Photorealistic Sketch to Image Synthesis using Conditional GANs

CHRISTOPHER WILSON, Deep Learning For Computer Graphics, University of Florida

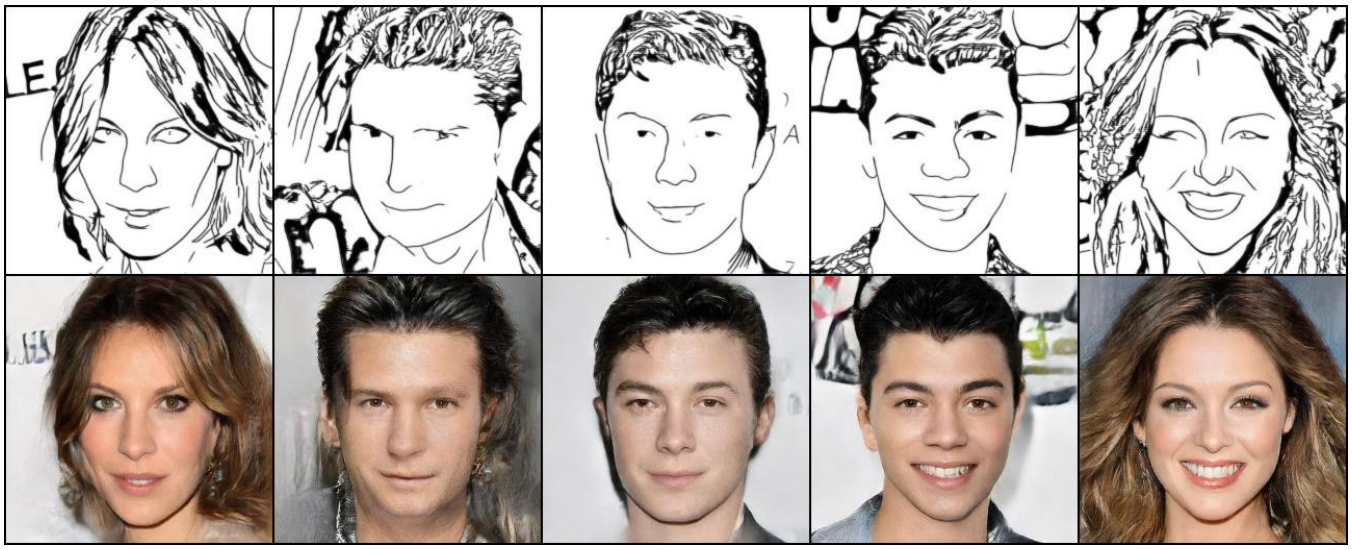

Fig. 1. Pairs of input sketches with photo outputs generated by our model.

This paper presents a three stage deep learning pipeline for translating sketches of human faces into realistic face images. By learning feature encodings for each major facial feature separately (left eye, right eye, nose, and mouth), a generative adversarial network (GAN) can be used to accurately synthesize images of faces that are almost imperceptible from real images. Encoding the major facial structures as input to a GAN provides the generator a framework to construct images of new human faces from the basic structure common to all faces. This approach avoids the multiplicity problem common to many GAN-based networks by always ensuring there are two eyes, a nose and a mouth, and that they are all positioned correctly relatively to each other.

ACM Reference Format:
Christopher Wilson. 2021. Photorealistic Sketch to Image Synthesis using Conditional GANs. 1, 1 (April 2021), 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Conditional image synthesis, the generation of a new image based on given information, is an important area of study in deep learning. The generation of a full-detail image from a simple, sparse representation is of particular interest in the field of deep learning because of the large set of plausible results it gives networks to learn. In this paper, we implement an end-to-end deep learning pipeline for the synthesis of a photorealistic human face from a simple sparse sketch. This is a difficult problem because for a single sketch of a human face, there are countless scores of plausible human faces that are a good match for the sketch. In other words, when translating from low detail to high detail, there are many ways to fill in the details. As a result, there is no one-to-one function that a neural network can learn to translate sketches to faces. However, it is only necessary to produce a face that is a "good enough" translation of the sketch, so a generative adversarial network is used for this task.

Author's address: Christopher Wilson, Deep Learning For Computer Graphics, University of Florida.

XXXX-XXXX/2021/4-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 PREVIOUS WORK

This paper is based on the network proposed in DeepFaceDrawing[1]. Both this paper and DeepFaceDrawing implement a 3 stage sketch to image pipeline, but this paper does not include the k-nearest-neighbors feature vector projection implemented by Chen et al. As a result, this paper aims to learn a more deterministic mapping than DeepFaceDrawing.

