Abstract: Research on face recognition with masked faces has been increasingly important due to the prolonged COVID-19 pandemic. To make face recognition practical and robust, a large amount of face image data should be acquired for training purposes. However, it is difficult to obtain masked face images for each human subject. To cope with this difficulty, this paper proposes a simple yet practical method to synthesize a realistic masked face for an unseen face image. For this, a cascade of two convolutional auto-encoders (CAEs) has been designed. The former CAE generates a pose-alike face wearing a mask pattern, which is expected to fit the input face in terms of pose view. The output of the former CAE is readily fed into the secondary CAE for extracting a segmentation map that localizes the mask region on the face. Using the segmentation map, the mask pattern can be successfully fused with the input face by means of simple image processing techniques. The proposed method relies on face appearance reconstruction without any facial landmark detection or localization techniques. Extensive experiments with the GTAV Face database and Labeled Faces in the Wild (LFW) database show that the two complementary generators could rapidly and accurately produce synthetic faces even for challenging input faces (e.g., low-resolution face of 25 × 25 pixels with out-of-plane rotations).

1. Introduction

Face recognition (FR) is one of the most important problems in computer vision area as it has a wide range of real-world applications. Conventional FR functions are generally performed with primary facial features such as eyes, nose, and mouth, assuming non-occluded faces [1]. However, one may encounter situations where faces are largely occluded with unwanted objects such as accessories. Facial occlusion is one of the most challenging problems in FR because the occluded parts on the human face can be arbitrary in position, size and shape [2,3]. Due to the prolonged COVID-19 pandemic, wearing masks has been recommended in many countries to avoid spreading of the virus. Masked face recognition (MFR) is a special case of occluded FR [3]. It has been known that MFR is very challenging because the features of mouth and nose are severely damaged; thus, the discriminating features are greatly reduced [3]. To address the challenges of MFR, some research efforts with deep learning approaches have been made [4,5]. One of the most straightforward methods for MFR is discarding occluded face region. In general, this approach first detects the occluded regions and discard them as preprocessing [4,6]. The authors in [4] have proposed to apply three pre-trained deep convolutional neural networks (CNNs) [7,8], called VGG-16 [9], AlexNet [7], and ResNet-50 [10], aiming to extract deep features from the remaining facial regions (mostly eyes and forehead regions). The main drawback of the occlusion discarding based approach is limited discriminating capability. The authors in [5] have proposed a deep learning method based on Generative Adversarial Network (GAN) [11] that removes the mask region and reconstructs the missing region with synthetic facial appearance. They have tried to preserve structural and shape consistency in the recovered face using two discriminators where one seeks to learn the global structure of
the face and then the other focuses on learning deep missing regions (such as mouth and teeth regions which are far away from the occlusion boundary caused by wearing the mask) [5].

MFR methods often need large datasets with masked face images to make full use of the learning capability of deep learning technologies. To address this issue, the authors in [12] have constructed Real-world Masked Face Recognition Dataset (RMFRD) by collecting 2203 images of 525 subjects wearing masks, and 90,000 images of the same 525 subjects without masks [12]. However, in this dataset, face image pairs of with and without mask are not available because the collected face images are basically real-world images. In order to expand the volume and diversity of the masked face dataset, the authors in [12] have generated synthetic images of masked faces in the existing public large-scale datasets (i.e., LFW [13] and Webface [14] datasets). As a result, this simulated datasets (called SMFRD) contain about 500,000 face images of 10,000 subjects. To improve the data manipulation efficiency during the construction of SMFRD dataset, an automatic mask wearing software based on Dlib library [15] has been utilized. Similarly, in [16], 68 facial landmarks are detected using Dlib library and various synthetic masks are generated by connecting the points which form a polygon filled with a color. In [17], a Dlib based face landmark detector is used to identify the face tilt and six key features for applying mask. According to the face tilt, a template mask is selected from the mask library. Next, the selected template mask is warped based on the key features to fit properly the face appearance. Such synthetically masked face images [12,16,17] are advantageous over real masked images in a sense that the synthetic face images can be used along with their original (i.e., unmasked) face images. For example, face image pairs (masked/unmasked) are required in the training stage of deep networks for unmasking of masked face and filling the missing region. In addition, the face image pairs can be useful for investigating the effect of FR performance degradation caused by wearing masks. However, the geometric based methods in [12,16,17] that rely on landmark localization could suffer from low resolution or occlusion in real-world face images.

