Unmasked: Reconstructing Masked Faces with Machine Learning

Tian Yu Fan, Grant Perkins, Jean Claude Zarate, Mingjie Zeng

I. Abstract

The accuracy of existing face detection and recognition models are greatly compromised when obstructions, like surgical masks, occlude facial features of the individual. This paper introduces a method which uses facial reconstruction as a new entry for face detection in a mask-wearing context. In the first phase of the model, an object detection network, called EfficientDet-D0, locates the position of the mask. In the second phase, a Gated Convolutional Network and SN-PatchGAN model, in a Generative Adversarial Network, work collaboratively to reconstruct the occluded region of the face. The EfficientDet-D0 model successfully detects masks with a prediction score of 0.966 mAP with an IoU threshold of 50%. The GAN was trained successfully and reconstructed the outline of a human face, but failed to reproduce detailed facial features due to time and hardware constraints.

II. Introduction

Facial detection and recognition have been important recent milestones in artificial intelligence and have significant influence on various fields and applications. This includes identifying individuals for security or attendance purposes. For example, police officers have used facial detection to analyze security footage collected at a crime scene. Modern smartphones also integrate facial recognition to unlock the device for its designated owner.

Unfortunately, however, the accuracy of face detection and recognition models become greatly compromised by obstructions as the number of facial feature landmarks are significantly reduced [1]. As a result of the recent COVID-19 pandemic, the prevalence of wearing face masks on a regular basis has drastically skyrocketed. This new practice has been a great inconvenience for existing facial detection and recognition models. Once this limitation is addressed, existing models will be able to perform their designated tasks despite any masked regions in the input data, improving their robustness and performance under nonideal environments.

III. Background

Within the past three decades, extensive research has been conducted on facial detection and recognition. This has been accompanied by significant advancements and notable breakthroughs. However, obstruction remains to be a serious limitation because important facial landmarks are removed from the dataset, decreasing the accuracy of existing models. In application, this problem arises when accessories, like masks and hats, obstruct facial features on the target individual. Many research groups have previously attempted to address this problem by proposing their own models, which will be discussed in this section.

Weng et. al proposed a partial face recognition approach that takes a pair of masked and unmasked images as inputs [2]. For each image, the model extracts prominent facial feature landmarks based on the various geometric and textural attributes using the Scale Invariant Feature Transform (SIFT) algorithm. Then, each feature point is mapped between the masked and unmasked image using Lowe's matching scheme. The two images are iteratively aligned using a robust point set matching (RPSM) procedure such that they have the same orientation. Finally, the similarity of the two faces is assessed based on the distance between the two aligned feature sets. This method operates on a relatively simple concept, but it requires an unmasked image of the individual to perform proper detection. This information is not always available, making this method ineffective in some situations.

Bagchi et. al proposed a face recognition model that takes a 3D range image as input and processes it using Iterative Closest Point (ICP) algorithm [3]. The obstructed regions are extracted by thresholding depth map values and reconstruction is performed through Principal Component Analysis (PCA). This output is passed onto a recognition system for classification, reaching an accuracy of 91.30%. While this methodology is accurate, the performance of the ICP heavily relies on the initial conditions of the algorithm. Furthermore, the construction of the 3D face requires significant processing power to achieve good results.

Many of the previous facial recognition methodologies constructed a custom dataset containing obstructed faces to fine-tune their original facial detection model [7,8]. This allowed for the models to achieve better precision scores for detection of masked-faces, but the accuracies were still generally undesirable. Other studies preferred to focus on several masked facial features independently, such as eyes and forehead, to further enhance facial recognition models and improve results [6].

This paper introduces a method which uses facial reconstruction as a new entry point for face detection in a mask-wearing context. This transforms the masked-face recognition problem into a general face reconstruction problem through unmasking the occluded face. In the first phase, an object detection network, called EfficientDet-D0, captures the masked region. Additionally, the EfficientDet-D0 model uses transfer learning, which compared to training the model from scratch allows for training starting with a pre-trained model to detect new features in a new dataset yielding better detection rates [5]. In the second phase, a Generative Adversarial Network (GAN) is trained for generative inpainting to obtain a reconstructed face. The methodology is dedicated to reconstructing the facial features of a masked face while maintaining the authenticity of the generated faces. To address this problem, the Gated Convolutional Network and SN-PatchGAN models were selected as the main architectures for facial reconstruction in this paper. This

methodology was inspired by Yu et al.'s research that addresses a free-form image inpainting problem, where the images have masks that can take any shape and position [4]. The models use a custom face dataset that originates from OpenImages V6. The masked face dataset is generated using the MaskTheFace software which overlays masks at different angles to human faces. This computational approach eliminates the requirement of having an unmasked face for performing facial recognition. Furthermore, it addresses the problem of time efficiency since it does not reconstruct a 3D model of the face.

