Deep Secure Encoding for Face Template Protection

Rohit Kumar Pandey Yingbo Zhou Bhargava Urala Kota Venu Govindaraju
University at Buffalo, SUNY
{rpandey, yingbozh, buralako, govind}@buffalo.edu

Abstract

In this paper we present a framework for secure identification using deep neural networks, and apply it to the task of template protection for face password authentication. We use deep convolutional neural networks (CNNs) to learn a mapping from face images to maximum entropy binary (MEB) codes. The mapping is robust enough to tackle the problem of exact matching, yielding the same code for new samples of a user as the code assigned during training. These codes are then hashed using any hash function that follows the random oracle model (like SHA-512) to generate protected face templates. The algorithm makes no unrealistic assumptions and offers high template security, cancelability, and matching performance comparable to the state-of-the-art. The efficacy of the approach is shown on CMU-PIE, Extended Yale B, and Multi-PIE face databases. We achieve high ($\sim$ 95%) genuine accept rates (GAR) at zero false accept rate (FAR) while maintaining a high level of template security.

1. Introduction

Authentication on the basis of “who we are" instead of “something we possess" or “something we remember", offers convenience and often, stronger system security. Template protection is one of the important factors related to making biometric passwords as widespread as text based ones. In general biometrics based passwords offer lower template protection in comparison to text passwords due to difficulties in exact matching. Given the sensitive nature of biometric data, algorithms that provide the same level of template security without compromising on matching accuracy would be ideal.

A typical password authentication system would use a sample of the user’s password to extract and store a template from it. It is desirable that this template is stored in a protected and cancelable manner for the purpose of system security. During authentication, a new template is extracted from the presented password and matched to the stored template. Depending on the matching score, access is granted or denied. In the case of text based passwords, a one way non-invertible transform (i.e. a hash) of it is stored as the template. During verification, a password is entered and its hash value is calculated. The hash is compared with the stored hash and if the two strings matched exactly, their hashes would match as well, and access would be granted. In such a scenario, the stored hash reveals no information about the original password (protection) and also, if the password is compromised, it can be changed and a new password can be registered (cancelability).

This kind of security would be ideal for biometric based authentication as well but, unlike text passwords, biometric modalities lack two important aspects. 1) They rarely match exactly between different readings, and 2) they cannot be changed if compromised. Thus, the objective of cancelable biometrics approaches is to extract template from biometric modalities that are 1) protected i.e. given the template, it should be infeasible to extract any information about the original modality, and 2) cancelable i.e. if compromised, it should be possible to extract a new template from the same modality.

1.1. Contribution

We tackle these objectives by using a deep convolutional neural network (CNN) to learn a robust mapping of face classes to maximum entropy binary (MEB) codes. The mapping is robust enough to tackle the problem of exact matching, yielding the same code for new samples of a user as the code assigned during training. This exact matching enables us to store a hash of the code as the template of the user. The hash function used could be any function that follows the random oracle model, and in our case we choose SHA-512 since it is the current standard for string based passwords, and offers strong security. Once hashed, the template has no correlation with the code assigned to the user. Furthermore, the codes assigned to users are bitwise randomly generated and thus, possess maximum entropy, and have no correlation with the original biometric modality (the user’s face). These properties make attacks on the template very difficult, leaving brute force attacks in the code domain and complex dictionary attacks in the input domain as the only feasible options. Cancelability is achieved by changing the codes assigned to users and re-learning the mapping.

Exploiting the large learning capacity of the CNN with powerful regularization, we also achieve matching performance comparable with the state-of-the-art on PIE, Extended Yale B and Multi-PIE databases. Note that, in this work, we focus on the use-case of using faces as passwords and thus, validate our results on data collected in controlled environments.

