to Appear in the IEEE Trans. PAMI, 2018

On the Reconstruction of Face Images from Deep Face Templates

Guangcan Mai, Kai Cao, Pong C. Yuen*, *Senior Member, IEEE,* and Anil K. Jain, *Life Fellow, IEEE*

Abstract—State-of-the-art face recognition systems are based on deep (convolutional) neural networks. Therefore, it is imperative to determine to what extent face templates derived from deep networks can be inverted to obtain the original face image. In this paper, we study the vulnerabilities of a state-of-the-art face recognition system based on template reconstruction attack. We propose a neighborly de-convolutional neural network (*NbNet*) to reconstruct face images from their deep templates. In our experiments, we assumed that no knowledge about the target subject and the deep network are available. To train the *NbNet* reconstruction models, we augmented two benchmark face datasets (VGG-Face and Multi-PIE) with a large collection of images synthesized using a face generator. The proposed reconstruction was evaluated using type-I (comparing the reconstructed images against the original face images used to generate the deep template) and type-II (comparing the reconstructed images against a different face image of the same subject) attacks. Given the images reconstructed from *NbNets*, we show that for verification, we achieve TAR of 95.20% (58.05%) on LFW under type-I (type-II) attacks @ FAR of 0.1%. Besides, 96.58% (92.84%) of the images reconstructed from templates of partition *fa* (*fb*) can be identified from partition *fa* in color FERET. Our study demonstrates the need to secure deep templates in face recognition systems.

Index Terms—Face recognition, template security, deep networks, deep templates, template reconstruction, neighborly de-convolutional neural network.

1 Introduction

Face recognition systems are being increasingly used for secure access in applications ranging from personal devices (e.g., iPhone X[1] and Samsung S8[2]) to access control (e.g., banking[3] and border control[4]). In critical applications, face recognition needs to meet stringent performance requirements, including low error rates and strong system security. In particular, the face recognition system must be resistant to spoofing (presentation) attacks and template invertivility. Therefore, it is critical to evaluate the vulnerabilities of a face recognition system to these attacks and devise necessary countermeasures. To this end, several attack mechanisms (such as hill climbing [1], [2], [3], spoofing [4], [5], [6], [7], [8], and template reconstruction (template invertibility) [9]) have been proposed to determine the vulnerabilities of face recognition systems.

In this paper, we focus on the vulnerability of a face recognition system to template invertibility or reconstruction attacks. In a template reconstruction attack (Fig. 1), we want to determine if face images can be successfully reconstructed from the face templates of target subjects and then used as input to the system to access privileges. Fig. 2 shows examples of face images reconstructed from their deep templates by the proposed method. Some of these reconstructions are successful in that they match well with the original images (Fig. 2 (a)), while others are not successful (Fig. 2 (b)). Template reconstruction attacks generally assume that templates of target subjects and the corresponding *black-box* template extractor can be accessed. These are reasonable assumptions because: (a) templates of target users can be exposed in hacked databases[5,6], and (b) the corresponding *black-box* template extractor can potentially be obtained by purchasing the face recognition SDK. To our knowledge, almost all of the face recognition vendors store templates without template protection, while some of them protect templates with specific hardware (e.g., Secure Enclave on A11 of iPhone X [10], TrustZone on ARM[7]). Note that unlike traditional passwords, biometric templates cannot be directly protected by standard ciphers such as AES and RSA since the matching of templates needs to allow small errors caused by intra-subject variations [17], [18]. Besides, state-of-the-art template protection schemes

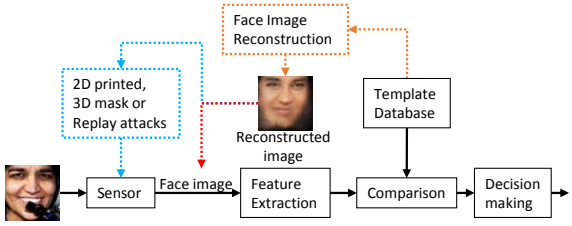


Fig. 1. Face recognition system vulnerability to template reconstruction attacks. Face image of a target subject is reconstructed from the corresponding template to gain system access by (a) creating a fake face (for example, a 2D printed image or 3D mask) (blue box) or (b) injecting a reconstructed face image directly into the feature extractor (red box).

- *Guangcan Mai and Pong C. Yuen are with the Department of Computer Science, Hong Kong Baptist University, Hong Kong SAR, CHINA. E-mail:{csgcmai,pcyuen}@comp.hkbu.edu.hk;*
- *Kai Cao and Anil K. Jain are with the Department of Computer Science and Engineering, Michigan State University, MI, 48824, USA. E-mail: {kaicao,jain}@cse.msu.edu*

* *Corresponding author*

1. https://www.apple.com/iphone-x/#face-id
2. http://www.samsung.com/uk/smartphones/galaxy-s8/
3. https://goo.gl/6TGcrr
4. https://goo.gl/ViVdDY
5. https://goo.gl/QUMHpv
6. https://goo.gl/KdxzqT
7. https://www.arm.com/products/security-on-arm/trustzone

TABLE 1
Comparison of major algorithms for face image reconstruction from their corresponding templates

Algorithm	Template features	Evaluation	Remarks
MDS [11]	PCA, BIC, COTS	Type-I attack[a]: TAR of 72% using *BIC*[b] and 73% using *COTS*[c] at an FAR of 1.0% on FERET	Linear model with limited capacity
RBF regression [9]	LQP [12]	Type-II attack[d]: 20% rank-1 identification error rate on FERET; EER = 29% on LFW;	RBF model may have limited generative capacity
CNN [13]	Final feature of FaceNet [14]	Reported results were mainly based on visualizations and no comparable statistical results was reported	**White-box** template extractor was assumed
Cole et. al., [15]	Intermediate feature of FaceNet [14][e]		High-quality images (e.g., front-facing, neutral-pose) are required for training.
This paper	Final feature of FaceNet [14]	Type-I attack: TAR[f] of 95.20% (LFW) and 73.76% (FRGC v2.0) at an FAR of 0.1%; rank-1 identification rate 95.57% on color FERET Type-II attack: TAR of 58.05% (LFW) and 38.39% (FRGC v2.0) at an FAR of 0.1%; rank-1 identification rate 92.84% on color FERET	Requires a large number of images for network training

[a] Type-I attack refers to matching the reconstructed image against the face image from which the template was extracted.
[b] *BIC* refers to Bayesian intra/inter-person classifier [16].
[c] *COTS* refers to commercial off-the-shelf system. A local-feature-based COTS was used in [11].
[d] Type-II attack refers to matching the reconstructed image against a face image of the same subject that was not used for template creation.
[e] Output of 1024-D 'avgpool' layer of the "NN2" architecture.
[f] TAR for LFW and FRGC v2.0 cannot be directly compared because their similarity thresholds differ.

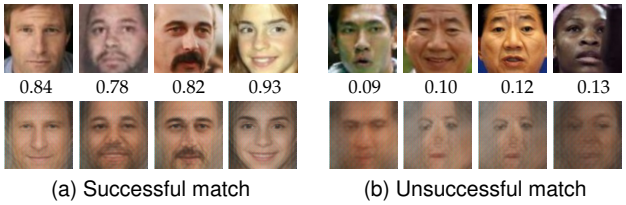


Fig. 2. Example face images reconstructed from their templates using the proposed method (VGG-NbB-P). The top row shows the original images (from LFW) and the bottom row shows the corresponding reconstructions. The numerical value shown between the two images is the cosine similarity between the original and its reconstructed face image. The similarity threshold is 0.51 (0.38) at FAR = 0.1% (1.0%).

are still far from practical because of the severe trade-off between matching accuracy and system security [19], [20].

Face templates are typically compact binary or real-valued feature representations[8] that are extracted from face images to increase the efficiency and accuracy of similarity computation. Over the past couple of decades, a large number of approaches have been proposed for face representations. These representations can be broadly categorized into (i) shallow [12], [21], [22], and (ii) deep (convolutional neural network or CNN) representations [14], [23], [24], according to the depth of their representational models[9]. Deep representations have shown their superior performances in face evaluation benchmarks (such as LFW [25], YouTube Faces [14], [26], and NIST IJB-A [24], [27]). Therefore, it is imperative to investigate the invertibility of deep templates to determine their vulnerability to template reconstruction attacks. However, to the best of our knowledge, no such work has been reported.

In our study of template reconstruction attacks, we made no assumptions about subjects used to train the target face recognition system. Therefore, only public domain face images were used to train our template reconstruction model.

8. As face templates refer to face representations stored in a face recognition system, these terms are used interchangeably in this paper.
9. Some researchers [23] refer to shallow representations as those that are not extracted using deep networks.

The available algorithms for face image reconstruction from templates [9], [11][10], [13], [15] are summarized in Table 1. The generalizability of the published template reconstruction attacks [9], [11] is not known, as all of the training and testing images used in their evaluations were subsets of the same face dataset. No statistical study in terms of template reconstruction attack has been reported in [13], [15].

To determine to what extent face templates derived from deep networks can be inverted to obtain the original face images, a reconstruction model with sufficient capacity is needed to invert the complex mapping used in the deep template extraction model [28]. De-convolutional neural network (D-CNN)[11] [29], [30], [31] is one of the straightforward deep models for reconstructing face images from deep templates. To design a D-CNN with sufficient model capacity[12], one could increase the number of output channels (filters) in each de-convolution layer [32]. However, this often introduces noisy and repeated channels since they are treated equally during the training.

To address the issues of noisy (repeated) channels and insufficient channel details, inspired by *DenseNet* [33] and *MemNet* [34], we propose a neighborly de-convolutional network framework (*NbNet*) and its building block, neighborly de-convolution blocks (NbBlocks). The NbBlock produces the same number of channels as a de-convolution layer by (a) reducing the number of channels in de-convolution layers to avoid the noisy and repeated channels; and (b) then creating the reduced channels by learning from their neighboring channels which were previously created in the same block to increase the details in reconstructed face images. To train the *NbNets*, a large number of face images are required. Instead of following the time-consuming and expensive process of collecting a sufficiently large face dataset [35], [36],

10. MDS method in the context of template reconstructible was initially proposed for reconstructing templates by matching scores between the target subject and attacking queries. However, it can also be used for template reconstruction attacks [11].
11. Some researchers refer to D-CNNs as CNNs. However, given that its purpose is the inverse of a CNN, we distinguish D-CNN and CNN.
12. The ability of a model to fit a wide variety of functions [28].

we trained a face image generator, DCGAN [37], to augment available public domain face datasets. To further enhance the quality of reconstructed images, we explore both pixel difference and perceptual loss [38] for training the *NbNets*. In summary, this paper makes following contributions:

- An investigation of the invertibility of face templates generated by deep networks. To the best of our knowledge, this is the first such study on security and privacy of face recognition systems.
- An *NbNet* with its building block, NbBlock, was developed for reconstructing face images from deep templates. The *NbNets* were trained by data augmentation and perceptual loss [38], resulting in maintaining discriminative information in deep templates.
- Empirical results show that the proposed face image reconstruction from the corresponding templates is successful. We show that we can achieve the following, verification rates (*security*), TAR of 95.20% (58.05%) on LFW under type-I (type-II) attack @ FAR of 0.1%. For identification (*privacy*), we achieve 96.58% and 92.84% rank one accuracy (partition *fa*) in color FERET [39] as gallery and the images reconstructed from partition *fa* (type-I attack) and *fb* (type-II attack) as probe.