The remainder of this paper will describe the acquisition of the image dataset, capturing masked and unmasked individuals, the implementation for mask detection of an image using EfficientDet-D0, the implementation of the GAN network that reconstructs the facial features of the masked face, and the integration of an evaluator that determines the success of generated images. Furthermore, the following sections will report the experimental findings, formulate an analysis on its performance, and make conclusions on its impact to broader applications.

IV. Methods

This section describes the process for obtaining a dataset of images that capture the face of various masked and unmasked individuals, the implementation of mask detection through EfficientDet-D0, the implementation of facial reconstruction in masked areas using a GAN, and the implementation of an evaluator which determines the quality and success of the generated face.

A. Environment Setup

The successful implementation of this project required three important development tools — a Docker environment, a GitHub organization, and a Google Drive folder.

Docker is a software platform that allows an application to run in an isolated environment from the local host, called a container. The Docker container for this project emulated an Ubuntu 18.04 LTS workspace with various dependencies, including TensorFlow 2.5 and OpenCV 4.5.3. This ensured that all collaborators operate in the same environment and are able to share their work without software conflicts.

GitHub is a website offering cloud-based Git version control and code management services. A GitHub organization was created and shared among all collaborators, allowing for effortless code sharing. The organization contains multiple repositories that serve different purposes — dataset generation, mask detection, and facial reconstruction.

A Google Drive folder was created and shared such that the authors can share large quantities of masked and unmasked images for training neural networks.

B. Dataset Acquisition

The models for mask detection and facial reconstruction were trained on a dataset of images capturing masked and unmasked individuals.

2,932 images of unmasked individuals were downloaded from Open Images V6 under the "Human Face" category. The data is composed of images with one or multiple faces at different angles, zoomed-in at different distances, brightened under different light conditions, and captured with different camera qualities. This allowed the models to train with more diverse environments such that they can perform their assigned tasks under nonideal conditions.

Figure 1. Image from Open Images V6 dataset

For every unmasked image, a corresponding masked image was generated using a computer vision script authored by Aqeel Anwar, called MaskTheFace. This tool uses facial detection to determine the tilt of the head and pinpoint six important facial landmarks. This information is used to warp the image of a mask to optimize its overlay over the face. The script offered various options for mask type, patterns, and color. This project only used blue surgical masks with no special patterns.

Figure 2. Image after MaskTheFace processing

To facilitate data processing during model implementations, the MaskTheFace script was edited such that important information could be written to a CSV file. For each detected face, the image title, image dimensions, coordinates of the facial bounding box, and coordinates of the six facial landmarks were saved to the CSV file.

C. Data Formatting

Each masked and unmasked image were processed by a Python script which resized them to an 800x800 format while preserving the original aspect ratio by adding black filler bars to the appropriate dimension.

Figure 3. Unmasked image resized to 800x800 while preserving aspect ratio.

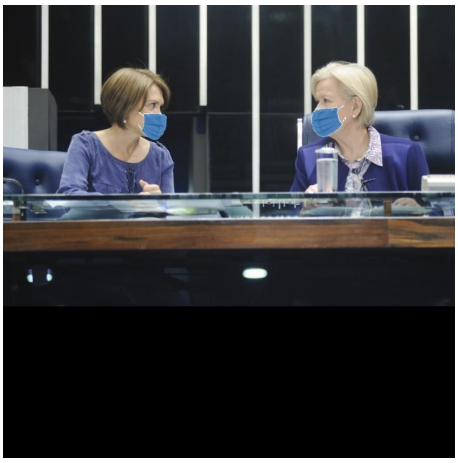

Figure 4. Masked image resized to 800x800 while preserving aspect ratio.

A python script used the mask coordinates stored in the CSV file, which was generated from the MaskTheFace script, to generate black-and-white mask channels, for each mask in each image. These generated images were subsequently stored in a folder for reporting the mask channels to the GAN. Additionally, another CSV file was created which contains the identification of the training images and the masks, and the scaled coordinates of each mask. Since all images were scaled to an 800x800 format while preserving aspect ratio, the coordinates of the mask were recalculated accordingly. This entails taking the coordinates of the point, multiplying them by 800, and finally dividing by the largest dimension of the image.

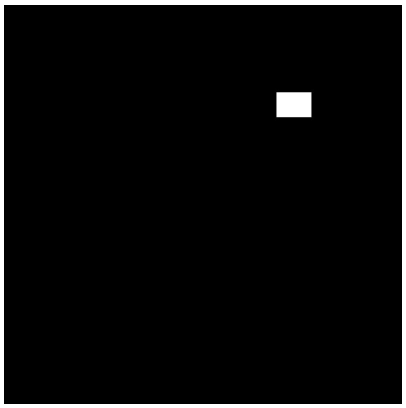

Figure 5. The mask channel for the right mask.