1.2. Related Work

A variety of template protection algorithms have been applied to faces. Schemes that used cryptosystem based approaches include Fuzzy commitment schemes by Ao and Li [1], Lu *et al*. [11] and Van Der Veen *et al*. [24], and fuzzy vault by Wu and Qiu [25]. In general, the fuzzy commitment schemes suffered from limited error correcting capacity or short keys. In Fuzzy vault schemes the data is stored in the open between chaff points, and this also causes an overhead in storage space. Some quantization schemes were used by Sutcu *et al*. [17, 18] to generate somewhat stable keys. There were also several works that combine the face data with user specific keys. These include combination with a password by Chen and Chandran [2], user specific token binding by Ngo *et al*. [12, 22, 23], biometric salting by Savvides *et al*. [14], and user specific random projection schemes by Teoh and Yuang [21] and Kim and Toh [9]. Hybrid approaches that combine transform based cancelability with cryptosystem based security like [5] have also been proposed but give out user specific information to generate the template creating possibilities of masquerade attacks. Pandey and Govindraju [13] proposed a security centric scheme that used features extracted from local regions of the face to obtain exact matching and thus, benefited from the security of hash functions. Although more secure, the matching accuracy of the scheme suffered and the feature space being hashed was not uniformly distributed.

On the image recognition side, deep CNN based algorithms like Facenet [15] and Deepface [19] have shown exceptional performance holding the current state-of-the-art results for face recognition. There is also some recent work that seeks to map data to binary codes using deep neural networks like [3]. Although mapping to binary codes (or learning hash functions) in this manner may seem similar to our approach, these methods are fundamentally different from what we are trying to achieve. Algorithms such as [3] seek to learn a natural binary representation of the data and thus, the binary codes they map to are correlated to the data distribution. Our MEB codes have no correlation to the original data distribution. This adds to template security, but also makes it a more challenging problem since the mapping function we seek to learn is more complex.

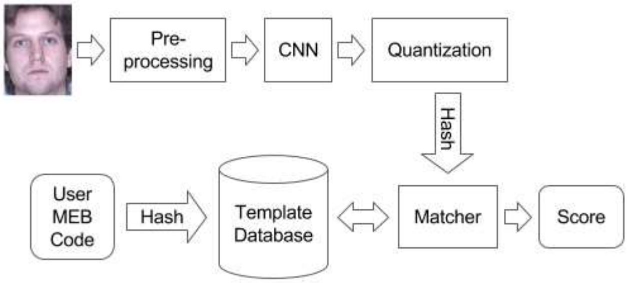

Figure 1. Overview of the algorithm.

2. Algorithm

In this section of the paper we describe the individual components of our architecture in more detail. An overview of the algorithm is shown in Figure 1.

2.1. Convolutional Neural Networks

Convolutional neural networks (CNNs) [10] are biologically inspired models, which contain three basic components: convolution, pooling and fully connected layers. In the convolution layer one tries to learn a filter bank given input feature maps. The input of a convolution layer is a 3D tensor with d number of 2D feature maps of size $n_1 \times n_2$. Let x_{ijk} denote the component at row j and column k in the ith feature map, and we use $x_i^{(l)}$ to denote the complete ith feature map at layer l. If one wants to learn h_f set of filters of size $f_1 \times f_2$, the output $x^{(l+1)}$ for the next layer will still be a 3D tensor with h_f number of 2D feature maps of size $(n_1 - f_1 + 1) \times (n_2 - f_2 + 1)$. More formally, the convolution layer computes the following:

$$x_j^{(l+1)} = s(\sum_i F_{ij}^{(l)} * x_i^{(l)} + b_j^{(l)}) \qquad (1)$$

where $F_{ij}^{(l)}$ denotes the filter that connects feature map x_i to output map $x_j^{(l)}$ at layer l, $b_j^{(l)}$ is the bias for the jth output feature map, $s(\cdot)$ is some element-wise non-linearity function and $*$ denotes the discrete 2D convolution.

The pooling (or subsample) layer takes a 3D feature map and tries to down-sample/summarize the content with less spatial resolution. Pooling is commonly done for every feature map independently and with non-overlapping windows. An intuition of such operation is to have some built in invariance against small translations as well as reduce the spatial resolution and thus save computation for the upper layers. For average (mean) pooling, the output will be the average value inside the pooling window, and for max pooling the output will be the maximum value inside the pooling window.

