Stacked Denoising Autoencoders for Face Pose Normalization

Yoonseop Kang[1], Kang-Tae Lee[2], Jihyun Eun[2],
Sung Eun Park[2] and Seungjin Choi[1]

[1]Department of Computer Science and Engineering
Pohang University of Science and Technology
77 Cheongam-ro, Nam-gu, Pohang 790-784, Korea

[2]KT Advanced Institute of Technology
17 Woomyeon-dong, Seocho-gu, Seoul, Korea
[1]{e0en,seungjin}@postech.ac.kr
[2]{kangtae.lee,eunjihyun,sungeun.park}@kt.com

Abstract. The performance of face recognition systems are significantly degraded by the pose variations of face images. In this paper, a global pose normalization method is proposed for pose-invariant face recognition. The proposed method uses a deep network to convert non-frontal face images into frontal face images. Unlike existing part-based methods that require complex appearence models or multiple face part detectors, the proposed method relies only on a face detector. The experimental results using the Georgia tech face database demonstrate the advantages of the proposed method.

Keywords: Pose normalization, face recognition, autoencoder, stacked denoising autoencoder

1 Introduction

Unlike traditional face recognition systems that recognize large number of people, mobile devices or televisions need to recognize only a small set of people with high accuracy. Although there are large amount of benchmark datasets available, collecting enough amount of training images of the set of people of interest is still very difficult. Therefore, we need a face recognition system that guarantees high accuracy with a small amount of training data.

When given a small number of training data, it is hard to generalize over the many kinds of variations including pose and illumination. While effect of changes in illumination can be reduced by using preprocessing techniques including histogram normalization, the changes in pose is difficult to handle with simple pre-processing. One of the popular approaches to overcome the difficulty of generalization over pose changes is to use pose normalization on face images.

Pose normalization refers to methods that infers a frontal face when given a non-frontal face images. Most of pose normalization algorithms are part-based:

They locate face parts, and re-locate them to their standard positions [1][2]. The main drawback of the part-based methods is that they require all face parts to be located with high accuracy. Locating face parts is done by sophisticated models including AAMs [3], and these models are not suitable for mobile devices with limited processor speed and memory size.

Instead of part-based methods, we suggest a global method that only uses a single detector: a face detector. The proposed method takes a whole non-frontal face image and converts it into a frontal face image. To learn the complex mapping between non-frontal and frontal faces, the proposed method uses a deep network called stacked denoising autoencoder.

In this paper, we first review the stacked denoising autoencoders, and then describe our framework for pose normalization and face recognition. Experiments on a benchmark dataset shows the usefulness of the proposed method in improving face recognition accuracy.

2 Stacked Denoising Autoencoder

2.1 Denoising Autoencoder

The term *autoencoder* refers to an unsupervised, deterministic neural network that 1) generates a hidden representation from input, 2) and then reconstructs input from the hidden representation [4]. Therefore, an autoencoder is composed of two parts: an encoding function and a decoding function (Fig. 1).

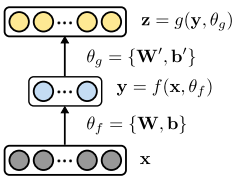


Fig. 1. Structure of an autoencoder with encoding function $f(\boldsymbol{x}, \theta_f)$ and decoding function $g(\boldsymbol{y}, \theta_g)$.

An encoding function $f(\boldsymbol{x}, \theta_f)$ maps an input $\boldsymbol{x}$ to a hidden representation $\boldsymbol{y}$ using a affine transformation with a projection matrix $\boldsymbol{W}$ and a bias $\boldsymbol{b}$, followed by a non-linear squashing function. Sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$ is typically used as a squashing function.

$$\boldsymbol{y} = f(\boldsymbol{x}, \theta_f) = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{1}$$

Then a decoding function $g(\boldsymbol{y}, \theta_g)$ maps the hidden representation back to a reconstruction of input $\boldsymbol{z}$. A decoding function can be either linear or nonlinear. Affine transformation is often used when the input takes real values, and sigmoid

squashing function is applied when the input is binary:

$$\boldsymbol{z} = g(\boldsymbol{y}, \theta_g) = \begin{cases} \boldsymbol{W}'\boldsymbol{y} + \boldsymbol{b}' & \text{or} \\ \sigma(\boldsymbol{W}'\boldsymbol{y} + \boldsymbol{b}') \end{cases} \tag{2}$$

Training an autoencoder is done by minimizing the mean-squared reconstruction error with respect to parameters $\theta_f = \{\boldsymbol{W}, \boldsymbol{b}\}$ and $\theta_g = \{\boldsymbol{W}', \boldsymbol{b}'\}$.

$$\arg\min_{\theta_f, \theta_g} \mathbb{E}\{||\boldsymbol{x} - \boldsymbol{z}||_2^2\} \tag{3}$$

Although autoencoders learn an effective encodings that are capable of reconstructing inputs, they suffer from overfitting when the dimension of hidden representations becomes higher. Moreover, it is likely that such autoencoders to learn a trivial identity mapping, instead of learning useful features from data.