2 Related Work

In this section, we describe the current practice of storing face templates, the limitations of current methods for reconstructing face images from deep templates and introduce GANs for generating (synthesizing) face images.

2.1 Face Template Security

Unlike traditional passwords that can be matched in their encrypted or hash form with standard ciphers (e.g., AES, RSA, and SHA-3), face templates cannot be simply protected by standard ciphers because of the intra-subject variations in face images [17], [18]. Due to the avalanche effect[13] [40] of standard ciphers, the face templates protected by standard ciphers need to be decrypted before matching. This introduces another challenge, (decryption) key management. In addition, decrypted face templates can also be gleaned by launching an authentication attempt.

Face template protection remains an open challenge. To our knowledge, either the vendors ignore the security and privacy issues of face templates, or secure the encrypted templates and the corresponding keys in specific hardware (e.g., Secure Enclave on A11 of iPhone X [10], TrustZone on ARM[14]). Note that the requirement of specific hardware limits the range of biometric applications.

2.2 Reconstructing Face Images from Deep Templates

Face template reconstruction requires the determination of the inverse of deep models used to extract deep templates from face images. Most deep models are complex and are typically implemented by designing and training a network with sufficiently large capacity [28].

Shallow model based [9], [11]: There are two shallow model based methods for reconstructing face images from templates proposed in the literature: multidimensional scaling (MDS) [11] and radial basis function (RBF) regression [9]. However, these methods have only been evaluated using shallow templates. The MDS-based method [11] uses a set of face images to generate a similarity score matrix using the target face recognition system and then finds an affine space in which face images can approximate the original similarity matrix. Once the affine space has been found, a set of similarities is obtained from the target face recognition system by matching the target template and the test face images. The affine representation of the target template is estimated using these similarities, which is then mapped back to the target face image.

Deep model based [13], [15]: Zhmoginov and Sandler [13] learn the reconstruction of face images from templates using a CNN by minimizing the template difference between original and reconstructed images. This requires the gradient information from target template extractor and cannot satisfy our assumption of *black-box* template extractor. Cole et. al. [15] first estimate the landmarks and textures of face images from the templates, and then combine the estimated landmarks and textures using the differentiable warping to yield the reconstructed images. High-quality face images (e.g., front-facing, neutral-pose) are required to be selected for generating landmarks and textures in [15] for training the reconstruction model. Note that both [13] and [15] does not aim to study vulnerability on deep templates and hence no comparable statistical results based template reconstruction attack were reported.

2.3 GAN for Face Image Generation

With adversarial training, GANs [37], [41], [42], [43], [44], [45], [46], [47], [48], [49] are able to generate photo-realistic (face) images from randomly sampled vectors. It has become one of the most popular methods for generating face images, compared to other methods such as data augmentation [50] and SREFI [51]. GANs typically consist of a generator which produces an image from a randomly sampled vector, and a discriminator which classifies an input image as real or synthesized. The basic idea for training a GAN is to prevent images output by the generator be mistakenly classified as real by co-training a discriminator.

DCGAN [37] is believed to be the first method that directly generates high-quality images (64×64) from randomly sampled vectors. PPGN [44] was proposed to conditionally generate high-resolution images with better image quality and sample diversity, but it is rather complicated. Wasserstein GAN [45], [46] was proposed to solve the model collapse problems in GAN [42]. Note that the images generated by Wasserstein GAN [45], [46] are comparable with those output by DCGAN. BEGAN [47] and LSGAN [48] have been proposed to attempt to address the model collapse, and non-convergence problems with GAN. A progressive strategy for training high-resolution GAN is described in [49].

In this work, we employed an efficient yet effective method, DCGAN to generate face images. The original DCGAN [37] is easy to collapse and outputs poor quality high-resolution images (e.g., 160×160 in this work). We address the above problems with DCGAN (Section 3.6.2).

13. https://en.wikipedia.org/wiki/Avalanche_effect
14. https://www.arm.com/products/security-on-arm/trustzone

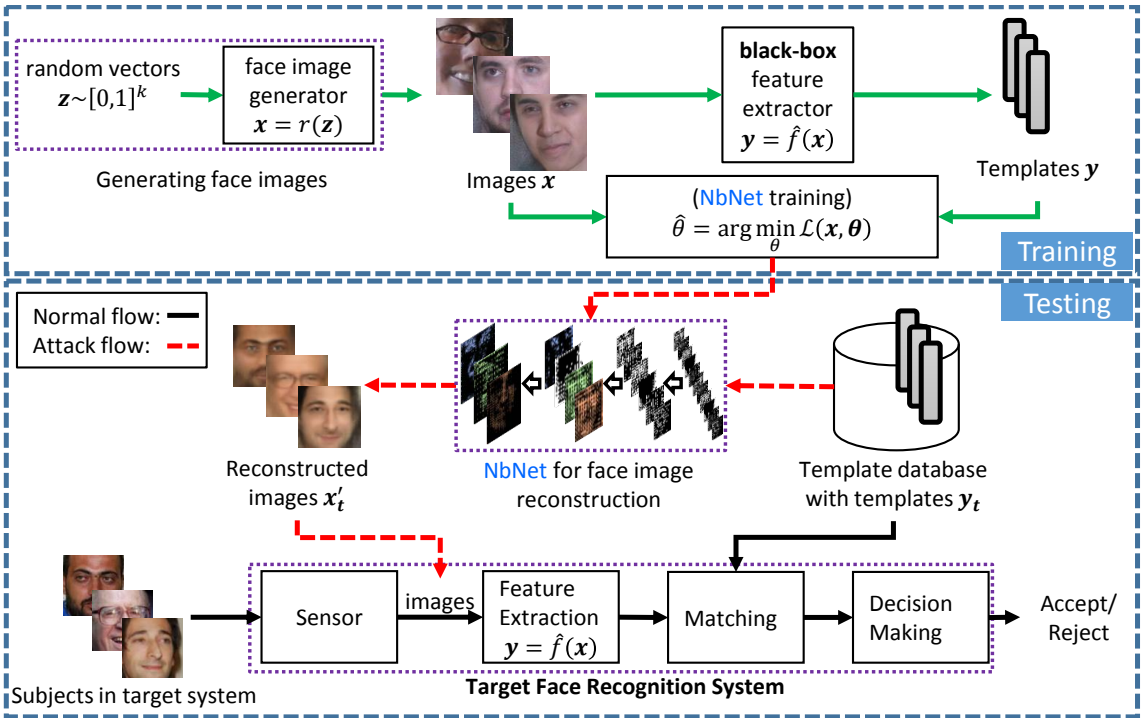


Fig. 3. An overview of the proposed system for reconstructing face images from the corresponding deep templates.

3 Proposed Template Security Study

An overview of our security study for deep template based face recognition systems under template reconstruction attack is shown in Fig. 3; the normal processing flow and template reconstruction attack flows are shown as black solid and red dotted lines, respectively. This section first describes the scenario of template reconstruction attack using an adversarial machine learning framework [52]. This is followed by the proposed *NbNet* for reconstructing face images from deep templates and the corresponding training strategy and implementation.

3.1 Template Reconstruction Attack

The adversarial machine learning framework [52], [54] categorizes biometric attack scenarios from four perspectives: an adversary's goal and his/her knowledge, capability, and attack strategy. Given a deep template based face recognition system, our template reconstruction attack scenario using the adversarial machine learning framework is as follows.

- *Adversary's goal:* The attacker aims to impersonate a subject in the target face recognition system, compromising the system integrity.
- *Adversary's knowledge:* The attacker is assumed to have the following information. (a) The templates $\boldsymbol{y}_t$ of the target subjects, which can be obtained via template database leakage or an insider attack. (b) The *black-box* feature extractor $\boldsymbol{y} = \hat{f}(\boldsymbol{x})$ of the target face recognition system. This can potentially be obtained by purchasing the target face recognition system's SDK. The attacker has neither information about target subjects nor their enrollment environments. Therefore, no face images enrolled in the target system can be utilized in the attack.
- *Adversary's capability:*(a) Ideally, the attacker should only be permitted to present fake faces (2D photographs or 3D face masks) to the face sensor during authentication. In this study, to simplify, the attacker is assumed to be able to inject face images directly into the feature extractor as if the images were captured by the face sensor. Note that the injected images could be used to create fake faces in actual attacks. (b) The identity decision for each query is available to the attacker. However, the similarity score of each query cannot be accessed. (c) Only a small number of trials (e.g., < 5) are permitted for the recognition of a target subject.
- *Attack strategy:* Under these assumptions, the attacker can infer a face image $\boldsymbol{x}_t$ from the target template $\boldsymbol{y}_t$ using a reconstruction model $\boldsymbol{x}_t = g_{\boldsymbol{\theta}}(\boldsymbol{y}_t)$ and insert reconstructed image as a query to access the target face recognition system. The parameter $\boldsymbol{\theta}$ of the reconstruction model $g_{\boldsymbol{\theta}}(\cdot)$ can be learned using public domain face images.

3.2 *NbNet* for Face Image Reconstruction

3.2.1 Overview

An overview of the proposed *NbNet* is shown in Fig. 4 (a). The *NbNet* is a cascade of multiple stacked de-convolution blocks and a convolution operator, ConvOp. De-convolution blocks up-sample and expand the abstracted signals in the input channels to produce output channels with a larger size as well as more details about reconstructed images. With multiple (D) stacked de-convolution blocks, the *NbNet* is able to expand highly abstracted deep templates back to channels with high resolutions and sufficient details for generating the output face images. The ConvOp in Fig. 4 (a) aims to summarize multiple output channels of D-th de-convolution block to the target number of channels (3 in this work for RGB images). It is a cascade of convolution, batch-normalization [53], and tanh activation layers.