The fully connected layer connects all the input units from the lower layer l to all the output units in the next layer

$l + 1$. In more detail, the next layer output is calculated by:

$$x^{(l+1)} = s(W^{(l)}x^{(l)} + b^{(l)}) \tag{2}$$

where $x^{(l)}$ is the vectorized input from layer l, $W^{(l)}$ and $b^{(l)}$ are the parameters of the fully connected layers at layer l.

A CNN is commonly composed of several stacks of convolution and pooling layers followed by a few fully connected layers. The last layer is normally associated with some loss to provide training signals, and the training for CNN can be done by doing gradient descent on the parameters with respect to the loss. For example, in classification the last layer is normally a softmax layer and cross entropy loss is calculated against the 1 of K representation of the class labels. In more detail, let $x^{(L)} = Wx^{(L-1)} + b$ be the pre-activation of the last layer, $\mathbf{t}$ denotes the final output and t_k the kth component of $\mathbf{t}$, and $\mathbf{y}$ denote the target 1 of K vector and y_k the kth dimension of that vector, then

$$t_k = \frac{\exp\{x_k^{(L)}\}}{\sum_j \exp\{x_j^{(L)}\}} \tag{3}$$

$$L(\mathbf{t}, \mathbf{y}) = \sum_j y_j \log t_j \tag{4}$$

where L is the loss function.

2.2. Maximum Entropy Binary Codes

Our first step of training is to assign unique codes to each user to be enrolled. Note that these codes are internally used for training during enrollment, and are not supplied to the user or retained in an unprotected form after training. From a template security point of view, these codes should ideally possess two properties. First, they should posses high entropy. Since a hash of these codes is the final protected template, the higher the entropy of the codes, the larger the search space for a brute force attack would be. In order to make brute force attacks in the code domain infeasible, we use binary codes with a minimum dimensionality $K = 256$ and experiment with values up to $K = 1024$. The second desirable property of the codes is that they should not be correlated with the original biometric modality. Any correlation between the biometric samples and the secure codes can be exploited by an attacker to reduce the search space during a brute force attack. One example to illustrate this can be to think of binary features extracted from faces. Even though the dimensionality of the feature vector may be high, given the feature extraction algorithm and type of data, the number of possible values the vector can take is severely reduced. In order to prevent such reduction of entropy, the codes we used are bit-wise randomly generated and have no correlation with the original biometric samples. This makes the space to be hashed uniformly distributed. More precisely, let $c_i \sim \mathbf{B}(1, 0.5)$ be the binary variable for each bit of the code, where $\mathbf{B}(1, 0.5)$ is the maximum entropy Bernoulli distribution, and the resultant MEB code with K independent bits is thus $\mathbf{C} = [c_1, c_2, \ldots, c_K]$. We denote the code for user u by $\mathbf{C}_u$.

2.3. Learning the Mapping

In order to learn a robust mapping of a user's face samples to the codes, we make some modifications to the CNN training procedure. The 1 of K encoding of the class labels is replaced by the MEB codes $\mathbf{C}_u$ assigned to each user. Since we now want several bits of the network output to be one instead of a single bit, we use sigmoid activation instead of softmax. In more detail:

$$t_k = \frac{1}{1 + \exp\{-x_j^{(L)}\}} \tag{5}$$

$$L(\mathbf{t}, \mathbf{C}) = \sum_j \{c_j \log t_j + (1 - c_j) \log(1 - t_j)\} \tag{6}$$

where t_k is the kth output from the last layer and L is the binary cross entropy loss.

2.3.1 Data Augmentation

Deep learning algorithms generally require a large number of training samples whereas, training samples are generally limited in the case of biometric data. In order to magnify the number of training samples per user, we perform the following data augmentation. For each training sample of size $m \times m$ we extract all possible crops of size $n \times n$. Each crop is also flipped along its vertical axis yielding a total of $2 \times (m - n + 1) \times (m - n + 1)$ crops. The crops are then re-sized back to $m \times m$ and used for training the CNN.