To cope with the aforementioned limitations, this paper proposes a compact and practical method to synthesize masked face. The main contributions of this paper are twofold:

1. A two-stage generative model based on cascading two convolutional auto-encoders (CAEs) [18] is introduced in this paper. The goal of the former CAE is to obtain a virtual mask pattern suitable for an input face image in terms of pose view. The latter CAE aims to generate a segmentation map to localize the mask pattern region. Based on the segmentation map, the mask pattern can be successfully fused with the input face by means of simple image processing techniques (e.g., pixel-wise sum). Different from the methods in [12,16,17], the proposed generative model relies on face appearance reconstruction without any facial landmark detection or localization techniques.

2. With the appearance-based approach described above, the proposed generative model can be practically used for constructing large datasets of masked face images. As demonstrated in experiments, the proposed method is able to process low resolution faces of 25 × 25 pixels with acceptable synthesis results without high loss in recognition rate (refer to Section 4.3). Additionally, the proposed method works well with moderately rotated face images. It is possible to complete learning of the two complementary CAEs with tens of seconds using a PC with a single GPU (refer to Table 1). Hence, one could easily extend the cascaded network for applying various mask patterns.
Table 1. Computation Time For $G_p$ and $G_r$. Here, $80 \times 80$ was selected as facial resolution.

| Generator | Learning (s) | Inference (for Processing 397 Face Images) (s) |
|-----------|-------------|---------------------------------------------|
| $G_p$     | 25.66       | 0.14                                        |
| $G_r$     | 25.04       | 0.14                                        |

The rest of this paper is organized as follows. Section 2 presents an overview of the proposed method. Section 3 describes in detail the cascading of CAEs to obtain a synthetic masked face. Section 4 presents experimental results to demonstrate the robustness and efficiency of the proposed method. The conclusion is drawn in Section 5.

2. Overview of Proposed Method

The proposed method aims to transfer an unmasked face image (input) to a masked face image (output). Specifically, for a given unmasked face image, a masked face image is generated by overlaying a fake mask pattern on the corresponding face. The proposed method contains two sequential stages, each of which utilizes a CAE. Auto-encoder is a method specialized in unsupervised learning, consisting of an encoder and a decoder. The encoder learns to translate the input into an internal representation. The decoder learns to convert the internal representation into the same form as the input. The best-known neural network for image modeling is CNN which enables to effectively retain the structural information connected between pixels of an image. For this reason, CNN has been successfully applied to an auto-encoder where CNN is deployed in both the encoder and decoder. Figure 1 shows an overview of the proposed method. In the stage of mask pattern generation, the CAE takes an unmasked face image as input (e.g., Figure 2a), and produces a virtual image of face wearing a mask. For this purpose, the CAE needs to be pretrained for the source domain and target domain, which are unmasked face and masked face, respectively. Note that the difference between the two domains is whether mask is present or not. It should be noted that the main objective of this stage is to obtain a mask pattern image that accurately fits the input face in terms of position, size, and shape. To this end, the CAE is required to faithfully reconstruct a pose-alike face for the input face image. In this paper, the face detection function in the Dlib library is used to detect face region. The input face image is obtained by aligning the detected face region based on the two eye-coordinates. This allows the CAE to mainly focus on the out-of-plane rotation (pitch or yaw) during reconstruction of a pose-alike face while the effect of the in-plane rotation (roll) could be suppressed.
Figure 1. An overall framework of the proposed method. In this figure, an RGB face image of 80 × 80 pixels is used for input as an example. The stage of mask pattern generation aims to produce pose-alike face with a mask pattern reflecting the pose view of an input unmasked face. The stage of mask region extraction focuses on localizing the mask pattern by producing the segmentation map. In the stage of image masking and fusion, the mask pattern was fused with the input unmasked face, resulting in a realistic masked face.