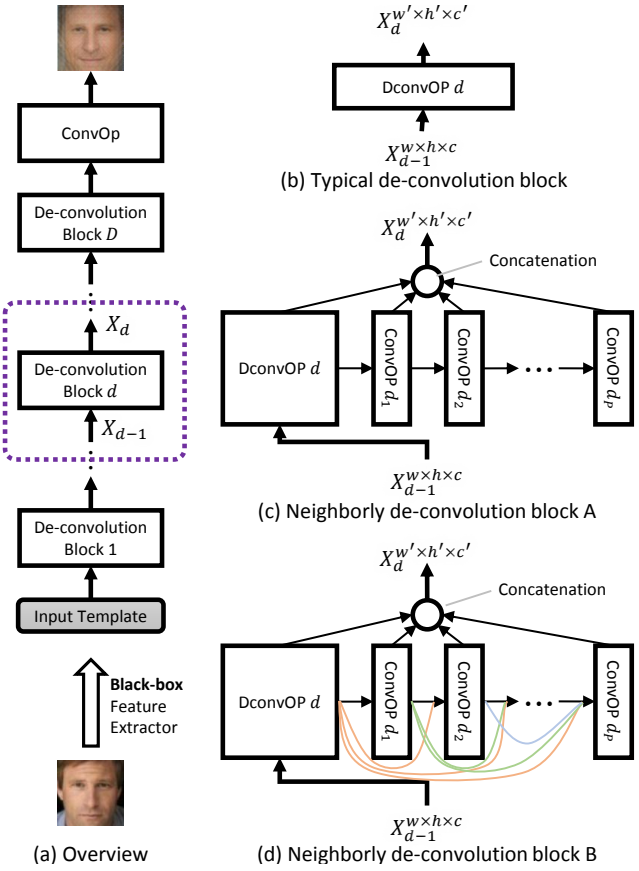


Fig. 4. The proposed *NbNet* for reconstructing face images from the corresponding face templates. (a) Overview of our face reconstruction network, (b) typical de-convolution block for building de-convolutional neural network (D-CNN), (c) and (d) are the neighborly de-convolution blocks (NbBlock) A/B for building NbNet-A and NbNet-B, respectively. Note that ConvOP (DconvOP) denotes a cascade of a convolution (de-convolution), a batch-normalization [53], and a ReLU activation (tanh in ConvOP of (a)) layers, where the width of ConvOp (DconvOP) denotes the number of channels in its convolution (de-convolution) layer. The black circles in (c) and (d) denote the channel concatenation of the output channels of DconvOP and ConvOPs.

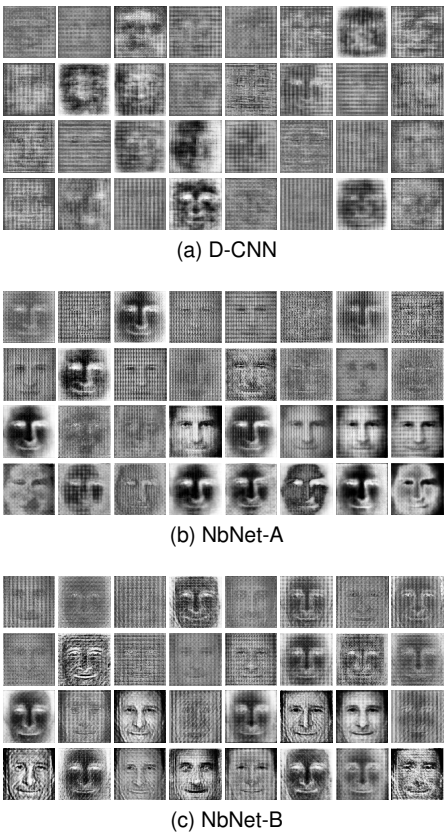


Fig. 5. Visualization of 32 output channels of the *5th* de-convolution blocks in (a) D-CNN, (b) NbNet-A, and (c) NbNet-B networks, where the input template was extracted from the bottom image of Fig. 4 (a). Note that the four rows of channels in (a) and the first two rows of channels in (b) and (c) are learned from channels from the corresponding *4th* block. The third row of channels in both (b) and (c) are learned from their first two rows of channels. The fourth row of channels in (b) is learned from the third row of channels only, where the fourth row of channels in (c) is learned from the first three rows of channels.

3.2.2 Neighborly De-convolution Block

A typical design of the de-convolution block [29], [37], as shown in Fig. 4 (b), is to learn output channels with up-sampling from channels in previous blocks only. The number of output channels c' is often made large enough to ensure sufficient model capacity for template reconstruction [32]. However, the up-sampled output channels tend to suffer from the following two issues: (a) noisy and repeated channels; and (b) insufficient details. An example of these two issues is shown in Fig. 5 (a), which is a visualization of output channels in the fifth de-convolution block of a D-CNN that is built with typical de-convolution blocks. The corresponding input template was extracted from the bottom image of Fig. 4 (a).

To address these limitations, we propose NbBlock which produces the same number of output channels as typical de-convolution blocks for the face template reconstruction. One of the reasons for noisy and repeated output channels is that a large number of channels are treated equally in a typical de-convolution block; from the perspective of network architecture, these output channels were learned from the same set of input channels and became the input of the same forthcoming blocks. To mitigate this issue, we first reduce the number of output channels that is simultaneously learned from the previous blocks. We then create the reduced number of output channels with enhanced details by learning from neighbor channels in the same block.

Let $G_d(\cdot)$ denote the d-th NbBlock, which is shown as the dashed line in Fig. 4 (a) and is the building component of our *NbNet*. Suppose that $G_d(\cdot)$ consists of one de-convolution operator (DconvOP) $\mathcal{N}'_d$ and P convolution operators (ConvOPs) $\{\mathcal{N}_{d_p} | p = 1, 2, \cdots, P\}$. Let X'_d and $X_{d,p}$ denote the output of DconvOP $\mathcal{N}'_d$ and p-th ConvOP $\mathcal{N}_{d_p}$ in d-th NbBlock $G_d(\cdot)$, then

$$X_d = G_d(X_{d-1}) = [\mathcal{X}_P] \tag{1}$$

where X_{d-1} denotes the output of the $(d-1)$-th NbBlock, $[\cdot]$ denotes a function of channel concatenation, and $\mathcal{X}_P$ is the set of outputs of DconvOP and all ConvOPs in $G_d(\cdot)$,

$$\mathcal{X}_P = \{X'_d, X_{d,1}, X_{d,2}, \cdots, X_{d,P}\} \tag{2}$$

where X'_d and $X_{d,p}$ denotes the output of DconvOP and the p-th ConvOP in d-th block, resp., and satisfy

$$X'_d = \mathcal{N}'_d\left(X_{d-1}\right), X_{d,p} = \mathcal{N}_{d_p}\left([\hat{\mathcal{X}}_p]\right) \tag{3}$$

where $\hat{\mathcal{X}}_p$ is a non-empty subset of $\mathcal{X}_p$.

Based on this idea, we built two NbBlocks, A and B, as shown in Figs. 4 (c) and (d), where the corresponding reconstructed networks are named NbNet-A and NbNet-B, respectively. In this study, the DconvOp (ConvOp) in Figs. 4 (b), (c), and (d) denotes a cascade of de-convolution (convolution), batch-normalization [53], and ReLU activation layers. The only difference between blocks A and B is the choice of $\hat{\mathcal{X}}_p$,

$$\hat{\mathcal{X}}_p = \begin{cases} \{X'_d\}, & \text{blocks A \& B with } p = 1; \\ \{X_{d,p-1}\}, & \text{block A with } p > 1; \\ \mathcal{X}_p, & \text{block B with } p > 1. \end{cases} \quad (4)$$

In our current design of the NbBlocks, half of output channels ($\frac{c'}{2}$ for block d) are produced by a DconvOP, and the remaining channels are produced by P ConvOPs, each of which gives, in this study, eight output channels (Table. 2). Example of blocks A and B with 32 output channels are shown in Figs. 5 (b) and (c). The first two rows of channels are produced by DconvOp and the third and fourth rows of channels are produced by the first and second ConvOps, respectively. Compared with Fig. 5 (a), the first two rows in Figs. 5 (b) and (c) have small amount of noise and fewer repeated channels, where the third and fourth row provide channels with more details about the target face image (the reconstructed image in Fig. 4 (a)). The design of our NbBlocks is motivated by *DenseNet* [33] and *MemNet* [34].

3.3 Reconstruction Loss

Let us denote $\mathcal{R}(\boldsymbol{x}, \boldsymbol{x}')$ as the reconstruction loss between an input face image $\boldsymbol{x}$ and its reconstruction $\boldsymbol{x}' = g_{\boldsymbol{\theta}}\left(\hat{f}(\boldsymbol{x})\right)$, where $g_{\boldsymbol{\theta}}(\cdot)$ denotes an *NbNet* with parameter $\boldsymbol{\theta}$ and $\hat{f}(\cdot)$ denotes a *black-box* deep template extractor.

Pixel Difference: A straightforward loss for learning reconstructed image $\boldsymbol{x}'$ is pixel-wise loss between $\boldsymbol{x}'$ and its original version $\boldsymbol{x}$. The Minkowski distance could then be used and mathematically expressed as

$$\mathcal{R}_{pixel}(\boldsymbol{x}, \boldsymbol{x}') = ||\boldsymbol{x} - \boldsymbol{x}'||_k = \left(\sum_{m=1}^{M} |x_m - x'_m|^k\right)^{\frac{1}{k}} \quad (5)$$

where M denotes number of pixels in $\boldsymbol{x}$ and k denotes the order of the metric.

Perceptual Loss [38]: Because of the high discriminability of deep templates, most of the intra-subject variations in a face image might have been eliminated in its corresponding deep template. The pixel difference based reconstruction leads to a difficult task of reconstructing these eliminated intra-subject variations, which, however, are not necessary for reconstruction. Besides, it does not consider holistic contents in an image as interpreted by machines and human visual perception. Therefore, instead of using pixel difference, we employ the perceptual loss [38] which guides the reconstructed images towards the same representation as the original images. Note that a good representation is robust to intra-subject variations in the input images. The representation used in this study is the feature map in the VGG-19 model [55][15]. We empirically determine that using the output of *ReLU3_2* activation layer as the feature map leads the best image reconstruction, in terms of face matching accuracy. Let $F(\cdot)$ denote feature mapping function of the *ReLU3_2* activation layer of VGG-19 [55], then the perceptual loss can be expressed as

$$\mathcal{R}_{percept}(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{2} ||F(\boldsymbol{x}) - F(\boldsymbol{x}')||_2^2 \quad (6)$$

15. Provided by https://github.com/dmlc/mxnet-model-gallery

3.4 Generating Face Images for Training

To successfully launch an template reconstruction attack on a face recognition system without knowledge of the target subject population, *NbNets* should be able to accurately reconstruct face images with input templates extracted from face images of different subjects. Let $p_{\boldsymbol{x}}(\boldsymbol{x})$ denote the probability density function (pdf) of image $\boldsymbol{x}$, the objective function for training a *NbNet* can be formulated as

$$\begin{aligned} \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) &= \arg\min_{\boldsymbol{\theta}} \int \mathcal{R}(\boldsymbol{x}, \boldsymbol{x}') \, p_{\boldsymbol{x}}(\boldsymbol{x}) d\boldsymbol{x} \\ &= \arg\min_{\boldsymbol{\theta}} \int \mathcal{R}\left(\boldsymbol{x}, g_{\boldsymbol{\theta}}(\hat{f}(\boldsymbol{x}))\right) p_{\boldsymbol{x}}(\boldsymbol{x}) d\boldsymbol{x}. \end{aligned} \quad (7)$$

Since there are no explicit methods for estimating $p_{\boldsymbol{x}}(\boldsymbol{x})$, we cannot sample face images from $p_{\boldsymbol{x}}(\boldsymbol{x})$. The common approach is to collect a large-scale face dataset and approximate the loss function $\mathcal{L}(\boldsymbol{\theta})$ in Eq. (7) as:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{N} \sum_{i}^{N} \mathcal{R}\left(\boldsymbol{x}_i, g_{\boldsymbol{\theta}}(\hat{f}(\boldsymbol{x}_i))\right) \quad (8)$$

where N denotes the number of face images and $\boldsymbol{x}_i$ denotes the i-th training image. This approximation is optimal if, and only if, N is sufficiently large. In practice, this is not feasible because of the huge time and cost associated with collecting a large database of face images.