2.1 Image Synthesis

The topic of image synthesis has been studied in depth for several decades. Traditionally, image synthesis was performed using explicitly written algorithms. Much research was focused on learning and understanding the relationship between lighting and image color, and physics-based models that simulated the effect of light rays dominated [9]. Isola et al. proposed a convolutional GAN and

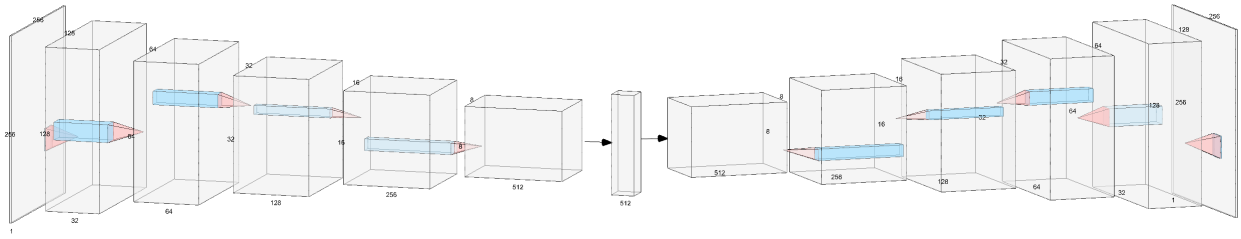

Fig. 2. The autoencdoer architecture of the Component Embedding module. It features 5 convolution blocks on the encoder half and 5 transpose convolution blocks on the decoder half. There are ResNet blocks immediately following each convolution and transpose convolution block, and there is a fully connected layer directly before and after the feature vector in the middle. There are 5 autoencoders in the Component Embedding module, one for the left eye, right eye, nose, mouth, and rest of face, and the dimensions of each are shown in table 1.

applied it to general-purpose image synthesis, opening the door to applying deep learning and GANs to image synthesis [3]. The same architecture was able to transform images of street maps into satillite view images, photos taken during day to night, edge maps of purses to photos of purses, building façade segmentation maps to building photos, and black-and-white images to colorful images. This paper takes the edge-map-to-purse-photo task of Isola et al. and applies it to the much more complex problem space of human faces, presenting a network that translates sketches of human faces into photorealstic images. Since Isola et al's paper in 2016, a variety of other papers have come out iterating on the concept and proposing more and more complex networks for image synthesis. An example of this is Zhang et al's paper which iterates on Isola et al's convolutional GAN architecture by applying the concepts of recurrent neural networks to generate new images based on a text description [15].

2.2 Conditional GANs

When GANs were first proposed in 2014 by Goodfellow et al, they were a revolutionary new method of generating new, unseen samples by learning the underlying distribution of the training data and randomly sampling from it [2]. The main problem with this was that the original GANs could only accept random noise as inputs, meaning it was good for randomly generating data that comes from an underlying distribution, but not very useful for generating data with specific properties. Soon after, Mirza and Osindero published a proof of concept that it was possible to build GANs to generate a specific MNIST digit given only a single numerical input [8]. This opened the door to far more useful networks, where the generative power of GANs could be leveraged to generate data conditioned on a specific input, not just a random noise vector. Over time, resarchers started publishing more and more networks using conditional GANs, including using semmantic image labels to generate high resolution images [13], as well as general purpose image to image translation like Isola et al's work mentioned earlier [3]. This paper combines several of the latest developments in conditional GAN research including multi-scale discriminators [7, 14] and discriminators as perceptual loss functions [4] to implement a system that translates sketches of human faces into detailed photos.

3 BODY

3.1 Technical Description

3.1.1 Data Preparation. Data preparation was a significant challenge. To begin, the dataset prepared by CelebAMask-HQ [6] was used as a starting point. This dataset contains 30,000 1024x1024 pixel images of celebrity faces. Also included with the dataset are labels of accessories for each image (is wearing glasses, is wearing hat, etc) as well as bit masks for each major facial feature. For example, each image had a left eye bit mask that had ones over the photo's left eye and zeros everywhere else. These bit masks were summed to generate heat maps of the major facial features (left eye, right eye, nose, and mouth) and the heat maps were used to derive crop areas for each facial feature as described in section 3.1.3. To prepare the data, each image was resized to 256x256 pixels, and images with glasses or hats were discarded for simplicity. The transformed face images are used to generate the sketch dataset and act as real-image inputs to the discriminator during training (section 3.1.5.

To get sparse sketches of human faces, various approaches were attempted. Open-CV's Python library contains a function, `cv2.pencilSketch`, but this method created sketches that were far too dense and discolored. The best approach found was the following algorithm [10] using a Gaussian blur:

```
procedure makeSketch:
    convert to grayscale
    invert pixel colors
    blur with Gaussian kernel
    invert pixel colors
    element-wise divide grayscale image by result
    return
end
```

The resulting image was far better than any other attempt, but it still had some shading.

To reduce the shaded sketch to crisp lines, a pre-trained sketch simplification network[11, 12] was applied to the images produced by the Gaussian blur algorithm to generate the final sketches. These approaches are depicted in figure 4.