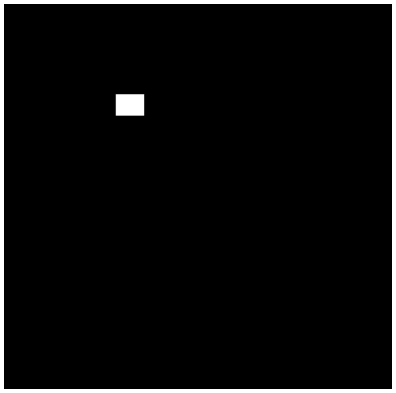

Figure 6. The mask channel for the left mask.

In order to train the mask detection model, the entries stored in the CSV were repurposed such that each recorded image is mapped to its corresponding set of masks, based on the number of previously detected faces. The six landmarks outlining the contour of each mask were used to calculate the coordinates of a rectangular bounding box which encloses the mask. The profile of this bounding box is described by its top left and bottom right coordinates, which reduces the computation complexity of the mask model. The image and its metadata, including height, width, and coordinates of the bounding boxes, were all written to a large TFRecord for the sake of efficiency. The TFRecord format contains large datasets and allows for optimization of the computational speed for the Tensorflow API. The dataset for mask detection was split up such that 80% of images were used for training and the remaining 20% were used for testing.

D. Facial Reconstruction

The purpose of the facial reconstruction model is to remove the masks from the images and recreate these occluded areas with appropriate facial features, derived from machine learning, that is consistent with the remainder of the image and keeps the authenticity of the original image.

To adapt the model for different usages, a YAML file is provided in the implementation such that users can specify the desired shape of their images, the batch size for training, and other specific configurations suitable for their

application. The training model loads these configurations into the program such that they can influence the behavior of the program. The program must also know the location of the directories containing the training dataset, which stores the images of unmasked individuals, and the testing dataset, which stores the images of masked individuals.

The training of the model requires the unmasked images and the location of the masks as inputs. For each image, the coordinate of every mask was retrieved from a CSV file such that the mask channel can be calculated based on the size of the output image. This data is subsequently passed onto the generator to train for facial reconstruction.

The training model is overall composed from a generator and discriminator. In a general sense, the generator learns from the input data and generates images of reconstructed faces based on acquired knowledge. The discriminator attempts to identify whether this output image is real or fake, creating a closed-feedback loop that leads to gradual improvements in learning. A satisfactory generator implementation is achieved when the discriminator can no longer distinguish the difference between a generated face and a real one.

The generator uses gated convolutions and was modeled for image inpainting, under the specific usage of unmasking faces. Unlike vanilla convolutions, gated convolutions discriminate between input pixels, treating some as invalid, to produce more reasonable and accurate results [4]. The occluded regions from the masked images were treated as invalid. Otherwise, the masked regions would cause ambiguity in training, leading to visual artifacts, including inconsistent colors, blurring, and noticeable responses to edges.

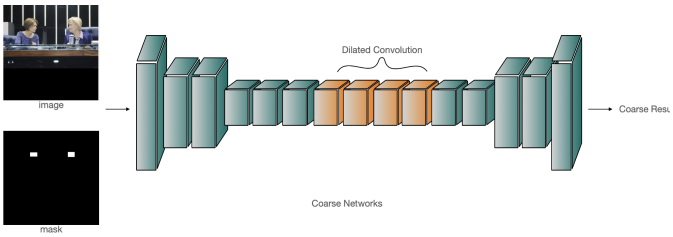

Figure 7. First stage of the convolutional network.

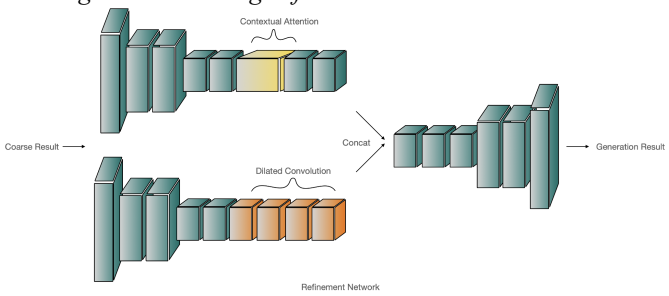

Figure 8. Second stage of the convolutional network.

The generator is composed of two stages. The first stage has 15 convolutional layers and produces coarse results. In the second stage, the pipeline splits into two parallel networks. The first network has 10 convolutional layers while the second one has 8 convolutional layers. The results of these two networks are concatenated and processed through 5 more convolutional layers, which finally outputs the inpainting results.

A Spectral-Normalized Markovian Discriminator, or SN-PatchGAN, was implemented as the discriminator. It utilizes Markovian patches to extract features [9] and uses spectral normalization as its weight normalization method [10]. Through using Markovian patches, the patch size can be small enough such that the model gets fewer parameters, resulting in faster runtime and better results at the same time [9]. Since the discriminator iterates through the entire image patch-by-patch to decide the authenticity of the image, the global discriminator is not necessary in this case. The spectral normalization controls the Lipschitz constant of the discriminator function [4] to stabilize the training of the GAN. Finally, by using SN-PatchGAN, the model uses fewer loss terms to evaluate its performance.

