

# CSCI 353 Final Project

## Sketch to Face Translation with Resolution Assisted GANs

Anthony Sokolov   Joe Suzuki   Jason Li

December 2020

## **Project Checklist**

Report - This document

Github - <https://github.com/AnthonySokolov/sketch-to-face-translation>

Video - <https://youtu.be/IsFKBmBwVDQ>

# Contents

<b>1. Problem Description</b>	<b>4</b>
<b>2. Team Member Roles</b>	<b>5</b>
<b>3. State-of- the-art/Related Work</b>	<b>5</b>
Generative Adversarial Networks	5
Image translation	6
Residual Networks	9
Super Resolution	9
<b>4. Approach</b>	<b>10</b>
Dataset	10
Environment	10
Realistic Sketches	10
Conditional GAN	11
CGAN Improvements	11
Cycle Consistent GAN	13
GAN + Super Resolution	13
<b>5. Experiments/Evaluation</b>	<b>15</b>
Conditional GAN	15
CGAN Improvements	18
CycleGAN	19
Conditional GAN + Super Resolution.	21
Evaluations	23
<b>6. Discussion of results</b>	<b>25</b>
<b>7. Lessons Learned about ML topics due to project</b>	<b>26</b>
<b>8. Challenges faced and goals that were not achieved.</b>	<b>27</b>
<b>Bibliography</b>	<b>29</b>

# 1. Problem Description

With the increased amounts of image data, the problem of translating images to other images has been at the forefront of convolutional neural networks research. In our project, we focus on the task of exploring methods to take in a sketched image of a human face and return a realistic looking face conditioned to the sketch. Our goal was to create a model capable of finding the most effective mapping between the domains of facial sketches and realistic facial images. Formally, we wish to train a network  $G$  capable of taking an input  $x$  from the domain  $X$  of sketch images and outputting an image  $y$  that is representative of the target domain  $Y$  of realistic face images while still retaining the structural qualities of the input.

Due to the complex nature of the problem, we focused our efforts on several approaches, all of which involved the use of Generative Adversarial Networks (GANs) [1]. We chose to focus our approach on GANs for two central reasons:

- 1) Most of the recent breakthroughs we found in the domain of image translation used GAN based approaches. In our research we found GANs to be widely used for this task, and found few papers using other approaches.
- 2) We felt that effectively exploring GANs would require the majority of our attention for this project. GANs are complex models with many working parts, and training for image translation tasks can be time intensive.

We focus our project on finding the best approach to generating high quality translated images of size  $256 \times 256$ . We first implement two existing approaches and tune them to our dataset. We then investigate an approach involving training a smaller model that generates images of size  $128 \times 128$ , and then scaling the images to  $256 \times 256$ .

## 2. Team Member Roles

Anthony explored the performance of conditional GANs cycle consistent GANS under different learning rates and generator types. He implemented a ResNet generator and set up the scaling algorithm to be used for our resolution assisted GAN. He computed the Frechet Inception Distance (FID) to measure image quality evaluation of each GAN's output.

Joe implemented markovian (PatchGAN) discriminator and a U-Net encoder-decoder generator for a conditional GAN architecture. He also implemented a conditional GAN with a U-net + residual blocks + DCGAN generator, and a loss that consists of adversarial loss, identity loss, and L1 loss.

Jason preprocessed the dataset and used OpenCV to generate sketch images. He trained conditional GAN models on different styles of input sketches to determine the ideal dataset to use. He visualized performance and evaluated training loss by using the loss log for each model to compare results.

## 3. State-of- the-art/Related Work

### Generative Adversarial Networks

GAN's were first introduced in Ian Goodfellow's 2014 paper [1], which describes generative adversarial networks as consisting of two models G and D that are simultaneously trained. G aims to generate synthetic data, and D estimates the probability that a sample comes from the training data as opposed to G. The GAN's training can be likened to a game between the two networks G and D: G aims to maximize the odds of successfully fooling D while at the same time D aims to minimize the chance of being fooled by G. The adversarial setup enables G to learn parameter values capable of generating complex distributions because D is constantly identifying error in G's output. In this way, GANs allow us to specify a high level goal

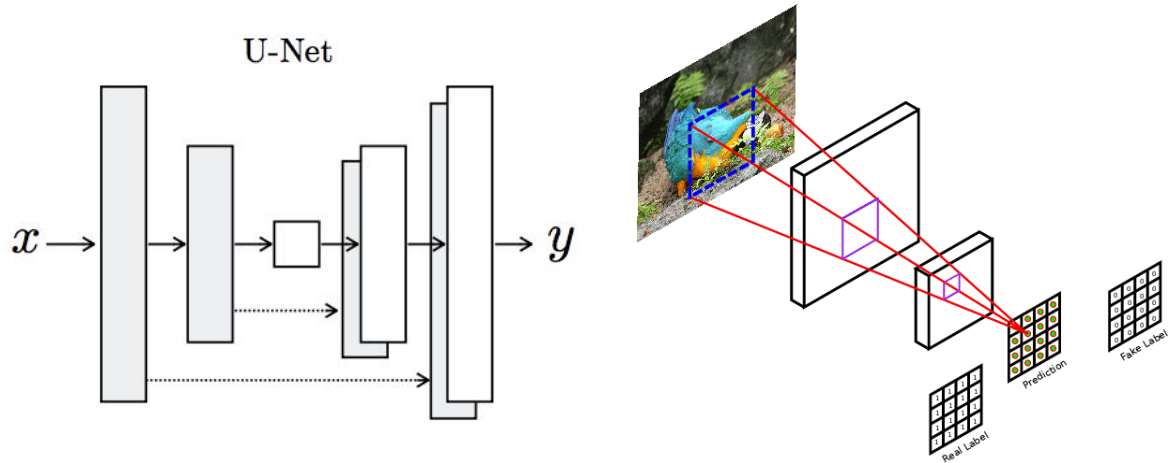
of learning to generate a realistic output and have the model learn the exact loss measure needed for this operation. A generator trained without adversarial loss would lead to overfitting and memorization of training instances by the network.