2.3.2 Regularization

The large learning capacity of deep neural networks comes with the inherent risk of over-fitting. The number of parameters in the network are often enough to memorize the entire training set, and the performance of such a network does not generalize to new data. In addition to general concerns, mapping to MEB codes is equivalent to learning a highly complex function, where each dimension of the function output can be regarded as an arbitrary binary partition of the classes. This further increases the risk of over-fitting and powerful regularization techniques need be employed to achieve good matching performance.

We apply dropout [8] on all fully connected layers with 0.5 probability of discarding one hidden activation. Dropout is a very effective regularizer and can also be regarded as training an ensemble of an exponential number of neural networks that share the same parameters, therefore reducing the variance of the resulting model.

2.4. Protected Template

Even though MEB codes assigned to each user have no correlation with the original samples, another step of taking a hash of the code is required to generate the protected template. Given the parameters of the network, it is not possible to entirely recover the original samples from the code (due to the max pooling operation in the forward pass of the network) but, some information is leaked. Using a hash digest of the code as the final protected template prevents any information leakage. The hash function used can be any function that follows the random oracle model. For our experiments we utilized SHA-512, yielding the final protected template $\mathbf{T}_u = \text{SHA512}(\mathbf{C}_u)$.

During verification, a new sample of the enrolled user is fed through the network to get the network output $\mathbf{y}_{out} = \mathbf{t}$. We then binarize this output via a simple thresholding operation yielding the code for the sample $\mathbf{s}_{out} = [s_1, s_2, \ldots, s_K]$, where $s_i = \mathbf{1}(t_i > 0.5)$ and $\mathbf{1}(\cdot)$ is the indicator function. At this point, the SHA-512 hash of the code, $\mathbf{H}_{out} = \text{SHA512}(\mathbf{s}_{out})$ could be taken and compared with the stored hash $\mathbf{T}_u$ for the user. Due to the exact matching nature of the framework, this would yield a matching score of true/false nature. This is not ideal for a biometric based authentication system since it is desirable to obtain a tunable score in order to adjust the false accept (FAR) and false reject rates (FRR). In order to obtain an adjustable score, several crops and their flipped counterparts are taken for the new sample (in the manner described in Section 2.3.1) and $\mathbf{H}_{out}$ is calculated for each one, yielding a set of hashes $\mathbb{H}$. We define the final matching score as the number of $\mathbf{H}_{outs}$ in $\mathbb{H}$ that match the stored template, scaled by the cardinality of $\mathbb{H}$. Thus, the score for matching against user u is given by,

$$score = \frac{\sum_{\mathbf{H}_i \in \mathbb{H}} \mathbf{1}(\mathbf{H}_i = \mathbf{T}_u)}{|\mathbb{H}|} \tag{7}$$

Now the score can be set to achieve the desired value of FAR/FRR. Note that, the framework provides the flexibility to work in both verification and identification modes. For identification $\mathbb{H}$ can be matched against templates of all the users stored in the database.

3. Experiments

We now describe the databases, evaluation protocols, and specifics of the parameters used for experimental evaluation.

3.1. Databases

In this study we tackle the problem of using faces as passwords and thus, choose face databases that have been collected in controlled environments for experimentation. We use evaluation protocols including variations in lighting, session and pose that would be typical to applications like face unlock since a reasonable degree of user compliance is expected.

The CMU PIE [16] database consists of 41,368 images of 68 people under 13 different poses, 43 different illumination conditions, and with 4 different expressions. We use 5 poses (c27, c05, c29, c09 and c07) and all illumination variations for our experiments. 10 images are randomly chosen for training and the rest are used for testing.

The extended Yale Face Database B [6] contains 2432 images of 38 subjects with frontal pose and under different illumination variations. We use the cropped version of the database for our experiments. Again, we use 10 randomly selected images for training and the rest for testing.

