2021

Partial Facial Re-imaging Using Generative Adversarial Networks

Derek Desentz
Wright State University

PARTIAL FACIAL RE-IMAGING USING GENERATIVE ADVERSARIAL NETWORKS

A Thesis submitted in partial fulfillment of the

requirements for a degree of

Master of Science in Computer Engineering

By

DEREK DESENTZ

B.S.C.S., Wright State University, 2019

B.S.C.E., Wright State University, 2019

2021

Wright State University

WRIGHT STATE UNIVERSITY

GRADUATE SCHOOL

April 30, 2021

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Derek Desentz ENTITLED Partial Facial Re-Imaging Using Generative Adversarial Networks BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Computer Engineering.

_____

Yong Pei, Ph.D.
Thesis Director

_____

Mateen M. Rizki, Ph.D.
Chair, Department of Computer Science and Engineering

Committee on Final Examination:

_____

Yong Pei, Ph.D. (Advisor)

_____

Jeff Clark, Ph.D. (Co-Advisor)

_____

Mateen M. Rizki, Ph.D.

_____

Barry Milligan, Ph.D.
Vice Provost for Academic Affairs
Dean of the Graduate School

ABSTRACT

Desentz, Derek. M.S.C.E. Department of Computer Science and Engineering, Wright State University, 2021. Partial Facial Re-Imaging Using Generative Adversarial Networks.

Existing facial recognition software relies heavily on using neural networks to extract key facial features to accurately classify known individuals. Some of these key features include the shape, size, and distance between an individual's eyes, nose, and mouth. When these key features cannot be extracted due to facial coverings, existing applications become inaccurate and unreliable. The accuracy and reliability of these technologies are growing concerns as the facial recognition market continues to grow at an exponential rate.

In this thesis, we have developed a web-based application service that is able to take in a partially covered face image and generate a new image of what this person could look like without any facial coverings. This service is aimed to be used as an intermediary step between obtaining partially covered face imagery and using facial recognition to accurately classify the individual. This research uses various Generative Adversarial Networks (GAN) to generate facial images of an individual based on pre-processed data extracted from the original image. The web application also allows users the ability to upload data that will be used to build a new GAN model that more accurately represents their needs. This highly scalable service will enable transfer learning and encourage a community of research to be built around this topic.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Existing Facial Recognition Approaches

Facial recognition refers to the process of using either still imagery or videos streams of an individual's face and correctly classifying their identity [1]. Applications that use facial recognition typically achieve this by first detecting a face from their given input. Facial analysis will then calculate the geometry of the face by extracting key features such as the shape and size of an individual's eyes, nose, or mouth and the distances between them. This can be done via a neural network, most commonly a Convolutional Neural Network (CNN), where these extracted features generate embeddings from known labels of the training data. Once a network has been properly trained, that network is then able to take in an unseen image and calculate the embedding. The result from the unseen test image is then compared with the results of the trained images by calculating the distances between their embedded features. The classification from the image with the closest embedding is then returned along with a confidence value based on how far away the closest embedding was from the test image [2].

Whether facial recognition is computed using neural networks or other approaches, feature extraction is the critical step in generating accurate results. The use of CNNs for facial recognition is growing in popularity due to their ability to extract features of interest and produce multiple feature maps of this data [3]. These feature maps are then able to be reduced to only the key features by implementing pooling, which improves both computational and statistical efficiency. Therefore, since feature

extraction is of such high importance, having these key features visible is of equal importance.

1.2 Facial Recognition Market Size & Growth

The global facial recognition market size as of 2020 reached USD 3.8 billion and is expected to grow to USD 8.5 billion by 2025 [4]. With applications in market shares such as Healthcare, Government/Legislative, Media/Entertainment, and Retail/E-commerce, facial recognition is becoming a fundamental practice in society today. One major use case is law enforcement utilizing facial recognition to identify criminals caught on camera. More recently, law enforcement is shifting from legacy systems to utilizing facial recognition solutions to reduce the spread of the COVID virus [4]. Use cases for facial recognition are growing rapidly, therefore, as facial recognition becomes more and more abundant in society, the accuracy and reliability of these applications become even more important.

The major issue facial recognition faces today is being unable to extract key facial features due to distorted or partially covered imagery. Unfortunately, as the use cases for facial recognition grow, so does the frequency of this major issue. For example, due to the recent COVID pandemic, society has normalized wearing masks and other facial coverings while in public. When wearing masks or any facial covering, existing facial recognition software is unable to extract the key facial features needed to classify individuals accurately and reliably. To utilize this existing software, all facial features used to generate the trained embeddings need to be able to be extracted from the image.

Without all the features, the previously trained embeddings will not map to those extracted from an image that only contains a subset of those features. Therefore, to utilize any model trained on images without facial coverings with a test images that do, these test images must be altered to show all key facial features.

2. GENERATIVE ADVERSARIAL NETWORKS

2.1 What is a Generative Adversarial Network?

A Generative Adversarial Network (GAN) is a framework introduced by Ian Goodfellow in 2014 that can generate realistic "fake" data based on similarly trained data. This generative approach simultaneously trains two separate models (neural networks) that use the results from one another to properly learn and adjust their weights. These two models are known as the Generator and the Discriminator, each with opposing objectives. The generative model G captures the data distribution, and the discriminative model D estimates the probability that the result from G came from the training data [5].

2.2 Generator

The goal of the Generator is to maximize the probability the Discriminator makes a mistake. This means that the Generator is constantly trying to "fool" the Discriminator by continuously learning to generate more and more realistic data given the training dataset. By itself, the Generator can be any type of neural network. The Generator proposed by Goodfellow was a simple multilayer perceptron or artificial neural network. The input layer of the Generator is just random noise of an arbitrary dimension size. All hidden layers are subject to change to best fit a model for the desired purpose, while the output of the Generator must be same dimensionality of the desired image.

2.3 Discriminator

The goal of the Discriminator is to continuously learn how to accurately distinguish between data from the training set and the results from the Generator.

The Discriminator is also a simple multilayer neural network, with an input equal to the dimensions of the output of the Generator. All hidden layers are also subject to change while the output layer will have a single output node. This output node will be a prediction score (value between 0 – 1) of how "realistic" the Generators result was.

2.4 Adversarial Learning

What allows these two networks to work in unison is the fact they are playing a two-player minimax game with the value function V (G, D) [5]. This is done by training D to maximize the probability of correctly classifying both the train example and the sample from G while simultaneously training G to minimize the error of D.

Therefore,

$$\text{Error Function of G: } -\ln(D(G(\zeta))$$

$$\text{Error Function of D: } -\ln(1 - D(G(\zeta))$$

[5]

If the Discriminator is "too good" and the results always yield that the Generator data is fake, then the GAN will overfit. For example, let us say that the GAN was being trained to generate faces from a given dataset, and eventually the Generator learned enough to make a very realistic image. If this image was not an exact copy of any image found in the training dataset, and the Discriminator classified the image as fake, the Generator will continuously learn until it makes exact copies of images in the training set. This creates a problem since the purpose of this GAN is to produce images of faces that could exist, not generate copies of ones that already do. In building a GAN that will be

used as a predictive tool, it is imperative to not overfit the GAN model, therefore, neither the Generator nor Discriminator can have near 100% accuracy or loss near 0.

2.5 Types of GANs

In years since this adversarial network was introduced, various other types GANs have been developed using different types of neural networks for both the Generator and Discriminator. Most notably are Deep Convolutional GANs (DCGAN) which are composed of two CNNs for the Generator and Discriminator [6]. DCGANs become much more useful when working with high resolution data because the additional features that can be extracted from higher quality imagery can be better utilized in a CNN. For this initial research, a traditional GAN using two ANNs will be used to better demonstrate the entire process of partial facial re-imaging. This process can later easily substitute a DCGAN for a traditional GAN if higher resolution imagery is being used.

3. FACIAL IMAGERY

3.1 Facial Imagery Used in Existing Facial Recognition

In existing facial recognition applications, models are typically trained with images from a wide variety of resolution (pixel height x width). The two main factors that affect how well a model can be trained are the absolute resolution of the face in the image, and the face size relative to the total size of the image [7]. Therefore, many algorithms use face detection on an image to only train on the subset of the image that contains the face. This ensures that no features in the background of the image are being used in the embedding.

Facial recognition models perform better when the image has higher resolution, and most models used in real world applications today are trained with images that have at least 320 pixels in width and 240 pixels in height [7]. Although models can be trained using lower resolution, the accuracy of these model will suffer on larger datasets due to less distinguishable features able to be extracted.

3.2 Facial Imagery in GANs

Unlike facial recognition training, when training a GAN, the resolution used throughout does not affect the model's ability to produce realistic results. This is because the effectiveness of a GAN only depends on the images used to train it. This means that a GAN can generate low- or high-resolution data depending on the desired results. Therefore, a GAN can easily be configured to match the same resolution of any size training set available. The only caveat being the higher dimensionality of images used to

train a GAN, the longer the training process will take. In both the Generator and Discriminator networks, a fully connected multilayer neural network is being used with either output or input neurons equal to the number of pixels in the desired image. For example, when training with 32x32 RBG images the output size of the Generator and the input size of the Discriminator would each have 3,072 neurons (32 x 32 x 3). If 1280x720 RBG images are being trained with this number increase to 2,764,800 neurons. This means there would be 900 times more computations required at each layer in these networks, and typically each subsequent hidden layer either increases or decreases proportionally towards its desired output. This means that all hidden layers in a properly train GAN will also have more neurons when working with high resolution images. This exponential growth in neurons cause exponential growth in computation time.

3.3. Facial Imagery Used in Research

When learning new techniques and approaches to utilizing different types of neural network models, it is very beneficial to simplify the research and the time it takes to build new models by using lower resolution imagery. When training GAN models it can take upwards of an hour just to train one simplified model. Therefore, this research will be done using 32x32 RBG images to train the different GAN models. These images will represent an 8-bit styled face that have been custom generated for this research.

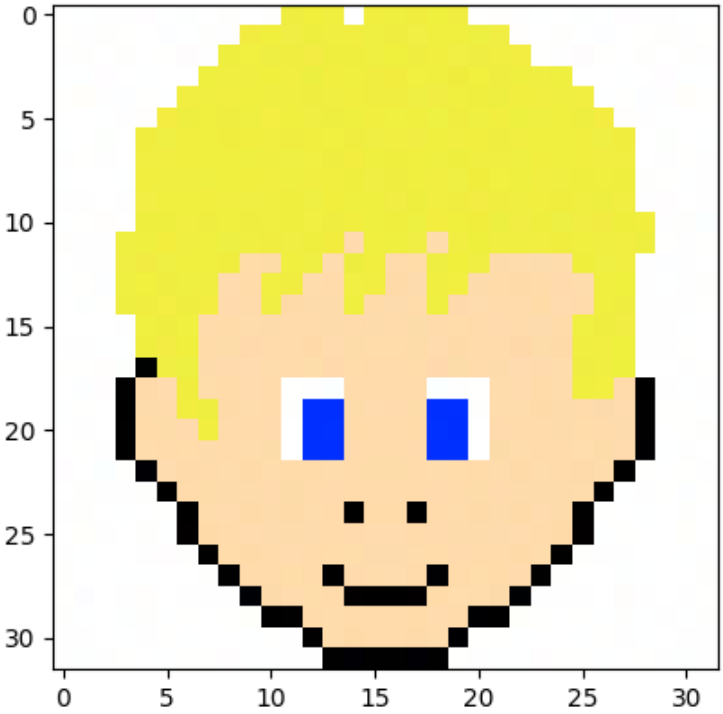

Figure 1- Example training image with easily extractable features

In Figure 1, you can see the 8-bit styles representation of a face with distinguishable key features such as eye shape, eye color, nose shape, mouth shape, skin color, hair color, and hair style. Using 8-bit styled images of faces allows these key features to be much more distinguishable as opposed to lower resolution images of a real face like shown in Figure 2.