To train a generalizable *NbNet* for reconstructing face images from their deep templates, a large number of face images are required. Ideally, these face images should come from a large number of different subjects because deep face templates of the same subject are very similar and can be regarded as either single exemplar or under large intra-user variations, a small set of exemplars in the training of *NbNet*. However, current large-scale face datasets (such as VGG-Face [23], CASIA-Webface [56], and Multi-PIE [57]) were primarily collected for training or evaluating face recognition algorithms. Hence, they either contain an insufficient number of images (for example, 494K images in CASIA-Webface) or an insufficient number of subjects (for instance, 2,622 subjects in VGG-Face and 337 subjects in Multi-PIE) for training a reconstruction *NbNet*.

Instead of collecting a large face image dataset for training, we propose to augment current publicly available datasets. A straightforward way to augment a face dataset is to estimate the distribution of face images $p_{\boldsymbol{x}}(\boldsymbol{x})$ and then sample the estimated distribution. However, as face images generally consist of a very large number of pixels, there is no efficient method to model the joint distribution of these pixels. Therefore, we introduced a generator $\boldsymbol{x} = r(\boldsymbol{z})$ capable of generating a face image $\boldsymbol{x}$ from a vector $\boldsymbol{z}$ with a given distribution. Assuming that $r(\boldsymbol{z})$ is one-to-one and smooth, the face images can be sampled by sampling $\boldsymbol{z}$. The

TABLE 2
Network details for D-CNN and *NbNets*. "$[k_1 \times k_2, c]$ DconvOP (ConvOP), stride s", denotes cascade of a de-convolution (convolution) layer with c channels, kernel size (k_1, k_2) and stride s, batch normalization, and ReLU (tanh for the bottom ConvOP) activation layer.

Layer name	Output size ($c \times w \times h$)	D-CNN	NbNet-A, NbNet-B
input layer	$128 \times 1 \times 1$		
De-convolution Block (1)	$512 \times 5 \times 5$	$[5 \times 5, 512]$ DconvOP, stride 2	$[5 \times 5, 256]$ DconvOP, stride 2 $\{[3 \times 3, 8]$ ConvOP, stride $1\} \times 32$
De-convolution Block (2)	$256 \times 10 \times 10$	$[3 \times 3, 256]$ DconvOP, stride 2	$[3 \times 3, 128]$ DconvOP, stride 2 $\{[3 \times 3, 8]$ ConvOP, stride $1\} \times 16$
De-convolution Block (3)	$128 \times 20 \times 20$	$[3 \times 3, 128]$ DconvOP, stride 2	$[3 \times 3, 64]$ DconvOP, stride 2 $\{[3 \times 3, 8]$ ConvOP, stride $1\} \times 8$
De-convolution Block (4)	$64 \times 40 \times 40$	$[3 \times 3, 64]$ DconvOP, stride 2	$[3 \times 3, 32]$ DconvOP, stride 2 $\{[3 \times 3, 8]$ ConvOP, stride $1\} \times 4$
De-convolution Block (5)	$32 \times 80 \times 80$	$[3 \times 3, 32]$ DconvOP, stride 2	$[3 \times 3, 16]$ DconvOP, stride 2 $\{[3 \times 3, 8]$ ConvOP, stride $1\} \times 2$
De-convolution Block (6)	$16 \times 160 \times 160$	$[3 \times 3, 16]$ DconvOP, stride 2	$[3 \times 3, 8]$ DconvOP, stride 2 $\{[3 \times 3, 8]$ ConvOP, stride $1\} \times 1$
ConvOP	$3 \times 160 \times 160$	$[3 \times 3, 3]$ ConvOP, stride 1	
Loss layer	$3 \times 160 \times 160$	Pixel difference or perceptual loss [38]	

loss function $\mathcal{L}(\boldsymbol{\theta})$ in Eq. (7) can then be approximated as follows:

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) &= \int \mathcal{R}\left(\boldsymbol{x}, g_{\boldsymbol{\theta}}(\hat{f}(\boldsymbol{x}))\right) p_{\boldsymbol{x}}(\boldsymbol{x}) d\boldsymbol{x} \\ &= \int \mathcal{R}\left(r(\boldsymbol{z}), g_{\boldsymbol{\theta}}\left(\hat{f}(r(\boldsymbol{z}))\right)\right) p_{\boldsymbol{z}}(\boldsymbol{z}) d\boldsymbol{z}.\end{aligned} \tag{9}$$

where $p_{\boldsymbol{z}}(\boldsymbol{z})$ denotes the pdf of variable $\boldsymbol{z}$. Using the *change of variables* method [58], [59], it is easy to show that $p_{\boldsymbol{z}}(\boldsymbol{z})$ and $r(\boldsymbol{z})$ have the following connection,

$$p_{\boldsymbol{z}}(\boldsymbol{z}) = p_{\boldsymbol{x}}(r(\boldsymbol{z})) \left|\det\left(\frac{d\boldsymbol{x}}{d\boldsymbol{z}}\right)\right|, \text{where} \left(\frac{d\boldsymbol{x}}{d\boldsymbol{z}}\right)_{ij} = \frac{\partial x_i}{\partial z_j}. \tag{10}$$

Suppose a face image $\boldsymbol{x} \in \mathbb{R}^{h \times w \times c}$ of height h, width w, and with c channels can be represented by a real vector $\boldsymbol{b} = \{b_1, \cdots, b_k\} \in \mathbb{R}^k$ in a manifold space with $h \times w \times c \gg k$. It can then be shown that there exists a generator function $\boldsymbol{b'} = \hat{r}(\boldsymbol{z})$ that generates $\boldsymbol{b'}$ with a distribution identical to that of $\boldsymbol{b}$, where $\boldsymbol{b}$ can be arbitrarily distributed and $\boldsymbol{z} \in [0, 1]^k$ is uniformly distributed (see Appendix).

To train the *NbNets* in the present study, we used the generative model of a DCGAN [37] as our face generator $r(\cdot)$. This model can generate face images from vectors $\boldsymbol{z}$ that follow a uniform distribution. Specifically, DCGAN generates face images $r(\boldsymbol{z})$ with a distribution that is an approximation to that of real face images $\boldsymbol{x}$. It can be shown empirically that a DCGAN can generate face images of unseen subjects with different intra-subject variations. By using adversarial learning, the DCGAN is able to generate face images that are classified as real face images by a co-trained real/fake face image discriminator. Besides, the intra-subject variations generated using a DCGAN can be controlled by performing arithmetic operations in the random input space [37].

3.5 Differences with *DenseNet*

One of the related work to *NbNet* is *DenseNet* [33], from which the *NbNet* is inspired. Generally, *DenseNet* is based on convolution layers and designed for object recognition, while the proposed *NbNet* is based on de-convolution layers and aimed to reconstruct face images from deep templates. Besides, *NbNet* is a framework whose NbBlocks produce output channels learned from previous blocks and neighbor channels within the block. The output channels of NbBlocks consist of fewer repeated and noisy channels and contain more details for face image reconstruction than the typical de-convolution blocks. Under the framework of *NbNet*, one could build a skip-connection-like network [60], NbNet-A, and a *DenseNet* -like network, NbNet-B. Note that NbNet-A sometimes achieves a comparable performance to NbNet-B with roughly 67% of the parameters and 54% running time only (see model VGG-NbA-P and VGG-NbB-P in Section 4). We leave more efficient and accurate *NbNets* construction as a future work.

3.6 Implementation Details

3.6.1 Network Architecture

The detailed architecture of the D-CNN and the proposed *NbNets* is shown in Table. 2. The NbNet-A and NbNet-B show the same structure in Table. 2. However, the input of the ConvOP in the de-convolution blocks (1)-(6) are different (Fig. 4), where NbNet-A uses the nearest previous channels in the same block, and NbNet-B uses all the previous channels in the same block.

3.6.2 Revisiting DCGAN

To train our *NbNet* to reconstruct face images from deep templates, we first train a DCGAN to generate face images. These generated images are then used for training. The face images generated by the original DCGAN could be noisy and sometimes difficult to interpret. Besides, the training as described in [37] is often collapsed in generating high-resolution images. To address these issues, we revisit the DCGAN as below (as partially suggested in [42]):

- *Network architecture*: replace the batch normalization and ReLU activation layer in both generator and discriminator by the SeLU activation layer [61], which performs the normalization of each training sample.

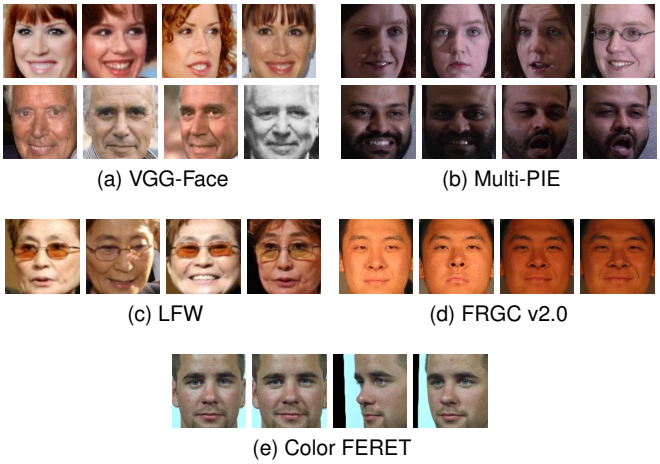


Fig. 6. Example face images from the training and testing datasets: (a) VGG-Face (1.94M images) [23], (b) Multi-PIE (151K images, only three camera views were used, including '14_0', '05_0' and '05_1', respectively) [57], (c) LFW (13,233 images) [25], [62], (d) FRGC v2.0 (16,028 images in the target set of Experiment 1) [63], and (e) Color FERET (2,950 images) [39].

- *Training labels*: replace the hard labels ('1' for real, and '0' for generated images) by soft labels in the range [0.7, 1.2] for real, and in range [0, 0.3] for generated images. This helps smooth the discriminator and avoids model collapse.
- *Learning rate*: in the training of DCGAN, at each iteration, the generator is updated with one batch of samples, while the discriminator is updated with two batches of samples (1 batch of 'real' and 1 batch of 'generated'). This often makes the discriminator always correctly classify the images output by the generator. To balance, we adjust the learning rate of the generator to 2×10^{-4}, which is greater than the learning rate of the discriminator, 5×10^{-5}.

Example generated images were shown in Fig. 7.