The use of GANs is very popular in image translation due to their ability to learn effective loss for mappings between domains and improve their generation over time. In the context of image translation the discriminator  $D$  aims to rate the probability that an input-output pair, rather than just an output sample, came from the training data. We incentivize our generator  $G$  to learn network parameters such that the discriminator  $D$  is fooled into classifying pairs containing generated images as authentic pairs.

An issue in training GANs is that larger models tend to be far less stable in their ability to generate realistic output. This is due to the increase in network parameters as size scales; larger networks require more resources to train compared to smaller networks.

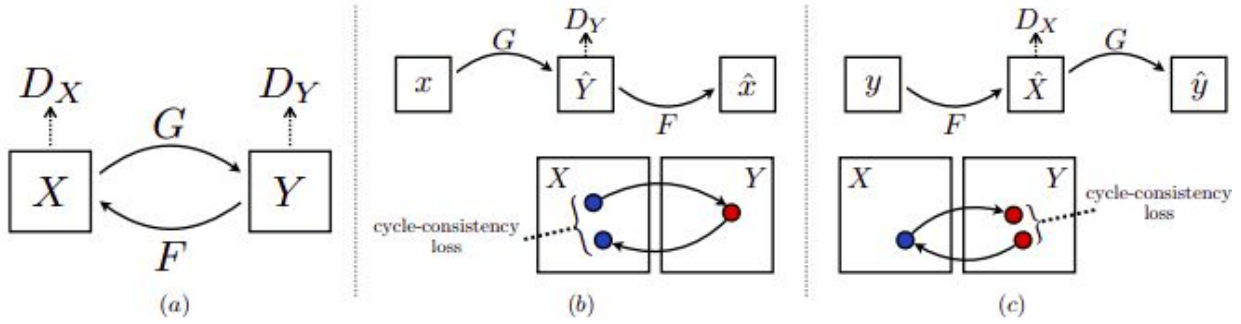
## Image translation

The paper “Image-to-Image Translation with Conditional Adversarial Networks” [2] put forth the Pix2Pix architecture, regarded as a breakthrough for being able to learn high quality translations across many domains given paired training data. The Pix2Pix architecture is a conditional GAN, which is a GAN that takes in an input image on which it is conditioned in addition to the random vector input traditionally taken in by a GAN. The generator of the Pix2Pix GAN uses a modified encoder-decoder network. An encoder-decoder network consists of a series of downsampling layers followed by a series of upsampling layers. The encoder and decoder sections of the network both have the same number of layers; the U-Net modifies this network by adding skip connections between each downsampling layer and its complement upsampling layers. This allows the data of the input photograph to flow directly from the input layer of the network to the output layer, which is vital in preserving the structure of an input image in an image translation model.



The Pix2Pix discriminator, called PatchGAN, outputs a matrix of classifications instead of a single value(real or fake). Each value in this matrix is on a scale of 0 to 1 where 0 represents fake and 1 represents real. PatchGAN slides its field of view across all the "patches" in the input image in a convolutional manner and then gives feedback on each region or patch of the image. Since this architecture calculates the probability of each patch being real or not, binary cross-entropy loss can be used to train PatchGAN. The resulting output for a fake image should be a matrix of all zeros, while the output for a real image should be a matrix of all one's. Overall this means that the discriminator will give a lot more feedback back to the U-net generator.

Defined in the paper "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks" [3], CycleGAN is an architecture that builds upon the Pix2Pix conditional GAN approach. Unlike Pix2Pix, CycleGAN does not use paired sets of images to train the network. The training set defines two domains X and Y upon which the model aims to learn a translation. To most effectively learn the translation from X to Y, CycleGAN trains two separate GANs, to learn the joint distribution of translations from X to Y as well as from Y to X.



CycleGANs utilizes two different GANs that encourage cycle consistency, which is an extra loss term to the loss function. Cycle consistency expects the generated image  $Y$  to look just like the input image  $X$ . The circle consistency loss is the sum of the pixel difference from both directions. This can be modeled as Least Squares Adversarial Loss (MSE) +  $\lambda$  \* Cycle Consistency Loss, where  $\lambda$  is a constant hyperparameter and cycle consistency loss is the adversarial loss similar to Pix2Pix in two directions.

One optional loss that CycleGANs proposed is the identity loss, which mainly helps preserve color in the output image. This loss helps make sure that the output image and real input have as low pixel distance from each as possible. When the identity loss is zero, then the generator knows that the image it has produced is correct, discouraging this mapping from being anything but identity. The identity loss is then added to the overall loss function.