3.1.2 Pipeline. Our network is an end-to-end pipeline that takes a 256x256 pixel sketch of a face and transforms it into a 256x256 pixel realistic image of a human face. To accomplish this, we describe

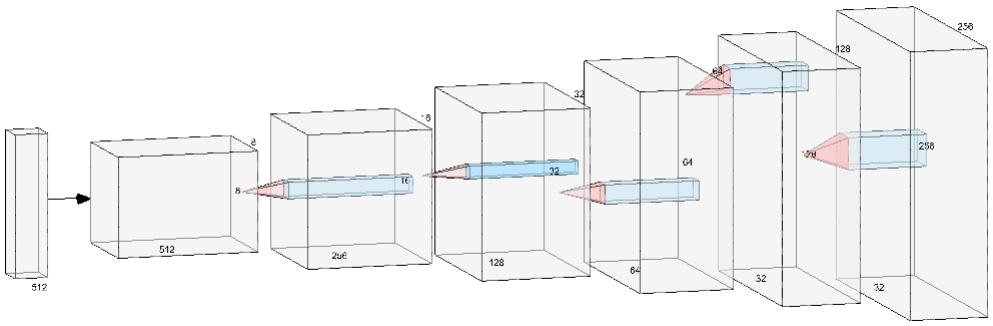

Fig. 3. The architecture of the Feature Mapping module. Similar to the decoder half of the autoencoders in the CE module, each of the 5 FM networks is composed of 6 transpose convolution blocks, each followed by a ResNet block with skip connections. The main difference between the two is that each FM network outputs a 32 channel feature map instead of a 1 channel sketch.

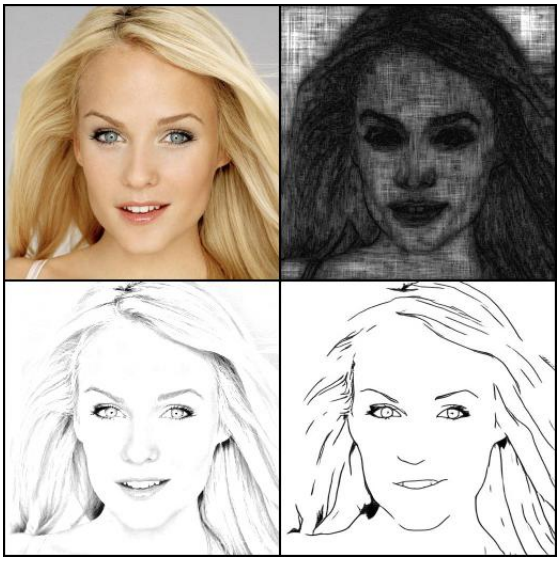

Fig. 4. From top left to bottom right: original image, image created from original with `cv2.pencilSketch`, image created from original with Gaussian blur algorithm, and Gaussian blur image after sketch simplification network.

three modules in the pipeline: Component Embedding (CE), Feature Mapping (FM), and Image Synthesis (IS).

3.1.3 The Component Embedding Module. The component embedding module contains 5 separate autoencoders that each learn the relation $x = D(E(x))$, where $E(x)$ is the encoder half of the network and $D(x)$ is the decoder half of the network. Each autoencoder is trained on a cropped section of a facial sketch for the facial features detailed in table 1.

For the relation $x = D(E(x))$, the goal is for the encoder to learn the mapping $E : \mathbb{Z}^{k\times k} \rightarrow \mathbb{R}^{512}$ and for the decoder to learn the mapping $D : \mathbb{R}^{512} \rightarrow \mathbb{Z}^{k\times k}$, where $k \in \{64, 84, 96, 256\}$, effectively mapping a subset of an image into a 512 dimension feature vector, then back into the original image subset.

Table 1. Table of feature windows for the CE module. The "Rest of Face" input is created by zeroing out all pixels that overlap with any other window. This helps capture the outer details of the image, including the hair, forehead, ears, and neck/shoulders.

Feature	Top Left of Crop	Bottom Right of Crop	Size
Left Eye	(90,64)	(154,128)	64x64px
Right Eye	(90,126)	(154,190)	64x64px
Nose	(103,86)	(187,170)	84x84px
Mouth	(143,80)	(239,176)	96x96px
Rest of Face	(0,0)	(255,255)	256x256px

These windows were derived from the semantic bit masks provided by the CelebAMask-HQ dataset. There are multiple masks per face image, one for each major facial structure (nose, mouth, etc). The crop windows were derived by computing the weighted sum of all bit masks for each facial feature, and then centering a square of the appropriate size over each feature as seen in figure 6.

The purpose of the component embedding module is to generate feature vectors representing each major facial landmark that will be passed to subsequent modules in the pipeline. This ensures that all major facial structures are accounted for in the network, and in the proper multiplicity, since many GANs can problematically generate images with multiple mouths or one eye for example.

Each autoencoder in the CE module takes on a convolutional architecture, with ResNet blocks and skip connections between each convolution block. These ResNet blocks are added to provide additional capacity to learn a relationship while also countering the vanishing gradient effect of deep neural networks. Further steps taken to minimize vanishing gradients include replacing ReLU activations with LeakyRelu, which allows a small gradient through if the preactivation value is less than zero, compared to ReLU, which can saturate at zero.