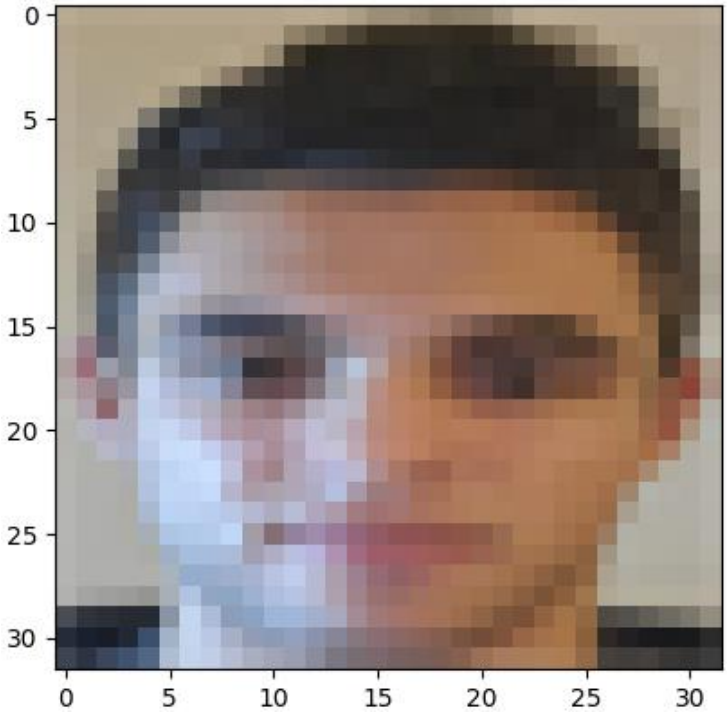

Figure 2 - Example training image with non-easily extractable features

By using images like Figure 1, the GAN can train on much lower resolution images that still have all the key features of a face easily extractable. If the GAN was to train on images like those found in Figure 2, then the results from the GAN would also generate incredibly blurry images that could not be used in visual analysis. Using 8-bit styled imagery allows for rapid prototyping of different GAN model configurations.

4. PARTIAL FACIAL RE-IMAGING USING GANS

4.1 Partial Facial Re-imaging Approach

Partial Facial Re-imaging refers to the action of taking a subset of data in a facial image and replacing it with new data while all other data values in the image remain the same. Partial Facial Re-imaging can be used to take a picture of an individual with any type of facial covering and return a resulting image of that same individual without any facial covering. To accomplish this, a GAN will be trained on similar faces to that found in the original image, but without any facial coverings. The result of the GAN will return a generated "fake" image of an individual that can be superimposed into the original image. Since the result of the GAN will always be a non-real entity, only the unknown or covered portions of the original image will be replaced with the corresponding mapping of the GAN result. This will produce a best guess prediction of what that induvial would look like without any facial coverings.

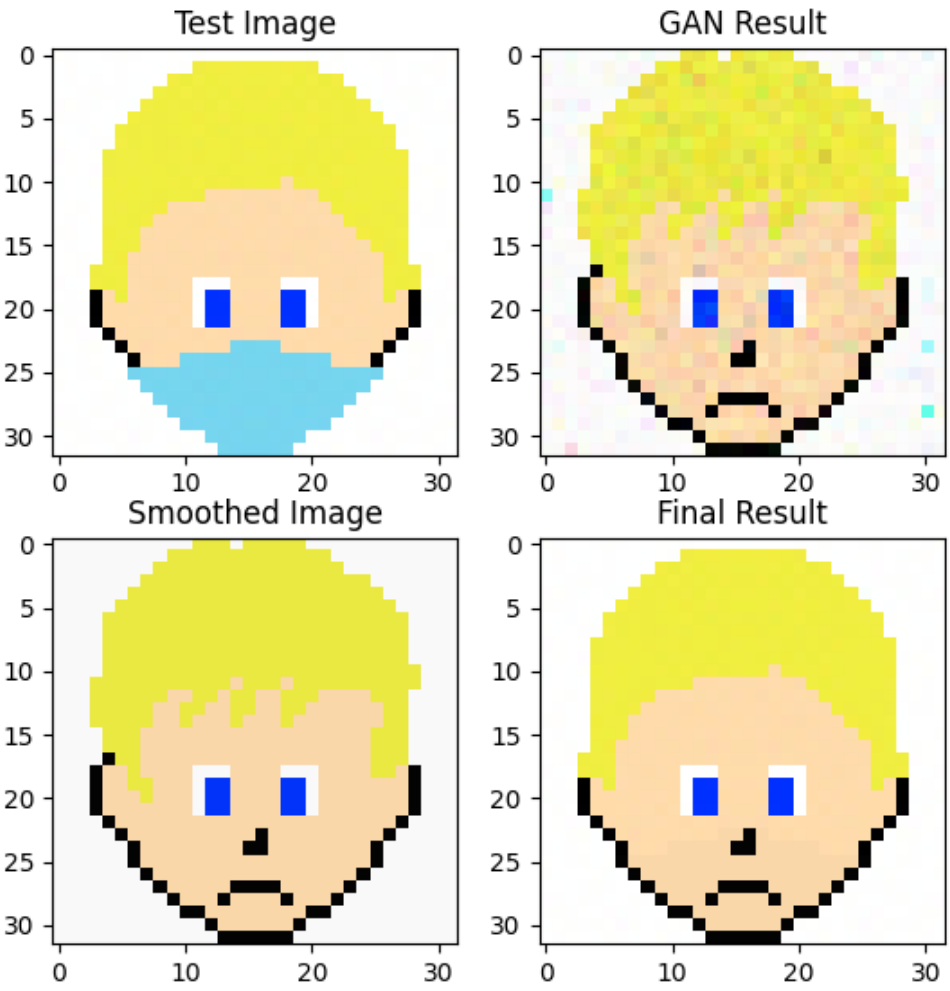

Figure 3 – Partial Facial Re-imaging result with perfect resolution

Figure 3 shows the result of the Partial Facial Re-imaging process with perfect resolution. This figure shows taking a test image with a mask as a facial covering and returning a GAN result of a face that was generated from a dataset of faces that contain similar features to those found in the test image. The features extracted from the test image were the color of the hair, eyes, and skin. The GAN result can then be smoothed using a K-Means approach. This approach groups all the pixels in the image into several clusters and then takes the average of every pixel in that

cluster to that average pixel value. This can scale well to high resolution images, because as you increase the K value in the K-Means algorithm more clusters will be created. With a larger variety of colors, the images will then look less animated and closer to a realistic image. After the smoothing has taken place, the mask clipping will calculate which pixels in the test image need to be updated. Once a clipping mask has be created, only the pixels from the smoothed image will be superimposed onto the final result. Figure 3 shows that the nose and mouth structure from the GAN was placed into final result along with the correct skin tone of the face. It can be verified that only a subset of the data from the GAN result was placed into the final result because of the hair style differences shown between the test image and the GAN result. The final result kept the hair of the test image, and only superimposed on the pixels that was needed.

4.2 Key Factors

A properly trained GAN can generate data that represents anything found in the training set it was developed with. For example, if a GAN were trained with facial images of every person in a neighborhood, the results from this GAN would produce images of people from all different races, with all different shapes and sizes of facial features. Since the input of the Generator is random noise, the result of a GAN will also be random. Therefore, it would be unreliable to always return a desired facial feature from this GAN since a single feature is not guaranteed to be found amongst all samples of the training dataset.

Since the result of a GAN is random, it is imperative to train a model with images that all contain a desired facial feature. This research primarily focuses on the key feature of skin tone. Skin tone is a major key facial feature that must be consistent across

the entire face. If the top portion of a face has much lighter skin tone than the bottom portion, the face would not look realistic.

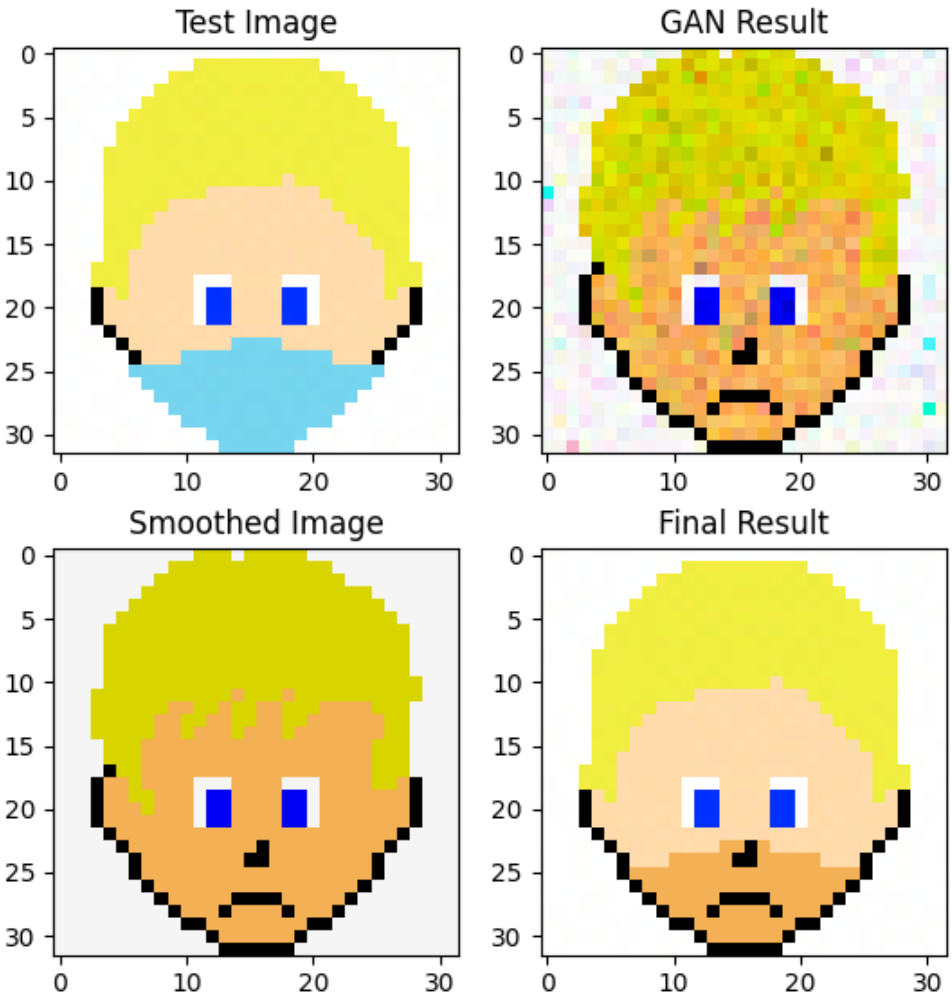

Figure 4 - Example result using a GAN model trained on different skin tone.

Figure 4 shows an example of utilizing a GAN that was trained on images of a different skin tone than the test image. Although the GAN result depicted a realistic 8-bit face in terms of facial features, this result is not realistic due to the skin tone mismatch. This shows the importance of training models to suit the desired outcome. Since we can rapidly design new models, we can prepare a wide array of different GAN models and

track meta data on these models that can later be used to select the model that best fits the test image. The meta data for a model refers to the key facial features that the model is guaranteed to produce. For example, the model used in Figure 4 would have meta data on the skin tone it produced, but it would not have any meta data on the shapes or sizes of the other key facial features.

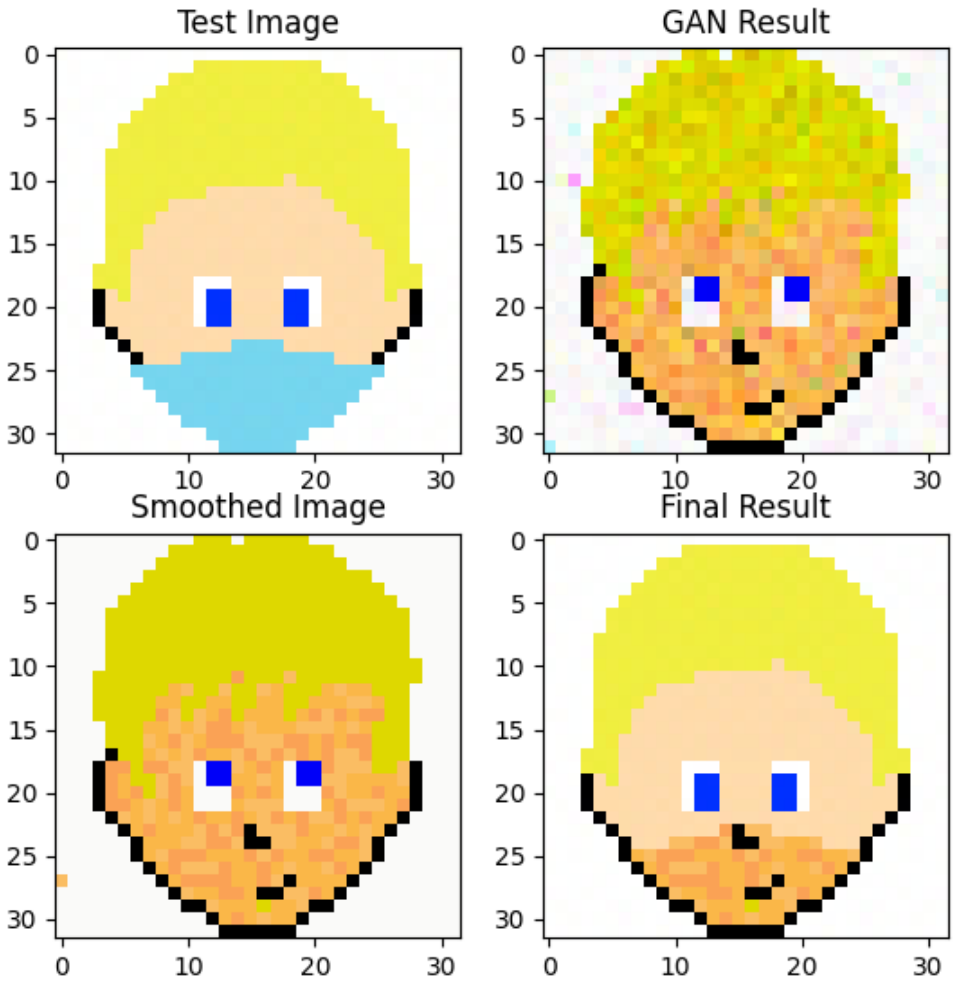

Figure 5 - Example of guaranteed meta data on a model.