Therefore, the loss function of CycleGAN in one direction will look like this:

$$\begin{aligned} \mathcal{L}(G, F, D_X, D_Y) = & \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ & + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ & + \lambda \mathcal{L}_{\text{cyc}}(G, F), \end{aligned}$$

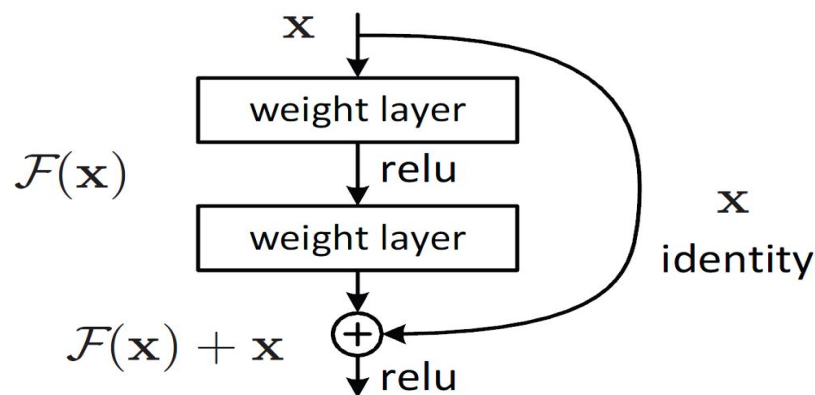
Least Squares Adversarial Loss +  $\lambda_1$  \* Cycle Consistency Loss +  $\lambda_2$  \* Identity Loss.

The generator loss can be modeled by the sum of the loss function described above for both directions, input to output and vice versa.



## Residual Networks

The ResNet architecture was originally defined in the paper “Deep Residual Learning for Image Recognition” [4]. A ResNet is composed of several residual blocks. A residual block consists of two convolutional layers, with an additional identity connection that takes the input to the residual block and adds it to the output of the two layers. Similar to the U-Net, the skip connection between layers allows data to flow more freely through the network, though in this case the skip connection only skips two layers and adds the output rather than concatenating it. The residual connection makes it easier to compute the gradients of the network, which allows us to use residual blocks to create very deep networks that can still be trained using gradient descent.



A residual block consisting of two layers with an additional identity connection [4]

## Super Resolution

Our approach leverages advancements in image scaling and super resolution to train a smaller image translation GAN and then upscale our output to the desired resolution of 256x256. The paper “Enhanced Deep Residual Networks for Single Image Super-Resolution” [7] uses a deep convolutional network with residual blocks to achieve state of the art results in image upscaling. We use OpenCV’s implementation with this network along with a set of pretrained weights to scale the images generated by our GAN.

## 4. Approach

### Dataset

We used the Celebrity Faces Dataset [8] downloaded from Kaggle. We selected this dataset as it offered a wide range of faces to train our model and had been published by Hong Kong University for the purpose of being used as training data for training computer vision models.

### Environment

Google Colab is a cloud-based Jupyter notebook environment that is suited for machine learning and research purposes. We largely used Google Colab to train and test our models using the available cloud GPUs.

### Realistic Sketches

We initially used OpenCv's, canny edge detection algorithm to generate edges of the faces in our dataset. Below are some outputs using edges maps corresponding to the images used. This method did well to capture edges when color was distinct enough for it to differentiate. Comparing the two images below, one captured too many edges while the other captured not enough edges



Upon observing our models struggling to learn with the data, we used OpenCV to generate more realistic looking “sketches”. The process was to convert the color image to grayscale. Using the “bitwise\_not” function, we were able to invert the colors of the grayscale

image that was generated. Then we used the “GaussianBlur” function as an image smoothing technique to reduce the noise and edges from the image. This is done by taking the weighted averages of the kernel while giving higher weights to pixels near the center of the kernel. The new value for the center pixel will be the weighted average. To get the final image, we combined the grayscale image with the smoothed image. Below are some examples of the “sketches” we used to train our models.