The loss for the generator was calculated through the summation of the hinge loss and the L1 loss functions. The loss for the discriminator was calculated based on the positive and negative weights produced by the spectral normalization.

The testing of the model requires the masked images and the location of the masks as inputs. Once again, for each image, the coordinate of every mask was retrieved from a CSV file such that the mask channel can be calculated based on the size of the output image. This data is subsequently passed on to the generator for facial reconstruction.

E. Mask Detection

The EfficientDet-D0 model was chosen to accomplish face mask detection. An EfficientDet family is a scalable convolutional neural network (CNN) with a weighted bi-directional feature pyramid network (BiFPN) that both feed into a class and box predictor [12]. EfficientDet's CNN is an EfficientNet, the predecessor to EfficientDet. The architecture is shown below.

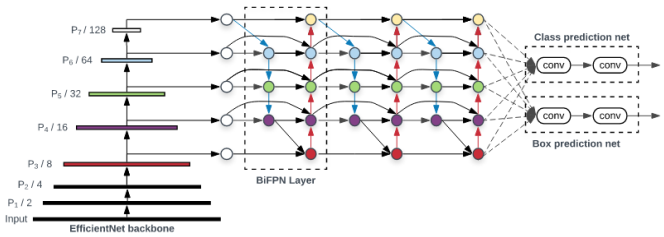

Figure 9. The architecture of EfficientDet. EfficientNet is used as the backbone network, BiFPN is used for the feature network, and there is a shared class-box prediction network. The number of layers for BiFPN and prediction net are dependent on computational resources available. [12]

The EfficientDet-D0 model is the most lightweight from its family since this section of the project only needs reasonable accuracy. The baseline performance of EfficientDet-D0 is comparable to YoloV3 [13].

The mask detection program uses the TensorFlow Object Detection API for transfer learning with the EfficientDet model. The model was trained using the COCO 2017 dataset, and then every layer except the last few were

frozen[14]. This base model runs interference at 35ms per image, with a COCO mean average precision (mAP) of 33.6. The last few layers of this model were re-trained with our custom dataset. This re-training makes the resulting model train faster, more accurate, and require less training data than starting from scratch. The custom dataset contained images of people wearing face masks and the coordinates of those masks.

The model employs a cosine decay learning rate to encourage large initial changes and smaller changes as the model becomes more accurate. To reach an initial learning rate of 0.08, 250 warm-up steps were used.

F. Performance Measurement

The EfficientDet-D0's performance can be evaluated quantitatively by calculating both mAP and recall. These metrics are used universally[12] with object detection neural networks.

Unfortunately, there is no such industry standard for quantitatively evaluating the performance of a GAN, due to the "creativity" a GAN can have with its output. Instead of reaching for a possibly confusing metric, we will evaluate the L1 and L2 distance between real images and generated images, as well as quantitatively evaluate our generated images by visually comparing them with the original. Through these two evaluation methods, we will get an informed sense of the performance of our GAN.

V. Results

This section describes the analysis for the face reconstruction and mask detection results.

A. Facial Reconstruction GAN

The following images showcase the outputs of the GAN when the facial reconstruction GAN was trained for 100 epochs.

Figure 10. Sample output of the GAN with masked individual (left), occluded region (middle), and reconstructed face (right).

In each test case, the outline of the face can be recognized but the facial features are difficult to distinguish. The performance of the model is restricted by the time constraint of this project and the available hardware. Each epoch takes a considerable amount of time, so training the model such that it can reconstruct refined facial features is difficult. In the first row of Figure 10, the shape of the individual's jaw structure is distinguishable and the model has started constructing their mouth as the whites of the teeth can be observed. The obstructed hair region is almost perfectly reconstructed as it is almost indistinguishable from the original image. In the second row of Figure 10, the cheeks and nose bridge of the individual is notable. However, the mask channel does not cover the entirety of the surgical mask, causing the blue hues in the reconstructing image. For each individual, the model was able to match their skin tone. However, further testing is needed to determine whether there is training bias due to the input dataset. In other words, the model might not consider diverse race, age, and gender.