Figure 5 shows the results of the exact same test image and model used in Figure 4. The only difference is that a different sample was selected from the output of the GAN. The GAN can generate as many "fake" images as needed; therefore, different samples can be selected if one sample fits the test image better than another. Comparing the two GAN Results in Figure 4 and 5 you can see that each result has a different eye, nose, and mouth shape. The only similarity is the skin and hair tone. This highlights the importance of properly understanding the training set being used to develop a particular GAN model.

A separate model should be designed specifically for any set of facial features that are a known desire. For example, if yielding a result of an individual smiling is important, the GAN should only train on smiling faces. Similarly, if a smiling person with a beard is of interest, the appropriate model should be only bearded, smiling persons. If a subset of this training data were of smiling faces without beards, the GAN result would guarantee a smile but there would be a small chance the result does not have a beard.

5. DEVELOPMENT OF A GAN

The implementation of the GAN for this research was created using the Tensorflow Keras library to build out multiple multi-layer neural networks. The overall GAN consists of 3 total models: the Generator, the Discriminator, and the combined model known as the GAN. Both the Generator and Discriminator are built using the Sequential model from Keras. The Sequential model linearly stacks layers using the output from one layer as the input to the next. There will be three main types of layers that will be used throughout each model.

Layer	Functionality
Dense	Fully Connected ANN Layer: Fully connected NN Layers added after flattening the results from the Convolution Layers in the network. The Dense layers are used to down sample the output to an output space that can be correlated to the labels of the sample data.
BatchNormalization	Normalization Layer: Applies a transformation to the data that maintains a near zero mean and zero variance of the output.
LeakyReLU	Rectified Linear Unit: Allows small gradients to when a unit not active, creates a dampening effect on non-active units.

Table 1 – Keras API Layers

5.1 Designing the Generator

In designing the Generator, only the output layer is not subject to change. The output layer must be a Dense layer with 3,072 neurons due to the known image dimensions of the training images (32 x 32 x 3). The activation function on this layer is set to use tanh to allow for neurons values between -1 and 1. This allows the LeakyReLU layers used throughout each model to be able to dampen the neurons with values less than 0.

The input layer to the Generator is a Dense layer with 512 neurons with an input dimension of 100. These 100 initial nodes will always be random noise values. By using random noise every time as the input, the Generator will continuously generate different images since the weights at each node will always have a different effect on the random noise input. All subsequent layers follow the general pattern of: Dense > LeakyReLU > BatchNormalization. The final network structure of the Generator is the following:

Layer Type	Output Shape	Parameters
Dense	[512]	51712
LeakyReLU	[512]	0
BatchNormalization	[512]	2048
Dense	[512]	262656
LeakyReLU	[512]	0
BatchNormalization	[512]	2048
Dense	[1024]	525312
LeakyReLU	[1024]	0
BatchNormalization	[1024]	4096
Dense	[3072]	3148800
Reshape	[32 32 3]	0

Table 2 – Generator Layers

This network configuration results in 3,996, 672 total parameters, with 3,992,576 of those parameters being trainable and 4,096 being non-trainable.

5.2 Designing the Discriminator

In designing the Discriminator, both the input and output layers are known and not subject to change. The input layer to the Discriminator must have the same number of neurons as the output of the Generator, which is a total of 3072 neurons. The output of the Discriminator must have a single neuron with the sigmoid activation function to produce a value between 0 and 1 that will be used as the percent likelihood that the image being tested if real. All the subsequent layers follow the general pattern of: Dense > LeakyReLU. The final network structure of the Discriminator is the following:

Layer Type	Output Shape	Parameters
Flatten	[3072]	0
Dense	[1024]	3146752
LeakyReLU	[1024]	0
Dense	[256]	262400
LeakyReLU	[256]	0
Dense	[64]	16448
LeakyReLU	[64]	0
Dense	[1]	65

Table 3 – Discriminator Layers

This network configuration results in 3,425,665 total parameters, with all parameters being trainable.

5.3 Training the GAN

The combined GAN model is created using the same input as the Generator and the same output of the Discriminator. Creating a combined GAN model allows for the overall loss of the GAN to be monitored as the GAN trains. The training process of the GAN begins by accessing the directory of images the GAN is to be trained with. These images are then converted to an array like structure to be normalized and compared with the results of the Generator. The training process beings by creating random, normalized noise in the shape of the input of the Generator [100]. Using the built in Keras function to

train a Sequential model, the combined GAN model is then trained using this noise as input and a value of 1 as the output. Typically, when training a GAN model, the expected output is set to ~0.9 instead of 1 to ensure that the model does not overfit. Since these GANs are training on a dataset of very similar images the expected output value can remain a 1 to ensure the desired meta data for this model is present. After the combined model finishes the forward pass and back propagation for the current epoch, the function will return the loss associate with the results.

The next step is to predict the Generator with the same noise used as the input to the combined model. This will produce an array of images equal to the batch size used during training. These generated results are then used to do batch training on the discriminator, using the generated images as the input and a 0 value as the expected output. This allows the discriminator to learn how fake images appear. The discriminator is then trained again with real images from the training dataset provided, but with an expected output value of 1. Similarly, this allows the discriminator to learn how real images appear. The loss from each step of training the discriminator is also collected to visualize the learning over epochs.

5.4 Results from the GAN Training

The first GAN model that was developed was trained on 125 images of 8-bit characters like Figure 1. In all these images the skin tone, eye color, hair style, hair color, and shape of the head remain the same amongst all images in the training set. All these features would be used as the meta data for this model.

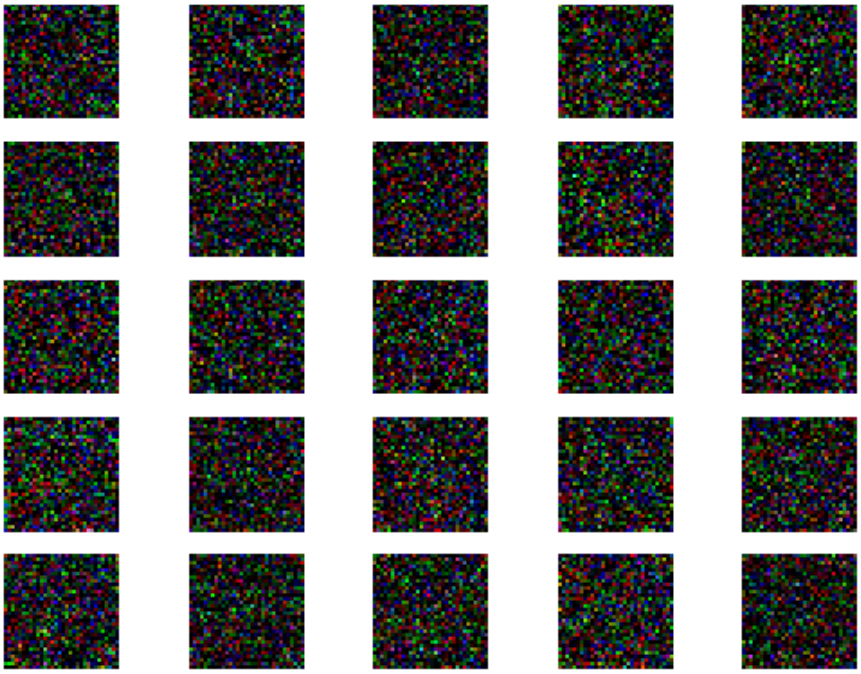

Figure 6 - Sample Generator results at the first epoch

Figure 6 shows the results of the Generator after the first epoch of training. As previously mentioned, the input to the Generator is simply just noise, therefore the output resulted in a [32x32x3] image of noise.

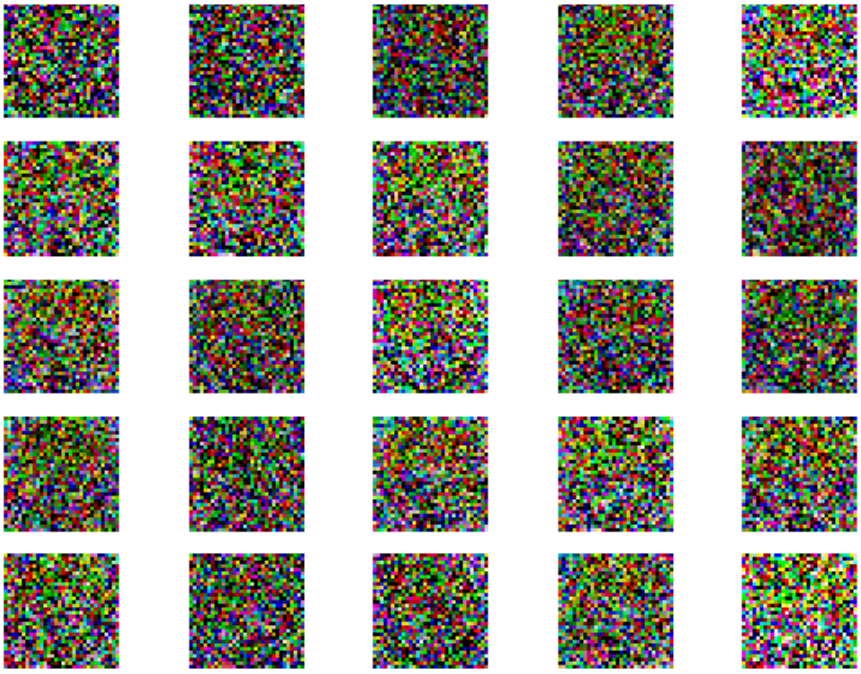

Figure 7 - Sample Generator results after epoch 200

By just the 200th epoch the GAN already began to learn the color and general structure of the hair. This is to be expected since these were part of the meta data attributes for this first model. Since all the images had these key features, these were the first features the GAN learned.

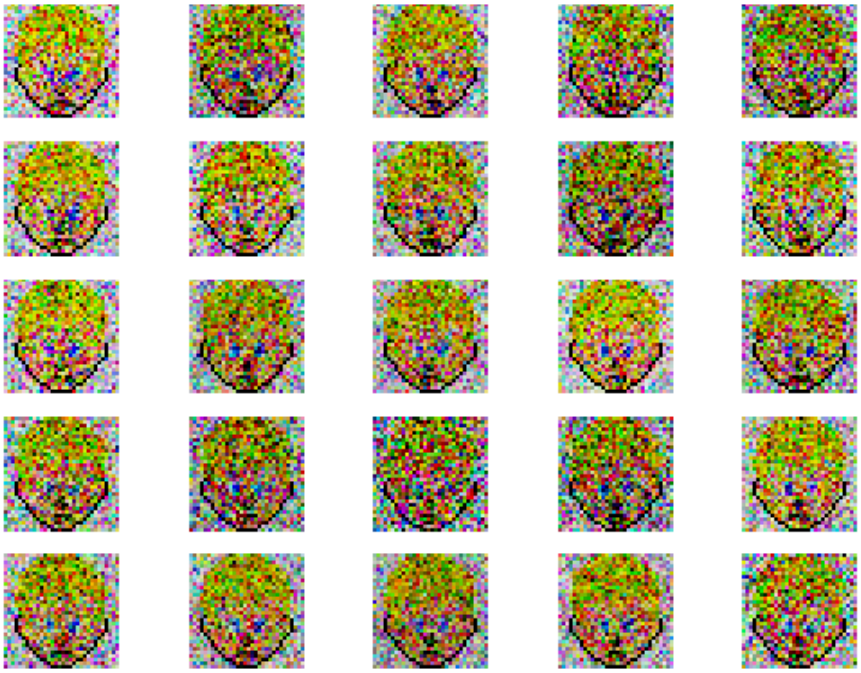

Figure 8 - Sample Generator results after epoch 700

Figure 8 shows that by the 700th epoch the shape and location of the head remains constant throughout all generated images. The shape and color of the hair has become more prominent, and the final meta data attribute of blue eyes started to show. There is still too much noise in these images to ever "fool" the Discriminator, so the learn rate remains high.