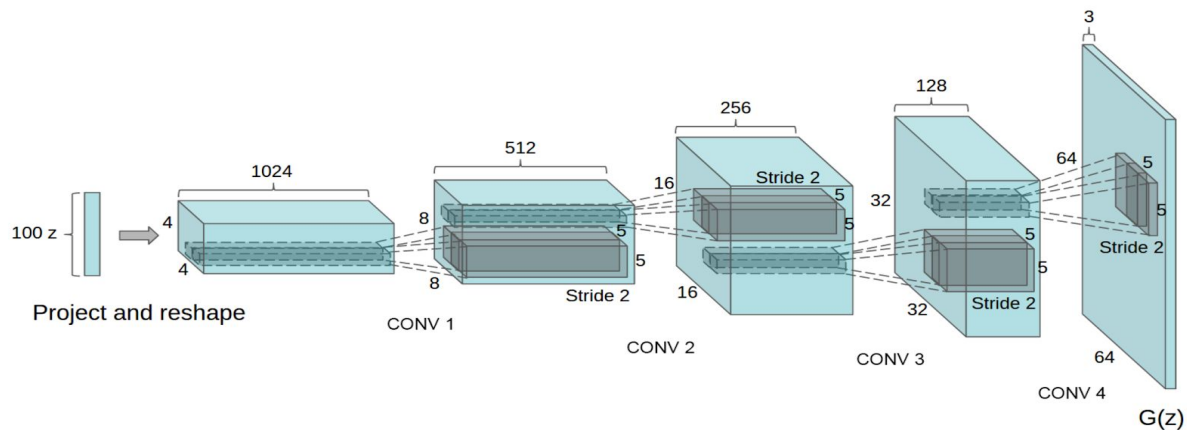
### Conditional GAN

We used code published by the Pix2Pix authors as a starting point for fitting a conditional GAN to our data. We used the utility functions to load and process our dataset and modified the code defining the training loop and architecture to support eager execution in Tensorflow. Our changes enabled us to fit multiple instances of the model and compare different hyperparameters. We focused on experimenting with learning rates of the GAN and varying the depth of our generator network.

### Conditional GAN + DCGAN + Residual Blocks + Identity Loss + Instance Normalization

We based this conditional GAN on the Pix2Pix architecture, but aimed for more stability. In order to achieve this goal, we kept the discriminator the same using PatchGAN but implemented more layers in our U-net generator similar to the architecture of Deep Convolutional Generative Adversarial Networks (DCGAN). Since training stability was an important goal for us, we used the stable architecture that was first introduced in a 2015 paper

by Alec Radford, et al. titled “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”



Under this network architecture, the following guidelines are:

Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).

Use batchnorm in both the generator and the discriminator.

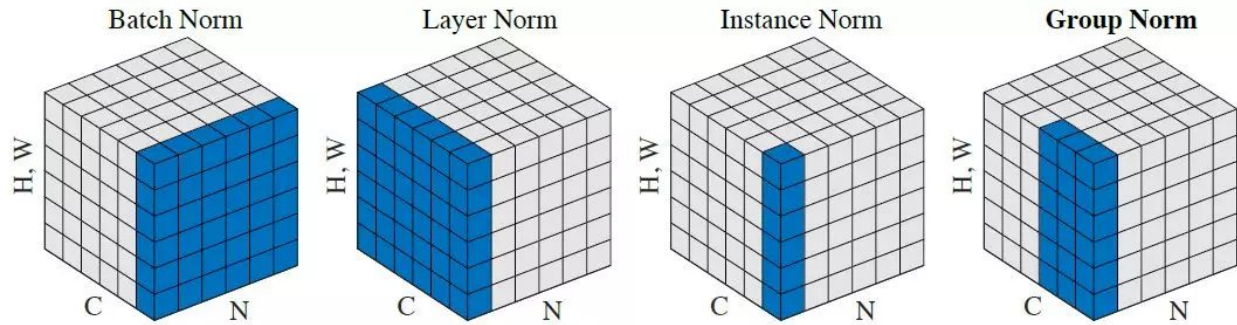
Remove fully connected hidden layers for deeper architectures.

Use ReLU activation in generator for all layers except for the output, which uses Tanh.

Use LeakyReLU activation in the discriminator for all layers.

According to the same paper, DCGANs does not produce high quality samples via simply overfitting/memorizing training examples and thus provides stability.

To guide out conditional GAN towards the right direction, we implemented a loss function that consists of the adversarial loss plus identity loss. Identity loss is an extra loss term that in theory will help with color preservation in our outputs.



Tackling the vanishing gradients problem is an important problem in deep neural networks and GANs. To avoid this, normalization is often used to a layer to have zero mean and unit variance, making training more efficient. Pix2Pix uses Batch normalization to improve generalization performance. Adding  $\beta$  to the normalized input and scaling it by  $\gamma$  ensures the model does not lose representational power as a result of the normalization. Instance normalization on the other hand normalizes each batch independently as shown above.

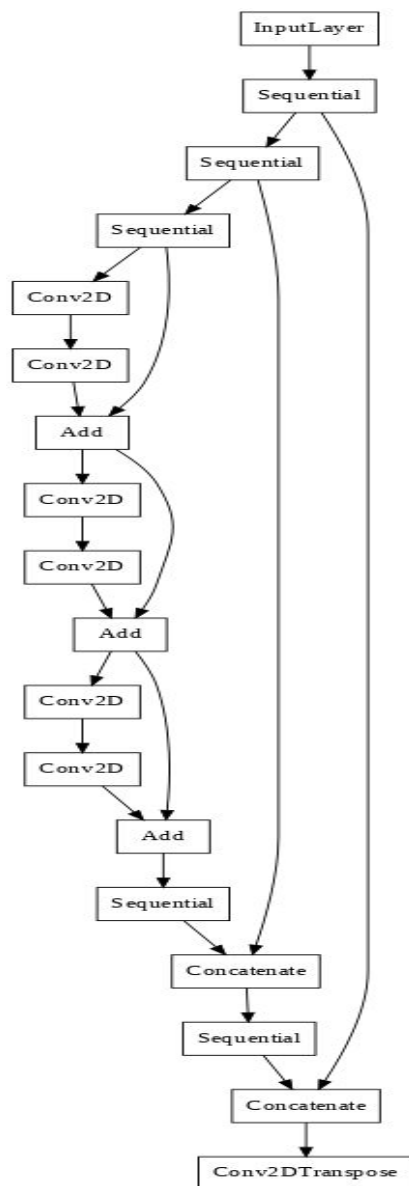
### Cycle Consistent GAN

We used code published by the CycleGAN authors as a starting point for fitting a cycle consistent GAN to our data. We used the same utility as our Conditional GAN to load and process our dataset. We modified the code defining the training loop and architecture to support eager execution in Tensorflow and allow us to fit work with multiple instances of the model in the same notebook. We used the same U-Net architecture used for our Conditional GAN rather than the ResNet used in the CycleGAN paper. We focused on experimenting with learning rates of the entire GAN and the relative learning rates of the generator and discriminator.