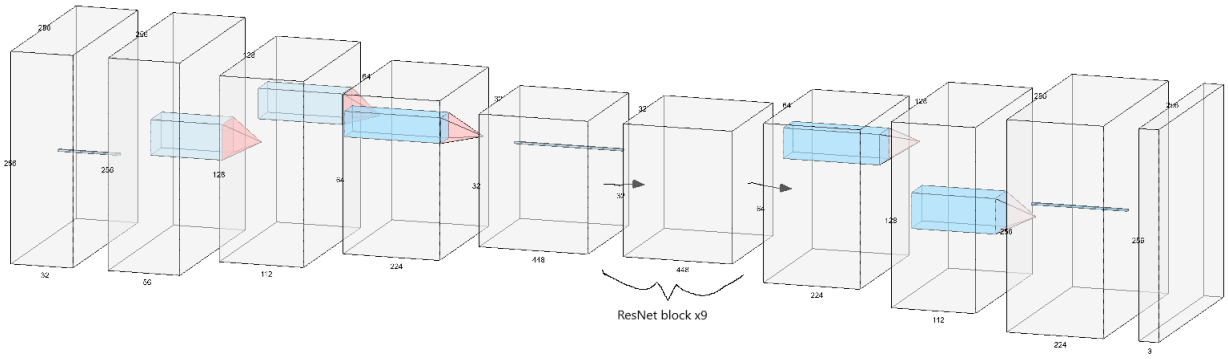

Fig. 5. The architecture of the IS module generator. The first 4 layers are convolution blocks, the last 4 are transpose convolution blocks, and the middle of the network contains 9 ResNet blocks.

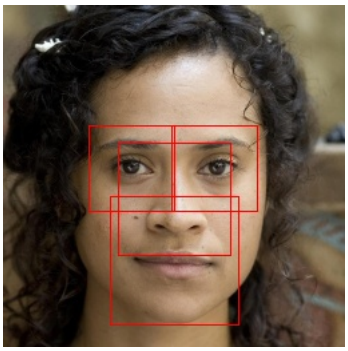

Fig. 6. Sample image with crop windows overlaid in red.

3.1.4 The Feature Mapping Module. The Feature Mapping module largely resembles the decoder half of the CE module, with the notable exception that instead of learning to reconstruct each image crop with 1 channel, it learns to construct a feature map of each image crop in 32 channels. Effectively, it learns the relation $FM : \mathbb{R}^{512} \rightarrow \mathbb{Z}^{32 \times k \times k}$, where $k \in \{64, 84, 96, 256\}$. After it generates image-scale feature maps (one for each facial feature listed in table 1), the FM module then recombines the feature in the same locations they were cropped from in the CE module. This image-resolution feature map is then passed to the Image Synthesis module.

The FM module is composed of 5 sub-networks, one for each facial feature. The feature vectors generated by the CE module are passed to their respective sub-networks to generate feature maps. Once the 5 feature maps are generated, the 4 facial features are inserted into the rest-of-face feature map with the following depth order:

(1) Left Eye
(2) Right Eye
(3) Nose
(4) Mouth
(5) Rest of Face

A depth order is necessary because of the overlap between the crop windows centered on each feature. A constant order allows the network to handle the boundaries between crop windows easier, as it can learn to give precedence to the facial features higher in the list.

3.1.5 The Image Synthesis Module. The Image Synthesis module is implemented as a Generative Adversarial Network [2] with two parts, the generator, which learns to generate realistic face images given the feature maps from the FM module, and the discriminator, which learns to tell real images apart from generated images and is used to guide the training of the generator.

The architecture of the generator is a vaguely autoencoder-shaped deep convolutional neural network, shown in figure 5, with ResNet blocks in the middle for additional capacity and reduce the impact of vanishing gradients.

Traditionally, GAN discriminators are binary classifiers, outputting 0 for images it thinks are generated and 1 for images it thinks are genuine (or vice versa). Here, the discriminator acts as a perceptual loss function [4], generating high-level features for each image, which are then compared to the features generated for genuine images using the mean squared error (see section 3.1.6). Traditionally, the discriminator takes in only an image to classify, but here the discriminator takes in both the image to classify and the feature maps output by the FM module. It then concatenates the two together and uses the resulting 35 channel (32+3) feature map to compute its output. This decision was made for two reasons. The first is to give the discriminator additional information to generate is output, and since the feature maps output by the FM module are high-level enough for the generator to create an image from, they are sufficient for this purpose. The second reason is to implement a skip connection between the FM module and the discriminator, partially bypassing the generator. Not only does this help with vanishing gradients, but it also helps the FM module learn independently of the generator by providing another path for gradients to backpropagate. Thus, when the generator's accuracy suffers for a few iterations during the GAN min-max training process, the FM module can still train to increase accuracy.