(a) \( f \)
(b) \( \hat{f}_m \)
(c) \( \hat{s} \)
(d) \( \hat{f}_m \)

Figure 2. (a) Input of the generator \( G_p \). (b) Output of the generator \( G_p \). (c) Output of the generator \( G_r \). (d) The synthesized face.

In the stage of mask region extraction, the secondary CAE takes the output of the masked pattern generation and predicts a segmentation map which localizes the mask region on the face. As illustrated in Figure 1, the bright region corresponds to the mask region. The segmentation map aims to fuse the mask pattern with the input face for the purpose of mask overlay on face. Similar to the previous stage, the CAE needs a pretraining step for two different domains which are masked face and its segmentation map indicating the mask region, respectively. The inputs (i.e., the unmasked face and the pose-alike face wearing a mask) to the two CAEs and the output (i.e., segmentation map) of the secondary CAE are readily fed into the image masking and fusion stage which combines the unmasked face and the mask pattern, resulting in a realistic face wearing mask (e.g., Figure 2d). Details on the proposed method for the masked face synthesis will be given in Section 3.
3. Masked Face Synthesis Using the Proposed Generative Model

3.1. Mask Pattern Generation

This section describes in detail the mask pattern generation which is the former stage of the proposed generative model. The CAE used in this stage is called mask pattern generator (denoted by \( G_p \)). Given an input (unmasked) face \( f \), \( G_p \) generates a pose-alike face \( \hat{f}_m \) with a mask pattern. It is expressed as

\[
\hat{f}_m = G_p(f).
\]

Figure 2b shows some examples of \( \hat{f}_m \). As is seen, the pose view of each \( \hat{f}_m \) is similar to that of the associated face \( f \) in Figure 2a. Note that, during the face generation, \( G_p \) is likely to pay more attention to the global appearance (e.g., pose view) than the local details due to the compressed representation at the middle of the CAE.

Figure 1 shows the network architecture of \( G_p \) used in this paper. An RGB face image of \( 80 \times 80 \) pixels is converted to a cubic (Conv1) consisting of 16 feature maps of \( 78 \times 78 \) pixels, which becomes another smaller cubic (Conv2). These convolution layers are designed to retain the spatial relationships in the image data. After the encoding process, in the middle, there is a fully connected auto-encoder and its hidden layer consists of 225 nodes. The 225 nodes are called pose vector in this paper because those are expected to encode pose-related information. The decoder is symmetric to the encoder in terms of layer structure.

In order to learn \( G_p \), let \( f^{(i)}_m \) denotes the \( i \)-th unmasked face (source domain) and \( f^{(i)}_m \) denotes the corresponding masked face (target domain). Note that \( i \) is index of a face image pair \((f^{(i)}_m, f^{(i)}_m)\) contained in the training set. To guarantee that the only difference between \( f^{(i)}_m \) and \( f^{(i)}_m \) is the presence of mask, it is possible to create a synthetic face \( \hat{f}^{(i)}_m \) by overaying a virtual mask pattern on \( f^{(i)}_m \) with a manual process (e.g., using Adobe Photoshop) or an automatic software tool. In this paper, a smart phone camera application called SNOW [19] is used for automatically producing the synthetic masked faces. The application is characterized by virtual stickers using augmented reality and photographic filters [19]. Figure 3 shows some image pairs (masked/unmasked) used for learning \( G_p \). To construct the training set, face image pairs have been collected from GTAV Face database [20]. Various facial pose views are included in the training set to deal with different pose views when generating mask patterns. The used faces can be roughly categorized into seven groups as shown in Figure 3. To learn \( G_p \), the loss function \( L_p \) is defined as follows:

\[
L_p = \frac{1}{N} \sum_{i=1}^{N} \text{MSE}(f^{(i)}_m, \hat{f}^{(i)}_m),
\]

where \( \text{MSE}(f^{(i)}_m, \hat{f}^{(i)}_m) \) is the mean squared error [21] which computes the average of the squares for the pixel differences between the \( i \)-th target masked face \( f^{(i)}_m \) and the \( i \)-th pose-alike faces denoted by \( \hat{f}^{(i)}_m \) (i.e., output of \( G_p \) for \( f^{(i)}_m \)). \( N \) is the size of a mini batch.

