Face Unmasking Using Image Segmentation and Inpainting

Jagannathan, R[1]., Mupparapu, S[1]., Hernandez Mejia, J. L.[1], Pyrcz, M.[2]

[1]College of Natural Sciences, The University of Texas at Austin, Austin, Texas

[2]Jackson School of Geosciences, The University of Texas at Austin, Austin, Texas

Abstract

This presents the sequential application of three convolutional neural network architectures for the purpose of realistic and automatic large-object removal from facial images. Due to the significant increase people wearing face masks in visual media in recent years, this work attempts to unmask facial images through a three stage process where at each stage, the output of a neural network becomes the input for the next. This paper demonstrates the practical use of a three network pipeline that identifies and removes the exact region of the face covering, reconstructs the facial features underneath, and then enhances the new facial image for a more natural appearance. First, a U-Net model is trained to perform a binary image segmentation of a masked facial image into face-mask/non-mask regions. Then, an autoencoder that reconstructs the entire face using the non-mask region as input, recovering the missing facial features in the process. Lastly, the resulting image is smoothened by a super-resolution convolutional network, and a final image is produced by merging the smoothened output with the non-mask region of the input image. When tested on the CelebA dataset, the U-Net is able to segment the masked image with an accuracy of over 98%, and the autoencoder is able to produce realistic results with a mean squared error loss of 0.0024.

Introduction

The purpose of this work is to achieve automatic large-object removal from facial images, where the large object is a face mask covering. The problem of unmasking a masked face has great practical value, given the current state of the world considering a global pandemic and widespread mask usage. With a significant number of people wearing masks in public due to necessity and safety, mask removal technology can help to bridge the gap between maskless and masked appearances. There are potential applications in identity recognition technology, from improving surveillance media to criminal identification. There are also personal use cases for this technology, as pictures taken with masks on can be enhanced by removing face coverings.

Multiple non-learning methodologies have been used in the past to remove objects from images. Exemplar-based image inpainting can be used to perform region filling and object removal, where unwanted objects are replaced by synthesized content created by data sampled from the remainder of the image (Criminisi et al. 2004). With scene completion using millions of photographs, holes in an image are patched by pasting parts of similarly patterned images from a database of millions of images (Hays et al. 2007). Recursive error compensation using Principal Component Analysis (PCA) reconstructions has been successful in removing glasses from facial images (Park et al. 2005). However, these algorithms are largely limited to small objects and images of scenery. Facial images are highly unique, and removing a face mask typically involves generating the entire lower half of the face.

In recent years, research has been done into the use of Generative Adversarial Networks (GANs) trained on large datasets to perform object removal by artificially generating the image underneath the object. In particular, the company JoliBrain has attempted to remove face masks from images with a GAN specialized in image-to-image translation for domain adaptation ("Removing Face Masks with Joligan: DeepDetect." 2021). JoliBrain's generator architecture produces photorealistic results, demonstrating the validity of learning-based methods to the problem at hand. However, their approach to object removal involves inpainting a large rectangular region of the face where the mask lies, leading to irregularities near facial borders. Moreover, generated faces sometimes possess characteristics inconsistent with the rest of the face, such as a bearded mouth area inpainted on a woman's face, indicating a lack of sufficient training on a large enough dataset. To address these issues and solve the problem of applying learning techniques to mask removal, our work makes four main contributions:

1) We construct a large synthetic dataset that contains 50,000 unmasked facial images, 50,000 corresponding masked facial images, and 50,000 segmentation maps.
2) We utilize a U-Net CNN architecture to accurately segment the mask and non-mask parts of an image.

3) We utilize an autoencoder architecture to inpaint just the segmented area of the image with the mask, preserving facial structure and minimizing irregularities at the facial boundaries.
4) We utilize a small CNN architecture to improve the quality of the inpainted region produced by the autoencoder, achieving resolution-boosting effects and a more natural post-processed result.

A brief enumeration of our approach and data is as follows: our approach to the problem of unmasking a masked facial image consists of three steps: dataset creation, mask detection, and mask removal/inpainting. Due to the lack of existing datasets consisting of unmasked and masked facial pairs, we have generated a synthetic dataset of masked faces and corresponding segmentation maps from facial images to use for image segmentation and image inpainting. For mask detection, we will train a convolutional neural network to perform semantic image segmentation on pictures of masked faces and identify the region of the face covered by a face mask. For mask removal, we will post-process the mask detection model's output to obtain a clean segmentation map, then train an autoencoder to inpaint the masked region. Finally, for face resolution-boosting, we will train a convolutional neural network to perform super-resolution on the inpainted image.

Methods

Synthetic Dataset Creation Methodology

To train a convolutional neural network to segment a masked image into mask and non-mask parts, a masked facial image is required as input, and corresponding labels (mask or non-mask) for each pixel are needed as output. To train a model to inpaint the removed area of a face where a mask resided with a realistic and accurate generated face, image pairs of unmasked and masked faces are needed, where the only difference between the two images is that a mask exists in one and not the other. In other words, the characteristics and exact positioning of the face must be the same in both images. However, no publicly available dataset meets the restrictive constraints for training the models needed for our work. Therefore, we have constructed a synthetic dataset of 50,000 image trios using the CelebFaces Attributes Dataset (Liu et al. 2015). Some examples of these image trios are shown in Figure 1.