### GAN + Super Resolution

For a novel approach, we aimed to generate sharper 256x256 images with an approach that decreases the number of parameters in our model's sketch to face translation module. We train a conditional GAN designed to input and output 128x128 images. We then scale the translated image from our GAN to 256x256 using a pre trained Super Resolution CNN. By

reducing the complexity of our GAN we are able to train a more stable network and generate images with a better high level representation of facial features and textures. We leverage pre existing methods of deep learning based super resolution to scale our image to 256x256 while minimizing and reducing blur. In our application the super resolution network does not need to be specifically trained for the task, and does not need information about the domains of X and Y. We are only learning the translation from domain X and Y in our GAN.



In our conditional GAN, we implemented a ResNet based generator instead of a U-Net generator. The original CycleGAN paper used a ResNet based generator (we used a U-Net in ours)

In our implementation of the ResNet we included 3 downsampling layers followed by 9 residual blocks and 3 upsampling layers. As in the U-Net, we kept the skip connections between the encoder and decoder sections of our network . We tuned for different numbers of residual blocks and found 9 to be optimal. There was a visible difference between the results of the ResNet based models and the U-Net based model; there was less variation among the outputs of the ResNet generators with different numbers of residual blocks.

The image on the right shows a diagram of our ResNet generator with 3 residual blocks, activation layers have been omitted. There are two skip connections from the first two downsampling layers to the last two upsampling layers; this is the same as the U-Net and allows structural data to flow

through the network There are three identity connections for each of the residual blocks.

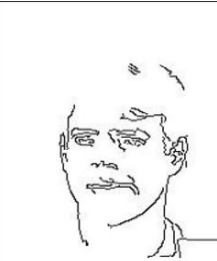













## 5. Experiments/Evaluation

### Conditional GAN

In the initial stages of working on our project we used the Pix2Pix architecture to compare different styles of input “sketches” to use. We compared to two methods of processing the raw data we outlined in the Approach section. We found that using only edges as input led to more distortion in generator output. Many of the input edge images had at least one of the following issues: a lack of defined edges, too many edges, background distortion captured by edges.

We found that the more realistic sketches we created with OpenCV produced more stable results. The additional shading data gives the generator more to work from and requires less inferences to be made about features connecting edge components. The table below compares outputs from models trained on the same datasets. We used 100 images for our training set and trained for 200 epochs.

Edge Map			Realistic Sketches		
 Input Image	 Generated Image	 Actual Image	 Input Image	 Generated Image	 Actual Image
 Input Image	 Generated Image	 Actual Image	 Input Image	 Generated Image	 Actual Image

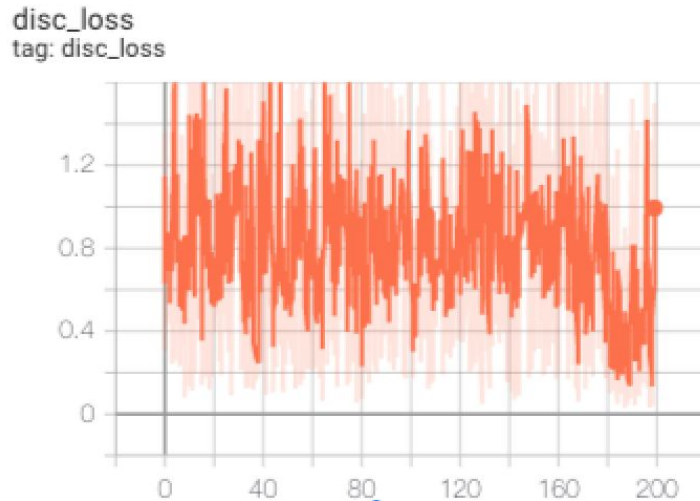
In our fitting of the Pix2Pix architecture to our dataset we experimented with varying the number of hidden layers in the U-Net architecture and the learning rate of our network. We found that learning rate carried a greater influence than generator depth. We found that a learning rate of 0.0001 performed the best qualitatively. We observed that varying network depth had less of an impact, we found that generators with maximal captured slightly more detail after being trained on a subset of the data. We chose to train a model with maximal depth, a generator learning rate of 0.0002 and a discriminator learning rate of 0.0001.