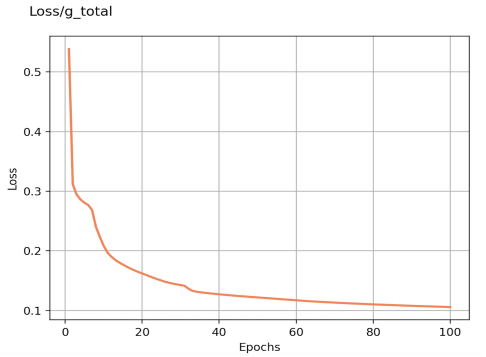

Figure 11. Total loss of GAN vs. Epochs

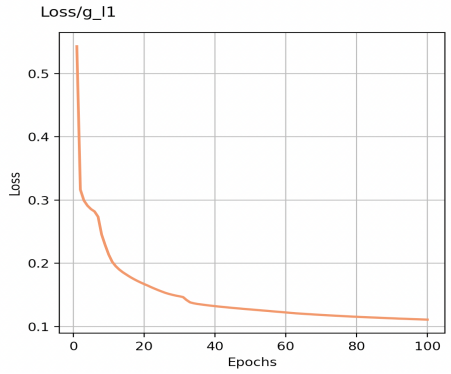

Figure 12. L1 loss of GAN vs. Epochs

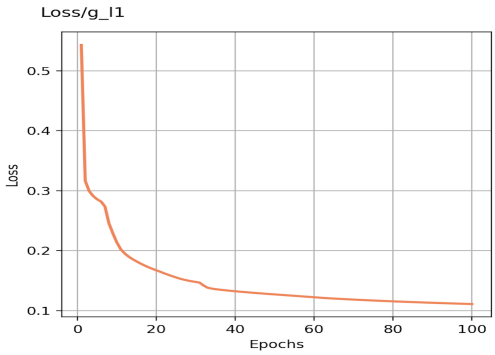

Figure 13.Hinge loss of GAN vs. Epochs

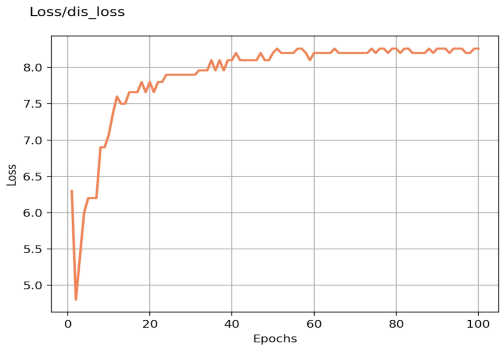

Figure 14.Discriminator loss of GAN vs. Epochs

Figure 11 and Figure 12 demonstrate the steady decline of the L1 and hinge loss functions for the GAN at each iteration. Figure 13 indicates a certain level of the instability of the GAN network and shows, to some extent, the complexity of the features required for face generation. Fortunately, however, the loss variation of our discriminator still tends to be flat. This suggests that the model learned successfully throughout the training as the error decreased consistently. This further insinuates the potential of a successful facial reconstruction model.

And after 300 epochs, we recompare the reconstructed face images with the original unmasked face images to show the performance of GAN in a more intuitive way. The following images demonstrate every stage of GAN sequence and provide a visual comparison of each stage.

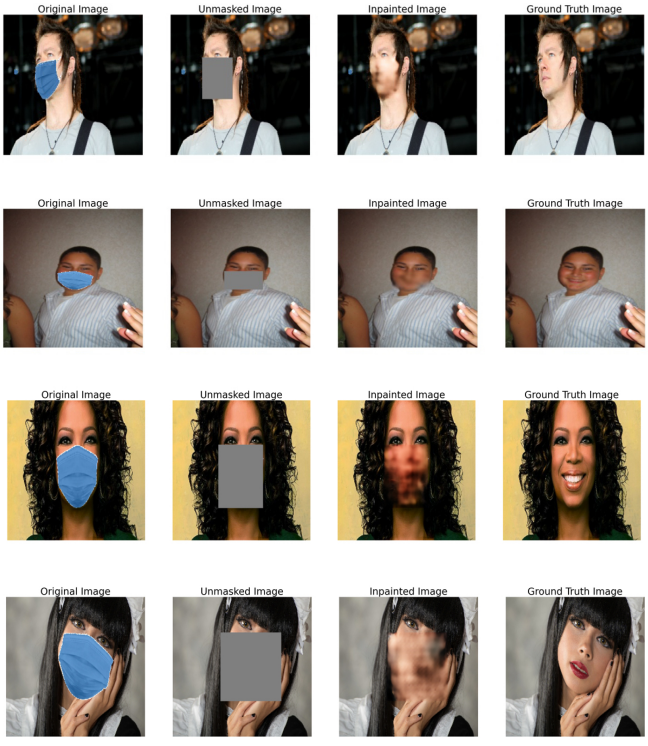

Figure 15. Sample output of the GAN with masked individual (1st), occluded region (2nd), reconstructed face (3rd), and original unmasked face (4th).

Figure 15 contains two positive examples that the GAN reconstructed the faces and their corresponding features properly. In the first Figure 15, the nose and mouth are formed in the suitable position and in their proper shape. The shape of the chin is correct and sideburns can also be generated correctly. Besides, there appears to be a beard between the nose and the mouth, perhaps because the overly long sideburns make the machine determine that the person has a beard. In the second row of Figure 15, the outline of the jaw and nose is highly clear and there is even a smiling mouth being formed. The tone of the generated skin and original one is so harmonious that it even produces a blush color as well.