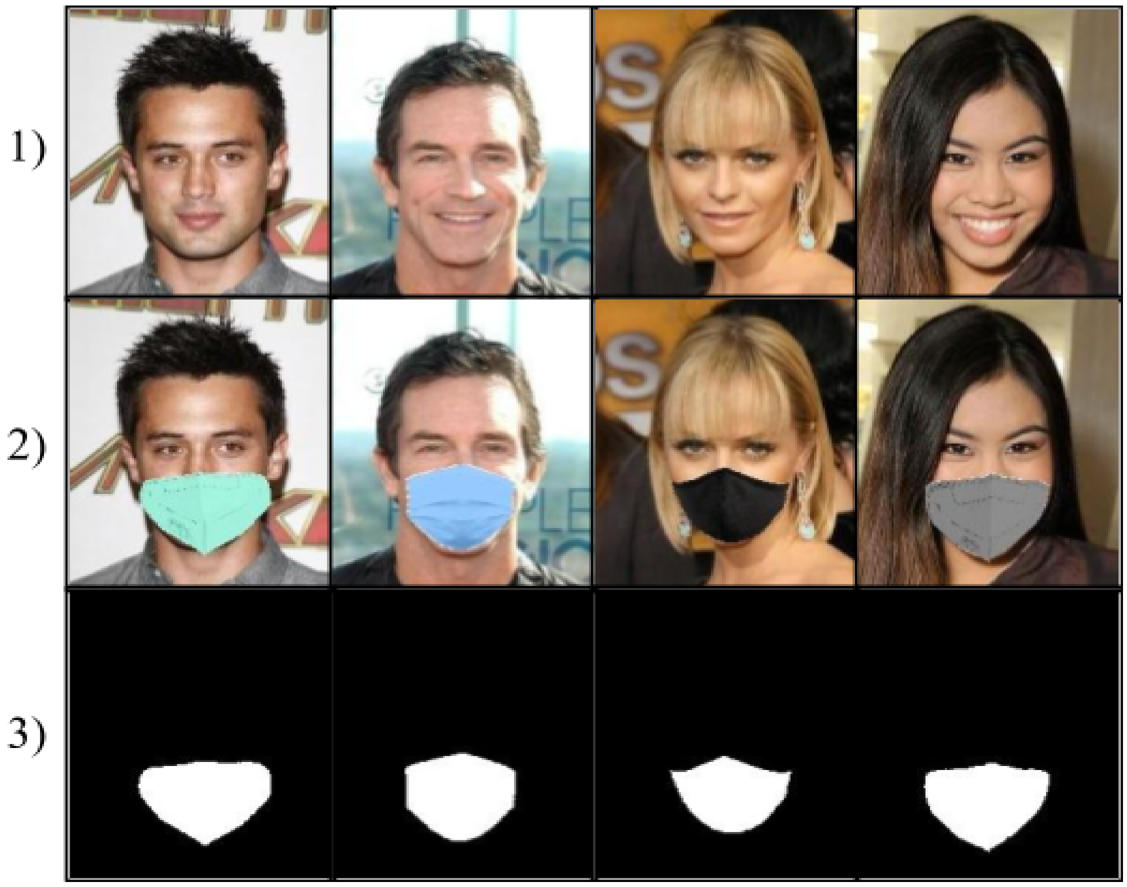

Figure 1. Our synthetic dataset consists of 50,000 image trios: 1) unmasked face, 2) masked face, and 3) segmentation map.

To create the synthetic images, it is necessary to have a large-scale image source for human faces that face forward and are cropped so that the faces are the central feature of the image. The CelebA dataset meets these criteria, having over 200,000 images of celebrity faces that are cropped or aligned to the center, of which a random 50,000 were used to train the models. To obtain images of these faces with masks on, a computer-vision-based script that overlays an image of a mask with the correct orientation and position onto a face was utilized (Anwar et al. 2020). To reflect the variety of face mask models and designs worn in the real world, the face-masking script was configured to overlay three different kinds of masks (surgical, KN95, cloth) in 4 different colors (blue, black, white, teal). For each facial image processed by the face-masking script, 1 of the 12 possible combinations of mask color and model was overlaid onto the face. If the face-masking script was unable to overlay a mask on the image, the image was not included in the synthetic dataset. To obtain the segmentation map, the pixels in the masked image that differed from the corresponding pixels in the unmasked image were labeled as face mask (white). Additionally, the OpenCV morphology library was used to clean noise and holes from the segmentation maps. Finally, all the RGB values were divided by 255 and normalized to a float between 0 and 1 in order to avoid exploding gradients during training. This completed the dataset consisting of 50,000 unmasked face images and their corresponding artificially masked face images and segmentation maps.

Mask Detection Methodology

The model architecture utilized for image segmentation is the U-Net (Ronneberger et al. 2015). The U-Net is comprised of two sections: the encoder, which is responsible for extracting information from the images, and the decoder, which upsamples the encoder's output tensor and creates the predicted segmentation map.

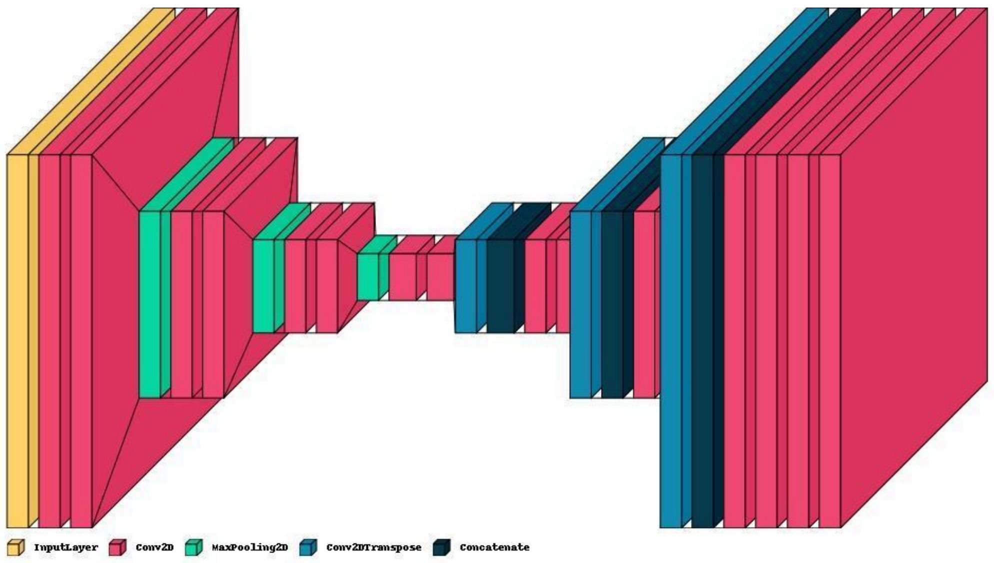

Figure 2. This is a 3D visualization of the UNet architecture implemented for semantic image segmentation.