The CMU Multi-PIE [7] face database contains more than 750,000 images of 337 people recorded in 4 different sessions, 15 view points and 19 illumination conditions. We use this database to highlight the algorithm's robustness to changes in session and lighting conditions. We chose two sessions (3 and 4) which have the most number of common users (198) between them. 10 randomly chosen frontal faces from session 3 are used for enrollment and all frontal faces from session 4 are used for testing.

3.2. Evaluation Metrics

We use the genuine accept rate (GAR) at 0 false accept rate (FAR) as the evaluation metric. We also report the equal error rate (EER) as an alternative operating point for the system. Since the train-test splits we use are randomly generated, we report the mean and standard deviation of the results for 10 different random splits.

3.3. Experimental Parameters

We use the same training procedure for all databases. The CNN architecture that we used is as follows: two convolutional layers of 32 filters of size 7×7 and 64 filters of size 7×7, each followed by max pooling layers of size 2×2. The convolutional and pooling layers are followed by two fully connected layers of size 2000 each, and finally the output. We use rectifier activation function $s(x) = \max(x, 0)$ for all layers, and apply dropout with 0.5 probability of discarding activations to both fully connected layers.

MEB codes of dimensionality $K = 256, 1024$ are assigned to each user. All training images are re-sized to $m \times m = 64 \times 64$ and roughly aligned using eye center locations. For augmentation we use $n \times n = 57 \times 57$ crops yielding 64 crops per image. Each crop is also illumination normalized using the algorithm in [20]. We train the network by minimizing the cross-entropy loss against user codes for 20 epochs using mini-batch stochastic gradient descent with a batch size of 200. 5 of the training samples are initially used for validation to determine the mentioned

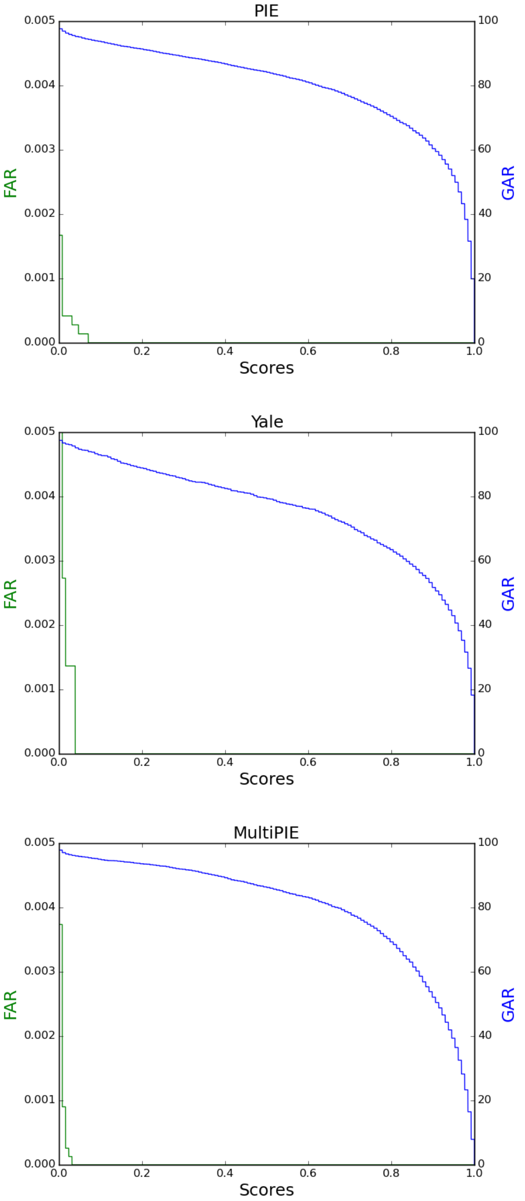

Figure 2. GAR and FAR with respect to matching score at K=256 for PIE (top), Yale (mid) and Multi-PIE (bottom) databases.

training parameters. Once the network is trained, the SHA-512 hashes of the codes are stored as the protected templates and the original codes are purged. During verification, crops are extracted from the new sample, pre-processed, and fed through the trained network. Finally, the SHA-512 hash of each crop is calculated and matched to the stored template, yielding the matching score in Equation 7.