The bottom two rows of Figure 15 are somewhat strange. In the second from bottom row, the generated face seems to have a mustache. Based on the features on the top half of the face and background information, there is perhaps too much information for the machine to think that this face should have a beard, and too little information that is not clear enough alive for the machine to recognize this as a female. The bottom row of Figure 15 is more special. The face in this picture is tilted. As the mask we detected would be a rectangular box area, one eye of this face is incorrectly obscured. We can tell that the machine tries to generate a mouth in the lower middle of the masked area, which is very unreasonable for a tilted face. Given that the general occlusion region does not contain eyes, the features that the machine can learn to generate eyes should be very poor, which makes it very difficult to generate eyes normally.

In order to measure the results more objectively, we calculated the L1 and L2 losses of the generated face images and the original masked face images. The average L1 score for the reconstructed face images after 300 training epochs is 6169 and the average L2 score is 1345.

The performance measure for facial reconstruction shows that the learning efficiency of this network seems to be quite good compared between the results of the faces generated by the 100 training epochs and the faces generated by the 300 training epochs. The reconstructed faces generated in the 300 epochs contain significantly more information, allowing more facial features to be recognized, such as nose, mouth, facial contours, etc.

Also in the process of measuring performance, we recognize that due to the limitations of the mask area detection, which can only be box area, there will be some special cases similar to Figure 18 arising. This will not only have an impact on the reconstruction of test images, but also have an unpredictable impact on the results if there are also some features in the training set that do not belong to the mask region being learned.

B. EfficientDet Face Mask Detection

The EfficientDet-D0 model was retrained for 2000 epochs on the custom face mask dataset for this project. Due to hardware limitations, the batch size was lower than ideal expectations, so the total loss graph is rougher than typically expected. Despite this roughness, the model's total loss

clearly converges to the 0.12 region. This loss was mostly influenced by the classification loss of detection candidates.

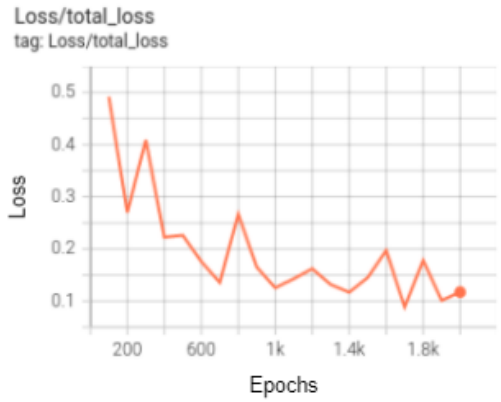

Figure 16. Total loss of EfficientDet-D0 vs. Epochs

The classification loss for the model converged to a region below 0.1, which means lower overall learning errors and is an indication of an accurate model.

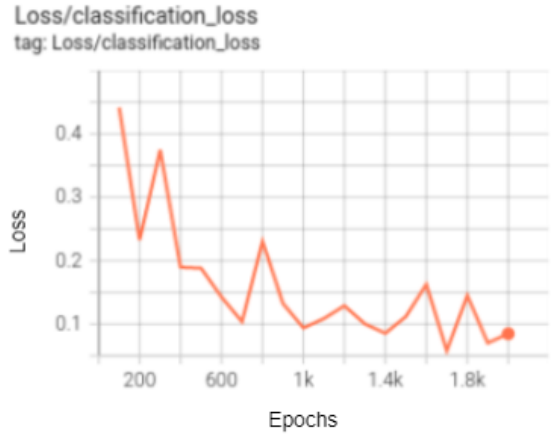

Figure 17. Classification loss of EfficientDet-D0 vs. Epochs

The localization loss for the model converged to 0.0032, which is sufficient for the use cases in this project.

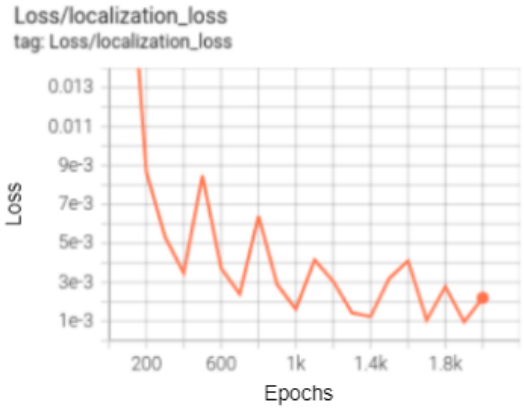

Figure 18. Localization loss of EfficientDet-D0 vs. Epochs

The two important metrics for model performance with object detection neural networks are mAP and recall. The metrics results are plotted below in Figure 19 with intervals of 400 epochs. The mAP reaches a peak of 0.69 when the intersection over union (IoU) threshold is 95%. With a lower IoU threshold of 50%, the model's performance improves significantly, reaching 0.966 mAP. Good mAP prediction scores are over 0.5 mAP, suggesting that the mask detection model is accurate.