Sample outputs at different learning rates

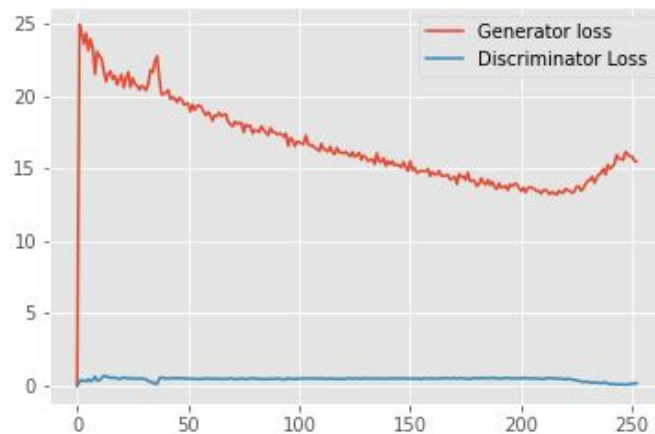


We found it difficult to determine if the Pix2Pix model (or any GAN we trained) converged or not. The discriminator loss of our network is plotted below; the loss value oscillates up and down over the training duration and does not clearly indicate model performance. As the generator learns to create better images the discriminator will be more likely to be fooled and thus the loss will go up.





In a separate training run we experienced the scenario in which the discriminator converged to a value close to zero ( $\sim .45$ ) and remained there, the graph of the generator and discriminator loss is shown below. This low value can indicate that the discriminator has overtaken the generator in terms of performance; the discriminator is able to accurately assess which image pairs are fake and the generator is failing to generate high quality realistic images.

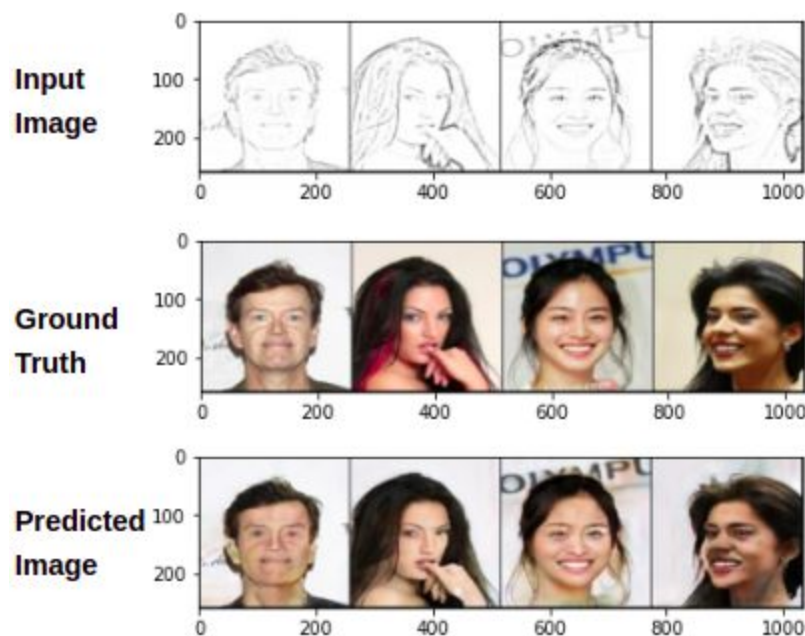


Conditional GAN + DCGAN + Residual Blocks + Identity Loss + Instance Normalization

After working with Pix2Pix in our initial stages of working on our project we found that the image generated had trouble with color distribution and training stability. To accomodate for these issues, we experimented with different ways to achieve better generation. Using the

cycleGAN paper as reference we were able to redesign the Pix2Pix generator by expanding up the bottleneck section with more convolutional layers similar to the DCGANs generator. Within the bottleneck section we also included additional skip connections using residual blocks from ResNet. According to the cycleGAN paper this combination did help the performance of their generator during training.

We chose to train this model in batches of 4, a generator learning rate of 0.0002 and a discriminator learning rate of 0.0001 over 40 epochs. Below are sample predictions from our input sketches.



Sample from our conditional GAN hybrid

## CycleGAN

In our training, CycleGAN was not the most effective approach to sketch to face translation. The translation from sketches to faces is defined by upsampling of pixel data while the translation from real faces to sketches involves downsampling of pixel data. We found that the GAN learning the mapping from faces to sketches converged after X epochs, far faster than

the mapping in the opposite direction. In the context of our problem, learning an additional mapping with CycleGAN doubles our model's parameter complexity and increases training time, but does not show an improvement in results or make efficient use of compute.

In our problem, the transformation from face to sketch is less complex than sketch to image. The downsampling operations required to convert a face photo to a realistic looking sketch is learned far quicker than the translation in the opposite direction. We show below an example of both translations on the same image pair. The generated sketch is far closer to the ground truth sketch than the generated face is to the ground truth face.

Translation from sketch to face



Translation from face to sketch



We did not feel that such a cycle consistent approach would be the most efficient for our final model. We found better results using the conditional GAN approach which is why we chose a simpler conditional GAN based approach to use in our final model.

We also investigated the effect of varying the relative learning rates of the generator and discriminator. We found that a generator that learns at twice the rate of the discriminator produced clearer outputs than a generator learning at the same rate as the discriminator. Below is an example of two outputs from generators with different relative learning rates.