Figure 9 - Sample Generator results after epoch 2,000

Figure 9 shows that by the 2000th epoch the noise in the background of the image has almost been completely removed. Visual analysis shows that some of these images could be considered realistic due to the head shape and average color of each pixel. Although all images still have quite a bit of noise, all the key facial features could be extracted from all these images. It would be around this time that the Discriminator accuracy for real images would begin to be decrease while the accuracy on fake images would increase. Since this Discriminator has not had much time to learn after only 2000 epochs, its ability to accurately distinguish between real and fake images would still not be very accurate, and therefore it is easier to "fool" with semi-realistic results.

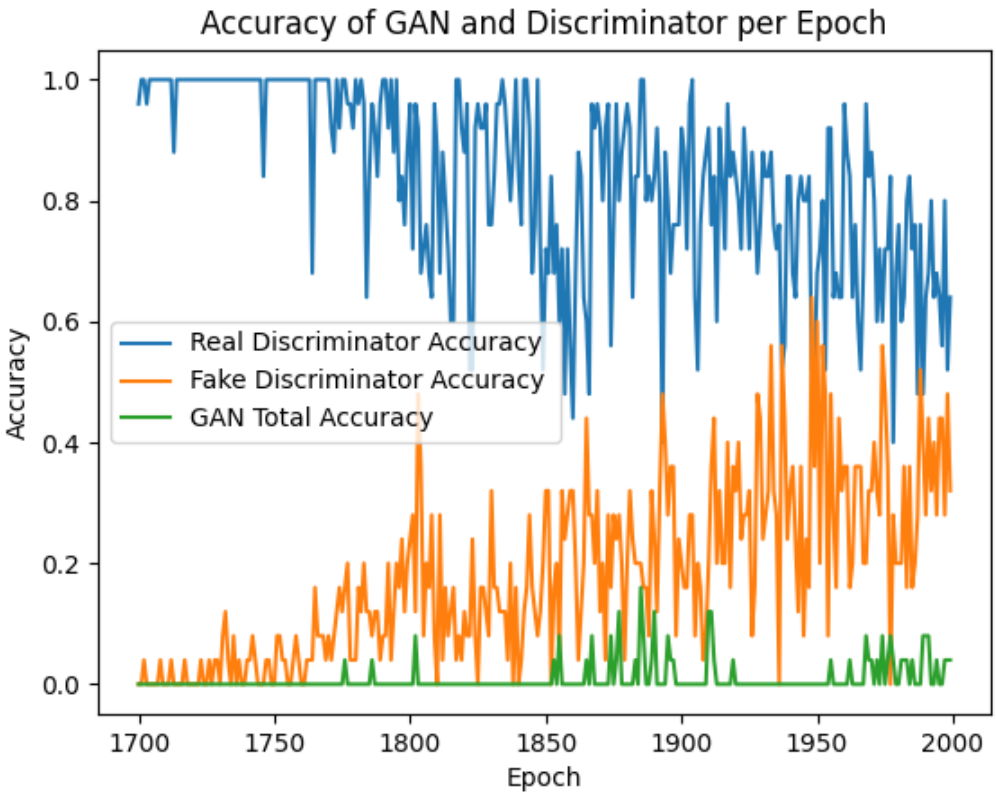

Figure 10 - Accuracy results at epoch 2,000

As expected, Figure 10 shows that the accuracy of the Discriminator for real images is beginning to trend downwards from 100%, while the accuracy for the Discriminator on fake images is beginning to trend upwards from 0%. This data shows that around the 1750th epoch the Discriminator started to get "fooled" more frequently by the Generator. The total accuracy of the GAN remains near zero since the Generator is only just now starting to fool the Discriminator.

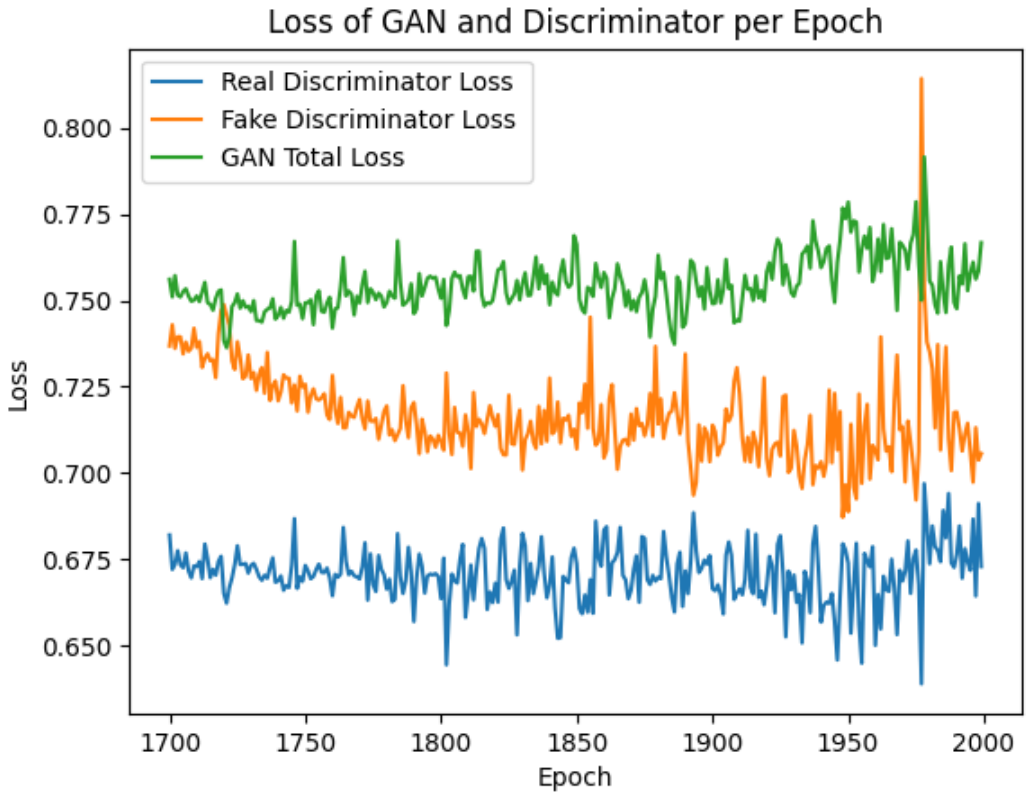

Figure 11 - Loss results at epoch 2,000

Similarly, Figure 11 shows that the total loss of the GAN at the 2,000th epoch is higher than that of the Discriminator values. With large loss values of the Discriminator for both datasets, the combined model of the GAN is still consistently far from the expected results. Since the loss of fake images is greater than the loss on real images, the GAN is classifying more images to be "real" than "fake". After only 2,000 epochs this is still to be expected, since the GAN has had enough time to generate key features, but the Discriminator has not had enough time to separate noisy images from real ones.

Now that the main facial structures of the images have been learned, the learn rate will decrease and more local learning can take place to reduce the noise of the generated faces.

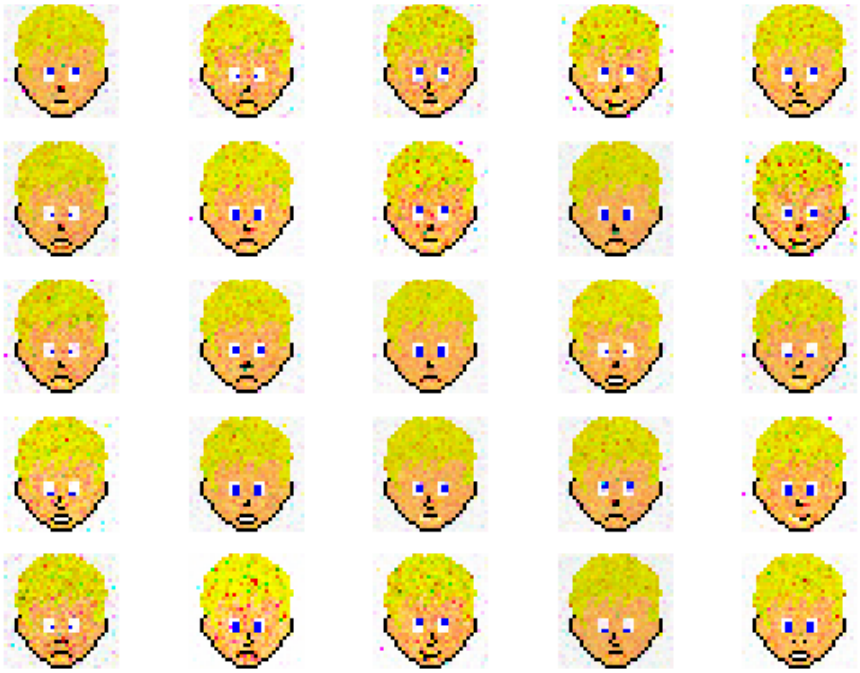

Figure 12 - Sample Generator results after epoch 54,000

Figure 12 shows highly realistic results with low noise by the 54,000th epoch. Results from the GAN at this point could be smoothed and potentially used to in the partial facial re-imaging process. It would be expected that around this iteration the Discriminator accuracy for both the real and fake images should both be much higher than before.