This encoder uses a series of downsampling blocks, each of which is composed of two convolutional layers with a 3x3 kernel and a ReLU activation after each convolution. The downsampling blocks additionally include a 2x2 max-pooling layer that can be called depending on function arguments to shrink the dimensionality of input images and analyze them with more convolutional filters, as well as skip-connections that save a version of each convolution where the subsequent max-pooling layers are skipped to facilitate model convergence. The encoder is made up of four of these downsampling blocks, with the first three implementing max-pooling and the last one without. The decoder is composed of a series of three upsampling blocks. Each upsampling block consists of one transpose convolutional layer and one convolutional layer that merges the expansive and contractive inputs from the previous block. Expansive inputs are the outputs of the last block, whether it be a downsampling block or an upsampling block, that passed through all the max-pooling layers in the encoding phase. Contractive inputs, on the other hand, refer to the outputs of the downsampling block that passed through the skip-connection with the same width and height dimensions as the expansive input after it has been upsampled. Because this U-Net max-pools after every downsampling block except for the final one, there is a bijective mapping between the skip-connection outputs from each downsampling block and the contractive inputs for each upsampling block. The implementation of these skip-connections drastically increased model performance by allowing it to extract more information from the images without hampering training efficiency. After both the encoding and decoding phases are complete, the inputs are passed through two final convolutional layers that create the model's predictions for each pixel in a two-element vector of logits. As the last step in the model, a softmax activation is applied to the model prediction tensor. The softmax activation function is defined by the following equation:

$$(1)\ \sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

This activation function normalizes all of the model's predictions by turning the vector of logit predictions $\vec{z}$ with length K for each pixel into a probability mass function representing the probability that the pixel belongs to each of the two mutually exclusive categories, mask and not a mask. In short, it returns a vector with the probabilities that the pixel is and is not a mask. This gives the model's certainty in whichever class it determines for the pixel. The softmax function also makes to converting the model's predictions to a segmentation map trivial, since the index of the greatest value of the vector returned by the softmax function is the pixel's classification, 0 or 1.

Mask Detection Training

The mask detection U-Net is trained with 2000 image and segmentation map pairs for 80 epochs with a batch size of 16. These images were split into train, test, and validation splits in a 60:20:20 ratio, for a total of 1200 training images, 400 validation images, and 400 test images. Additionally, the batches were randomly shuffled each epoch to avoid overfitting. The categorical cross-entropy loss function was used and is defined below:

$$(2)\ CE = -\sum_{i=1}^{C} t_i \log(\sigma(s)_i)$$

This function returns the difference between the probability distribution yielded by the softmax activation function, $\sigma(s)_i$, in the last layer of the neural network and the true one-hot encoded labels, t_i, for each of the C pixels. To propagate the changes through the weights in the model, the Adam optimization function (Kingma et al. 2014) was used with with a learning rate of 0.0001, obtained from Hyperband hyperparameter optimization, and the default β_1, β_2, and ϵ values of 0.9, 0.999, and 0.0000001, respectively. In addition to tracking the categorical cross-entropy loss, the categorical accuracy of the model was also calculated according to the following equation:

$$(3)\ Accuracy = \frac{True\ Positives + True\ Negatives}{True\ Postives + True\ Negatives + False\ Positives + False\ Negatives}$$

After the 80th epoch, the model had a categorical cross-entropy loss of 0.0548 and an average accuracy of 0.9805, or over 98%.

Face Inpainting Methodology

The model architecture used for face inpainting is an autoencoder, shown in Figure 4. It takes in an image with the pixels classified as a face mask blackened out and attempts to recreate the image with the blackened sections replaced by filled in facial features. Before the model could begin training, the images needed to be preprocessed by removing the face masks using their segmentation maps. In order to do this, the segmentation maps were inverted so each face mask pixel was represented as a 0 and everything else by a 1. The masked images were then multiplied by the inverse segmentation maps, resulting in an image that was identical to the original image but with black pixels wherever a face mask was predicted to be. This process is illustrated in Figure 3 below.

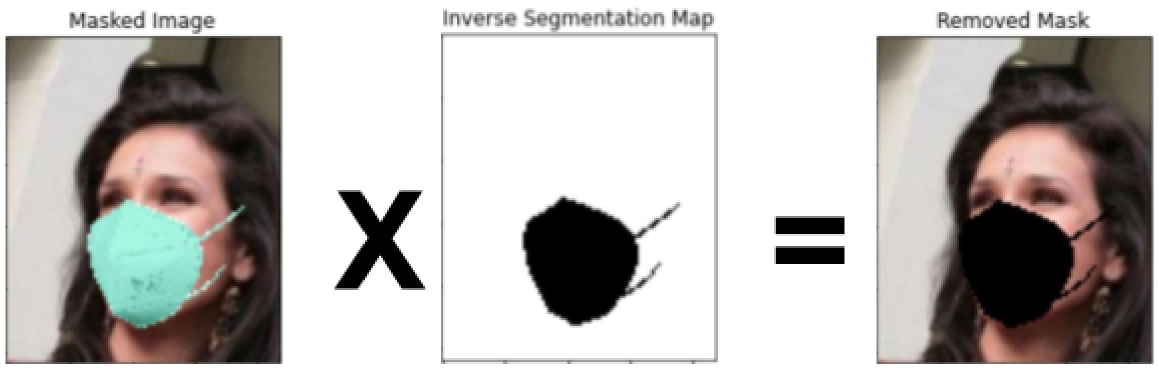

Figure 3. The model input is created by multiplying the masked image matrix with the inverse segmentation map.

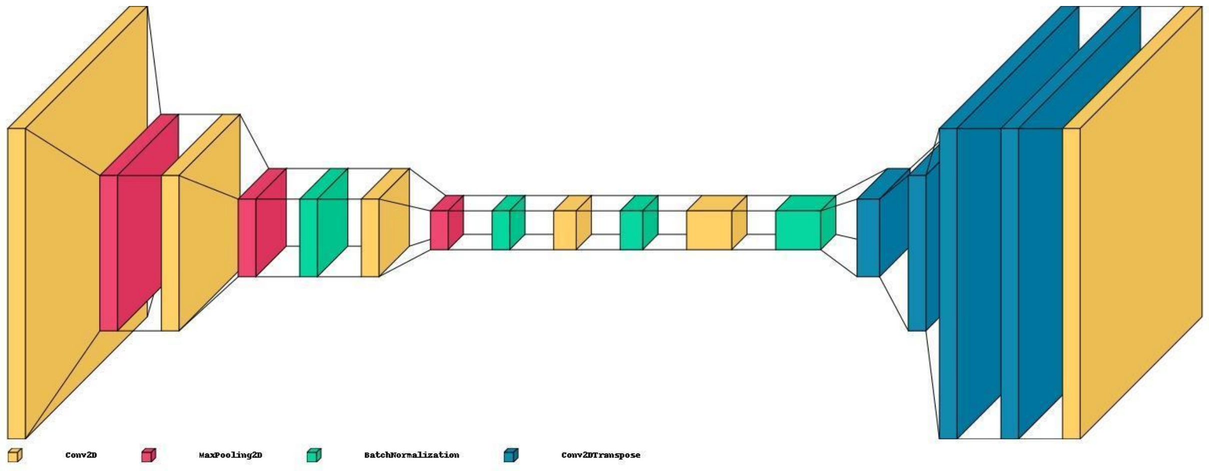

Figure 4. The autoencoder architecture used for face image generation.