![Image pairs of unmasked face and masked face used for learning the mask pattern generator.](image)

### Figure 3

The masked face images are obtained by using the SNOW application. The pose views of the used faces can be roughly categorized into seven groups (left-to-right): (1) \(-45\text{-}60\) degrees; (2) \(-15\text{-}30\) degrees; (3) frontal; (4) \(+15\text{-}30\) degrees; (5) \(+45\text{-}60\) degrees; (6) up; and (7) down.
3.2. Mask Region Extraction

As shown in Figure 2a,b the identity-related appearance in each masked face $f_m$ is clearly different from that in the input face $f$. This is due to the fact that if a subject in $f$ is not present during the learning of $G_p$, a conventional CAE is not capable of faithfully reconstructing the identity-related appearances. Furthermore, $f_m$ looks very blurry. To cope with the aforementioned problems, the generated mask pattern is fused with the input face $f$ in the proposed method. In order to obtain the only mask pattern region from $f_m$, this section describes the mask region extraction which is the latter stage of the generative model. The CAE used in this stage is called mask region extractor and denoted by $G_r$. During the inference (or prediction), the input to $G_r$ is $f_m$ which is the output of the $G_p$. Then, $G_r$ produces a segmentation map $\hat{s}$ as

$$\hat{s} = G_r(f_m),$$

where $\hat{s}$ aims to localize the mask region in $f_m$. Figure 2c shows some example images for $\hat{s}$. As illustrated in Figure 1, the mask region extractor is the same as the mask pattern generator in terms of network architecture.

To learn $G_r$, let $s^{(l)}$ denotes the segmentation map (target domain) associated with the $j$-th masked face $f_m^{(l)}$ (source domain). $j$ is index of a pair $(f_m^{(l)}, s^{(l)})$ used as training data. The segmentation map images have been manually created by the author, each of which is a binary image. Figure 4 shows examples of a masked face $f_m^{(l)}$ and its segmentation map $s^{(l)}$, which can be used as training data. Similar to $G_p$, the mean squared error is used to define the loss function $L_r$ for learning $G_r$ as follows:

$$L_r = \frac{1}{M} \sum_{j=1}^{M} \text{MSE}(s^{(l)}, \hat{s}^{(l)}),$$

where $\hat{s}^{(l)}$ denotes the output of $G_r$ for $f_m^{(l)}$. $M$ is the size of a mini batch.

![Figure 4](image-url). Examples for a pair of masked face image (left) and segmentation map image (right) that can be used to learn $G_r$.

3.3. Image Masking and Fusion

Because each segmentation map $\hat{s}$ generated using (3) is not a binary image as shown in Figure 5, it cannot be directly used for the purpose of fusing mask pattern with input face. Thus, it is converted into a binary image by using grayscale transformation and binarization. To make the binary segmentation map clean, it is processed via simple image morphology operations. Specifically, erosion operation is first applied, which is followed by dilation operation. Figure 5 shows an example of segmentation map denoted by $\tilde{s}_b$. Then, the synthetic masked face $\tilde{f}_m$ can be obtained by using the following operations:

$$\tilde{f}_m = BA(\tilde{s}_\text{binv}, \cdot f) \oplus BA(\tilde{s}_b, \cdot \tilde{f}_m),$$

where $BA$ is bitwise and operation used for image masking and $\oplus$ is image sum operation. $\tilde{s}_\text{binv}$ is the grayscale inverted image of $\tilde{s}_b$. 


4. Experiment

This section presents extensive experiments for the proposed method using Labeled Faces in the Wild (LFW) database [13] and GTAV Face database. To learn the mask pattern generator $G_p$ and the mask region extractor $G_r$, Adam [22] was used as optimizer and epoch was set to 100.