In addition to perceptual loss, the other major deviation from traditional GAN discriminators is the usage of multi-scale discriminators [7, 14] to pickup details at multiple levels of granularity. To

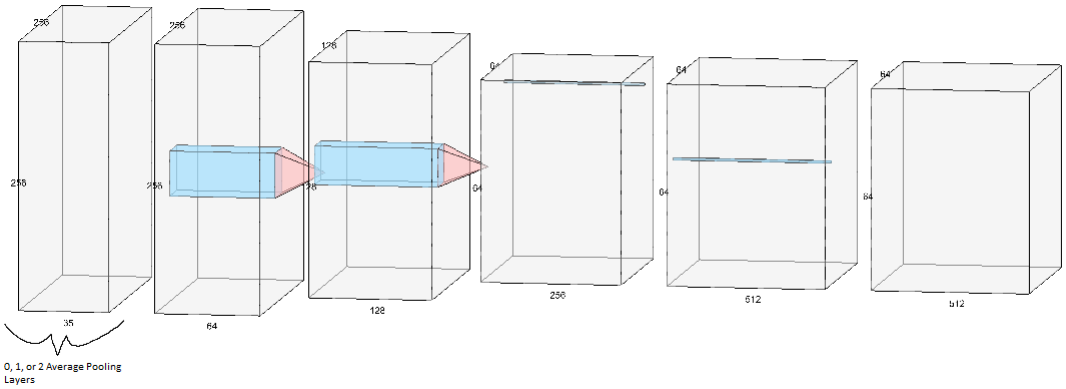

Fig. 7. The architecture of the IS module discriminator. The beginning of module has either 0, 1, or 2 average pooling layers to handle the image at multiple scales, and the rest of the network is a series of convolution blocks that reduce the image to feature maps used as a perceptual loss function.

implement this, three discriminators are trained simultaneously, with the total loss equaling the sum of the three individual losses. Each discriminator features the 5 convolution blocks at the end, only differing in the beginning layers. The first discriminator has two average pooling layers to reduce the image to 64x64, the second discriminator has one average pooling layer to reduce the image to 128x128, and the third discriminator has no pooling layers to operate at the full 256x256. A diagram explaining the usage of the discriminator can be found in figure 8.

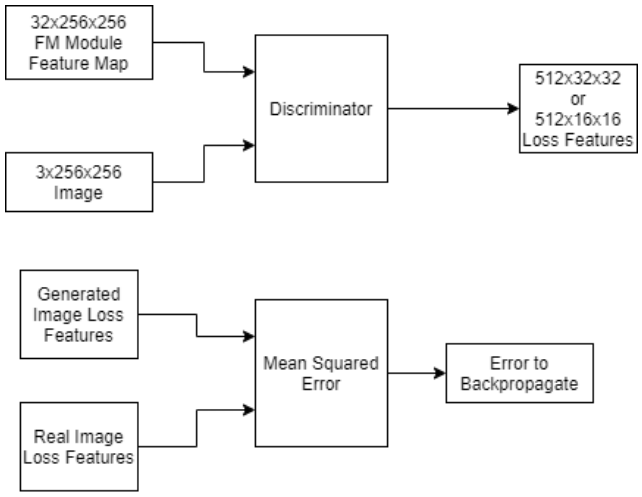

Fig. 8. How the discriminator is used. It takes in an image and the corresponding FM module features and produces a tensor representing loss features. To train, it takes the loss features from a generated image and the loss features from a real image and compares them using the mean squared error. The negative of the error is backpropagated to train the discriminator to maximize the difference between real and generated images. This is how the discriminator acts as a perceptual loss function.

The human face has details a multiple levels of granularity. Using a single discriminator would effectively pick up details at the full scale, but areas like the eyes require significantly more detail than the cheek or forehead, for example. Using three discriminators at three different scales allows the system to make better decisions based on all the details in a face image, increasing accuracy.

3.1.6 Training. The pipeline is trained in a two stage manner. In stage one, the CE autoencoders train on mini-batches of the prepared sketches, and in stage two, the FM and IS modules train on mini-batches of CE feature vectors and mini-batches of reference faces.

To train the CE autoencoders, each mini-batch of sketches is split into a component for each autoencoder (using the crop windows from table 1), and each autoencoder is trained to minimize the reconstruction loss, ℓ between the input sketch crop and the output of the CE module. This reconstruction loss takes the form of the mean squared error as seen in equation 1.

$$\ell_{\text{CE}} = \frac{1}{N}\sum[x - D(E(x))] \odot [x - D(E(x))] \tag{1}$$

where x is a mini-batch sketch crop, D is the decoder, E is the encoder, and N is the number of elements in the output image tensor.

Once the CE module is trained, the FM module and IS module are trained together while the CE module is held constant.

First, forward propagation occurs to generate images. A mini-batch of sketches is forward propagated through the encoder half of the CE module, generating 5 feature vectors. These vectors are then passed to the FM module, which transforms and joins them into 32 channel feature maps. This feature map is then passed to the generator, which generates images. This process is described by equations 2 and 3

$$m = FM(x_{\text{left eye}}, x_{\text{right eye}}, x_{\text{nose}}, x_{\text{mouth}}, x_{\text{face}}) \tag{2}$$

$$y_{\text{generated}} = G(m) \tag{3}$$

where $y_{\text{generated}}$ is the generated images, G is the generator, FM is the feature mapping module, m is the 32 channel feature map generated by the FM module, and x is an encoded component vector generated by the encoder half of the CE module.