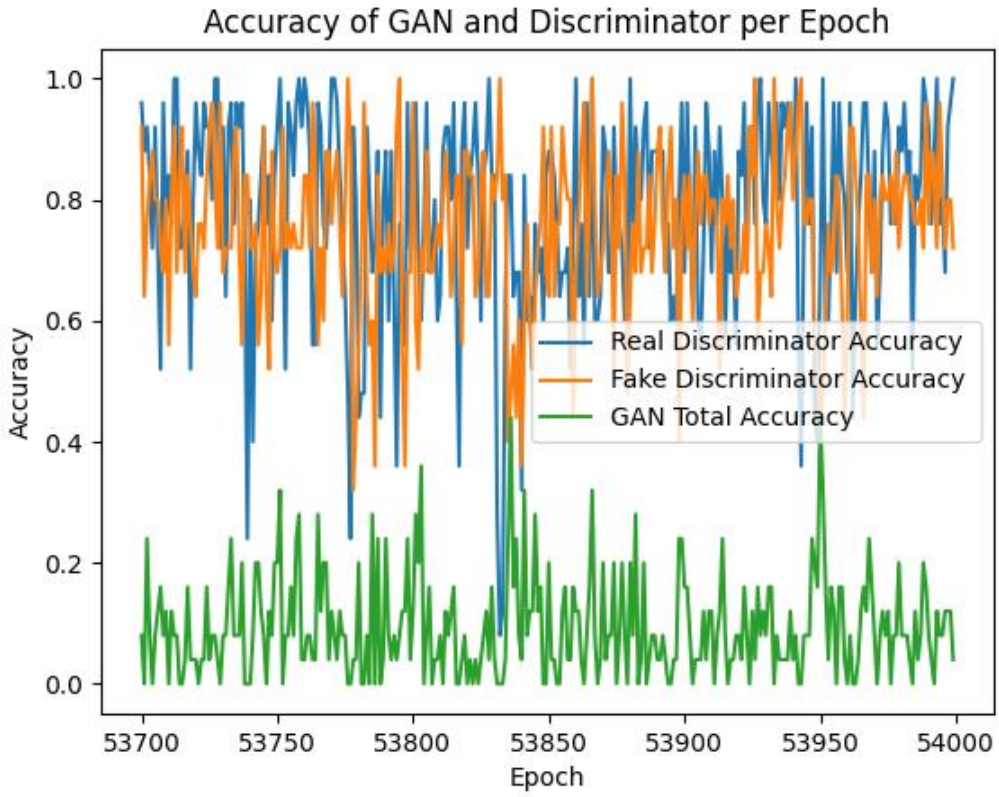

Figure 13 – Accuracy results at epoch 54,000

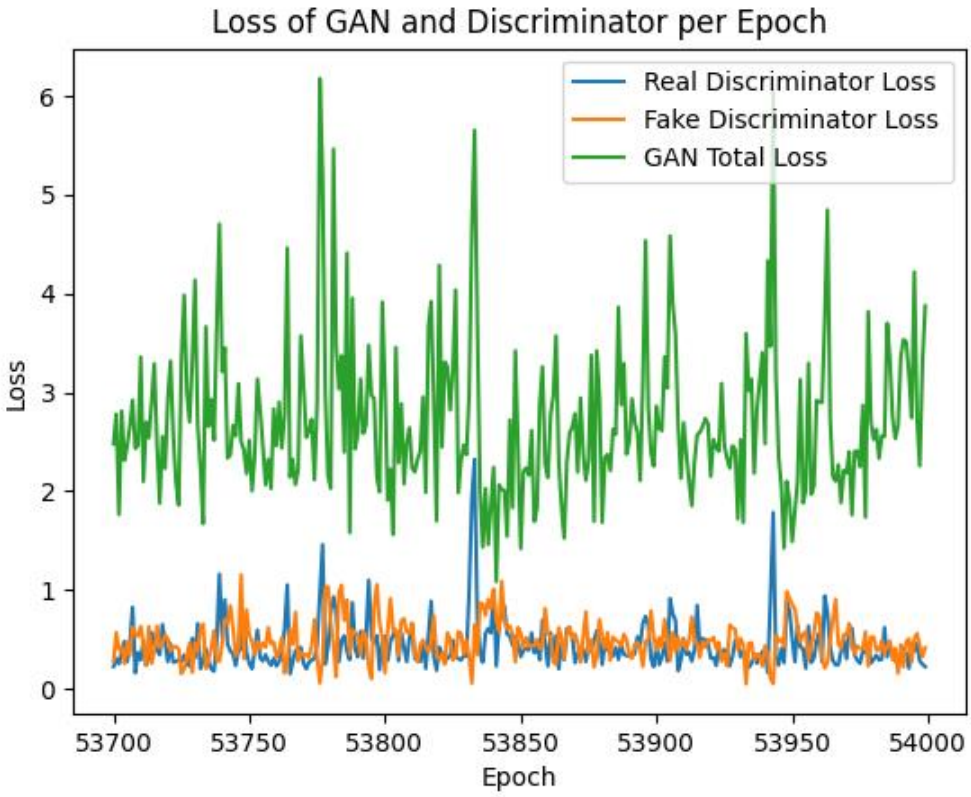

Figure 14 – Loss results at epoch 54,000

Figures 13 and 14 confirm this expectation showing that the accuracy on both the Generator and Discriminator are closer to the ideal range of ~90%. This is an ideal target for Generator and Discriminator models because it prevents overfitting while still having the ability to generate realistic results. As the loss for on the Discriminator for both real and fake images approach ~0.5, the total loss of the GAN will spike. This is because the GAN is being able to generate an equal amount of "fake" and "realistic" images so the loss on both the Generator and Discriminator are constantly fluctuating.

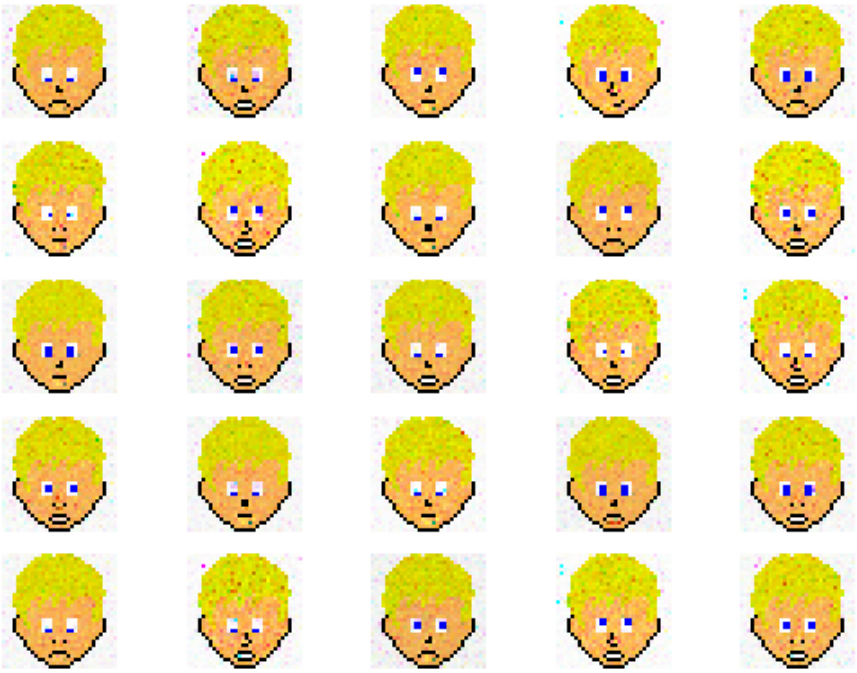

Figure 15 - Final Generator results after epoch 98,000

Figure 15 shows the results of the Generator after its training. Visual analysis can confirm that these images look highly realistic for this dataset. It is important to note that these images depict a slightly darker skin and hair tone than the training set. This is because the images consist of values between -1 and 1 due to the tanh activation function on the Generator. When plotting with Matplotlib, imagery data is expected to be in a range of [0 – 1] or [0 – 255]. If it is not, the data is clipped to fit this range. This produces this off-color visualization effect. To see the true values of the GAN the image data simply needs to be normalized between the values of [0 – 255] which is done during post-processing.

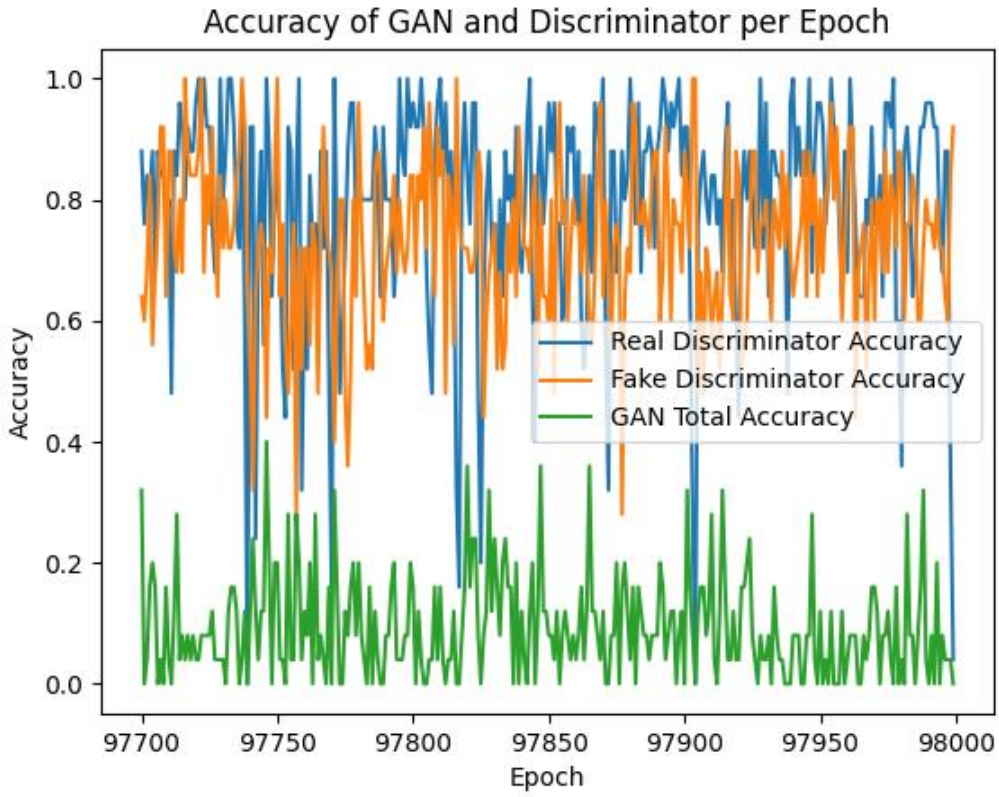

Figure 16 – Final Accuracy results at epoch 98,000

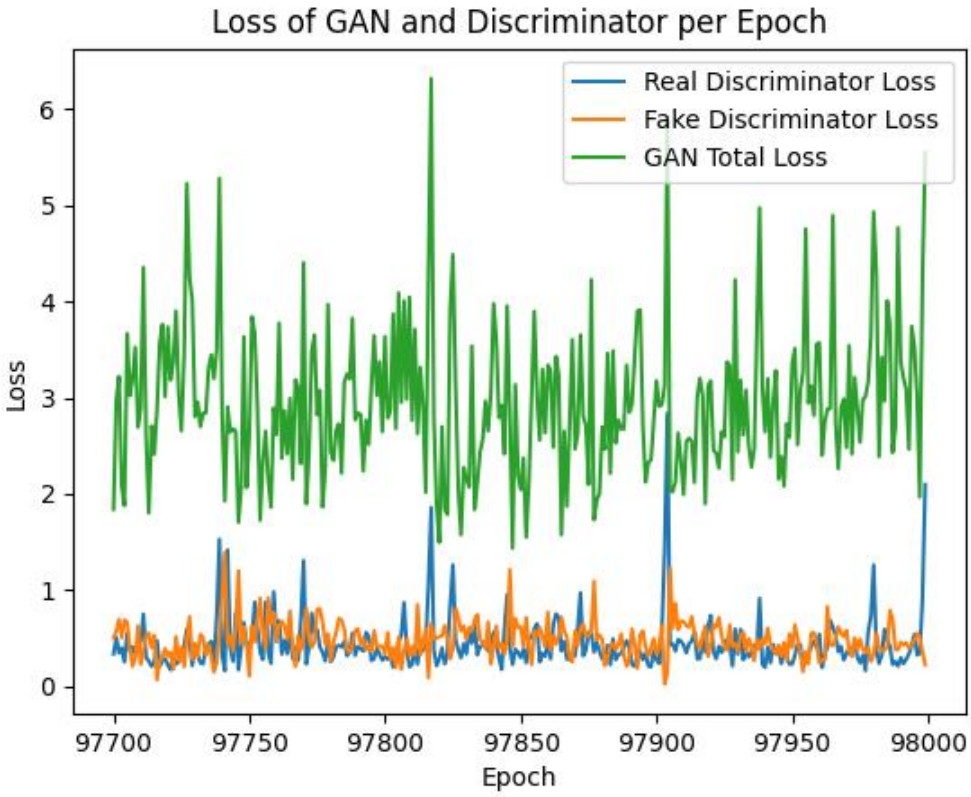

Figure 17 – Final Loss results at epoch 98,000

Figures 16 and 17 show only slight improvement compared to epoch 54,000 which is to be expected when comparing the differences between Figures 12 and 15. The noise reduction between the two epochs were the only difference which did not have a large effect on the overall accuracy or loss but does have a larger effect on visual analysis which is why the GAN was trained for so much longer.

6. PREPROCESSING USE CASE IMAGERY

With GAN models properly built, the next step of the partial facial re-imaging process is to preprocess the use case image that has a facial covering so that the appropriate model can be selected to yield the best results.

6.1 Determining Facial Coverings

Before a model is selected, the first step to preprocessing a use case image is to determine which pixels of the image contain a facial covering. The following examples will highlight finding a mask on the test characters. This step itself can be considered an entire project on its own. Many applications exist today can determine if a face has a mask or not, but very few of these solutions map out the pixels of the image that contain part of the mask. This step done properly will be added to future work, but for this initial research a simplified version of face mask extraction is being done by simply parsing the image and mapping the image pixels that contain the certain color. In real world scenarios this may not be a viable solution if the masked individual either has the same mask color shown anywhere else on their face, such as a hat, or if they are wearing a mask that does not fall within the color boundary the application is looking for.

6.2 Extracting Known Features and Obtaining Meta Data

Like the face mask extraction, obtaining meta data can also grow to be a large task. Depending on the number of models a user may have available, every unique type of meta data found within those models must be determined to select the best available model. One approach to doing this is to train a CNN for each unique GAN model. This CNN would be able to classify the probability that meta data feature is found in the test image. This research focused on building a CNN to classify which skin tone the test image most likely has. This CNN will be trained with every image used to train all GAN models available to the user. In this case three models have been trained, each with a unique skin tone found in every sample of their training data.

The CNN used to determine the appropriate GAN model is inspired by one of the more commonly known CNN models, VGG16 [8]. This model follows a typical layer pattern such as:

Conv > Conv > Pool > Conv > Conv > Pool > … > Flatten > Dense > Dense > Dense

This layer pattern has been proven to work well with high resolution images with RGB channels, therefore, for the purpose of this research the since the images are much lower resolution, a fewer number of Conv > Pool cycles will be used.