4.1. Results for Various Pose Views and Resolutions

This section aimed to investigate the effectiveness of the proposed method under different pose views and resolutions of input faces. GTAV Face database was used in this experiment. The training set consisted of 903 images (i.e., 301 unmasked faces, 301 masked faces, and 301 segmentation maps) from 22 (out of 44) subjects, and the testing set consisted of 397 unmasked face images from the remaining 22 subjects. Seven different pose views were considered: (1) $-45$~$60$ degrees; (2) $-15$~$30$ degrees; (3) frontal; (4) $+15$~$30$ degrees; (5) $+45$~$60$ degrees (6) down; and (7) up. For the case of facial resolution, three different pixel sizes were used: (1) $80 \times 80$; (2) $40 \times 40$; and (3) $25 \times 25$. Furthermore, $80 \times 80$ pixel images were downsampled to obtain $40 \times 40$ and pixel images. From the results of the masked face synthesis illustrated in Figure 6, the following observations were made:

1. The proposed method was able to generate similar mask synthesis results for the three different facial resolutions of an input face image. This demonstrated that the proposed method was robust to variation in facial resolution.
2. The generated mask pattern fitted accurately the faces with moderate out-of-plane rotations in pitch or yaw. This was because the generator $G_p$ could be learned to reconstruct pose-alike faces (refer to Section 4.2) by utilizing training faces with different pose views.
3. Thanks to the reconstruction ability of the appearance-based method, the proposed model was able to generate the masked faces even for the faces occluded by a hand (see Figure 6c).

Figure 5. Examples for a segmentation map $s$ (left) and the processed binary segmentation map $s_b$ (right).
Figure 6. Results for different facial poses and resolutions. (a) −45–60 degrees. (b) −15–30 degrees. (c) frontal. (d) +15–30 degrees. (e) +45–60 degrees. (f) down. (g) up.
Table 1 shows the computation time required for the learning and inference. Because the two generators could be learned within one minute, one could immediately retrain them for masked face synthesis by applying different mask patterns. Using the learned networks, it took only 0.28 s to process 397 face images. This result demonstrated the low computation cost of the proposed method.

4.2. Analysis on Facial Pose

The goal of this experiment was to investigate that the encoded pose vector in $G_p$ was useful for reconstructing similar faces to an input face in terms of pose view. As shown in Figure 7, three face images with different pose views were selected as the query faces from the testing set described in Section 4.1. Next, the encoded pose vector of each query face image was compared with those of the remaining face images within the testing set. Using the Euclidean distance, the most similar faces to the query faces were retrieved. From the result in Figure 7, one could see that the retrieved faces looked similar in pose view to the query face. This demonstrated that a pose vector contained pose-related information that could be useful for successfully generating a pose-alike face.

![Figure 7](image_url)

Figure 7. Face retrieval result. Faces were sorted in ascending order by pose vector distance (i.e., the smallest Euclidean distance value is ranked as 1st).

4.3. Analysis on Facial Resolution

This section aimed to quantitatively verify that the proposed generative model was robust against variation in facial resolution. For this, the FR accuracies (in recognition rate) were measured for the three different facial resolutions present in Figure 6. Similar to the method described in Section 4.1, $80 \times 80$ pixel images were downsampled to obtain $40 \times 40$ and $25 \times 25$ pixel images. After that, the resized face images were fed to the two-stage generator consisting of $G_p$ and $G_r$ in order to synthesize the masked face images. For each facial resolution, a total of 397 masked face images were obtained by applying the proposed method to 397 unmasked face images in the testing set described in Section 4.1. Five-fold cross validation was adopted to measure the recognition rates. Note that the recognition rates measured for the unmasked face images (e.g., face images of the ‘Input’ columns in Figure 6) were also included for comparison. As illustrated in Figure 8, a conventional CNN classifier was employed for measuring the facial recognition rates. The first convolution layer used 64 filters with sizes of $3 \times 3$ which was followed by a $2 \times 2$ max pooling layer to reduce the spatial size of the feature maps and computational cost of the network. Next, the second and third convolution layers employed 64 filters and 32 filters, respectively with sizes of $3 \times 3$. A fully connected layer was included at the end of the CNN network to classify subject class. The output layer had a total of 22 nodes corresponding to 22 subject classes. In order to learn the CNN model, epoch and batch size were set to 100 and 5, respectively. Adam was used as optimizer.
The recognition rates for the three facial resolutions were illustrated in Figure 9. The average performance gaps between the masked face images and the unmasked face images were around 3%. It was worth noting that the performance gap was maintained for face images with low resolution (i.e., $25 \times 25$ pixels).