In training stage two, loss computation, backpropagation, and weight updates occur three different times per mini-batch. Once to train the discriminator, once to train the generator with the discriminator, and once to train the generator with an L1 loss.

To train the discriminator, the generator and FM module are held constant. The discriminator is used as a perceptual loss function to generate loss feature maps. These loss feature maps (computed for both the generated and real face images) are computed by the discriminators at each of the three scales (64x64, 128x128, and 256x256) and compared using the mean squared error. The total error then becomes the sum of each discriminator's error. This process is described by the following equations:

$$z_{\text{generated}_i} = D_i(y_{\text{generated}}, m) \tag{4}$$

$$z_{\text{real}_i} = D_i(y_{\text{real}}, m) \tag{5}$$

where D is a discriminator, z is a loss feature map from the discriminator, y is an image, and m is a 32 channel feature map for the image (see equation 2). The mean squared error computation between real and generated loss features is described by the following:

$$\ell_{\text{discriminator}_i} = \frac{1}{N}\sum[z_{\text{generated}_i} - z_{\text{real}_i}] \odot [z_{\text{generated}_i} - z_{\text{real}_i}] \tag{6}$$

where N is the number of elements in the loss feature map, $\odot$ is the element-wise tensor product, and ℓ is the mean squared error between generated and real loss features.

Because the model is trained with corresponding sketch-image pairs, the same feature map m can be used as input to the discriminator for both the generated images and the real images (equations 4 and 5). Once the mean squared error between real image and generated image loss features are computed, the error to backpropagate becomes the opposite of the sum of the error for each of the three discriminators:

$$\epsilon_{\text{discriminator}} = -\sum_{i \in \{1,2,3\}} \ell_{\text{discriminator}_i} \tag{7}$$

Consequently, the error backpropagated to the generator is:

$$\epsilon_{\text{generator}} = \sum_{i \in \{1,2,3\}} \ell_{\text{discriminator}_i} \tag{8}$$

It's important to note that the error backpropagated to the discriminator is the negative sum of per-discriminator losses. This comes from the min-max game that GANs play. The objective of training the IS module GAN is to find the optimal generator, G^* such that:

$$G^* = \arg\min_G \max_{D \in \{D_1, D_2, D_3\}} V(D, G) \tag{9}$$

where V is the positive multi-discriminator loss (equation 8) parameterized by a given generator G and discriminator D.

By backpropagating the positive error to the generator, weight updates cause the generator to learn to minimize the difference between the real and generated images. Similarly, by backpropagating the negative error to the discriminator, the discriminator learns to maximize the difference between the real and generated images. This training proceeds in a zero-sum game, where the generator tries to fool the discriminator and the discriminator tries to identify the generator's images as fake.

Implementation wise, the discriminator is trained first, backpropagating gradients and updating weights, then the loss features are recomputed with the new discriminator, and then the generator is trained with the feature mapping module.

In addition to training with GAN loss, the generator and FM module are also trained with the L1 loss between the generated image and the target image as shown in equation 10.

$$\ell_{\text{L1}} = \frac{1}{N}\sum |y_{\text{generated}} - y_{\text{real}}| \tag{10}$$

The purpose of this extra weight update step is to take advantage of the available sketch-image pair dataset for more accurate image synthesis. The L1 norm is more computationally problematic than the L2 norm (mean squared error) since $f(x) = |x|$ has constant derivatives for positive and negative values, but is not differentiable at zero. These shortfalls are compensated for by the fact that minimizing the L1 norm better minimizes the euclidean distance between points in space compared to the L2 norm, and since the L1 norm is only being used as a secondary error in addition to the GAN loss, the nondifferentiability at zero won't affect the convergence of the generator since the GAN loss has no such differentiability concerns.

The network is trained in this way using the Adam optimization algorithm [5] because of its documented good performance with non-stationary learning targets (like GANs). Hyperparameters include the learning rate set to 0.0002, and the Adam-specific beta coefficients of $\beta_1 = 0.5$, and $\beta_2 = 0.999$.

3.2 Results and Analysis

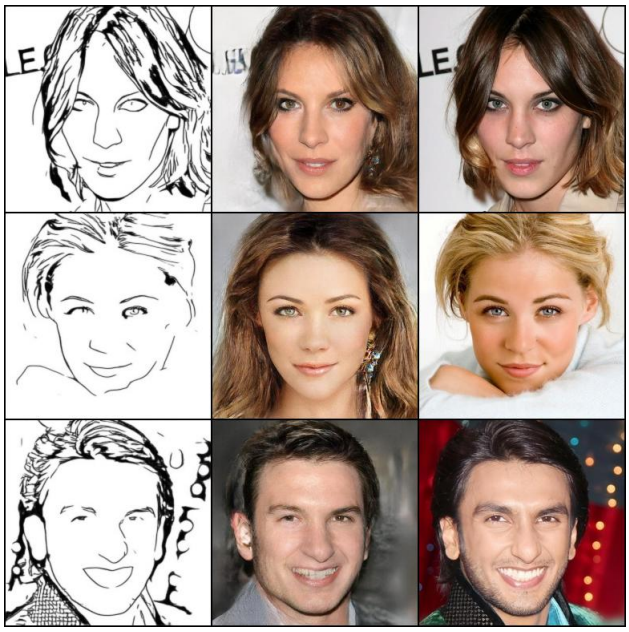

Fig. 9. Input sketches (left), generated output images (middle), and the real photo the sketch was derived from (right).

