained warped and inaccurate looking attributes. These qualities are caused by the lack of training the model has, and the quality can be increased by using enough computing resources to further train the model.
Chapter 17

Ethics

17.1 Senior Design and the Ethical Justification of the Project

Generative adversarial networks (GANs) are a recent development in the field of machine learning, so they have seen relatively little use beyond testing new methods or neural network architectures. This aligns with part of purpose of this project to advance scientific research with respect to GANs. GAN research has often been oriented towards aesthetic ends as producing images is often the end goal of much GAN research. While the arts and aesthetics have value in improving the quality of everyday life, GANs also have the side effect of enabling more people to express themselves creatively, making art more accessible. For example, Artbreeder is an online tool that allows anyone to create new works of art from their own images. To this end, GANs can democratize the creation of art and allow more people to cultivate their creative abilities. Our project working on conditional image generation extends this further by providing greater ability to control the outputs without necessarily investigating the inner workings of GANs. As GANs are able to create higher quality images and their outputs become more customizable, we will likely see them gain popularity for commercial use. In fact, there are already some commercial applications for generative machine learning models: Waifulabs has built an interface through which one may interact with a GAN to generate and make certain modifications to a portrait in a popular animation style and once satisfied, you can request a pillow or poster with your character’s portrait on it. There have also been experiments in interpolating images between frames of animation to increase the frame rate of the original animation and upscaling images in general to increase their resolution. These could lead to higher quality end products at lower costs for both artists and their patrons. With specific regard to our project, generating images of human faces based on text descriptions still has plenty of artistic applications. While other kinds of generative deep learning has the potential for deepfakes, imposing an image of a different person over the original person in an image or video, our project does not have any direct applications to this.
17.2 Senior Design Project and the Virtues of a Good Engineer

Machine learning is a highly technical field involving advanced mathematics and computer science. Working on this project, we have cultivated much deep technical knowledge of machine learning. Because of the technical depth of the field, we also need to be good communicators to those who are not familiar with the field or technically inclined, as they often have stake in the technology and how it is used. Machine learning is a young field with a great deal of breadth. New research is constantly published in the many sub-fields of machine learning. With respect to GANs, especially conditional GANs, they can almost always be important to note. For example, research in natural language processing is important to conditional GANs as the goal of many may be to use text as the condition. Because of how much breadth of knowledge is also required to work in the field, we as a team had to work together to effectively search for research relevant to our project that has already been done. Machine learning is also a highly interdisciplinary and applicable field, affecting many outside of those directly involved in research. This requires a great deal of technosocial sensitivity from us who have worked on the project. This directly influenced what kind of project we wanted to work on. We as a team had more affinity for the idea of contributing in an almost entirely positive manner to the world, therefore we decided to choose a project involving generative adversarial networks since they have great creative possibilities.

17.3 Senior Design Project and Safety, Risk, the Public, and Informed Consent

As mentioned earlier, we avoid many of the typical ethical pitfalls of machine learning projects in the formulation of project: our work is not really susceptible to falling into the ethical pitfalls of deepfakes, and even if it were, the faces we generate are of non-existent persons, and even if our system were to generate a face that bore the likeness of an existing human, it would very likely be that of a celebrity whose face already is already easily searchable and exists in higher quality images on the internet. This touches on one of the most important aspects of many machine learning projects: the dataset. Because we use the CelebA dataset, we use pictures from people who have consented to have their photos taken (9). We also are using the dataset in an academic context, keeping to the terms of those who make the dataset available. As we demonstrated in the testing section, because we rely on this one dataset, our model will reflect its distributions in the images it produces. The only way to get around this would be to add to a new dataset and use that to train our model as well. As mentioned earlier, image generation and conditional image generation have the potential help current artists; however, it also poses a threat to reduce the demand for artists. As image generation techniques produce higher quality images and become more configurable, the artist’s role may change from creating a work from scratch to using results from a system like ours as a base for their own work. With more improvement, their role may change to using a system similar to ours in order to create the desired result entirely. Once systems
like these become even more intuitive to use, the artist may not even be necessary to create the desired result. These scenarios, while possible and the subject of much contemporary discourse, are still unlikely to manifest in the near future. Even if a system becomes good enough and intuitive enough that they become preferable to financing an artist, the system based on contemporary machine learning models may only produce works in styles of art already created and would not be able to use new styles without new artists contributing to it. These risks are also resultant of general advancement of research in GANs rather than directly from our project. While we mentioned that the system may be used to aid in police investigations, this was mostly inspiration and metaphor to explain the mechanics of the system. While it may be true that it could produce images in the likeness of existent humans, it would not be likely. However, we can envision a use for our system where the system produces a base output from which the user may modify it in attempts to create an image in the likeness of an existent person. Aiding state police carries with its own ethical baggage of dubious merit, but it is not unreasonable to envision that this could also be used by private investigators or individuals who are simply looking for a lost loved one. What we can say is that there would still likely be a person like the sketch artist who instead uses our system in such a way. The effectiveness of a system based on a model like ours is difficult to say because that requires much careful testing.
Chapter 18