To further investigate the effect of using different facial resolutions of masked face synthesis on recognition rate, the masked face images generated from the unmasked face images of $80 \times 80$ pixels were resized to $25 \times 25$ pixels and their recognition rate was measured. The recognition rate was 73.54% which was very similar with the result (73.80%) for the masked face images generated from the unmasked face images of $25 \times 25$ pixels (see Table 2). These results demonstrated that the proposed generative model was able to preserve discriminative information (i.e., identity-related facial appearance) even when synthesizing masked face at low-resolution.

Table 2. Effect of using different facial resolutions of masked face synthesis on recognition rate.

| Performing Masked Face Synthesis | Performing Face Recognition | Recognition Rate (%) |
|----------------------------------|-----------------------------|-----------------------|
| at $80 \times 80$ pixels          | at $80 \times 80$ pixels    | 84.15                 |
| at $40 \times 40$ pixels         | at $40 \times 40$ pixels    | 81.11                 |
| at $25 \times 25$ pixels         | at $25 \times 25$ pixels    | 73.80                 |
| at $80 \times 80$ pixels         | (synthesizing masked face at $80 \times 80$ and resizing it to $25 \times 25$ pixels for face recognition) | 73.54 |

4.4. Results on Real-World Face Images (LFW Database)

In this section, the effectiveness of the proposed method was investigated using another dataset called LFW. LFW was a database of face photographs designed for studying the problem of unconstrained face recognition, which contained real-world face images of faces collected from the web. LFW database was much more challenging than GTAV Face database due to the uncontrolled conditions of the face acquisition. To consolidate
training data, in addition to the original training set consisting of 903 images, 397 faces (i.e., testing set in Section 4.1) were also included in the construction of a new training set. To this end, a total of 1191 images (i.e., 397 unmasked faces, 397 masked faces, and 397 segmentation maps) were obtained. To make the new training set more complete in terms of face pose view, 2094 (=1191 + 903) images were horizontally flipped. As a result, a total of 4188 images were used as the training data.

Three geometric based methods \cite{12,16,17} were present in this section for comparison purposes where the source codes provided by the authors were used to obtain reliable results. For the comparison methods and the proposed method, face images of 80 × 80 pixels were used as input. Figure 10 illustrates the comparison results on a number of face images from LFW database. For the case of \cite{12}, the synthetic faces looked unrealistic because they had cartoon-like mask patterns. It was shown in Figure 10d that the mask patterns obtained by using the method in \cite{16} looked binary masking maps due to the very simple algorithm (i.e., connecting the points which formed a polygon filled with a color). The method in \cite{17} achieved better-fitting mask patterns by warping the mask according to estimated mask key points. However, due to the warping of a two-dimensional mask pattern image, one might feel the small gaps between the face and the mask pattern (see the fifth and sixth face images in the fifth column of Figure 10). The proposed method was comparable with the method in \cite{17}. Because the proposed method was based on generating a pose-alike face wearing a mask pattern, the mask pattern could fit accurately and tightly to the input face with different pose views (see the fifth and sixth face images in the sixth column of Figure 10).

Figure 10. Comparison results on LFW database. (a) Input. (b) Method in \cite{12}. (c) Method in \cite{12}. (d) Method in \cite{16}. (e) Method in \cite{17}. (f) Proposed method.
Figure 11 presents more results for the proposed masked face synthesis, which were categorized into four different challenges: (1) Illumination variation; (2) Expression variation; (3) Pose variation; and (4) Occlusion. For constructing the testing set, 2115 faces of 400 + celebrities were collected from the image dataset downloaded in Kaggle [23]. As shown in Figure 11, the proposed method achieved acceptable results on real-world face images. For example, the proposed method worked accurately with severe illumination conditions including various outdoor lighting environments. Furthermore, the proposed method could yield stable synthesis results even with combined challenges (see the first face in the Expression variation challenge in Figure 11, which was also characterized by out-of-plane rotation). The results for 2115 synthetic face images can be found in the author’s github [24]. It should be noted that, for the proposed method, the two generators ($G_p$ and $G_r$) were learned using the training set derived from GTAV Face database and tested on LFW database. The results of the cross-database validation demonstrated the generalization capability of the proposed generative model.