Through visual inspection, the results generated by the model are quite realistic (figures 9 and 12). While visual inspection can give a good intuition for the efficacy of a model, it is hardly a robust or quantitative measure. As it happens, measuring the quality of GAN results is an open problem in machine learning. Because our system generates new faces based on learning a distribution of human faces, direct per-pixel comparisons between outputs and labels isn't as significant of a measure of accuracy as with other model types. Regardless, there is a degree of similarity in the underlying facial structure that can be partially represented by a per-pixel error. A batch of 50 sketch-image pairs was prepared and fed to the model. The mean squared error between output image and original image was computed and the error per image is shown in figure 10. The batch had a mean error of 5658.69 and a median error of 3927.27.

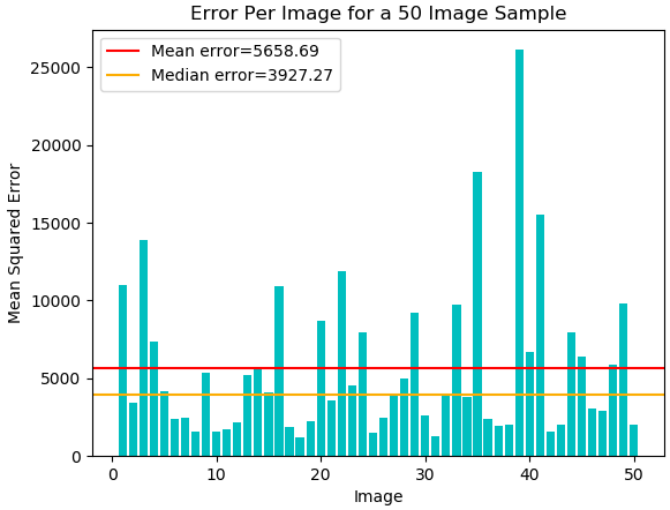

Fig. 10. Bar graph of the mean squared error between 50 output images and the real images the corresponding input sketches were derived from.

As can be seen in the graph, there was one image in the batch with an abnormally high error. This particular sketch produced a completely black image when fed into the model. This implies that some property of the input sketch results in activations of zero propagating through the model. Most likely, the tensor representing the sketch, x at some point along in the model falls within the null space of one of the layers, A, such that $Null(A) = \{v \mid Av = \vec{0}\}$ and $x \in Null(A)$. This outlier skews the mean of the errors, so the median is recorded as a more outlier-resistant measure of center.

The model implemented in this paper is quite resource intensive to train, test, and store. The development process was fraught with out-of-memory errors which required special techniques to deal with (see section 4.1). The three modules used for image image generation (CE, FM, and the IS generator) require over 5.2 GB of space when saved to disk, which doesn't include the discriminator or the space required to store gradients during training. The two-stage training process takes three to four days on an Nvidia Quadro RTX 6000. It takes about 6 seconds to predict on a single image, depending on the hardware. A bar graph of module size is depicted in figure 11.

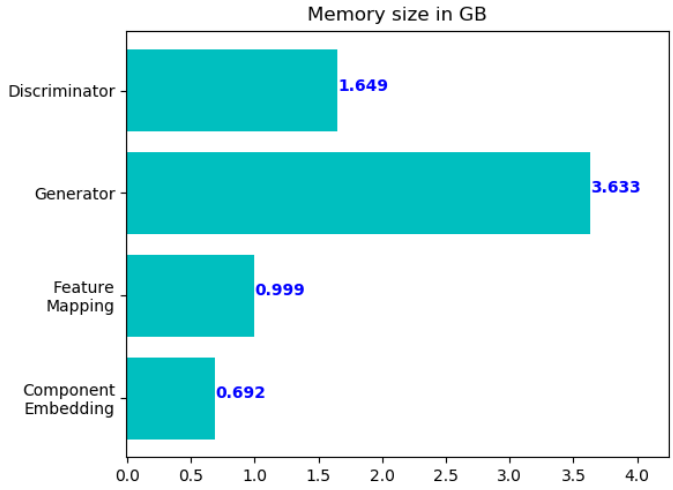

Fig. 11. Bar graph of the memory size of each module in GB.

Overall, the system is very successful at the task of sketch to image translation, and the model consistently produces images that can easily pass as real.

3.3 Applications

Our system can be utilized for a variety of practical applications, some of which are discussed below.