The autoencoder is comprised of two parts: the encoder and the decoder. The encoder consists of a series of five convolutional layers with kernel size 3x3, same padding, a normal kernel initialization, and a ReLU activation applied to the outputs. The first four of these layers are followed by 2x2 max-pooling layers, and the last four are followed by batch normalization layers. The number of filters after each convolution is initially 32, doubling each time the tensor passes through a convolutional layer, resulting in an autoencoder bottleneck with dimensions 13x11x512. The decoder is composed of a series of four convolutional transpose layers. Each layer has a kernel size of 3x3, a stride of 2, and produces a tensor with half the convolutional filters as the tensor passed into it. After the decoder, there is one final convolutional layer with a stride of 3 that produces a tensor of depth 3, one for each of the RGB color channels, with the model's prediction.

Face Inpainting Training

The face-inpainting autoencoder was trained on 1000 masked image-segmentation map pairs from the dataset. These images were split into training and testing datasets in a 90:10 ratio, resulting in a training dataset of 900 images and a testing dataset of 100 images. To perform backpropogation, an Adam optimizer (Kingma et al. 2014) with a learning rate of 0.0006, which was obtained from Hyperband hyperparameter tuning, and the mean squared error (MSE) loss function were used. The mean squared error is defined by the following equation:

$$(4)\ MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

In this equation, n represents the number of data points whose loss is being calculated, y_i represents the model's output, and $\hat{y}_i$ represents the expected value. When applied to the autoencoder, this calculates the difference between each pixel the model predicted and its expected value, squares each difference, and takes the average of all the pixel's losses. The inclusion of the squared error term adds a penalty to each pixel's loss that becomes more significant the more the predicted and expected values differ, which helps the model understand when it makes massive mistakes in its predictions. The autoencoder was trained for a total of 2500 epochs and was able to achieve a low mean squared error loss of 0.0024.

Face Super-Resolution Methodology

The model used for facial image super-resolution is a Super-Resolution Convolutional Neural Network (SRCNN), shown in figure 5 (Dong et al. 2015). It is a three-layer fully convolutional architecture that mimics sparse-coding image processing methodologies, intended for increasing image resolution while maintaining greater image quality that non-learning, interpolation-based methods. Our work utilizes the SRCNN to improve the image quality of the generated inpainted image produced by the autoencoder. However, our usage differs from traditional super-resolution methods in that we do not attempt to increase the dimensions of the autoencoder output. Instead image quality is enhanced without resizing. As input, our SRCNN receives the output of the autoencoder, which is a reconstruction of the entire facial image after mask removal. Using the corresponding original facial image as a training label, the SRCNN has been trained to learn the end-to-end mappings between the autoencoder's lower-quality reconstruction and the ground truth unmasked image.

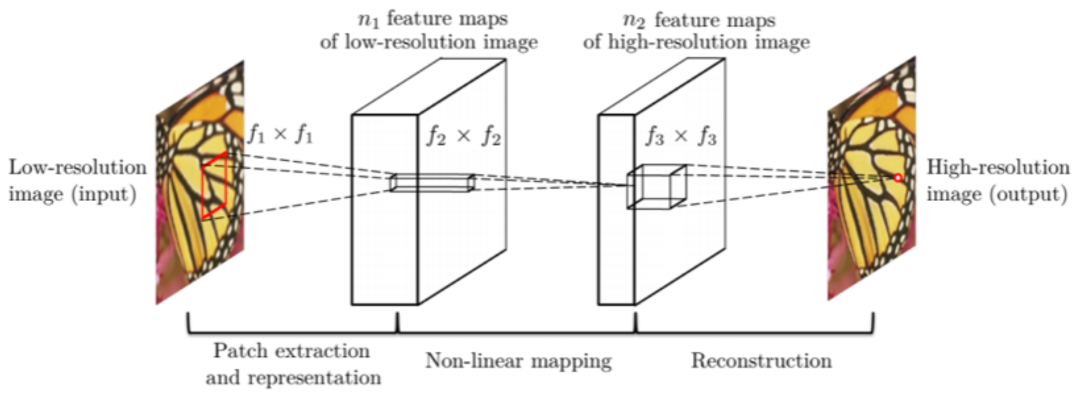
n_1 feature maps of low-resolution image
n_2 feature maps of high-resolution image
$f_1 \times f_1$
$f_2 \times f_2$
$f_3 \times f_3$
Low-resolution image (input)
High-resolution image (output)
Patch extraction and representation
Non-linear mapping
Reconstruction

Figure 5. A diagram describing the characteristics of the SRCNN architecture.

The SRCNN is comprised of three convolutional layers: 64 9x9 filters with no padding, 32 5x5 filters with padding, and 1 5x5 filter with no padding. ReLU is applied as the activation function for all three layer outputs. Unlike the previous model architectures used in this paper, the input to the SRCNN is not an RGB image but is rather the luminance channel (Y) of a YCrCb image. The autoencoder output is preprocessed to convert from the RGB color space to YCrCb, and the luminance (Y) channel is isolated and expanded to a 3D volume for input. The SRCNN output is a two-dimensional volume representing the high-resolution version of the luminance channel. The result is post-processed to an RGB image by combining the output with the Cr and Cb channels from the preprocessing stage.

To produce an optimal final result, the autoencoder output was multiplied by the input image's segmentation map, isolating the area of the image that was inpainted, and this was added to the model's input to merge the two images. This process is illustrated in Figure 6 below.