3.4. Results

Table 1. Verification results obtained from various datasets.

Database	K	GAR@0FAR	EER
PIE	256	$93.22 \pm 2.61\%$	$1.39 \pm 0.20\%$
	1024	$90.13 \pm 4.30\%$	$1.14 \pm 0.14\%$
Yale	256	$96.74 \pm 1.35\%$	$0.93 \pm 0.18\%$
	1024	$96.49 \pm 2.30\%$	$0.71 \pm 0.17\%$
Multi-PIE	256	$95.93 \pm 0.55\%$	$1.92 \pm 0.27\%$
	1024	$97.12 \pm 0.45\%$	$0.90 \pm 0.13\%$

Table 2. Performance comparison with other algorithms on PIE dataset.

Method	K	GAR@1FAR	EER
Hybrid Approach [5]	210	90.61%	6.81%
BDA [4]	76	96.38%	–
MEB Encoding	**1024**	**97.59%**	**1.14%**

The results of our experiments are shown in Table 1. We report the mean and standard deviation of GAR at zero FAR, and EER for the 10 different train-test splits at code dimensions, $K = 256, 1024$. We achieve GARs up to $\sim 90\%$ on PIE, $\sim 96\%$ on Yale, and $\sim 97\%$ on Multi-PIE with up to $K = 1024$ at the strict operating point of zero FAR. During experimentation we observed that our results were stable with respect to K, making the parameter selectable purely on the basis of desired template security. In order to get an idea of the system performance with respect to the operating point, we also show the GAR and FAR of the system with respect to the matching score for $K = 256$ in Figure 2. It is noteworthy that the system has very low FAR values even at low matching scores due to the strict exact matching requirements.

A comparison of our results to other face template protection algorithms on the PIE database is shown in Table 2. We compare GAR at an FAR of 1% as this is the reported operating point in [4]. For security level, we compare our code dimensionality parameter (K) to the equivalent parameter in the shown approaches. In absence of algorithm parameters, this is a good measure of the security level against brute force attacks. In terms of matching performance we outperform [5], which offers acceptable security level, and are comparable to [4], which offers lower security to brute force attacks.

4. Security Analysis

We analyze the security of the system in a stolen template scenario. We assume that the attacker has possession of the templates, and knowledge of the template generation algorithm. Given the templates, the attacker's goal is to extract information about the original biometric of the users. Since the hash function used to generate the templates is a one way transformation function, no information about the MEB codes can be extracted from the protected templates. Thus, the only way in which the attacker can get the codes is by brute forcing through all possible values the codes can take, hash each one, and compare them to the templates. We now analyze the search space for such brute force attacks.

In absence of the CNN parameters, the search space for brute force attacks would be 2^K where K is the number of dimensions of the MEB code. This is because the MEB codes are bit-wise randomly generated and uncorrelated to the original biometric data. Since we use a minimum of $k = 256$, the search space would be of the order of 2^{256} or larger, making brute force attacks computationally infeasible.

We now analyze an attack given the CNN parameters. With the CNN parameters, it would make sense to generate attacks in the input domain of the network and try to exploit the FAR of the system. Brute forcing through all possible values in the input domain would yield a search space much larger than 2^K. Thus, in a practical scenario, attackers would most likely perform a dictionary attack using a large set of faces that is available to them. Even though it is not straighforward to analyze the reduction of the attacker's search space due to the knowledge of the system parameters, the FAR of the system under the aforementioned attack scenario is arguably a good indicator of the template security. The genuine and imposter score distributions when all other users other than the genuine are treated as imposter are shown in Figure 3. It can be seen that the imposter scores are always zero, indicating that there are no false accepts in this scenario. The genuine and imposter distributions under a dictionary attack using an attacker database consisting of all frontal images of the Multi-PIE database and genuine database consisting of the smaller Yale database is shown in Figure 4. Again, it can be seen that there are no false accepts indicating that the model does not easily accept external faces even when they are preprocessed in the same manner as the enrolled ones. Separately, we also use a large number of random noise samples as an attack to verify that the CNN does not trivially learn how to map large portions of the input space to the learned codes. Here too we see that there are no false accepts verifying our intuition. Hence, even though it is not straighforward to quantify the reduction in the search space of codes due to knowledge of the CNN parameters, we empirically show that false accepts are difficult due to the strict exact matching requirements of the system.