Equal learning rates



Generator learns twice as fast

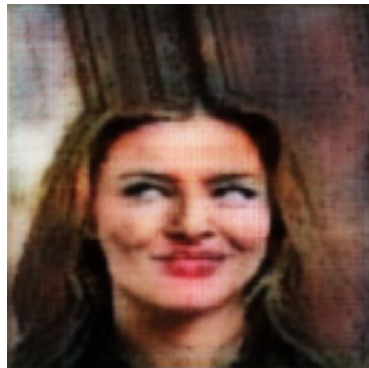


The example in which the learning rates are equal is far blurrier than the example in which the generator learns at a faster rate.

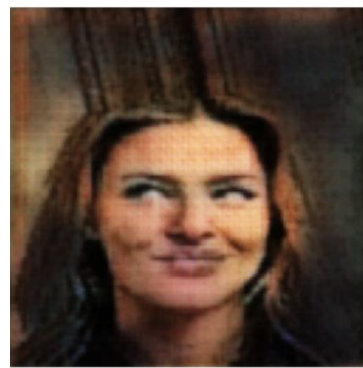
### **Conditional GAN + Super Resolution.**

For our final model, we trained a conditional gan based model for 100 epochs. We compared the performance of a ResNet generator to a U-Net generator. We trained both approaches on a subset of the data for 20 epochs. We found that the U-Net accentuated color more while teh ResNet did a good job of generating stable output. We chose to train the ResNet on the entire training set to experiment with a new architecture. Below is an example of U-Net and ResNet outputs.

U-Net



ResNet





Input Sketch	Ground Truth	Pix2Pix	CycleGAN	ResNet + Super Resolution
				
				
				
				
				

A table of the results from our different models after being trained on the full dataset of 2000 images

## Evaluations

The evaluation of results from GANs, especially in the setting of image translation, is still an open problem in research. Two reasons for this challenge in the evaluation of GANS are that:

1. Loss is uninformative of performance. Unlike other classifiers such as k-nearest neighbors or SVM, where loss is correlated with the models performance, the loss from the generator or discriminator of GANs only suggests whether learning has stopped or not. Our losses often oscillated throughout training and do not give us a clear indication about how well the model is performing.
2. There is no perfect non-human metric for evaluation. The goal of our GAN is to produce images that look realistic to humans. The Pix2Pix and CycleGAN research papers we reviewed used perceptual studies to holistically evaluate the realism of their model's outputs. The researchers crowdsourced evaluation in small studies in which participants looked at a mix of real and generated outputs and attempted to correctly classify them. This method is still prone to human bias but makes for a decent metric on which to evaluate the level of realness in generated images; unfortunately we did not have the time or resources to conduct such studies on our results.

In choosing the optimal hyperparameters and architectures of our models we largely relied upon qualitative evaluation of the results of different models. We used this approach to decide on which hyperparameters to use when fitting on the entire training set for our implementations of pix2pix, CycleGAN, and our super resolution ResNet GAN.

To rate the quality of the distributions generated by our models on test inputs, we evaluated them using the Frechet Inception Distance (FID score) [6] , a metric that uses a pre

trained Inception V3 image classifier to measure the difference in the distribution of last layer activations between the set of real images and the set of generated images. The FID score considers gaussian distributions for both the real and generated distributions based on the mean and covariance of their activations, and then computes the Frechet distance between the two curves. The FID score builds on the Inception Score metric [5] , which uses the Inception V3 network to measure the fidelity and diversity of GANs based on how the generated images are classified by the network..

A downside of the Inception Score is that the generated images should correspond to a class in the ImageNet dataset that the Inception network was trained on; the dataset does not contain faces which is why we did not focus on this metric. The paper that first proposed the metric recommends a sample size of 50,000 to be used, which would be a challenge for us to manage computationally. We chose to focus on the FID score due to the limitations of the inception score and the fact that the FID score finds the distance between the true distribution and the generated distribution.

Rather than relying on human perception, we rely on the perception of a computer vision model. We assume that similar images have similar activations in the network; a low FID score between a true distribution and generated distribution indicates that the two distributions are similar in quality. The paper in which the FID score was originally defined conducted experiments to demonstrate that a higher FID score correlates with increased disturbance in images and that the score is in line with human judgement. We use the penultimate layer of the inception net because it is the coding layer where representations of features are stored. Formally the paper defines the FID score between real images  $x$  and generated images  $g$  as:

$$FID(x, g) = ||\mu_x - \mu_g||_2^2 + Tr(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{\frac{1}{2}}),$$

We convert the activations to Gaussian distributions based on their mean  $\mu$  and covariance values. We then compute the frechet distance between these two curves.



In addition, we measured the average difference in pixel values between real images and generated images. This metric can give us some data about our generator's accuracy, but does not measure the diversity of our model's performance and rewards models that overfit.

	<b>Frechet Inception Distance</b>	<b>Average Pixel Difference</b>
<b>Pix2Pix</b>	142.44	0.9
<b>CycleGAN</b>	143.18	0.96
<b>ResNet + Bicubic Interpolation</b>	142.05	0.91
<b>ResNet + Super Resolution</b>	139.07	0.91

The FID Scores for different GAN architectures are largely similar. The scores indicate that the ResNet GAN with Super Resolution produces the best output compared to the real data. The output from the ResNet GAN scaled with a baseline upscaling algorithm has an FID score only slightly higher than the Pix2Pix and CycleGAN distributions.