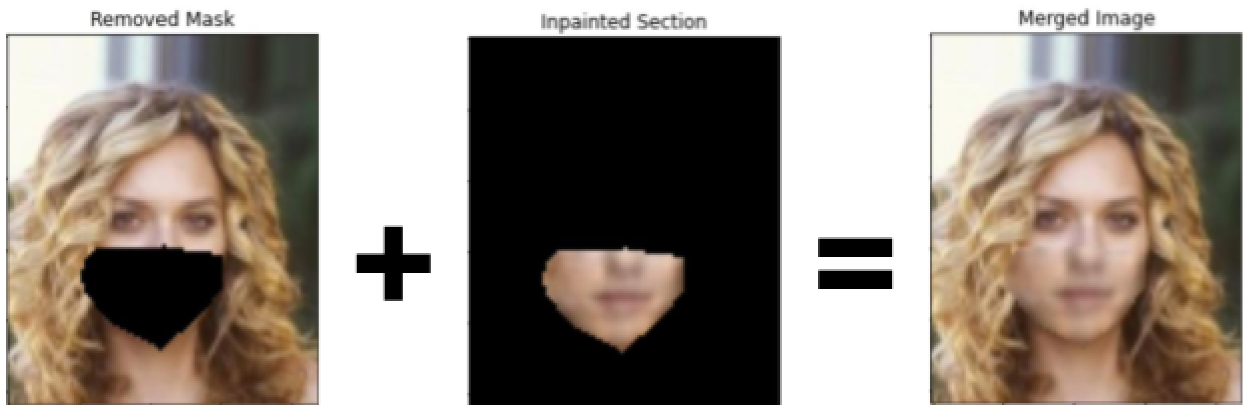

Figure 6. The postprocessed output is created by adding the removed masked image matrix with the isolated section from SRCNN postprocessing.

Face Super-Resolution Training

The SRCNN was trained with the luminance channels of the autoencoder outputs and the luminance channels of the corresponding original ground truth image as labels. The model was trined on 21600 shuffled 33x33 image patches generated from 900 full-sized autoencoder luminances. To describe this process further, a sliding window technique with a stride of 14 pixels was used to turn each full-sized image into numerous 33x33 image patches. The same process was applied to the ground truth label images, except the the label image patches were 21x21 pixels to account for the valid padding used in two of the SRCNN's layers. To perform backpropogation, an Adam optimizer (Kingma et al. 2014) with a learning rate of 0.0001 was used with the mean squared error (MSE) as the loss function. The model was trained for 150 epochs with a batch size of 128. Peak signal-to-noise ratio (PSNR) was used as the metric function in evaluating the SRCNN training results. PSNR is a measure of how well two images resemble each other, so a higher PSNR indicates a better result.

$$(5)\ MSE_{mat} = \frac{1}{mn} \sum_{0}^{m-1} \sum_{0}^{n-1} ||f(i,j) - g(i,j)||^2$$

$$(6)\ PSNR = 20\ log_{10}(\frac{MAX_f}{\sqrt{MSE_{mat}}})$$

In equations 5 and 6, *f* represents the matrix data of the output luminance, *g* represents the matrix data of the label luminance, *m* and *n* represent the number of rows and columns of pixels in the matrices, *i* and *j* represent the row and column indices, and MAX_f is the maximum signal value which is 255. After training, the SRCNN was able to achieve an MSE of 0.0036 and a PSNR of 24.44.

Results

Mask Detection Results

Sample results from the mask detection U-Net are shown in Figure 7. Negative pixel classifications are black and positive ones are white.