It is worth noting that even if a MEB code is obtained by the attacker, reconstructing the original biometric in an exact manner is not possible due to the pooling and dropout operations in the CNN. Furthermore, knowledge of one code reveals no information about the others since they are uncorrelated. Thus, if a security breach is detected, it is safe to use a new set of MEB codes to generate another set of templates.

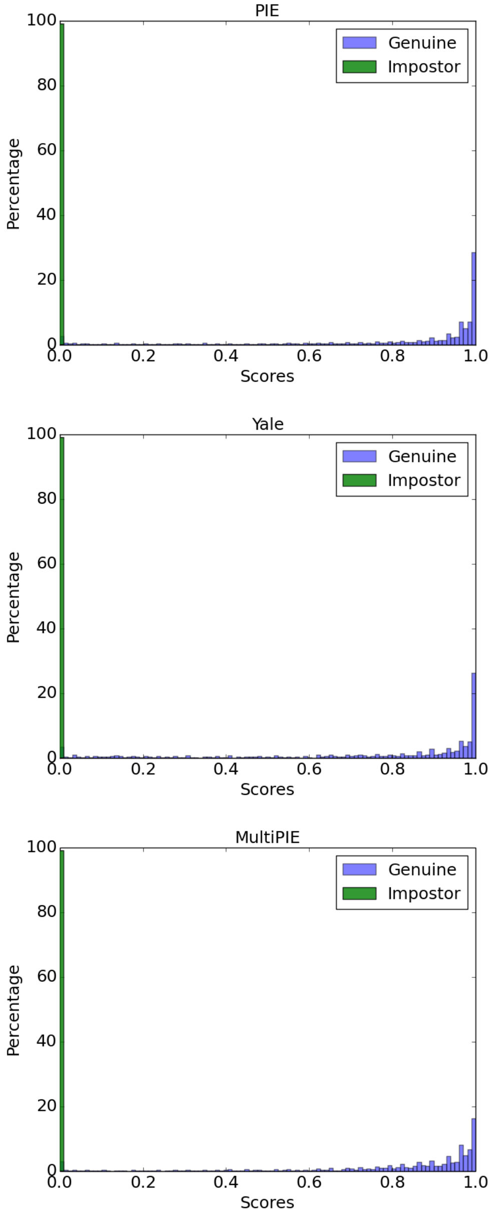

Figure 3. Genuine and imposter distributions from PIE (top), Yale (mid) and Multi-PIE (bottom) databases.

5. Conclusion and Future Work

We presented a template protection algorithm which achieves template security by using MEB codes to address the issue of uniformity, and relying on the strength of standard hash functions. We achieved high ($\sim 95\%$) GARs at the strict operating point of zero FAR and showed that the exceptional performance of deep CNNs can be utilized to minimize loss of matching accuracy in template protection algorithms. The current work deals with the problem of using faces as passwords in controlled environments, and we plan to extend our results to faces in uncontrolled environments, other biometric modalities, and broader applications like Microsoft Windows picture passwords.

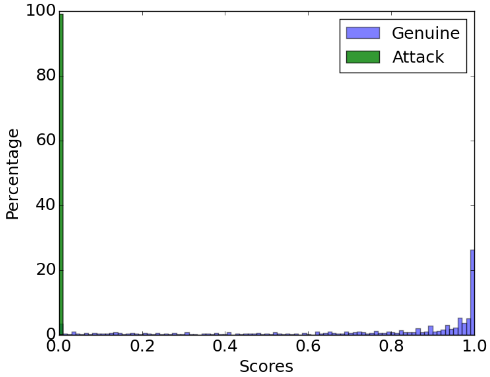

Figure 4. Genuine and imposter distributions under a dictionary attack in the input space.