3.6.3 Training Details

With the pre-trained DCGAN, face images were first generated by randomly sampling vectors $\boldsymbol{z}$ from a uniform distribution and the corresponding face templates were extracted. The *NbNet* was then updated with the generated face images as well as the corresponding templates using batch gradient descent optimization. This training strategy was used to minimize the loss function $\mathcal{L}(\boldsymbol{\theta})$ in Eq. (9), which is an approximation of the loss function in Eq. (7).

The face template extractor we used is based on FaceNet [14], one of the most accurate CNN models for face recognition currently available. To ensure that the face reconstruction scenario is realistic, we used an open-source implementation[16] based on TensorFlow[17] without any modifications (model *20170512-110547*).

We implemented the *NbNets* using MXNet[18]. The networks were trained using a mini-batch based algorithm, Adam [64] with batch size of 64, $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

16. https://github.com/davidsandberg/facenet
17. Version 1.4.0 from https://www.tensorflow.org
18. Version 0.1.0 from http://mxnet.io

Fig. 7. Sample face images generated from face generators trained on (a) VGG-Face, and (b) Multi-PIE.

The learning rate was initialized to 2×10^{-4} and decayed by a factor of 0.94 every 5K batches. The pixel values in the output images were normalized to $[-1, 1]$ by first dividing 127.5 and then subtracting 1.0. For the networks trained with the pixel difference loss, we trained the network with 300K batches, where the weights are randomly initialized using a normal distribution with zero mean and a standard deviation of 0.02. For the networks trained with the perceptual loss [38], we trained the networks with extra 100K batches by refining from the corresponding networks trained with the pixel difference loss. The hardware specifications of the workstations for the training are the CPUs of dual Intel(R) Xeon E5-2630v4 @ 2.2 GHz, the RAM of 256GB with two sets of NVIDIA Tesla K80 Dual GPU. The software includes CentOS 7 and Anaconda2[19].

4 Experimental Results

4.1 Database and Experimental Setting

The vulnerabilities of deep templates under template reconstruction attacks were studied with our proposed reconstruction model, using two popular large-scale face datasets for training and three benchmark datasets for testing. The training datasets consisted of one unconstrained datasets, VGG-Face [23] and one constrained dataset, Multi-PIE [57].

- **VGG-Face** [23] comprises of 2.6 million face images from 2,622 subjects. In total, 1,942,242 trainable images were downloaded with the provided links.
- **Multi-PIE** [57]. We used 150,760 frontal images (3 camera views, with labels '14_0', '05_0', and '05_1', respectively), from 337 subjects.

Three testing datasets were used, including two for verification (LFW [25] and FRGC v2.0 [63]) and one for identification (color FERET [39]) scenarios.

- **LFW** [25] consists of 13,233 images of 5,749 subjects downloaded from the Web.
- **FRGC v2.0** [63] consists of 50,000 frontal images of 4,003 subjects with two different facial expressions (smiling and neutral), taken under different illumination conditions. A total of 16,028 images of 466 subjects (as specified in the target set of Experiment 1 of FRGC v2.0 [63]) were used.
- **Color FERET** [39] consists of four partitions (i.e., *fa*, *fb*, *dup1* and *dup2*), including 2,950 images. Compared to the gallery set *fa*, the probe sets (*fb*, *dup1* and *dup2*) contain face images of difference facial expression and aging.

TABLE 3
Deep template face template reconstruction models for comparison

Model[a]	Training Dataset	Training Loss	Testing Dataset	Training and Testing Process
VGG-Dn-P VGG-NbA-P VGG-NbB-P	VGG-Face	Perceptual Loss	LFW FRGC v2.0 color FERET	Train a DCGAN using the training dataset, and then use face images generated from the pretrained DCGAN for training the target D-CNN. Test the trained D-CNN using testing datasets.
VGG-Dn-M VGG-NbA-M VGG-NbB-M		Pixel Difference (MAE[b])		
VGGr-NbB-M				Directly train the target D-CNN using face images from the training dataset, and then test the trained D-CNN using testing datasets.
MPIE-Dn-P MPIE-NbA-P MPIE-NbB-P	Multi-PIE	Perceptual Loss		Train a DCGAN using the training dataset, and then use face images generated from the pretrained DCGAN for training the target D-CNN. Test the trained D-CNN using testing datasets.
MPIE-Dn-M MPIE-NbA-M MPIE-NbB-M		Pixel Difference (MAE)		
MPIEr-NbB-M				Directly train the target D-CNN using face images from the training dataset, and then test the trained D-CNN using testing datasets.
Mixedr-NbB-M	VGG-Face CASIA-Webface Multi-PIE			
RBF [9]	LFW	N/A	LFW	Train and test the RBF regression based method using the training and testing images specified in the evaluation protocol.
	FRGC v2.0	N/A	FRGC v2.0	

[a] Dn, NbA, and NbB denote D-CNN, NbNet-A, and NbNet-B, respectively
[b] MAE denotes "mean absolute error"

The face images were aligned using the five points detected by MTCNN[20] [65] and then cropped to 160×160 pixels. Instead of aligning images from the LFW dataset, we used the pre-aligned deep funneled version [62]. Fig. 6 shows example images from these five datasets.

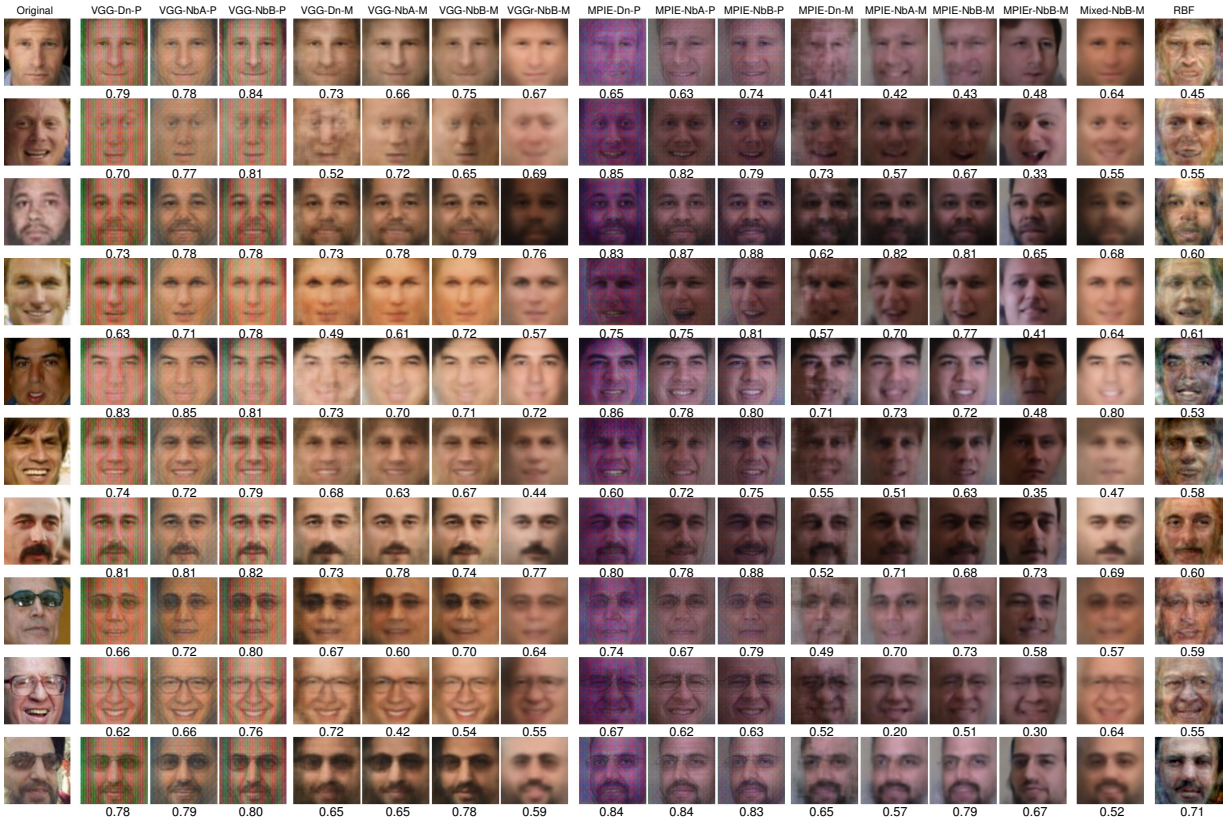
To determine the effectiveness of the proposed *NbNet*, we compare three different network architectures, i.e., D-CNN, NbNet-A, and NbNet-B, which are built using the typical de-convolution blocks, NbBlocks A and B. All of these networks are trained using the proposed generator-based training strategy using a DCGAN [37] with both pixel difference[21] and perceptual loss[22] [38]. To demonstrate the effectiveness of the proposed training strategy, we train the NbNet-B directly using images from VGG-Face, Multi-PIE, and a mixture of three datasets (VGG-Face, CASIA-Webface[23] [56], and Multi-PIE). Note that both VGG-Face and Multi-PIE are augmented to 19.2M images in our training. Examples of images generated using our trained face image generator are shown in Fig. 7. In addition, the proposed *NbNet* based reconstruction method was compared with a state-of-the-art RBF-regression-based method [9]. In contrast to the neural network based method, the RBF[24] regression model of [9] used the same dataset for training and testing (either LFW or FRGC v2.0). Therefore, the RBF-regression-based reconstruction method was expected to have better reconstruction accuracy than the proposed method. The MDS-based method [11] was not compared here because it is a linear model and was not as good as the RBF-regression-based method [9]. We did not compare [13], [15] because [13] does not satisfy our assumption of *black-box* template extractor and [15] requires to selecting high quality images for training. Table 3 summarizes the 16 comparison models used in this study for deep template inversion.

Examples of the reconstructed images of the first ten subjects in LFW and FRGC v2.0 are shown in Fig. 8. The leftmost column shows the original images, and the remaining columns show the images reconstructed using the 16 reconstruction models. For the RBF model, every image in the testing datasets (LFW and FRGC v2.0) has 10 different reconstructed images that can be created using the 10 cross-validation trials in the BLUFR protocol[25] [66]. The RBF-reconstructed images shown in this paper are those with the highest similarity scores among these 10 different reconstructions. The number below each image is the similarity score between the original and reconstructed images. The similarity scores were calculated using the cosine similarity in the range of $[-1, 1]$.