Future Work

The most obvious choice for future work would be the implementation of Skip-Thoughts vectors as the conditional input to the model. The Skip-Thoughts model would encode the user input into a vector representation, and each description would be considered by the discriminator when judging whether or not an image/description pair is real or fake. Because there have been developments with newer, more advanced language models like BERT by Devlin et. al., it would be prudent to try those in new experiments (3). To improve the quality of generated images, it would be prudent to experiment with other GAN models like those of StyleGAN and StyleGAN2 by Karras et. al. and BigGAN by Brock et. al (7; 5; 1). Trying various combinations of new language models and new GAN models would likely be useful. Because of the fine-grained nature of the problem, it would likely help to investigate methods used by Radford et. al. (8). Additionally, in turning the attributes into sentences, changing the order of the attributes in the deterministically generated sentences would better reflect patterns in natural language descriptions and may make the model more robust to human language input.
Bibliography


Appendix A

Installation Guide

A.1 Obtaining the code

Download the zip file containing our project from google drive: photorealistic_images_from_text_descriptions.zip

A.2 Creating an AWS instance

- First, select a 64bit “Ubuntu Server 18.04 LTS (HVM), SSD Volume Type” as your Amazon Machine Image
- Select a t2.medium instance as the instance type. Note that this instance costs money to run, but we chose it because the free tier instance did not have enough memory to actually run the model and generate pictures. Depending on how many people will be using the website at once, an instance with more storage may be required
- Choose 16GB of storage
- Add a custom TCP rule to the security group on port 5000 with ”anywhere” as the source.
- Then click ”review and launch.” The security group section should look like this:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>0.0.0.0/0</td>
</tr>
<tr>
<td>Custom TCP Rule</td>
<td>TCP</td>
<td>5000</td>
<td>0.0.0.0/0</td>
</tr>
<tr>
<td>Custom TCP Rule</td>
<td>TCP</td>
<td>5000</td>
<td>::/0</td>
</tr>
</tbody>
</table>

Figure A.1: Security Group

- After clicking ”launch” create a new key pair and download it

A.3 Accessing the instance

- In order to remotely access the instance via SSH, download PuTTY.
- We need to convert our private key from a .pem file to a .ppk file. To do this, launch PuTTYgen
Click "Load" to load your .pem file. When searching for the file, search for all file types and select the private key file.

After the private key has been loaded the PuTTYgen window should look like this:

Click "Save private key".

Launch PuTTY.

For the host name, enter ubuntu@, followed by the instance's public DNS which can be located in the instance description.

Under "Category" go to "Connection", then "SSH", then "Auth" and browse and select the .ppk file as the "private key file for authentication".

Next, go back to the "Session" category and under "Saved Sessions", type a name for the session and save it.