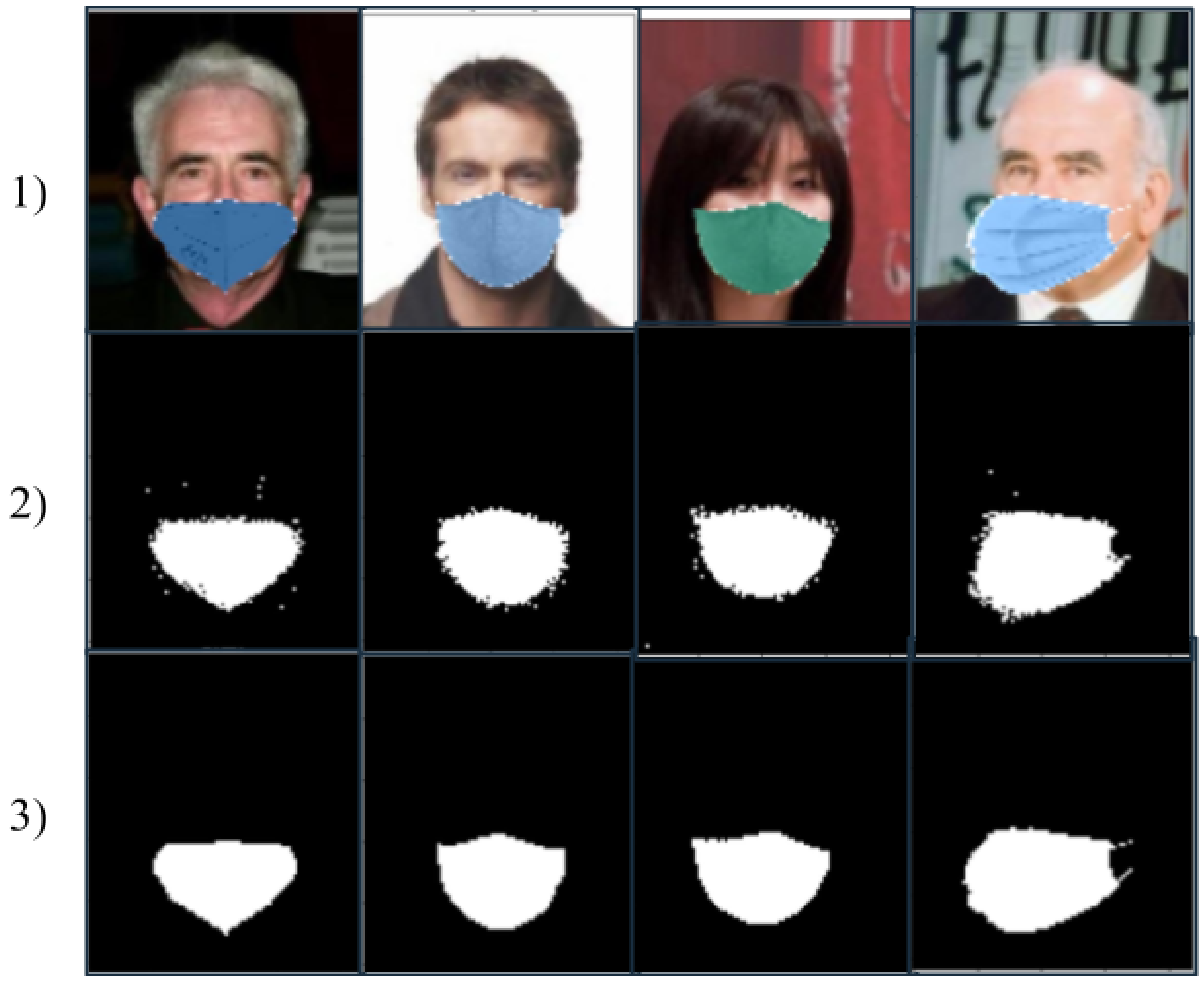

Figure 7. Image segmentation results from the UNet CNN: 1) image input, 2) image output, and 3) label image.

As can be seen in Figure 7, the model is highly successful in identifying the regions of the face that are covered by a face mask. In total, the UNet reached an accuracy of over 98%, so it is able to achieve near-perfect results. However, it does have some issues predicting the pixels on the border of the face and face mask, and it sometimes places small holes in the predicted segmentation map, so OpenCV was used to post-process the outputs as shown in Figure 8. The OpenCV morphology library's erosion and dilation functions performed a combination of denoising and hole patching on the raw UNet output.

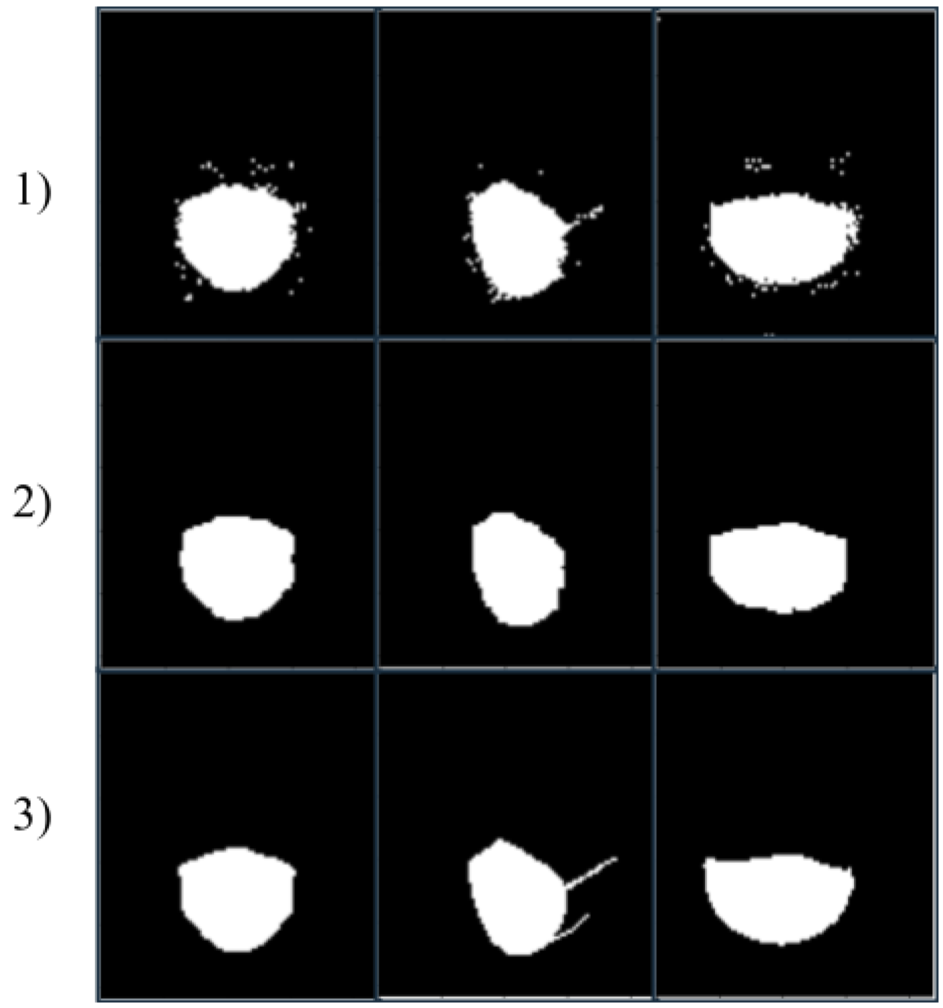

Figure 8. Postprocess the results using OpenCV Morphology: 1) UNet output, 2) post-processed output, and 3) label image.

Face Inpainting Results

As can be seen in Figure 9 below, the autoencoder is able to fairly successfully inpaint the faces in the testing dataset. Although there are some minor issues such as blurriness and slightly incorrect skin tones, the model is able to successfully recreate the major missing facial features such as the nose and mouth. It is also able to use the facial features that are not covered by a face mask to predict whether the mouth is open or closed as well as whether or not teeth are showing. For comparison purposes, the figure below also constructs a merged image where the section of the AE output corresponding to the face under the mask is appended to the removed mask image.