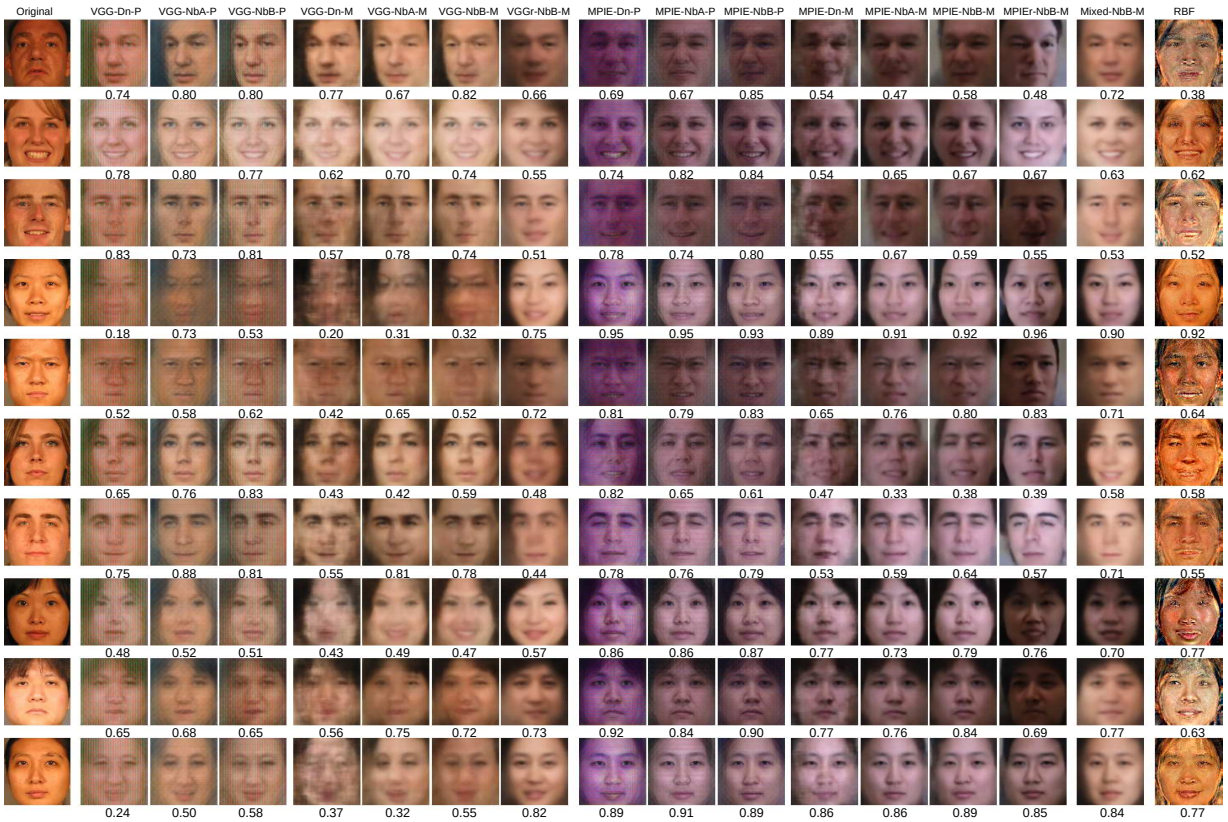
4.2 Verification Under Template Reconstruction Attack

We quantitatively evaluated the template security of the target face recognition system (FaceNet [14]) under type-I and type-II template reconstruction attacks. The evaluation metric was face verification using the BLUFR protocol [66]. The impostor scores obtained from the original face images were used in both of the attacks to demonstrate the efficacy of the reconstructed face images. The genuine scores in the type-I attack were obtained by comparing the reconstructed images against the original images. The genuine scores in the type-II attack were obtained by substituting one of the

19. https://www.anaconda.com
20. https://github.com/pangyupo/mxnet_mtcnn_face_detection.git
21. We simply choose mean absolute error (MAE), where order $k = 1$.
22. To reduce the training time, we first train the network with pixel difference loss and then fine-tune it using perceptual loss [38].
23. It consists of 494,414 face images from 10,575 subjects. We obtain 455,594 trainable images after preprocessing.
24. It was not compared in the identification task on color FERET.
25. http://www.cbsr.ia.ac.cn/users/scliao/projects/blufr/



(a) LFW



(b) FRGC v2.0

Fig. 8. Reconstructed face images of the first 10 subjects from (a) LFW and (b) FRGC v2.0. The original face images are shown in the first column. Each column denotes the reconstructed face images from different models used for reconstruction. The number below each reconstructed image shows the similarity score between the reconstructed image and the original image. The scores (ranging from -1 to 1) were calculated using the cosine similarity. The mean verification thresholds on LFW and FRGC v2.0 were 0.51 and 0.80, respectively, at FAR=0.1%, and 0.38 and 0.64, respectively, at FAR=1.0%.

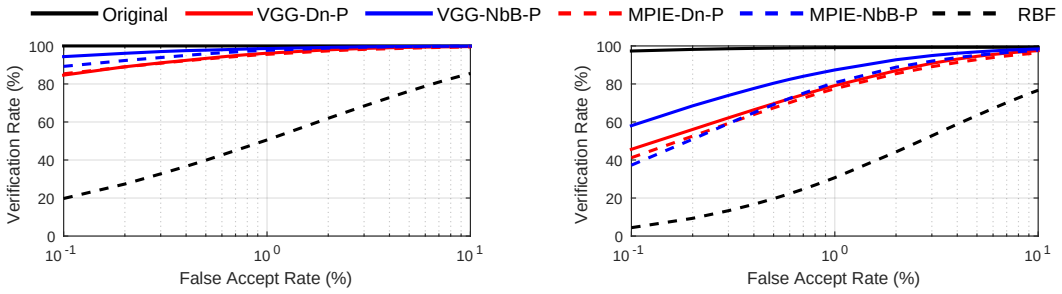


Fig. 9. ROC curves of (a) type-I and (b) type-II attacks using different reconstruction models on LFW. For the ease of reading, we only show the curves for D-CNN, NbNet-B trained with perceptual loss, and the RBF based method. Refer to Table 4 for the numerical comparison of all models. Note that the curves for VGG-Dn-P and MPIE-Dn-P are overlapping in (a).

TABLE 4
TARs (%) of type-I and type-II attacks on LFW for different template reconstruction methods, where "Original" denotes results based on the original images and other methods are described in Table 3. (**best**, second best)

Attack	Type-I		Type-II	
FAR	0.1%	1.0%	0.1%	1.0%
Original	100.00	100.00	97.33	99.11
VGG-Dn-P	84.65	96.18	45.63	79.13
VGG-NbA-P	**95.20**	**99.14**	53.91	87.06
VGG-NbB-P	94.37	98.63	**58.05**	**87.37**
VGG-Dn-M	70.22	88.35	26.22	64.88
VGG-NbA-M	79.52	94.94	30.97	68.14
VGG-NbB-M	89.52	97.75	37.09	79.19
VGGr-NbB-M	72.53	93.21	27.38	70.72
MPIE-Dn-P	85.34	95.57	41.21	77.51
MPIE-NbA-P	80.33	95.46	21.75	63.05
MPIE-NbB-P	89.25	97.69	37.30	80.67
MPIE-Dn-M	37.11	63.01	3.23	13.26
MPIE-NbA-M	50.54	78.91	6.11	33.26
MPIE-NbB-M	67.86	88.56	24.00	57.98
MPIEr-NbB-M	34.87	65.56	3.67	21.24
Mixedr-NbB-M	71.62	92.98	19.29	65.63
RBF [9]	19.76	50.55	4.41	30.70

original images in a genuine comparison (image pair) with the corresponding reconstructed image. For benchmarking, we report the "Original" results based on original face images. Every genuine score of "Original" in type-I attack was obtained by comparing two identical original images and thus the corresponding TAR stays at 100%. The genuine scores of "Original" in type-II attack were obtained by the genuine comparisons specified in BLUFR protocol. The BLUFR protocol uses tenfold cross-validation; the performance reported here is the 'lowest', namely $(\mu - \sigma)$, where μ and σ denote the mean and standard deviation of the attacking accuracy obtained from the 10 trials, respectively.

4.2.1 Performance on LFW

In each trial of the BLUFR protocol [66] for LFW [25], there is an average of 46,960,863 impostor comparisons. The average number of testing images is 9,708. Hence, there are 9,708 genuine comparisons in a type-I attack on LFW. The average number of genuine comparisons in a type-II attack on LFW is 156,915; this is the average number of genuine comparisons specified in the BLUFR protocol.

The receiver operator characteristic (ROC) curves of type-I and type-II attacks on LFW are shown in Fig. 9. Table 4 shows the TAR values at FAR=0.1% and FAR=1.0%, respectively. The ROC curve of "Original" in the type-II attack (Fig. 9b) is the system performance with BLUFR protocol [66] based on original images.

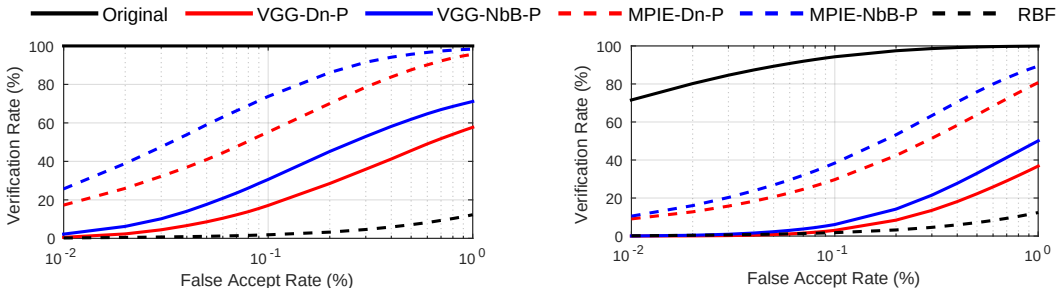
For both type-I and type-II attacks, the proposed *NbNets* generally outperform the D-CNN, where MPIE-NbA-P is not as effective as MPIE-Dn-P. Moreover, the models trained using the proposed strategy (VGG-NbB-M and MPIE-NbB-M) outperform the corresponding models trained with the non-augmented datasets (VGGr-NbB-M and MPIEr-NbB-M). The models trained using the raw images in VGG (VGGr-NbB-M) outperform the corresponding model trained using the mixed dataset. All *NbNets* trained with the proposed training strategy outperform the RBF regression based method [9]. In the type-I attack, the VGG-NbA-P model achieved a TAR of 95.20% (99.14%) at FAR=0.1% (FAR=1.0%). This implies that an attacker has approximately 95% (or 99% at FAR=1.0%) chance of accessing the system using a leaked template.

4.2.2 Performance on FRGC v2.0

Each trial of the BLUFR protocol [66] for FRGC v2.0 [63] consisted of an average of 76,368,176 impostor comparisons and an average of 12,384 and 307,360 genuine comparisons for type-I and type-II attacks, respectively.

The ROC curves of type-I and type-II attacks on FRGC v2.0 are shown in Fig. 10. The TAR values at FAR=0.1% and FAR=1.0% are shown in Table 5. The TAR values (Tables 4 and 5) and ROC plots (Figs. 9 and 10) for LFW and FRGC v2.0 cannot be directly compared, as the thresholds for LFW and FRGC v2.0 differ (e.g., the thresholds at FAR=0.1% are 0.51 and 0.80 for LFW and FRGC v2.0, respectively). The similarity threshold values were calculated based on the impostor distributions of the LFW and FRGC v2.0 databases.