## 6. Discussion of results

The method of training a small, stable GAN to learn a translation between domains and leveraging pre existing deep learning models to scale the output of the GAN appears to be an approach worth exploring further. As the realism of state of the art models trained on powerful GPUs continues to improve, it is also important to create architectures that most efficiently learn translations. The FID score of the GAN we trained with Super Resolution applied is better than the scores of the Pix2Pix and CycleGAN models. Qualitatively, the scaled model appears on par with the 256x256 Pix2Pix model. The CycleGAN architecture looks to be the weakest choice for our task of sketch to face translation. In addition to not making effective use of compute resources, the outputs from the model have less realism compared to our other GANs; many of the CycleGAN outputs contain distortion and random blips of color.

## 7. Lessons Learned about ML topics due to project

GANs are an example of a recent development in the still expanding field of Machine Learning. Researching GANs required us to sharpen our understanding of Information Theory and artificial neural networks.

Evaluating the quality of our GANs requires knowledge about entropy and KL divergence, as these metrics are used in calculating Inception Scores and Frechet Inception Distance. In working with GANs, we are often concerned about the entropy of our generated distribution.

GANs are a topic at the forefront of ML research. In studying them we were constantly exposed to the fact that Machine Learning is still an expanding field; we are still making new discoveries and growing our collective understanding. There are lots of paths of experimentation that have yet to be explored; there are also many novel approaches that lack extensive documentation.

There are many open problems that one may encounter when undertaking a ML project; in our case we did not have a clear evaluation metric to definitively rate our model's generated image. In the end, our evaluation is subjective. This project taught us about the importance of software engineering skills and familiarity with computing hardware when working on a ML project. We often found ourselves having to debug our models and refactor code to get the desired performance; coding skills and a solid understanding of a model's inner workings are important to have alongside theoretical knowledge of machine learning approaches. The ability to manage computer resources is essential. Deep learning, particularly the training of GANs, is very time and resource intensive. The ability to work with external GPUs and cloud GPUs, as well as managing active memory is necessary for efficient training of a model. We would not have been able to train our GANs had we been only relying on our local computer.

This project also taught us about the importance of interpretability in Machine Learning. GANs are very complex models with potentially billions of parameters to be tuned. The complexity of GANs allows them to generate realistic looking data but also makes it difficult to analyze the internal mechanics of the model; the network behaves largely as a black box.

The way in which we define loss can carry a major influence on our model's behavior. In our experiments we observed a large influence from modifying losses; defining what a model should optimize for is crucial in controlling the model's behavior.

## 8. Challenges faced and goals that were not achieved.

In order to train different GANs on our data, we first had to set up our environments to do so. There were many bugs that arose in the initial development of our notebooks; we initially ran into errors when processing our data to be taken in as input for our generator and discriminator. Since we were using a custom dataset not provided by the Pix2Pix authors, we had to fit our data into their model. This is a common part of the development process but posed a small challenge that took up our initial time.

In addition, our training was very time intensive; this limited the number of models, hyperparameters, and loss functions we were able to explore in training. We did most of our training on Google Colab, which disconnected or crashed during training on several occasions. Another issue is transferring files to Google Colab. One method was sending it to Google Drive where we can reuse the files but when we ran our models with increasing amounts of images trained and tested, we found that we quickly went over the available Google Drive storage limit. The alternative was importing the folders manually from our local file system. While this method was simple, each time we opened the colab notebook, we would need to reload all the folders from local storage as it does not save in Google Colab.

If we had more time, we would have liked to spend more time cleaning our dataset and removing distorted samples. We initially aimed to train our model using only raw edges as input, but we transitioned to using more realistic looking sketches. We would have liked to train and test with data from various sources and testing the fidelity and diversity of training with different distributions of sketch images.

In our proposal we said we would use FCN score to evaluate our models, we ended up using Frechet Inception Distance instead. The papers we read that used FCN score calculated the metric using a model that the researchers trained on a large dataset. To avoid training another large model we used the FID score because it also evaluated results using a computer vision model and can be computed using available pretrained weights.

Our evaluation of the results was also difficult because there is no one framework that we can follow to evaluate the quality of our generated images; forcing us to rely heavily on subjective evaluation. We used the FID score as an evaluation metric, but this score can only tell us so much as our results were fairly similar. If we had trained for longer with larger sample sizes perhaps there would have been more variance.

## Bibliography

- [1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In NIPS, 2014. 2
- [2] P. Isola, J. Zhu, T. Zhou and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 5967-5976, doi: 10.1109/CVPR.2017.632.
- [3] J. Zhu, T. Park, P. Isola and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, 2017, pp. 2242-2251, doi: 10.1109/ICCV.2017.244.
- [4] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.
- [5] Salimans, Tim & Goodfellow, Ian & Zaremba, Wojciech & Cheung, Vicki & Radford, Alec & Chen, Xi. (2016). Improved Techniques for Training GANs.
- [6] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *NIPS*.

[7] B. Lim, S. Son, H. Kim, S. Nah and K. M. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, 2017, pp. 1132-1140, doi: 10.1109/CVPRW.2017.151.

[8] Z. Liu, P. Luo, X. Wang and X. Tang, "Deep Learning Face Attributes in the Wild," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 3730-3738, doi: 10.1109/ICCV.2015.425.