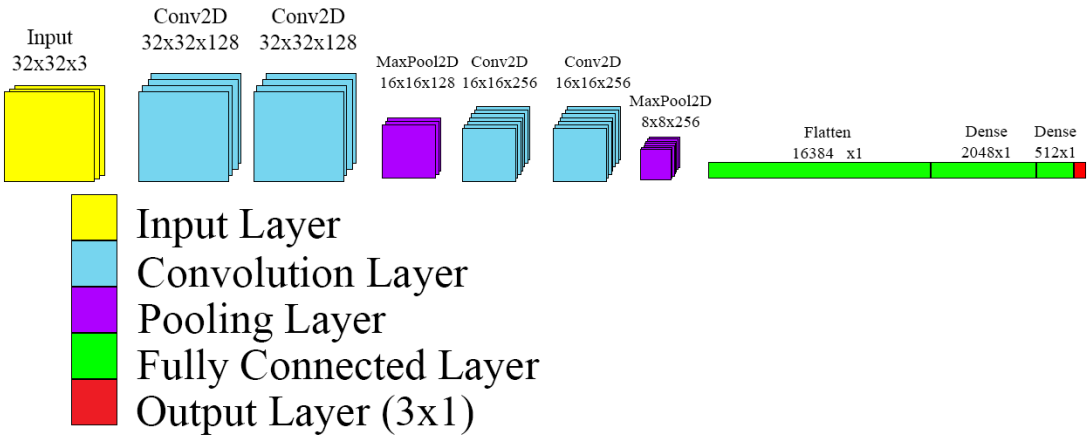

Figure 18 - Final CNN Model Structure for Model Classification

Layer Type	Output Shape	Parameters
Conv2D	[32 32 128]	3584
BatchNormalization	[32 32 128]	512
Conv2D	[32 32 128]	147584
BatchNormalization	[32 32 128]	512
MaxPooling2D	[16 16 128]	0
Conv2D	[16 16 256]	295168
BatchNormalization	[16 16 256]	1024
Conv2D	[16 16 256]	590080
BatchNormalization	[16 16 256]	1024
MaxPooling2D	[8 8 256]	0
Flatten	[16384]	0
Dense	[2048]	33556480
Dense	[512]	1049088
Dense	[3]	1539

Table 4 – CNN Layers

This network configuration resulted in 33,646,595 Total Parameters, with 33,645,059 being trainable and 1,536 being non-trainable.

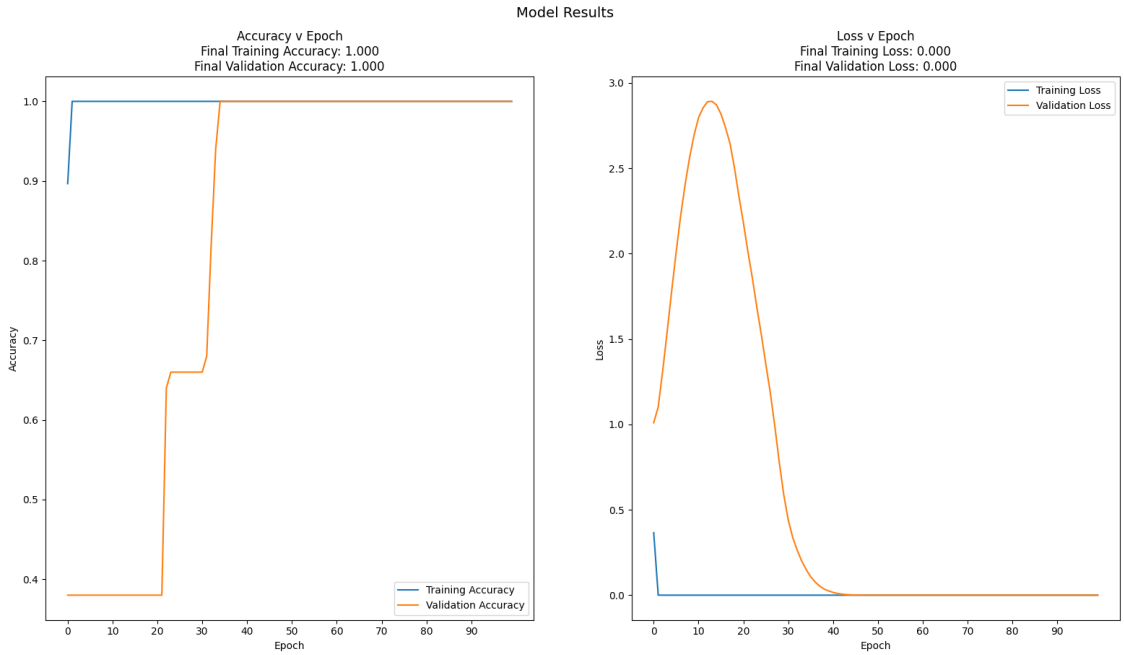

Figure 19 - Training and Validation Results for Model Classification

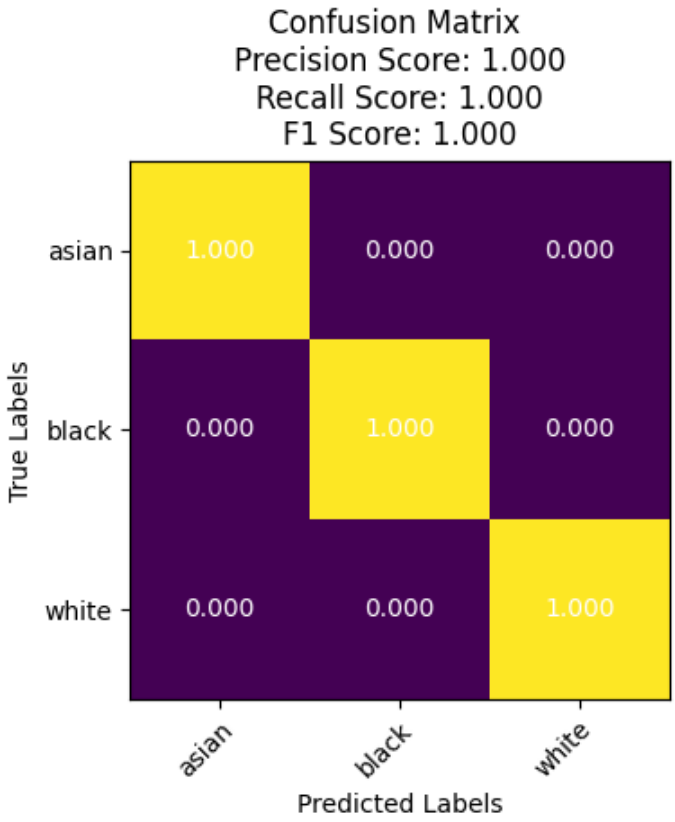

Figure 20 - Classification Report with Confusion Matrix for Model Classification

Figures 19 and 20 show that this CNN architecture was able to achieve perfect resolution with 100% training and validation accuracy as well as 0 training and validation loss by the end of its training. It is important to note these results can only be obtained when using a simplified and easily distinguishable training dataset. In the three different datasets depicting the three different skin tones (labeled ‘asian’, ‘black’, and ‘white’ for simplicity) each dataset is composed of 125 images with the exact same face structures and the only difference being the skin tone. One additional difference is that images in the ‘white’ class have blonde hair and blue eyes, while images from both the ‘asian’ and ‘black’ classes have black hair and black eyes. Therefore, this CNN in only classifying the skin tone of a given image with a bias on blonde hair/blue eyes for ‘white’.

375 total images were extracted from the three datasets. These images were first labeled and shuffled before being split into train, test, and validation sets. The training set consisted of 300 images while the testing set consisted of the remaining 75. The train set was then split again, resulting in 250 training images and 50 validation images. Figure 20 shows that after the training was complete, the CNN was able to have perfect recall as displayed in the confusion matrix as well as having precision, recall, and F1 scores of 1.0. Using a more complex dataset would likely not yield results as good as these, but given the simplicity of this dataset, anything less than perfect recall would be due to poor network configuration.

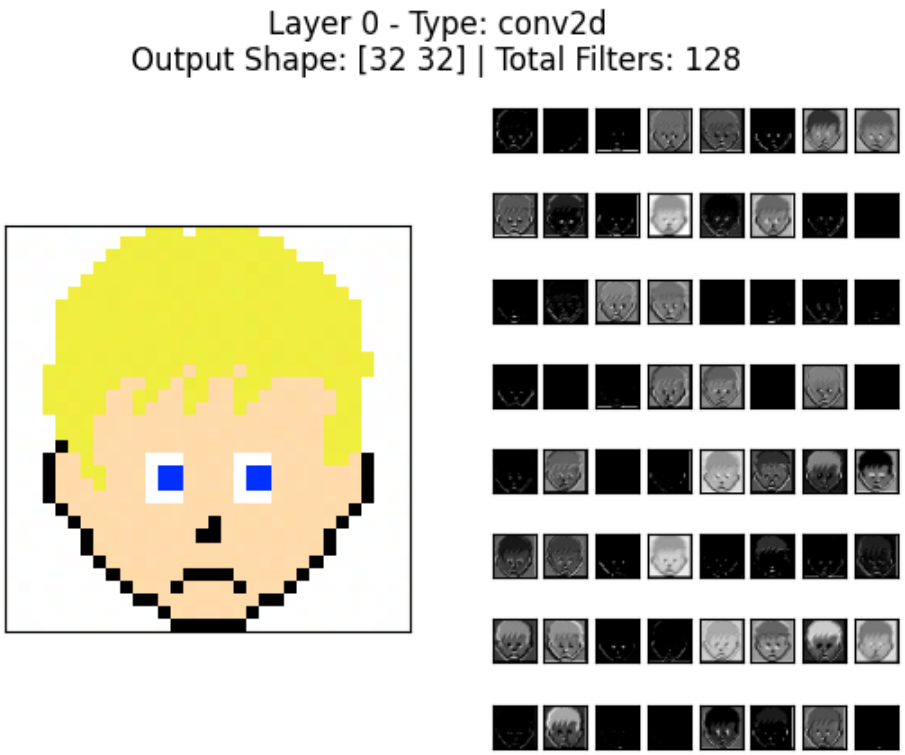

Figure 21 - First Convolution Layer of CNN

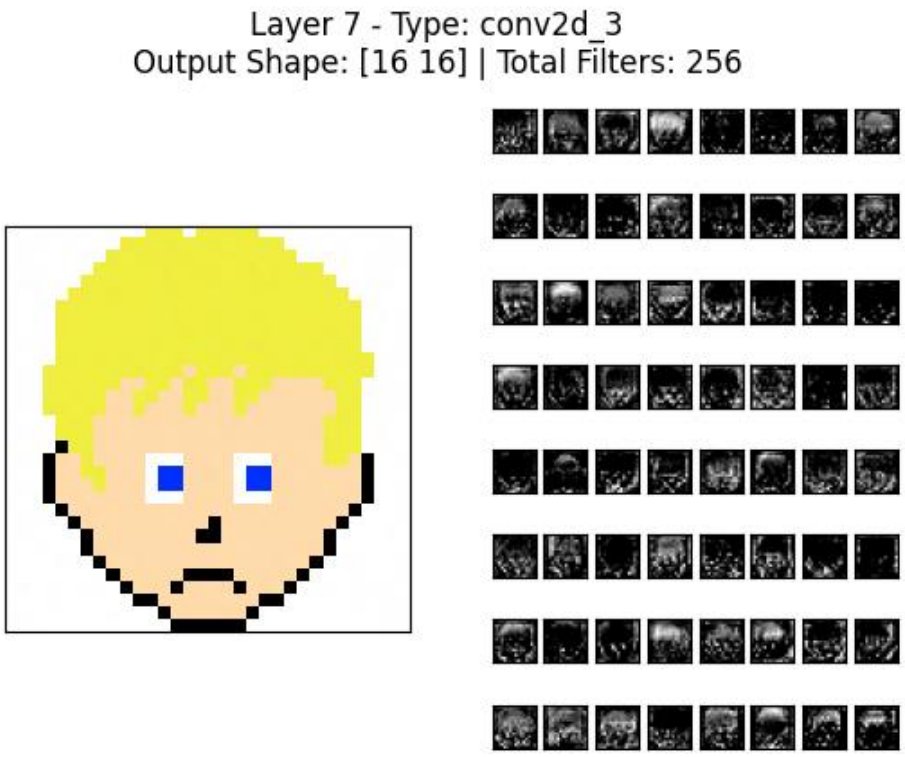

Figure 22 – Final Convolution Layer of CNN

During training, the images were passed through four separate convolution layers. Figure 21 shows a sample of the resulting filters (right) from the sample image (left). The filter results show the CNNs ability to extract key facial features by defining the geometry of the head, hair, eyes, nose, and mouth. Figure 22 shows that by the fourth convolution layer the CNN can extract features that may be unnoticeable to the human eye. By passing more kernels to the image, the CNN was able to extract smaller features that a human may not be able to distinguish. When using high resolution imagery on real facial images this is critical for accurate classification results. Using a CNN with a similar network configuration as the one used for this research could easily be applied to real facial data.