The session window should look like this:

Now, click "open" to remotely access the instance, and click "yes" on the warning message that appears.
A.4 Setting up the instance to run the website

- Now, login to the instance through putty and navigate to the following directory. One way to check which directory you are in is to type "pwd" (without the quotes)
  
  /home/ubuntu/

- Update apt-get
  
  ```bash
  sudo apt-get update
  ```

- Install pip for Python 3
  
  ```bash
  sudo apt-get install python3-pip
  ```

- Install Python 2 and pip2
  
  ```bash
  sudo apt install python-minimal
  sudo apt install python-pip
  ```

- Install virtualenv
  
  ```bash
  pip3 install virtualenv
  ```
- Install unzip
  
  ```
  sudo apt install unzip
  ```

- Using a tool like WinSCP, transfer the project zip folder to the instance. To do this, download WinSCP or your preferred tool and enter the public DNS as the host name. Enter "ubuntu" as the username. For the password, select "advanced", go to "authentication" and click the three dots under "Private Key File" to select your private key. Then, login and place the project zip folder in the following directory:

  ```
  /home/ubuntu/
  ```

- Unzip the project folder

  ```
  unzip photorealistic_image_synthesis_from_text_descriptions.zip
  ```

- Enter the project folder

  ```
  cd photorealistic_image_synthesis_from_text_descriptions
  ```

- Create a virtual environment

  ```
  python3 -m virtualenv venv
  ```

- Activate the virtual environment

  ```
  . venv/bin/activate
  ```

- Install the Python 3 and Python 2 library

  ```
  pip3 install -r python3_requirements.txt
  pip2 install -r python2_requirements.txt
  ```

- Download the zip folder containing the cWGAN model created by Cameron Fabbri from his Github page: https://github.com/cameronfabbri/cWGANs

- Using WinSCP or a similar tool, put the CWGAN zip folder on the instance in the following directory

  ```
  /home/ubuntu/photorealistic_image_synthesis_from_text_descriptions
  ```

- Go back to your PuTTY session and make sure you are in the following directory by typing "pwd" (without the quotes)

  ```
  /home/ubuntu/photorealistic_image_synthesis_from_text_descriptions
  ```

- Unzip the CWGAN folder

  ```
  unzip cWGANs-master.zip
  ```

- Now, we need to replace certain files from the model with ones we modified. First we will replace the file "interpolate.py" with our modified version

  ```
  rm cWGANs-master/celeba/interpolate.py
  ```

- Now, copy the version of "interpolate.py" over to the directory where the other version used to be. We will also do this for two other files

  ```
  cp interpolate.py cWGANs-master/celeba
  cp caller.py cWGANs-master/celeba
  cp helper_script.sh cWGANs-master/celeba
  ```
• Now we need to copy the checkpoints for the model into the correct directory
  
  \texttt{cp \ -r \ checkpoints/ \ cWGANs-master/celeba/}

• Remove the other copies of these files and directories
  
  \texttt{rm \ interpolate.py}
  \texttt{rm \ caller.py}
  \texttt{rm \ helper_script.sh}
  \texttt{rm \ -rf \ checkpoints/}

• Install python-tk and libsm6
  
  \texttt{sudo \ apt-get \ install \ python-tk}
  \texttt{sudo \ apt-get \ install \ libsm6 \ libxrender1 \ libfontconfig1}
Appendix B

User Manual

B.1 Running the app and accessing the website

- Ensure the current directory is as follows
  ```
  /home/ubuntu/photorealistic_image_synthesis_from_text_descriptions
  ```
- Ensure the virtual environment is active
  ```
  .venv/bin/activate
  ```
- Run the app
  ```
  python3 app.py
  ```
- Access the website via a web browser using the public DNS for the instance and port 5000. The URL should look like this (replace the public DNS listed with your own)
  ```
  ec2-3-17-5-29.us-east-2.compute.amazonaws.com:5000
  ```

B.2 Using the website

- In the text box, enter a description of the faces you would like to generate
- Click the "Generate Faces” button
Appendix C

Maintenance Guide

C.1 Removing old session folders

The website creates a new sessions folder with every request, so cleaning the sessions folder is required in order to ensure no file system problems appear during extended use of the website.

- Ensure the current directory is as follows
  
  ```
  cd home/ubuntu/photorealistic_images_from_text_descriptions/static
  ```

- Activate extglob (if not already enabled in bash)
  
  ```
  shopt -s extglob
  ```

- Use the following command to remove everything in the current folder except mainStyle.css, which is used for styling the website
  
  ```
  rm -rf !(*mainStyle.css)
  ```