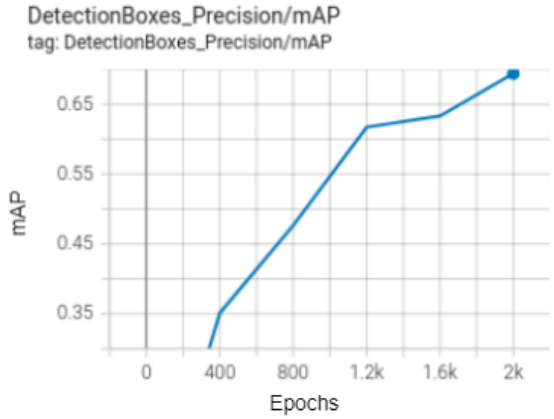

Figure 19. mAP score with IoU threshold of 95%

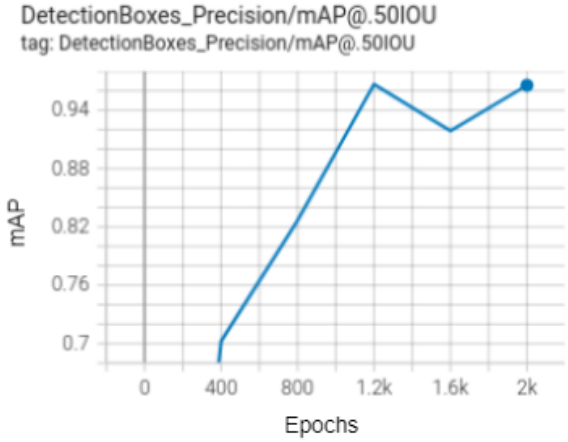

Figure 20. mAP score with IoU threshold of 50%

The average recall metric has a similar curve, reaching .732 by epoch 2000. Since most of the images have multiple face masks, the average recall only accounts for a maximum of ten detected masks.

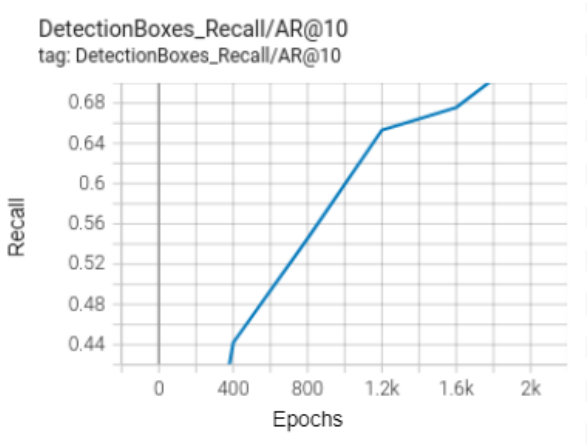

Figure 21. Average recall vs. Epochs

The image below demonstrates the output of the mask detector. The left bounding box is the region of the detected mask while the right bounding box is the ground truth label.

Figure 22. Detected mask (left) and truth label (right)

VI. Conclusion

In this project, we successfully implemented a novel model, involving state-of-the-art machine learning algorithms, to detect face masks and generate human faces. An EfficientDet-D0 model was trained to predict the bounding boxes of surgical masks from an image. Furthermore, a Gated Convolutional Network, composed of a generator and discriminator, was trained to reconstruct human faces in occluded regions, removed by the EfficientDet, and to distinguish the difference between generated and real human faces. These algorithms combined successfully to reconstruct human facial features from the occluded regions. This methodology eliminates the dependency on having the unmasked image in the input dataset, resolving the limitation from Weng et. al's experimentations. Furthermore, our proposed model does not require the reconstruction of a 3D range image, as proposed in Bagchi et al's paper, significantly reducing processing complexity.

The implementation of the mask detection model using EfficientDet-D0 was successful as shown by the 0.966 mAP prediction score with a IoU threshold of 50%. This is an extremely high mAP, which is more than suitable for our use-case. Figure 19 clearly illustrates the strength of our object detection model. It is worth discussing the quality of the object detection portion of our dataset when looking at model performance. As shown in Figure 22, in many cases the ground truth label is not a perfect bounding box to the true mask. The right side of the ground truth label in Figure 27 has a substantial gap of non-mask area. With a better quality dataset, our model may perform even better than the .966 mAP with 50% IoU threshold.

The implementation of the face reconstruction model using Gated Convolutional Network and SN-PatchGAN was able to successfully learn and generate overall bone structures and facial features. However, it failed to generate detailed and specific facial features because the training was limited by the time and available hardware constraints. The model should be able to generate more precise faces given more training time. And due to the existence of special results as Figure 15, we should think about the possibility of image classification and also about the diversity of training sets. For the problem that the mask area can only be rectangular and therefore contains some information that should not be in the physical mask area, image segmentation could be employed to better bound the face mask. Alternatively, more uniform face images could be used. If every face was facing towards the camera, was centered, and were of consistent size, then we may have better results. However, in a real-world setting people are not always facing the camera, perfectly vertical, centered in the frame, or otherwise perfect, making our model more robust in a real-world setting.