6.3 Selecting the Appropriate GAN Model

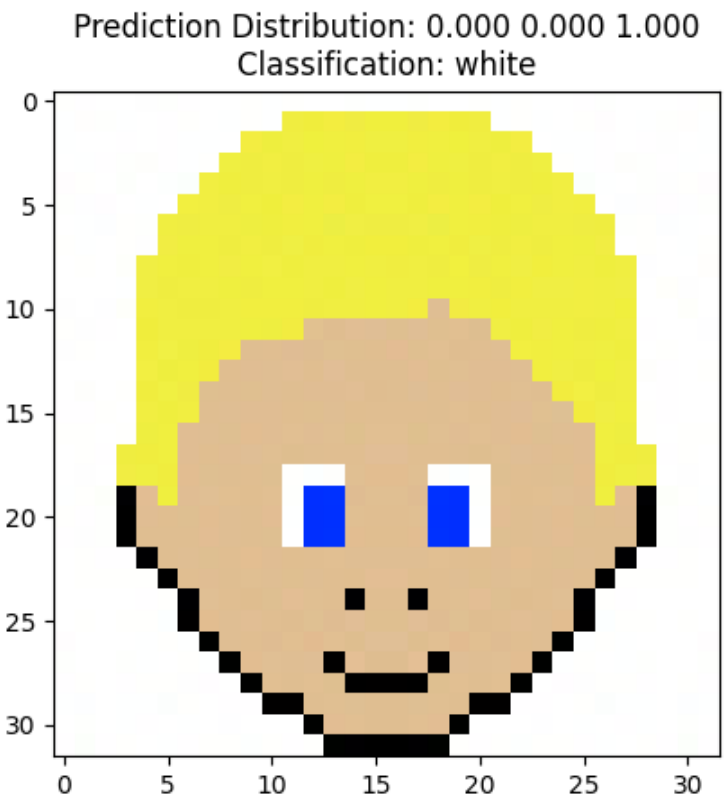

Figure 23 – Decision Distribution of CNN example 1

Figure 23 shows the decision distribution of this CNN on a test image with a skin tone that is different from all the skin tones used in the three datasets. With a skin tone that is a few shades darker the images from the 'white' dataset, the CNN was able to determine this image was closest to the 'white' dataset over 'asian' or 'black'. Although this is a reasonable prediction, this was heavily influenced due to the blonde hair and blue eyes in this test image.

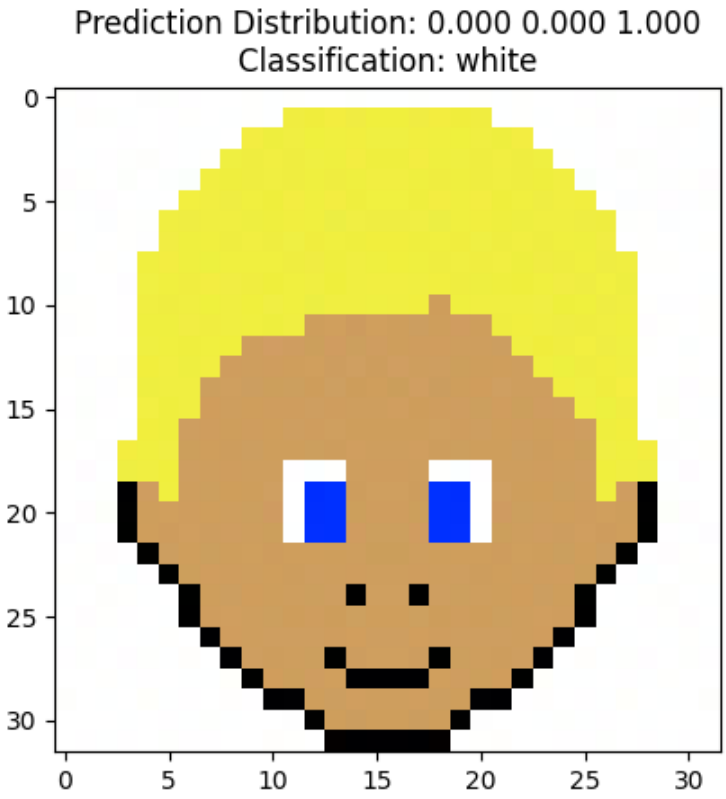

Figure 24 - Decision Distribution of CNN example 2

Figure 24 shows the decision distribution on a test image with a skin tone between ‘asian’ and ‘black’ but with blonde hair and blue eyes. Since these two features were unique to the ‘white’ dataset, the CNN classified this test image as ‘white’ even though that would not be the best skin tone match from the available models. This shows the importance of generating datasets with a minimal number of unique features. Since the ‘white’ class had more than one unique feature the additional features resulted in having a higher importance in the embedding of the final CNN model.

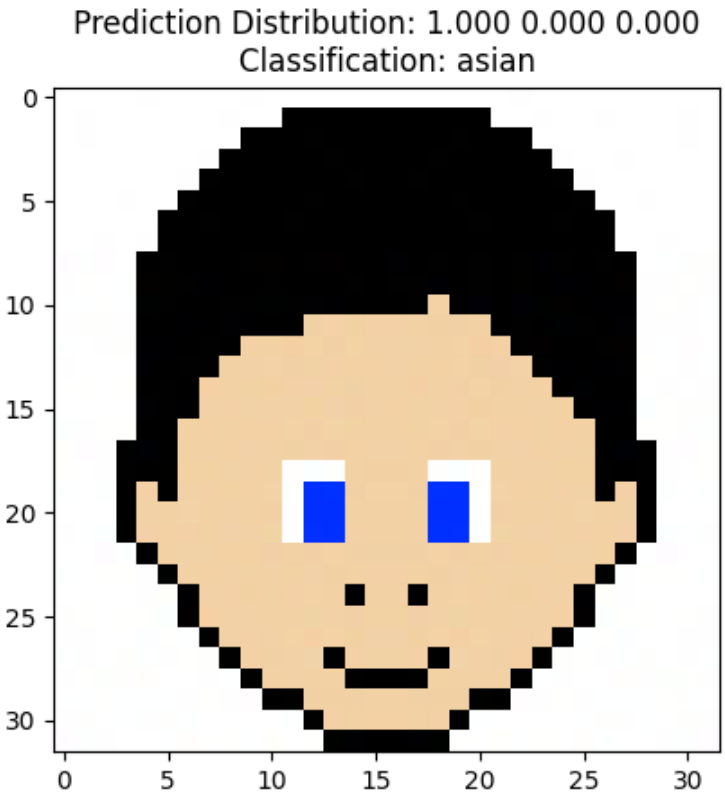

Figure 25 - Decision Distribution of CNN example 3

Figure 25 continues to highlight the importance of creating datasets based only one unique feature. This example has the exact skin tone as those found in the 'white' class and blue eyes, but does not have blonde hair, therefore, the CNN did not consider this image as 'white' rather 'asian'. Since the hair encompasses a larger portion of the image than the eyes do, the class that represents the hair feature had greater weight on the classification result.

7. PARTIAL FACIAL RE-IMAGING AS A SERVICE

7.1 MASKERAID

MASKERAID is a web-based tool developed to aid in the visualization of the partial facial re-imaging process. The idea of MASKERAID is what initiated this research. This tool was originally sought to be a solution to one of the major issues the COVID pandemic created. As previously mentioned, this issue is the fact that the use of face masks has been normalized throughout society which will ultimately lead to a massive decrease in accuracy and reliability of facial recognition applications used in the world today.

MASKERAID is a play on words of the word "masquerade" defined as a verb that "pretends to be someone one is not". In today's world as people wear masks every time they step out into public, they are hiding part of the identity. Whether the use of mask or any facial covering has malicious intent or not, this use causes an individual to appear to others as someone they are not. As facial recognition continues to grow and be more depended upon, knowing people's identity will be highly sought after. The solution to this problem is to "ASK AI" with mASKerAId.

7.2 MASKERAID Tech Stack

MASKERAID is a React application utilizing a Flask backend that connects to a Mongo database. React applications are highly responsive which is a highly valued feature due to this applications ability to visualize and interact with very large sets of data. A Flask server was chosen to easily utilize the Tensorflow and Keras libraries in

Python. Lastly, MongoDB is being used because it is a widely used and supported NoSQL database. Using a NoSQL is important for this application because many different types of files will be saved to this database. The input and output of the GANs being used are tensors, which are different datatypes than JPEG or PNG images. Being able to save both types of data is important when doing partial facial re-imaging since this process works with different images at different times throughout the process.

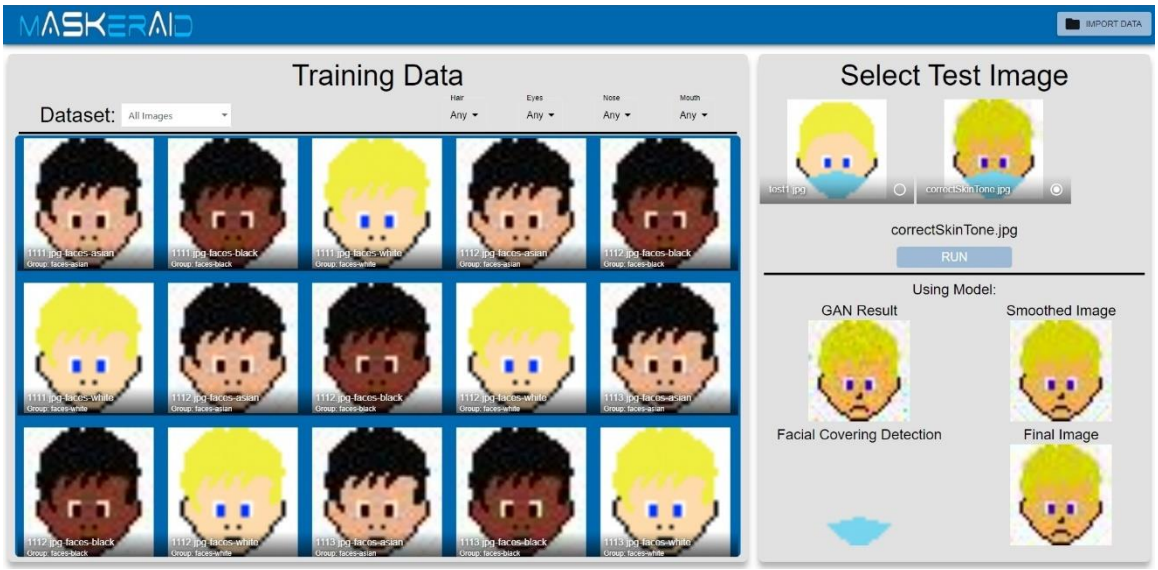

Figure 26 – MASKERAID web-based application

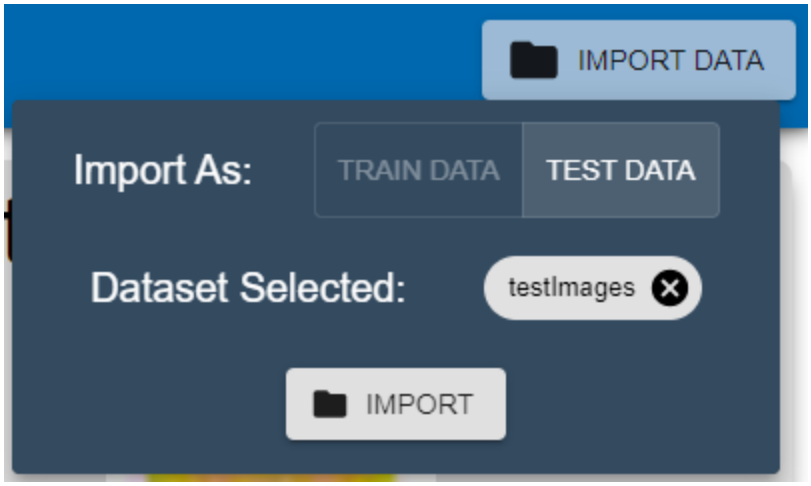

Figure 27 – MASKERAID data import

Figure 26 displays a screenshot of the MASKERAID tool. This tool allows the user to import data as either training or test data as shown in in Figure 27. Training data refers to all the images used to build any model that is available to the user. These images can easily be filtered by the dataset or model, and additionally by any of the meta data attributes that may be available from each image. Understanding the data used to build a particular GAN model will become very useful when developing new GANs and determining if a GAN model is suitable for a user's needs.