References

[1] M. Ao and S. Z. Li. Near infrared face based biometric key binding. In *Advances in Biometrics*, pages 376–385. Springer, 2009.

[2] B. Chen and V. Chandran. Biometric based cryptographic key generation from faces. In *Digital Image Computing Techniques and Applications, 9th Biennial Conference of the Australian Pattern Recognition Society on*, pages 394–401. IEEE, 2007.

[3] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2475–2483, 2015.

[4] Y. C. Feng and P. C. Yuen. Binary discriminant analysis for generating binary face template. *Information Forensics and Security, IEEE Transactions on*, 7(2):613–624, 2012.

[5] Y. C. Feng, P. C. Yuen, and A. K. Jain. A hybrid approach for generating secure and discriminating face template. *Information Forensics and Security, IEEE Transactions on*, 5(1):103–117, 2010.

[6] A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.

[7] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-pie. *Image and Vision Computing*, 28(5):807–813, 2010.

[8] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[9] Y. Kim and K.-A. Toh. A method to enhance face biometric security. In *Biometrics: Theory, Applications, and Systems, 2007. BTAS 2007. First IEEE International Conference on*, pages 1–6. IEEE, 2007.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[11] H. Lu, K. Martin, F. Bui, K. Plataniotis, and D. Hatzinakos. Face recognition with biometric encryption for privacy-enhancing self-exclusion. In *Digital Signal Processing, 2009 16th International Conference on*, pages 1–8. IEEE, 2009.

[12] D. C. Ngo, A. B. Teoh, and A. Goh. Biometric hash: high-confidence face recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 16(6):771–775, 2006.

[13] R. K. Pandey and V. Govindaraju. Secure face template generation via local region hashing. In *Biometrics (ICB), 2015 International Conference on*, pages 1–6. IEEE, 2015.

[14] M. Savvides, B. V. Kumar, and P. K. Khosla. Cancelable biometric filters for face recognition. In *ICPR 2004*, volume 3, pages 922–925. IEEE, 2004.

[15] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[16] T. Sim, S. Baker, and M. Bsat. The cmu pose, illumination, and expression (pie) database. In *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pages 46–51. IEEE, 2002.

[17] Y. Sutcu, Q. Li, and N. Memon. Protecting biometric templates with sketch: Theory and practice. *Information Forensics and Security, IEEE Transactions on*, 2(3):503–512, 2007.

[18] Y. Sutcu, H. T. Sencar, and N. Memon. A secure biometric authentication scheme based on robust hashing. In *Proceedings of the 7th workshop on Multimedia and security*, pages 111–116. ACM, 2005.

[19] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1701–1708. IEEE, 2014.

[20] X. Tan and B. Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *Image Processing, IEEE Transactions on*, 19(6):1635–1650, 2010.

[21] A. Teoh and C. T. Yuang. Cancelable biometrics realization with multispace random projections. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(5):1096–1106, 2007.

[22] A. B. Teoh, A. Goh, and D. C. Ngo. Random multispace quantization as an analytic mechanism for biohashing of biometric and random identity inputs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):1892–1901, 2006.

[23] A. B. Teoh, D. C. Ngo, and A. Goh. Personalised cryptographic key generation based on facehashing. *Computers & Security*, 23(7):606–614, 2004.

[24] M. Van Der Veen, T. Kevenaar, G.-J. Schrijen, T. H. Akkermans, F. Zuo, et al. Face biometrics with renewable templates. In *Proceedings of SPIE*, volume 6072, page 60720J, 2006.

[25] Y. Wu and B. Qiu. Transforming a pattern identifier into biometric key generators. In *ICME 2010*, pages 78–82. IEEE, 2010.