3.3.1 Artist Planning. Professional artists frequently draw quick sketches of their subjects to plan out the general design of what will become their final painting. Our system can be used to transform artists' initial sketches into photorealistic faces to give the artist an idea of the directions they could take their painting.

3.3.2 Law Enforcement. The most practical application of our system is use by law enforcement agencies to model sketches of suspected criminals according to eyewitness descriptions. For serious crimes, police officers usually collect suspect descriptions from eyewitnesses which then go to a police sketch artist. These sketches are frequently low-detail, and thus hard to match to potential suspects. Our system could be used to generate a variety of photorealistic images of faces based on a sketch from eyewitness descriptions, which would be far more useful to police work than low detail sketches.

4 CONCLUSION

4.1 Discussion

The network implemented in this paper is very intensive to store in memory. Significant difficulties were encountered in attempting to train the model in PyTorch. The network is so deep and there are so many feature map channels that several attempts to train the network generated errors because of an attempt to allocate 500GB

of memory. Models of this scale are pushing at the edge of video card hardware, and even running on a Quadro RTX 6000 with 23GB of VRAM proved insufficent for traditional training methods.

The solutions to these practical constraints included scaling down the input image size from the planned 512x512 pixels to 256x256 pixels, as well as implementing gradient checkpointing. Checkpointing in PyTorch trades memory for run time by recomputing gradients at each step during backpropagation. Instead of accumulating gradients in a computational graph as the forward pass occurs, PyTorch calculates each gradient right before it's needed during backpropagation. This slows down the model, but aliveiates the memory burden of storing large tensors of gradients for every computation. To further reduce memory consumption, a small batch size was used for training.

Despite the challenges, we have implemented an effective system for sketch-to-image translation, and the success of this project bodes well for other systems that infer complex details from very simple representations, showing the once again showing the inferential power of generative adversarial networks.

4.2 Future Work

The effectiveness of the system described in this paper to infer realistic details from a simplified sketch is promising for future research. Researchers wishing to expand upon this concept should attempt to train a similar architecture on general images to attempt to translate sketches of various scenes into realistic images. Such a system could have wide applications in the movie industry, as storyboards of scenes could be translated into photorealistic frames of a movie. Such a system could even be expanded to translate flip book or storyboard videos frame-by-frame into realistic looking movies. A system like this would require vast computational resources, which leads to the next major area for future research that this paper makes evident.

The memory issues encountered by our model are emblematic of a larger problem in deep learning research. New research keeps pushing the envelope of deeper and deeper networks with more and more memory intensive architectures. Without a shift to efficiency-focused research, new discoveries will stall as hardware becomes the bottleneck for deep learning. Future research should prioritize discovering new ways to make deep neural networks more efficient by reducing memory consumption and training time. By improving the efficiency of deep learning models, research can be made more accessible to those without significant institutional backing, which would drastically increase the amount of researchers working to advance the field, leading to more discoveries. It would also allow institutions with significant resources to get more value from their deep learning models, which would lead to more investment in deep learning research as a field.

REFERENCES

[1] Shu-Yu Chen, Wanchao Su, Lin Gao, Shihong Xia, and Hongbo Fu. 2020. DeepFaceDrawing: Deep Generation of Face Images from Sketches. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2020)* 39, 4 (2020), 72:1–72:16.

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. arXiv:1406.2661 [stat.ML]

[3] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2018. Image-to-Image Translation with Conditional Adversarial Networks. arXiv:1611.07004 [cs.CV]

[4] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. arXiv:1603.08155 [cs.CV]

[5] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[6] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. 2020. MaskGAN: Towards Diverse and Interactive Facial Image Manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[7] Michael Mathieu, Camille Couprie, and Yann LeCun. 2016. Deep multi-scale video prediction beyond mean square error. arXiv:1511.05440 [cs.LG]

[8] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. arXiv:1411.1784 [cs.LG]

[9] Fabio Pellacini, James A. Ferwerda, and Donald P. Greenberg. 2000. Toward a Psychophysically-Based Light Reflection Model for Image Synthesis *(SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., USA, 55–64. https://doi.org/10.1145/344779.344812

[10] randerson112358. 2020. Convert A Photo To Pencil Sketch Using Python In 12 Lines Of Code. https://python.plainenglish.io/convert-a-photo-to-pencil-sketch-using-python-in-12-lines-of-code-4346426256d4

[11] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. 2018. Mastering Sketching: Adversarial Augmentation for Structured Prediction. *ACM Transactions on Graphics (TOG)* 37, 1 (2018).

[12] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. 2016. Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup. *ACM Transactions on Graphics (SIGGRAPH)* 35, 4 (2016).

[13] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-Resolution Image Synthesis and Semantic Manipulation With Conditional GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[14] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. arXiv:1711.11585 [cs.CV]

[15] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N. Metaxas. 2017. StackGAN: Text to Photo-Realistic Image Synthesis With Stacked Generative Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Fig. 12. Supplemental results: 45 output images generated by the model.