7.3 Visualizing the Training Data

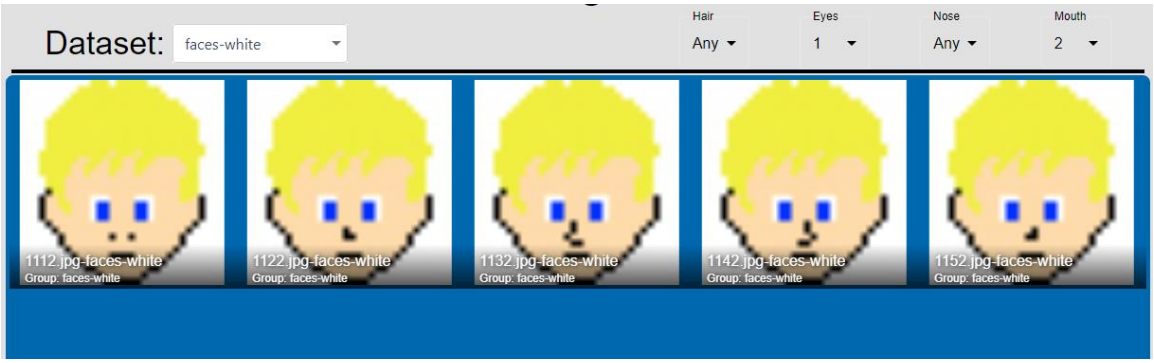

Figure 28 - Example MASKERAID filter on Train Data

Figure 28 shows an example of how filtering works withing the training data. The dataset selected is “faces-white” where you can see all resulting images have the same skin tone. These were sample images used to build the “faces-white” model. Additionally, each image has meta data on certain key features associated with it. The four attributes known from each image are the style of hair, the eye shape, the nose shape, and the mouth shape. This filter shown in Figure 28 shows the results of every image in the “faces-white” dataset that have any time of hair style, type 1 eye shape, any type of nose shape, and type 2 of mouth shape. From this you can see that all images have the same hair style even though the “any” type filter was set. This shows that all images in this dataset had the same hair style. Similarly, every image has the same eye and mouth shape, but the difference between all these images is the nose shape. This dataset shows that there are 5 different types of nose shapes used throughout training.

This means the generated GAN results could potentially return 1 of 5 different shapes of noses.

7.4 Visualizing the Testing Data

Once the training data is understood the user can select an image from the test dataset and run the partial facial re-imaging algorithm. These test images should contain an image of face with some type of facial covering.

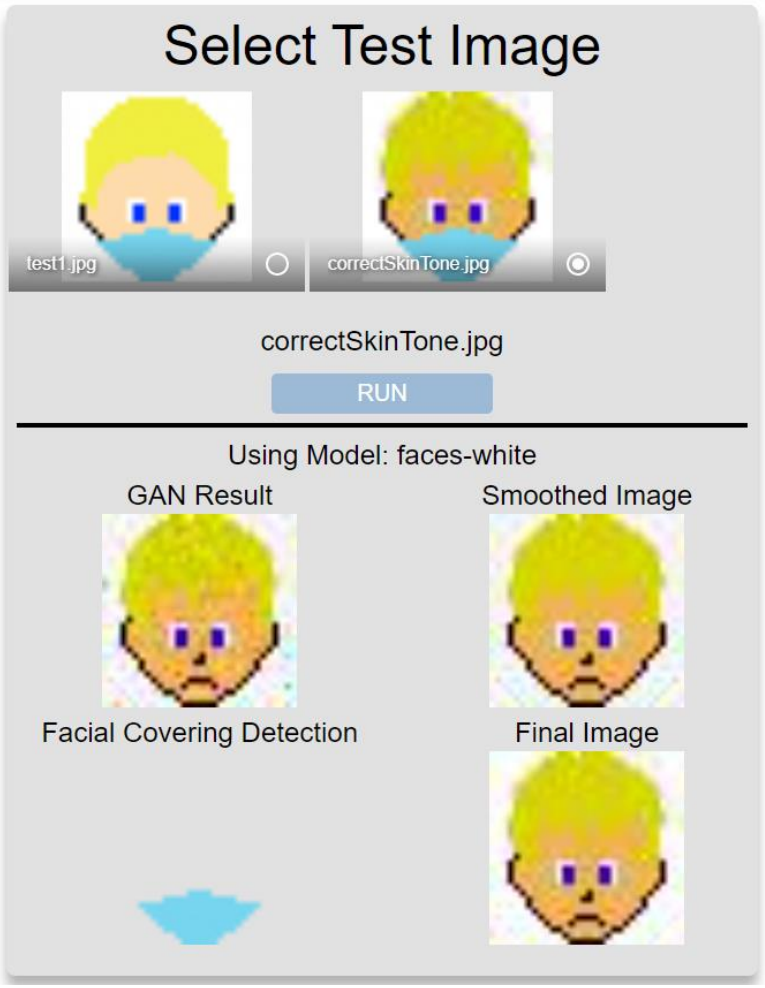

Figure 29 - Example MASKERAID results on Test Data

Figure 29 shows the expected results from the “correctSkinTone.jpg” test. After this image was selected and ran, pre-processing was done on the selected JPEG to determine that the “faces-white” model was the appropriate model to use. The results then show four important steps of the facial re-imaging process. The first image shows the unsmoothed result from the “faces-white” model. This result has a square shaped nose and a frowning mouth. The next image is a smoothed version of this GAN result. The third image is the clipped facial covering portion from the test JPEG. This image represents the pixels in the test JPEG that will be replaced with the corresponding pixels found from the smoothed result. The final image is the product of this occurring which shows that the square nose and frowning mouth were appended to this original image. The nose type and mouth shape could have been 1 of 25 different combinations (5 different nose types and 5 different mouth shapes).

7.5 MASKERAID Database

The data saved to the database remains persistent and can be accessed again for developmental purposes. In a typical use case, the user would upload a dataset of facial images of people of interest. For example, a local police department could upload the images of all the known criminals in the area. This would allow the user to be able to generate a custom GAN model based on these criminals. The criminal dataset would then be added to the MASKERAID database and could later be used by this police department at any time. This data could also be used in tandem with other users’ datasets to build a more generic model of all the images found in the database. This would then

be used as the default model if a user does not have their own dataset to build their own model from. This allows MASKERAID to become a highly scalable application. Having high scalability is an important feature of this application for future work that is planned.

8. FUTURE WORK

8.1 Future Vision & Objective

This research has proven the plausibility of using partial facial re-imaging to remove facial coverings from any given facial image. The main objective in future work is to begin training GANs on real facial data. The reason this was not done for this research is due to the time complexity of training on higher resolution data. To train on high-resolution data, and yield realistic high-resolution results, more computing power (GPUs) would be required. Once this becomes accessible, testing could then begin by using MASKERAID on partially covered faces and then using these results as the input to existing facial recognition software to increase their accuracy and reliability.

8.2 Development of DCGANs

To begin model development for real facial images, the existing GAN structure would likely need to be modified. As previously discussed, DCGANs have the likelihood of being able to extract more features from a face than a traditional GAN could. When working with high-resolution data this is a must. If real facial imagery data were trained with a traditional GAN, not only would it take much longer to train, but the results would likely be very blurry due to its inability to extract high dimensionality features.

8.3 Post-Processing Results

Although this research was able to produce very compelling results, these results were not always perfect. When looking at the neighboring pixels between the clipped area of the face and the original imagery it is noticeable that the skin tone is not always the exact same between these two areas. Most of the time it was, but this was due to using ideal test images. The human face is highly complex, and when used on real data, there will never be only three distinct skin tones like in this initial research. Although it is possible to create more models to better match all possible skin tones, it is highly unlikely there will ever be hundreds of thousands of models available to a user for every skin tone there is. Therefore, to create more realistic results with a smoother transition between the original and superimposed pixels, post-processing will need to be done on the resulting image.

This post-processing should get the exact skin tone the original image had and update the superimposed pixels to match this skin tone. The problem is human faces have more than one shade to their skin. Therefore, this process would need to find patterns in the skin rather just selecting one tone. Applying these patterns in the correct locations is also not a straight-forward task. Tonal patterns are dependent on the lightning found in the original image. This means that the same person can have different skin tones based on how much lighting, and the direction of lighting that has been applied to their face. More research will need to be done in this area to be able to post-process the results as accurately as possible.

8.4 Training Models as a Service

Currently MASKERAID is only able to use models found locally on a machine. These model files can become quite large and for some users they will not be able store these models on their machines for use. There are plans for MASKERAID to be able to allow users to upload their datasets and be able to save their models on the MASKERAID servers. This would also allow other users to have access to this model if their use cases are similar. Therefore, users would be able to browse and view previously uploaded datasets and determine if this dataset is applicable to them. Users would be able to search for datasets that contain meta data of interest to them and easily test if the model returns accurate results. With the access to existing models, users can then apply transfer learning to develop a model that best fits their needs.

Creating a service like this would require a lot of storage space on the MASKERAID servers as well as many GPU nodes to be able to build users models as quickly as possible. Since the process of building a GAN model can take many hours, emails would be sent to users notifying them when their model is available.

8.5 Meta Data Research

This initial research focused primarily on the ‘skin tone’ meta data attribute of a model. To reduce as much post processing as possible, this is arguably the most important piece of meta data for any model. As partial facial re-imaging gets applied to real data, many other meta data attributes will have much greater importance. More research will need to be done to determine which facial attributes have the highest importance in creating GAN models. These attributes should then be prioritized when separating the training data for a new GAN.

There is also a lot of research that can be done in non-facial features attributes. One important attribute that could have significant value is geo-location. Given a location such a town or city, it is likely that the people from that area appear similar in some way. For example, people from Dayton, OH tend to have a different appearance than those from Boston, MA. This meta data field could be of great use to an example user such as a police department. Even though a police departments criminal database is likely to be composed of many different looking people, since all these people are from the same area the GAN may be able to extract key features that a human would not be able to distinguish alone.

9. CONCLUSION

Partial facial re-imaging is a multi-step process involving the usage of various neural networks. The development of these networks is non-trivial and require knowledge on the desired output. To produces the most realistic results the underlying GAN model must be trained with imagery as similar to the test image as possible. Since the results of partial facial re-imaging have no known truth, training models to better fit the desired outcome becomes difficult. The overall "accuracy" of partial facial re-imaging is subjective to user, and until these results are used to classify unknown persons, the effectiveness remains unknown.

This initial research has shown very promising results in eventually being able to be used as an intermediary step for facial recognition on partially covered faces. The potential usages extend far beyond facial recognition as partial facial re-imaging can be used in any application that requires creating a full facial image from a partially covered face. This same process is also not limited to just facial data, but any type of imagery.

Partial facial re-imaging can utilize an online tool such as MASKERAID to create a service for additional research. With more data and models being created and tested, more information can be learned on how to better this process. MASKERAID doubles not only as a service for its users but as a tool to create a community of research on the topic of partial facial re-imaging.

REFERENCES

[1] Kaspersky. "What Is Facial Recognition – Definition and Explanation." *Www.kaspersky.com*, 13 Jan. 2021,

www.kaspersky.com/resource-center/definitions/what-is-facial-recognition

[2] "Face Recognition with FaceNet and MTCNN." – *Ars Futura*,

https://arsfutura.com/magazine/face-recognition-with-facenet-and-mtcnn/

[3] Liu, Yu Han; Feature Extraction and Image Recognition with Convolutional Neural Networks (2018)

https://iopscience.iop.org/article/10.1088/1742-6596/1087/6/062032/pdf

[4] "Facial Recognition Market." *Market Research Firm*,

https://www.marketsandmarkets.com/Market-Reports/facial-recognition-market-995.html

[5] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative Adversarial Networks. arXiv:1406.2661 [cs] 2014

https://arxiv.org/pdf/1406.2661.pdf

[6] Mwiti, Derrick. "Introduction to Generative Adversarial Networks (GANs): Types, and Applications, and Implementation." *Medium*, Heartbeat, 8 Apr. 2021,

https://heartbeat.fritz.ai/introduction-to-generative-adversarial-networks-gans-35ef44f21193

[7] Danis, Nazli. "Best Practices for Developing with Face Recognition." *Kairos*,

https://www.kairos.com/docs/api/best-practices#:~:text=Under%20ideal%20circumstances%2C%20images%2Fvideos,not%20be%20detrimental%20to%20performance.

[8] Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 [cs] 2014

https://arxiv.org/pdf/1409.1556.pdf