As shown in Figure 3, SNOW application was able to generate realistic masked face images. However, SNOW was not suitable for processing a large number of unmasked face images from many different subjects because it was basically designed for processing a single selected photo stored in smart phone or a video frame captured by a smart phone camera. On the other hand, the proposed generative model was able to deal with a large number of face images due to its batch processing design. Furthermore, different from SNOW which provided users with limited number of mask patterns, the proposed generative model could be extended to other various mask patterns. For example, the synthetic face images in Figure 10e could be used as training data for learning the mask pattern generator $G_p$.

5. Conclusions

This paper proposed a two-stage generative model to synthesize masked face images. The former generator aimed to produce a pose-alike face with a mask pattern reflecting the pose view of an input face image. The latter one focused on localizing the mask pattern by
producing the segmentation map. Then, the mask pattern was fused with the input face, resulting in a realistic masked face.

Research on face recognition (FR) with masked faces has been becoming increasingly important due to the prolonged COVID-19 pandemic [25]. The proposed method aimed to generate a fake mask and overlay it on a real face. It was believed that the proposed method could be useful for constructing a large face recognition dataset that contained image pairs of unmasked face and synthesized masked face. Such synthetically masked face images were advantageous over real masked face images in a sense that the synthetic face images could be used along with their original (i.e., unmasked) face images. The face image pairs (masked/unmasked) could be used as training data for reducing the effect of wearing mask in MFR and used for investigating the effect of FR performance degradation caused by wearing masks.

The proposed method could be practically used for the following reasons:

1. The generative model was very compact and easy to implement. The two generator networks did not require complicated network settings (recall that the two generators shared the same network architecture which have 16 feature maps for convolution layer and 225 nodes for hidden layer).

2. As demonstrated in qualitative and quantitative experiments, the generative model could be robust against moderate out-of-plane rotation (up to \( \pm 60 \) degrees) and resolution variation present in face images (\( 80 \times 80, 40 \times 40 \), and \( 25 \times 25 \) pixels).

3. The generative model had very low computational cost for both training and inference. Because it took only 0.28 s to process 397 face images, the model could rapidly generate synthesized masked faces with real-time processing. Furthermore, due to the compact network architecture, the generative model could be easily extended to other mask patterns by simply retraining the generator networks.

Due to the efficiency of the proposed method, it could also be useful for research of video-based face recognition [26–28]. For future work, generative models to rapidly synthesize the masked faces for video sequence will be studied. In particular, a hardware format (such as Raspberry Pi with camera module) could be incorporated into the proposed method which allows collecting a large amount of real-world training image sequences of masked faces.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

**References**

1. Alzu’bi, A.; Albalas, F.; L-Hadhrami, T.A.; Younis, L.B.; Bashayreh, A. Masked face recognition using deep learning: A review. *Electronics* 2021, 10, 2666. [CrossRef]

2. Vu, H.N.; Nguyen, M.H.; Pham, C. Masked face recognition with convolutional neural networks and local binary patterns. *Appl. Intell.* 2022, 52, 5497–5512. [CrossRef] [PubMed]

3. Li, Y.; Guo, K.; Lu, Y.; Liu, L. Cropping and attention based approach for masked face recognition. *Appl. Intell.* 2021, 51, 3012–3025. [CrossRef] [PubMed]

4. Hariri, W. Efficient Masked Face Recognition Method during the COVID-19 Pandemic. 2020. Available online: https://www.researchsquare.com/article/rs-39289/v3 (accessed on 8 October 2022).

5. Din, N.U.; Javed, K.; Bae, S.; Yi, J. A novel GAN-based network for unmasking of masked face. *IEEE Access* 2020, 8, 44276–44287. [CrossRef]

6. Priya, G.N.; Banu, R.W. Occlusion invariant face recognition using a mean based weight matrix and support vector machine. *Sadhana* 2014, 39, 303–315. [CrossRef]

7. Krizhevsky, A.; Sutskever, I.; Hinton, G. ImageNet classification with deep convolutional neural networks. In Proceedings of the Neural Information and Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012.

8. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation applied to handwritten zip code recognition. *Neural Comput.* 1989, 1, 541–551. [CrossRef]

9. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* 2014, arXiv:1409.1556.

10. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
11. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2672–2680.

12. Wang, Z.; Wang, G.; Huang, B.; Xiong, Z.; Hong, Q.; Wu, H.; Yi, P.; Jiang, K.; Wang, N.; Pei, Y.; et al. Masked Face Recognition Dataset and Application. 2020. Available online: http://arxiv.org/abs/2003.09093 (accessed on 8 October 2022).

13. Huang, G.B.; Ramesh, M.; Berg, T.; Learned-Miller, E. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In Proceedings of the Workshop on Faces in ‘Real-Life’ Images: Detection, Alignment, and Recognition, Marseille, France, 17 October 2008.

14. Yi, D.; Lei, Z.; Liao, S.; Li, S.Z. Learning face representation from scratch. arXiv 2014, arXiv:1411.7923.

15. Boyko, N.; Basystiuk, O.; Shakhovska, N. Performance evaluation and comparison of software for face recognition, based on dlib and opencv library. In Proceedings of the IEEE Second Conference on Data Stream Mining and Processing, Lviv, Ukraine, 21 August 2018.

16. Ngan, M.; Grother, P.; Hanaoka, K. Ongoing Face Recognition Vendor Test (FRVT) Part 6A: Face Recognition Accuracy with Masks Using Pre- COVID-19 Algorithms; NLST: Gaithersburg, MD, USA, 2020. [CrossRef]

17. Anwar, A.; Raychowdhury, A. Masked face recognition for secure authentication. arXiv 2020, arXiv:2008.11104.

18. Masci, J.; Meier, U.; Cireajan, D.; Schmidhuber, J. Stacked convolutional auto-encoders for hierarchical feature extraction. In Proceedings of the International Conference on Artificial Neural Networks, Berlin, Germany, 14 June 2011; pp. 52–59.

19. Available online: https://en.wikipedia.org/wiki/Snow_(app) (accessed on 8 October 2022).

20. Tarrés, F.; Rama, A. GTAV Face Database. Available online: http://gps-tsc.upc.es/GTAV/ResearchAreas/UPCFaceDatabase/GTAVFaceDatabase.htm (accessed on 8 October 2022).

21. Wang, Z.; Bovik, A.C. Mean Squared Error: Love it or leave it? A new look at signal fidelity measures. IEEE Signal Process. Mag. 2009, 26, 98–117. [CrossRef]

22. Kingma, D.P.; Ba, J.L. Adam: A method for stochastic optimization. Proc. Int. Conf. Learn. Represent. 2015, 1–41. [CrossRef]

23. Available online: https://www.kaggle.com/datasets/akashguna/lfw-dataset-with-masks?resource=download (accessed on 8 October 2022).

24. Available online: https://github.com/leesh903/masked-face-dataset-LFW- (accessed on 8 October 2022).

25. Ullah, N.; Javed, A.; Ghazanfar, M.A.; Alsufyani, A.; Bourouis, S. A novel DeepMaskNet model for face mask detection and masked facial recognition. J. King Saud Univ.-Comput. Inf. Sci. 2022. [CrossRef]

26. Wang, H.; Wang, Y.; Cao, Y. Video-based face recognition: A survey. World Acad. Sci. Eng. Technol. 2009, 60, 293–302.

27. Stallkamp, J.; Ekenel, H.K.; Stiefelhagen, R. Video-based face recognition on real-world dataset. In Proceedings of the 2007 IEEE 11th International Conference on Computer Vision, Rio de Janeiro, Brazil, 14–21 October 2007; pp. 1–8.

28. Bashbaghi, S.; Granger, E.; Sabourin, R.; Parchami, M. Deep Learning Architectures for Face Recognition in Video Surveillance; Springer: Singapore, 2019; pp. 133–154.