It was observed that the proposed *NbNets* generally outperform D-CNN. The only exception is that the MPIE-NbA-P was not as good as MPIE-Dn-P. Significant improvements by using the augmented datasets (VGG-NbB-M and MPIE-NbB-M) were observed, compared with VGGr-NbB-M and MPIEr-NbB-M, for both the type-I and type-II attacks. All *NbNets* outperform the RBF regression based method [9].



(a) Type-I attack (b) Type-II attack

Fig. 10. ROC curves of (a) type-I and (b) type-II attacks using different reconstruction models on FRGC v2.0. For readability, we only show the curves for D-CNN, NbNet-B trained with perceptual loss, and the RBF based method. Refer to Table 5 for the numerical comparison of all models.

TABLE 5
TARs (%) of type-I and type-II attacks on FRGC v2.0 for different template reconstruction methods, where "Original" denotes results based on the original images and other methods are described in Table 3. (**best**, second best)

Attack	Type-I		Type-II	
FAR	0.1%	1.0%	0.1%	1.0%
Original	100.00	100.00	94.30	99.90
VGG-Dn-P	17.10	57.71	3.00	36.81
VGG-NbA-P	32.66	71.54	8.65	51.87
VGG-NbB-P	30.62	71.14	6.06	50.09
VGG-Dn-M	3.52	35.94	0.68	20.40
VGG-NbA-M	8.95	55.84	2.39	33.40
VGG-NbB-M	16.44	67.57	3.60	44.19
VGGr-NbB-M	6.75	55.51	4.05	36.18
MPIE-Dn-P	55.22	95.65	29.70	80.72
MPIE-NbA-P	49.75	94.41	28.46	78.71
MPIE-NbB-P	**73.76**	**98.35**	**38.39**	**89.41**
MPIE-Dn-M	12.82	47.84	10.47	38.39
MPIE-NbA-M	15.58	61.44	13.42	48.46
MPIE-NbB-M	28.48	80.67	19.85	63.04
MPIEr-NbB-M	12.72	49.53	11.75	40.59
Mixedr-NbB-M	9.65	63.82	8.15	45.10
RBF [9]	1.86	12.29	1.78	12.37

TABLE 6
Rank-one recognition rate (%) on color FERET [39] with partition *fa* as gallery and reconstructed images from different partition as probe. The partitions (i.e., *fa*, *fb*, *dup1* and *dup2*) are described in color FERET protocol [39]. Various methods are described in Table 3. (**best** and second best) of rank-one identification rate in each column.

Attack	Type-I	Type-II		
Probe	fa	fb	dup1	dup2
VGG-Dn-P	89.03	86.59	76.77	78.51
VGG-NbA-P	94.87	90.93	80.30	81.58
VGG-NbB-P	95.57	**92.84**	84.78	84.65
VGG-Dn-M	80.68	74.40	62.91	65.35
VGG-NbA-M	86.62	80.44	64.95	66.67
VGG-NbB-M	92.15	87.00	75	75.44
VGGr-NbB-M	81.09	74.29	61.28	62.28
MPIE-Dn-P	96.07	91.73	84.38	85.53
MPIE-NbA-P	93.86	90.22	79.89	79.82
MPIE-NbB-P	**96.58**	**92.84**	**86.01**	**87.72**
MPIE-Dn-M	73.54	64.11	53.26	49.12
MPIE-NbA-M	72.23	64.01	51.09	44.74
MPIE-NbB-M	85.61	78.22	71.06	68.42
MPIEr-NbB-M	63.88	54.54	44.57	35.96
Mixedr-NbB-M	82.19	76.11	62.09	58.77
Original	100.00	98.89	97.96	99.12

In the type-I attack, the best model, MPIE-NbB-P achieved a TAR of 73.76% (98.35%) at FAR=0.1% (FAR=1.0%). This implies that an attacker has a 74% (98%) chance of accessing the system at FAR=0.1% (1.0%) using a leaked template.

4.3 Identification with Reconstructed Images

We quantitatively evaluate the privacy issue of a leaked template extracted by target face recognition system (FaceNet [14]) under type-I and type-II attacks. The evaluation metric was the standard color FERET protocol [39]. The partition *fa* (994 images) was used as the gallery set. For the type-I attack, the images reconstructed from the partition *fa* was used as the probe set. For the type-II attack, the probe sets (*fb* with 992 images, *dup1* with 736 images, and *dup2* with 228 images) specified in the color FERET protocol were replaced by the corresponding reconstructed images.

The rank-one identification rate of both type-I and type-II attacks on color FERET are shown in Table 6. The row values under "Original" show the identification rate based on the original images. It stays at 100% for the type-I attack because the corresponding similarity score are obtained by comparing two identical images. It was observed that the proposed *NbNets* outperform D-CNN with the exception that the MPIE-Dn-P and MPIE-Dn-M slightly outperform MPIE-NbA-P and MPIE-NbA-M, respectively. Besides, significant improvements introduced by the proposed training strategy were observed, comparing models VGG-NbB-M and MPIE-NbB-M with the corresponding models trained with raw images (VGGr-NbB-M and MPIEr-NbB-M), respectively. It was observed that the best model, MPIE-NbB-P achieves 96.58% and 92.84% accuracy under type-I and type-II attacks (partition *fb*). This implies a severe privacy issue; more than 90% of the subjects in the database can be identified with a leaked template.

4.4 Computation Time

In the testing stage, with an NVIDIA TITAN X Pascal (GPU) and an Intel(R) i7-6800K @ 3.40 GHz (CPU), the average time (in microseconds) to reconstruct a single face template with D-CNN, NbNet-A, and NbNet-B is shown in Table 7.

TABLE 7
Average reconstruction time (ms) for a single template. The total number of network parameters are indicated in the last column.

	CPU	GPU	#Params
D-CNN	84.1	0.268	4,432,304
NbNet-A	62.6	0.258	2,289,040
NbNet-B	137.1	0.477	3,411,472

5 Conclusions and Future Work

We investigated the security and privacy of deep face templates by studying the reconstruction of face images via the inversion of their corresponding deep templates. A *NbNet* was trained for reconstructing face images from their corresponding deep templates and strategies for training generalizable and robust *NbNets* were developed. Experimental results indicated that the proposed NbNet-based reconstruction method outperformed RBF-regression-based face template reconstruction in terms of attack success rates. We demonstrate that in verification scenario, TAR of 95.20% (58.05%) on LFW under type-I (type-II) attack @ FAR of 0.1% can be achieved with our reconstruction model. Besides, 96.58% (92.84%) of the images reconstruction from templates of partition *fa* (*fb*) can be identified from partition *fa* in color FERET [39]. This study revealed potential security and privacy issues resulting from template leakage in state-of-the-art face recognition systems, which are primarily based on deep templates.

Our future research goals are two-fold: protecting the system from template reconstruction attacks and improving the proposed reconstruction. For the protection, we aim to design a template protection scheme [17], [19], [20], [67] by introducing user-specific randomness into deep networks for extracting secure and discriminative deep templates. Therefore, the extracted deep templates not only depend on the input images, but also subject-specific keys. To further enhance the system security, stronger anti-spoofing techniques [4], [5], [6], [7], [8] will also be sought. For the improvement of the reconstruction, we plan to (*i*) devise a more effective reconstruction algorithm by designing a more effective *NbNet* and considering the holistic contents in face images; and (*ii*) study cross-system attacks using face images reconstructed from the templates of a given face recognition system to access a different face recognition system (different from the one used to generate he template).

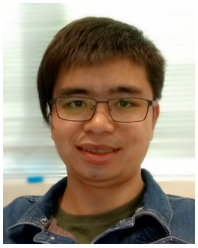
Acknowledgments

This research was partially supported by a Hong Kong RGC grant (HKBU 12201414) and the Madam Kwok Chung Bo Fun Graduate School Development Fund, HKBU. The authors would like to thank Prof. Wei-Shi Zheng, Dr. Xiangyuan Lan and Miss Huiqi Deng for their helpful suggestions.

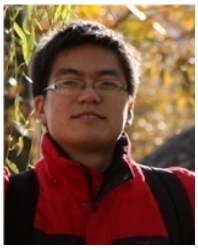
References

[1] A. Adler, "Sample images can be independently restored from face recognition templates," in *CCECE*, 2003.
[2] J. Galbally, C. McCool, J. Fierrez, S. Marcel, and J. Ortega-Garcia, "On the vulnerability of face verification systems to hill-climbing attacks," *Pattern Recognition*, 2010.
[3] Y. C. Feng, M.-H. Lim, and P. C. Yuen, "Masquerade attack on transform-based binary-template protection based on perceptron learning," *Pattern Recognition*, 2014.
[4] D. Wen, H. Han, and A. K. Jain, "Face spoof detection with image distortion analysis," *IEEE Transactions on Information Forensics and Security*, 2015.
[5] K. Patel, H. Han, and A. K. Jain, "Secure face unlock: Spoof detection on smartphones," *IEEE Transactions on Information Forensics and Security*, 2016.
[6] S. Liu, P. C. Yuen, S. Zhang, and G. Zhao, "3d mask face anti-spoofing with remote photoplethysmography," in *ECCV*, 2016.
[7] R. Shao, X. Lan, and P. C. Yuen, "Deep convolutional dynamic texture learning with adaptive channel-discriminability for 3d mask face anti-spoofing," in *IJCB*, 2017.
[8] Y. Liu, A. Jourabloo, and X. Liu, "Learning deep models for face anti-spoofing: Binary or auxiliary supervision," in *CVPR*, 2018.
[9] A. Mignon and F. Jurie, "Reconstructing faces from their signatures using rbf regression," in *BMVC*, 2013.
[10] "Face id security," *Apple Inc*, 2017. [Online]. Available: "https://images.apple.com/business/docs/FaceID_Security_Guide.pdf"
[11] P. Mohanty, S. Sarkar, and R. Kasturi, "From scores to face templates: a model-based approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.
[12] S. U. Hussain, T. Napoléon, and F. Jurie, "Face recognition using local quantized patterns," in *BMVC*, 2012.
[13] A. Zhmoginov and M. Sandler, "Inverting face embeddings with convolutional neural networks," *arXiv:1606.04189*, 2016.
[14] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*, 2015.
[15] F. Cole, D. Belanger, D. Krishnan, A. Sarna, I. Mosseri, and W. T. Freeman, "Synthesizing normalized faces from facial identity features," in *CVPR*, 2017.
[16] B. Moghaddam, W. Wahid, and A. Pentland, "Beyond eigenfaces: Probabilistic matching for face recognition," in *FG*, 1998.
[17] K. Nandakumar and A. K. Jain, "Biometric template protection: Bridging the performance gap between theory and practice," *IEEE Signal Processing Magazine*, 2015.
[18] A. K. Jain, K. Nandakumar, and A. Ross, "50 years of biometric research: Accomplishments, challenges, and opportunities," *Pattern Recognition Letters*, 2016.
[19] G. Mai, M.-H. Lim, and P. C. Yuen, "Binary feature fusion for discriminative and secure multi-biometric cryptosystems," *Image and Vision Computing*, 2017.
[20] Y. C. Feng, P. C. Yuen, and A. K. Jain, "A hybrid approach for generating secure and discriminating face template," *IEEE Transactions on Information Forensics and Security*, 2010.
[21] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *CVPR*, 1991.
[22] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006.
[23] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *BMVC*, 2015.
[24] M. Hayat, S. H. Khan, N. Werghi, and R. Goecke, "Joint registration and representation learning for unconstrained face identification," in *CVPR*, 2017.
[25] E. Learned-Miller, G. B. Huang, A. RoyChowdhury, H. Li, and G. Hua, "Labeled faces in the wild: A survey," in *Advances in Face Detection and Facial Image Analysis*. Springer, 2016.
[26] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *CVPR*, 2011.
[27] B. F. Klare, B. Klein, E. Taborsky, A. Blanton, J. Cheney, K. Allen, P. Grother, A. Mah, and A. K. Jain, "Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a," in *CVPR*, 2015.
[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.
[29] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *CVPR*, 2010.
[30] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *ICCV*, 2011.
[31] H. Gao, H. Yuan, Z. Wang, and S. Ji, "Pixel deconvolutional networks," *arXiv:1705.06820*, 2017.
[32] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *BMVC*, 2016.