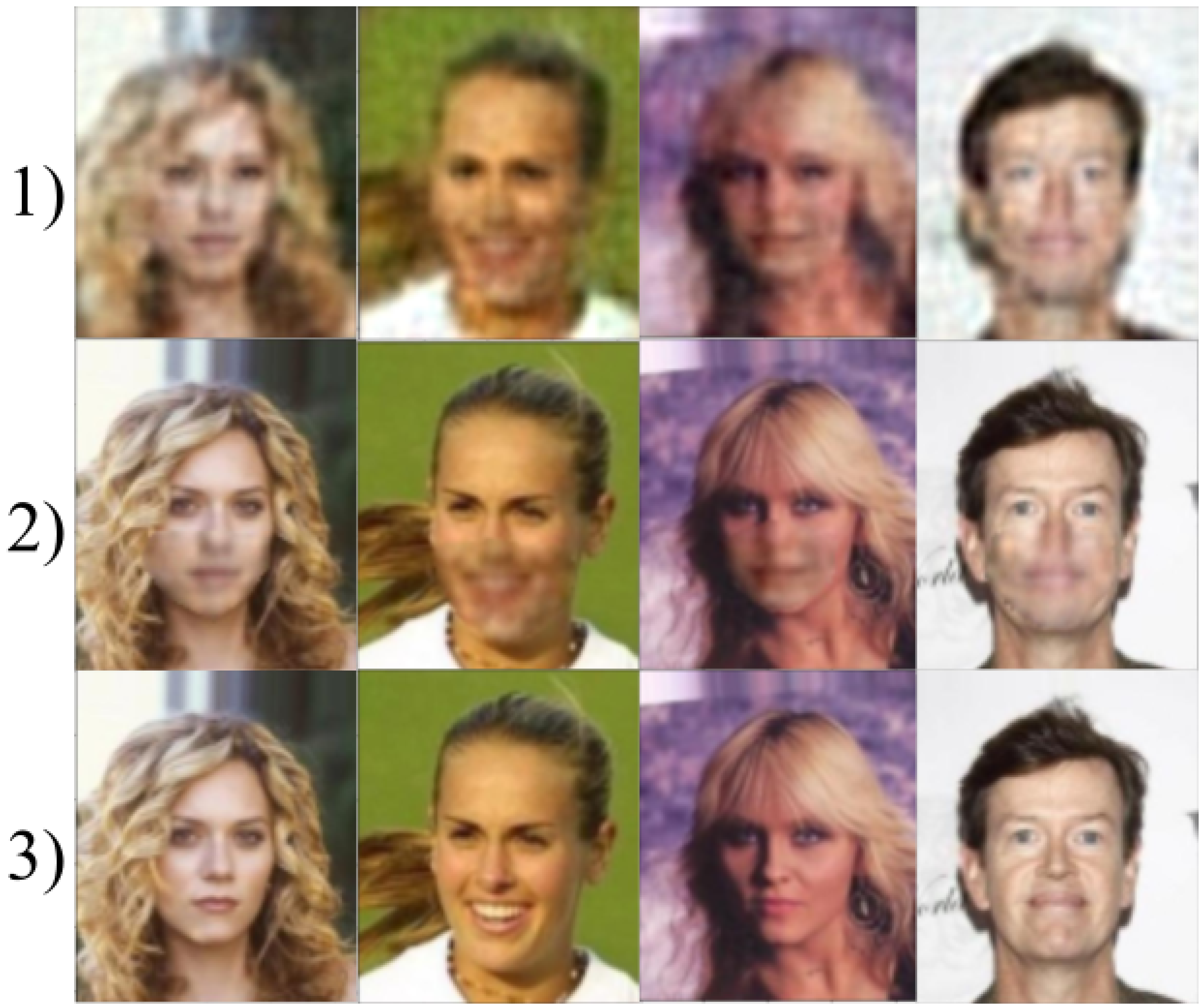

Figure 9. Face Inpainting results for the autoencoder (AE): 1) AE output, 2) merged result, 3) ground truth image.

Face Super-Resolution Results

As can be seen in Figure 10 below, the SRCNN enhances the photorealism of the resulting merged image representing the final product of the entire three-model pipeline. Compared to using the AE output without the SRCNN resolution boosting, the SRCNN merged image appears smoother and more natural, blending in with the rest of the face.

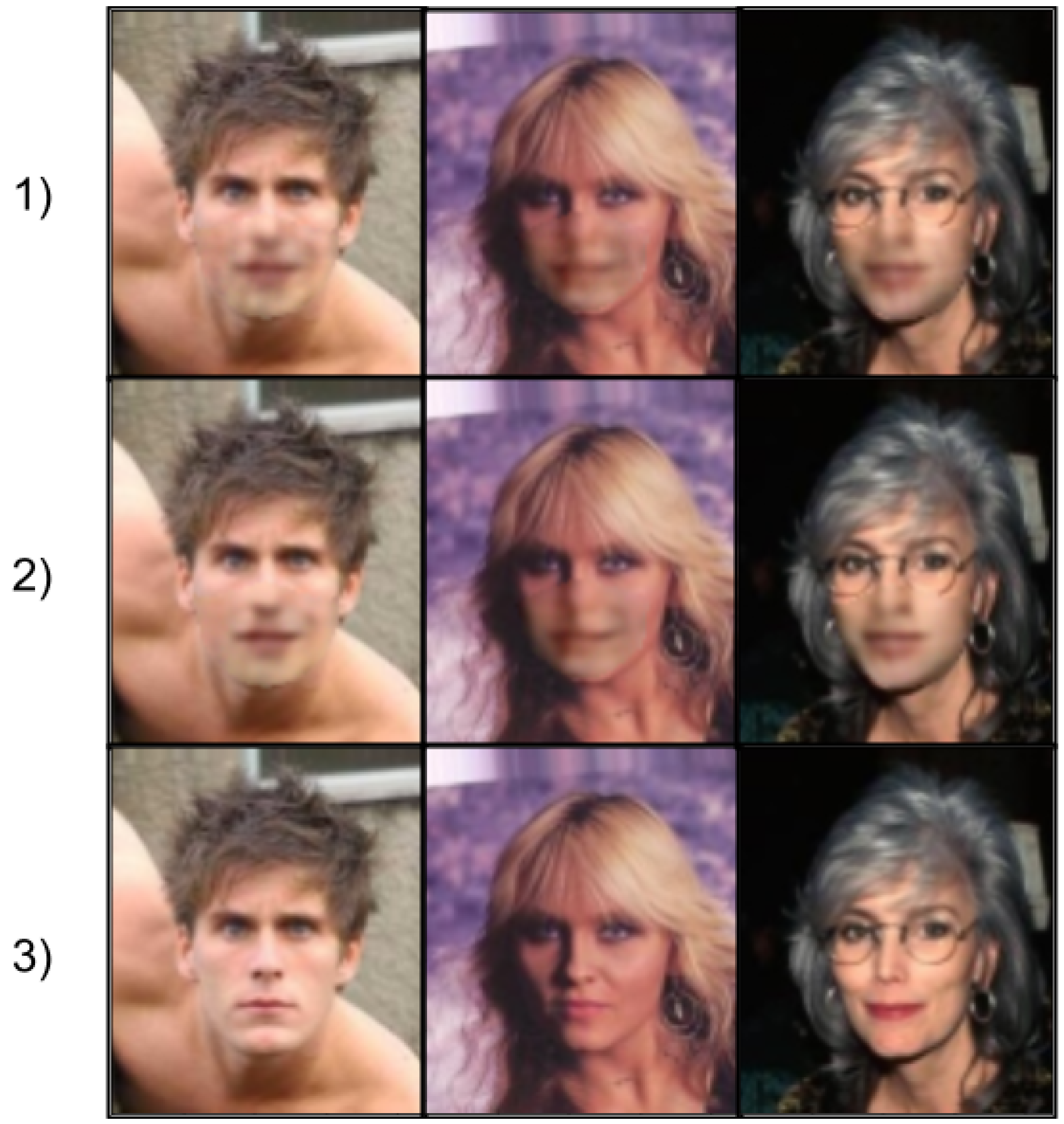

Figure 10. Face Super-Resolution results for the SRCNN: 1) AE merged result, 2) SRCNN merged result, 3) ground truth image.