The first significant improvement for future works is to acquire a larger dataset for the training of the Gated Convolutional Network. The current implementation of the GAN requires a very large neural network, so the sub-3000 images from the current dataset is insufficient for generating better images than the blurry faces which are currently produced. With a larger dataset, in the range of tens of thousands of images, more realistic facial features could be reconstructed. A side effect of the small dataset can be observed in Figure 17, where it is apparent that the dataset underrepresents people of color. The generator believes that the darker pigment of the woman's face is a beard, and not her skin. With more data, the generated images will have less blurry facial features, and more accurately reproduce more diverse faces. This includes the consideration of race, age, and gender of the individual, which can be achieved by obtaining a more diverse dataset. The results of the model were also limited by time and hardware constraints. The submission date of this project limited the training time of the model, resulting in premature results. The weights of the model could have converged to better values, if more time was available. Furthemore, the model was trained on a team member's personal computer, which severely increased the training time of each epoch. In the future, the training should be out-source such that the training time is significantly reduced.

There were also errors in the generated dataset for training the models. Occasionally, the mask channel failed to cover the entire surgical mask, leaving blue patches in the input image. This misleading information confuses the generator

as the surgical mask does not belong on a normal human face. Furthermore, the surgical masks sometimes covered the upper facial features of the individuals. Since the GAN is not specifically trained to generate those regions, the constructed face becomes distorted.

With a properly trained SN-PatchGAN, the next step would be to implement a real-time inference system. We believe it would be interesting to see both the performance and the realism of the models in a real use case. Developing a simple program that takes a camera feed, runs it through the two models, and outputs the generated images to a video stream would be ideal.

Recognizing the complexity of the two-model pipeline, it may be necessary to perform optimizations to have the pipeline perform inference at a reasonable speed. Such optimizations could include quantization-aware training[16] and model distillation[17]. The goal would be to run inference at at least 15 frames per second, as anything slower would be unreasonable.

VII. References

1. 1.Q. Mundial, M. S. Ul Hassan, M. I. Tiwana, W. S. Qureshi and E. Alanazi, "Towards Facial Recognition Problem in COVID-19 Pandemic," 2020 4rd International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM), 2020,pp. 210-214, doi: 10.1109/ELTICOM50775.2020.9230504.
2. R. Weng, J. Lu and Y. Tan, "Robust Point Set Matching for Partial Face Recognition," in *IEEE Transactions on Image Processing*, vol. 25, no. 3, pp. 1163-1176, March 2016, doi: 10.1109/TIP.2016.2515987.
3. Bagchi P., Bhattacharjee D., Nasipuri M. (2016) Registration of Range Images Using a Novel Technique of Centroid Alignment. In: Chaki R., Cortesi A., Saeed K., Chaki N. (eds) Advanced Computing and Systems for Security. Advances in Intelligent Systems and Computing, vol 396. Springer, New Delhi. https://doi.org/10.1007/978-81-322-2653-6_6
4. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., & Huang, T. (n.d.). *Free-Form Image Inpainting with Gated Convolution*.
5. S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.
6. 2.Anwar, A. and Raychowdhury, A., “Masked Face Recognition for Secure Authentication”, arXiv e-prints, 2020.
7. 3.Montero, D., Nieto, M., Leskovsky, P., and Aginako, N., “Boosting Masked Face Recognition with Multi-Task ArcFace”, *arXiv e-prints*, 2021.
8. 4.Vu, H, Nguyen, M, Pham, C. "Masked face recognition with convolutional neural networks and local binary patterns". Applied Intelligence 2021.
9. Li C., Wand M. (2016) Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9907. Springer, Cham. https://doi.org/10.1007/978-3-319-46487-9_43
10. Miyato, T., Kataoka, T., Kayoma, M., & Yoshida, Y. (2018, February 16). *Spectral Normalization for Generative Adversarial Networks*.
11. Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. (n.d.). *Image-to-Image Translation with Conditional Adversarial Networks*.
12. Tan, M., Pang, R., & Le, Q. (n.d.). *EfficientDet: Scalable and Efficient Object Detection*.
13. Wu, D. (n.d.). *Using channel pruning-based YOLO v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments*.
14. Rathod, V. (n.d.). *TensorFlow 2 Detection Model Zoo*. Retrieved November 30, 2021, from https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md
15. Geitgey, A. (2017). *Face Recognition*. Retrieved November 30, 2021, from https://github.com/ageitgey/face_recognition
16. Floropoulos, & Tefas, A. (2019). Complete vector quantization of feedforward neural networks. Neurocomputing (Amsterdam), 367, 55–63. https://doi.org/10.1016/j.neucom.2019.08.003
17. Heo, B., Lee, M., Yun, S., & Choi, J. Y. (2019). Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In Proceedings of the AAAI Conference on Artificial Intelligence (pp. 3779–3787).