[33] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *CVPR*, 2017.
[34] Y. Tai, J. Yang, X. Liu, and C. Xu, "Memnet: A persistent memory network for image restoration," in *ICCV*, 2017.
[35] D. Wang, C. Otto, and A. K. Jain, "Face search at scale," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
[36] C. Otto, D. Wang, and A. K. Jain, "Clustering millions of faces by identity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
[37] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv:1511.06434*, 2015.
[38] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *ECCV*, 2016.
[39] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The feret evaluation methodology for face-recognition algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.
[40] W. Stallings, *Cryptography and network security: principles and practice*. Pearson, 2016, vol. 7.
[41] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.
[42] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv:1701.00160*, 2016.
[43] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *NIPS*, 2016.
[44] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune, "Plug & play generative networks: Conditional iterative generation of images in latent space," *arXiv:1612.00005*, 2016.
[45] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv:1701.07875*, 2017.
[46] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv:1704.00028*, 2017.
[47] D. Berthelot, T. Schumm, and L. Metz, "Began: Boundary equilibrium generative adversarial networks," *arXiv:1703.10717*, 2017.
[48] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *ICCV*, 2017.
[49] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv:1710.10196*, 2017.
[50] I. Masi, A. T. Trn, T. Hassner, J. T. Leksut, and G. Medioni, "Do we really need to collect millions of faces for effective face recognition?" in *ECCV*, 2016.
[51] S. Banerjee, J. S. Bernhard, W. J. Scheirer, K. W. Bowyer, and P. J. Flynn, "Srefi: Synthesis of realistic example face images," in *IJCB*, 2017.
[52] B. Biggio, P. Russu, L. Didaci, and F. Roli, "Adversarial biometric recognition: A review on biometric system security from the adversarial machine-learning perspective," *IEEE Signal Processing Magazine*, 2015.
[53] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv:1502.03167*, 2015.
[54] G. Mai, M.-H. Lim, and P. C. Yuen, "On the guessability of binary biometric templates: A practical guessing entropy based approach," in *IJCB*, 2017.
[55] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, 2014.
[56] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv:1411.7923*, 2014.
[57] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker, "The cmu multi-pose, illumination, and expression (multi-pie) face database," *CMU Robotics Institute. TR-07-08, Tech. Rep*, 2007.
[58] Random: Probability, mathematical statistics, stochastic processes. [Online]. Available: http://www.math.uah.edu/stat/
[59] N. Dokuchaev, *Probability Theory: A Complete One Semester Course*. World Scientific, 2015.
[60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[61] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," *arXiv:1706.02515*, 2017.
[62] G. B. Huang, M. Mattar, H. Lee, and E. Learned-Miller, "Learning to align from scratch," in *NIPS*, 2012.
[63] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek, "Overview of the face recognition grand challenge," in *CVPR*, 2005.
[64] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.
[65] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, 2016.
[66] S. Liao, Z. Lei, D. Yi, and S. Z. Li, "A benchmark study of large-scale unconstrained face recognition," in *IJCB*, 2014.
[67] M.-H. Lim, S. Verma, G. Mai, and P. C. Yuen, "Learning discriminability-preserving histogram representation from unordered features for multibiometric feature-fused-template protection," *Pattern Recognition*, 2016.



Guangcan Mai received the B.Eng degree in computer science and technology from South China University of Technology, Guangzhou, China, in 2013. He is currently pursuing the Ph.D degree in Computer Science from Hong Kong Baptist University, Hong Kong. He was a Visiting Scholar at Michigan State University, USA. His research interests include biometric security and machine learning. He is a student member of IEEE.



Kai Cao received the Ph.D. degree from Key Laboratory of Complex Systems and Intelligence Science, Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2010. He is currently a Post Doctoral Fellow in the Department of Computer Science and Engineering, Michigan State University, East Lansing. His research interests include biometric recognition, image processing and machine learning

Pong C. Yuen is a Professor of the Department of Computer Science in Hong Kong Baptist University. He was the head of the department from 2009 to 2017. He is the Vice President (Technical Activities) of the IEEE Biometrics Council, Editorial Board Member of Pattern Recognition, Associate Editor of IEEE Transactions on Information Forensics and Security and Senior Editor of SPIE Journal of Electronic Imaging. He also serves as a Hong Kong Research Grant Council Engineering Panel Member. His current research interests include video surveillance, human face recognition, biometric security and privacy.

Anil K. Jain is a University distinguished professor in the Department of Computer Science and Engineering at Michigan State University, East Lansing, Michigan. He served as the editor-in-chief of the IEEE Transactions on Pattern Analysis and Machine Intelligence (1991-1994), a member of the United States Defense Science Board and a member of the Forensic Science Standards Board. He has received Fulbright, Guggenheim, Alexander von Humboldt, and IAPR King Sun Fu awards. He is a member of the United States National Academy of Engineering and a Foreign Fellow of the Indian National Academy of Engineering.

APPENDIX A
PROOF OF THE EXISTENCE OF A FACE IMAGE GENERATOR

Suppose a face image $\boldsymbol{x} \in \mathbb{R}^{h \times w \times c}$ of height h, width w, and c channels can be represented by a real vector $\boldsymbol{b} = \{b_1, \cdots, b_k\} \in \mathbb{R}^k$ in a manifold space with $h \times w \times c \gg k$, where $b_i \sim \mathcal{F}_{b_i}, i \in [1, k]$ and $\mathcal{F}_{b_i}$ is the cumulative distribution function of b_i. The covariance matrix of $\boldsymbol{b}$ is $\Sigma_{\boldsymbol{b}}$. Given a multivariate uniformly distributed random vector $z \in [0, 1]^k$ consisting of k independent variables, there exists a generator function $\boldsymbol{b'} = \hat{r}(\boldsymbol{z})$, $\boldsymbol{b'} = \{b'_1, \cdots, b'_k\}$ such that $b'_i \sim \mathcal{F}_{b_i}, i \in [1, k]$, and $\Sigma_{\boldsymbol{b'}} \cong \Sigma_{\boldsymbol{b}}$.

Proof. The function $\hat{r}(\cdot)$ exists and can be constructed by first introducing an intermediate multivariate normal random vector $\boldsymbol{a} \sim \mathcal{N}(\boldsymbol{0}, \Sigma_{\boldsymbol{a}})$, and then applying the following transformations:

(a) NORTA [1], [2], which transforms vector $\boldsymbol{a}$ into vector $\boldsymbol{b'} = \{b'_1, \cdots, b'_k\}$ with $b'_i \sim \mathcal{F}_{b_i}, i \in [1, k]$ and the corresponding covariance matrix $\Sigma_{\boldsymbol{b'}} \cong \Sigma_{\boldsymbol{b}}$ by adjusting the covariance matrix $\Sigma_{\boldsymbol{a}}$ of $\boldsymbol{a}$.

$$b'_i = \mathcal{F}_{b_i}^{-1}\left[\Phi(a_i)\right], i \in [1, k], \tag{11}$$

where $\Phi(\cdot)$ denotes the univariate standard normal cdf and $\mathcal{F}_{b_i}^{-1}(\cdot)$ denotes the inverse of $\mathcal{F}_{b_i}$. To achieve $\Sigma_{\boldsymbol{b'}} \cong \Sigma_{\boldsymbol{b}}$, a matrix $\Lambda_{\boldsymbol{a}}$ that denotes the covariance of the input vector $\boldsymbol{a}$ can be uniquely determined [3]. If $\Lambda_{\boldsymbol{a}}$ is a feasible covariance matrix (symmetric and positive semi-definite with unit diagonal elements; a necessary but insufficient condition), $\Sigma_{\boldsymbol{a}}$ can be set to $\Lambda_{\boldsymbol{a}}$. Otherwise, $\Sigma_{\boldsymbol{a}}$ can be approximated by solving the following equation:

$$\begin{aligned} &\arg\min_{\Sigma_{\boldsymbol{a}}} \mathcal{D}(\Sigma_{\boldsymbol{a}}, \Lambda_{\boldsymbol{a}}) \\ &\text{subject to } \Sigma_{\boldsymbol{a}} \geq 0, \Sigma_{\boldsymbol{a}}(i, i) = 1 \end{aligned} \tag{12}$$

where $\mathcal{D}(\cdot)$ is a distance function [2].

(b) Inverse transformation [4] to generate $\boldsymbol{a} \sim \mathcal{N}(\boldsymbol{0}, \Sigma_{\boldsymbol{a}})$ from multivariate uniformly distributed random vector $\boldsymbol{z} = \{z_1, \cdots, z_k\}$, where $z_i \sim \mathbf{U}(0, 1), i \in [1, k]$.

$$\boldsymbol{a} = \mathbf{M} \cdot \left[\Phi^{-1}(z_1), \cdots, \Phi^{-1}(z_k)\right]' \tag{13}$$

where M is a lower-triangular, non-singular, factorization of $\Sigma_{\boldsymbol{a}}$ such that $\mathbf{M}\mathbf{M}' = \Sigma_{\boldsymbol{a}}$, Φ^{-1} is the inverse of the univariate standard normal cdf [4].

This completes the proof. □

REFERENCES

[1] M. C. Cario and B. L. Nelson, "Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix," Department of Industrial Engineering and Management Sciences, Northwestern University, Tech. Rep., 1997.
[2] S. Ghosh and S. G. Henderson, "Behavior of the norta method for correlated random vector generation as the dimension increases," *ACM Transactions on Modeling and Computer Simulation*, vol. 13, no. 3, pp. 276–294, 2003.
[3] S. G. Henderson, B. A. Chiera, and R. M. Cooke, "Generating dependent quasi-random numbers," in *WSC*, 2000.
[4] M. E. Johnson, *Multivariate Statistical Simulation: A guide to selecting and generating continuous multivariate distributions*. John Wiley & Sons, 2013.