Full Pipeline

Figure 11 demonstrates demonstrate the results of the full pipeline, which is the sequential application of the three convolutional neural networks on the input masked image. In the three stage pipeline, the postprocessed output of a neural network becomes the input for the next. As a result, inconsistencies in one stage of the pipeline must be handled properly by the next stage. For example, a slightly incorrect segmentation of the face mask results in remnant mask pixels left in the AE input, which can distort the inpainted facial features. As can be seen in

Figure 11, although remnant mask pixels remain in the inpainted output image, the inpainted face remains intact, although with less clarity. Future work would require improvements to the U-Net segmentation model to prioritize complete removal of mask pixels, effectively improving the quality of the AE output.

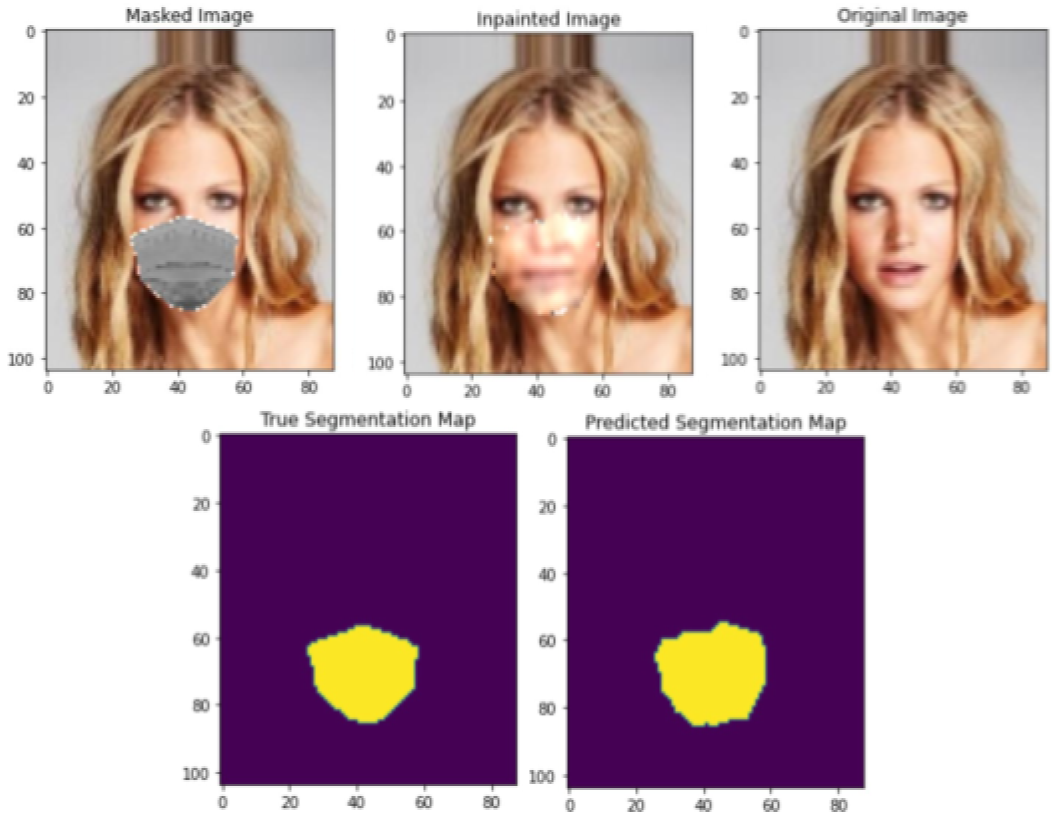

Figure 11. Face unmasking results by full pipeline. Inpainted image is the result of propagating the image thorugh the full pipeline.

Conclusion

The results from these models are promising, as the U-Net is able to accurately segment the masked image, the autoencoder generates realistic inpainted images, and the SRCNN successfully sharpens the autoencoder output. Not only do these components work individually, they are able to work together as well and ignore the imperfections in their input from the previous model's output. Overall, this work shows the promise of combining semantic image segmentation with image inpainting to remove face masks while preserving as many of the original features as possible.

References

A. Criminisi, P. Perez and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," in IEEE Transactions on Image Processing, vol. 13, no. 9, pp. 1200-1212, Sept. 2004, doi: 10.1109/TIP.2004.833105.

Anwar, Aqeel, and Arijit Raychowdhury. Masked Face Recognition for Secure Authentication. 2020.

Dong, C., Loy, C. C., He, K., & Tang, X. (2015). Image Super-Resolution Using Deep Convolutional Networks. arXiv. https://doi.org/10.48550/ARXIV.1501.00092

James Hays, Alexei A. Efros. Scene Completion Using Millions of Photographs. ACM Transactions on Graphics (SIGGRAPH 2007). August 2007, vol. 26, No. 3.

Jeong-Seon Park, You Hwa Oh, Sang Chul Ahn and Seong-Whan Lee, "Glasses removal from facial image using recursive error compensation," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 5, pp. 805-811, May 2005, doi: 10.1109/TPAMI.2005.103.

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv. https://doi.org/10.48550/ARXIV.1412.6980

Liu, Ziwei, et al. “Deep Learning Face Attributes in the Wild.” Proceedings of International Conference on Computer Vision (ICCV), 2015.

“Removing Face Masks with Joligan: DeepDetect.” DeepDetect by JoliBrain, 26 Mar. 2021, https://www.deepdetect.com/blog/15-face-masks-gan/.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv. https://doi.org/10.48550/ARXIV.1505.04597