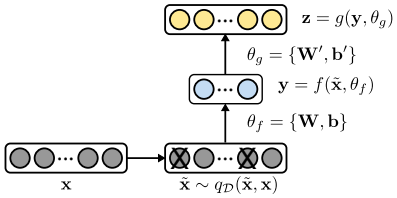


Fig. 2. Structure of an denoising autoencoder with encoding function $f(\boldsymbol{x}, \theta_f)$, decoding function $g(\boldsymbol{y}, \theta_g)$, and stochastic corruption $q_{\mathcal{D}}(\tilde{\boldsymbol{x}}|\boldsymbol{x})$.

Denoising autoencoder (DAE) was proposed to overcome the limitations of autoencoders by reconstructing denoised inputs $\boldsymbol{x}$ from corrupted, noisy inputs $\tilde{\boldsymbol{x}}$ [5]. DAEs avoids overfitting and learns better, non-trivial features by introducing stochastic noises to training samples. One may generate corrupted inputs $\tilde{\boldsymbol{x}}$ from their original value $\boldsymbol{x}$ with several different stochastic corruption criteria $q_{\mathcal{D}}(\tilde{\boldsymbol{x}}|\boldsymbol{x})$, including adding Gaussian random noise, randomly masking dimensions to zero, and adding salt-pepper noise (Fig. 2).

$$\tilde{\boldsymbol{x}} \sim q_{\mathcal{D}}(\tilde{\boldsymbol{x}}|\boldsymbol{x}) \tag{4}$$
$$\boldsymbol{y} = f(\tilde{\boldsymbol{x}}, \theta_f) = \sigma(\boldsymbol{W}\tilde{\boldsymbol{x}} + \boldsymbol{b}) \tag{5}$$
$$\boldsymbol{z} = g(\boldsymbol{y}, \theta_g) = \boldsymbol{W}'\boldsymbol{y} + \boldsymbol{b}' \tag{6}$$

The objective function of DAEs remains the same as typical autoencoders. Note that the objective function minimizes the discrepancy between reconstructions and original, uncorrupted inputs $\boldsymbol{x}$, not the corrupted inputs $\tilde{\boldsymbol{x}}$.

A DAE is trained using back-propagation just as ordinary multi-layer perceptrons.

2.2 Stacked DAEs

Stacking DAEs on top of each other allows the model to learn more complex mapping from input to hidden representations [6]. Just as other deep models including deep belief networks [7], training stacked DAEs is also done in two-phase: layerwise, greedy pre-training and fine-tuning.

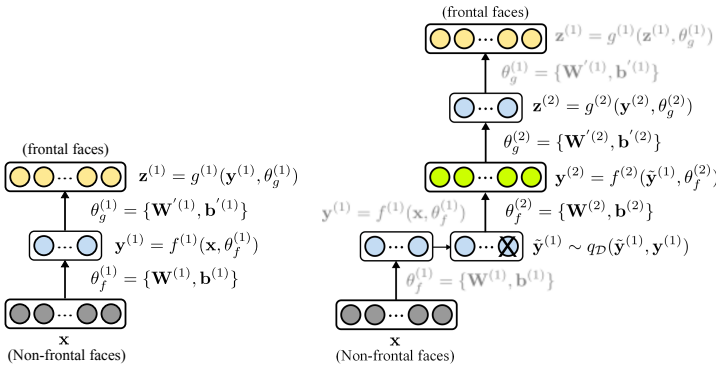


Fig. 3. Pre-training of stacked DAEs. (a) Train a bottom layer DAE with clean and corrupted inputs (in our case, frontal faces and non-frontal faces), then (b) train another DAE that reconstructs $\boldsymbol{z}^{(2)}$ from the hidden representation $\boldsymbol{y}^{(1)}$ extracted from the bottom layer DAE.

Unlike typical deep models that are extended by adding layers from bottom to top in pre-training, stacked DAEs are extended by adding layers in the middle of them. More specifically, the pre-training of stacked DAEs is done by the following steps.

First, train bottom layer DAE with encoding function $\boldsymbol{y}^{(1)} = f^{(1)}(\boldsymbol{x}, \theta_f^{(1)})$ and decoding function $\boldsymbol{z}^{(1)} = g^{(1)}(\boldsymbol{y}^{(1)}, \theta_g^{(1)})$ (Fig. 3(a)). Once the bottom layer DAE is trained, train a new DAE that takes the hidden representations of the bottom layer DAE $\boldsymbol{y}^{(1)}$ as training data. Stochastic noise $q_{\mathcal{D}}(\tilde{\boldsymbol{y}}^{(1)}|\boldsymbol{y}^{(1)})$ is added to $\boldsymbol{y}^{(1)}$ to generate corrupted input $\tilde{\boldsymbol{y}}^{(1)}$.

$$\boldsymbol{y}^{(1)} = f^{(1)}(\boldsymbol{x}, \theta_f^{(1)}) \tag{7}$$

$$\tilde{\boldsymbol{y}}^{(1)} \sim q_{\mathcal{D}}(\tilde{\boldsymbol{y}}^{(1)}|\boldsymbol{y}^{(1)}) \tag{8}$$

$$\boldsymbol{y}^{(2)} = f^{(2)}(\tilde{\boldsymbol{y}}^{(1)}, \theta_f^{(2)}) \tag{9}$$

$$\boldsymbol{z}^{(2)} = g^{(2)}(\tilde{\boldsymbol{y}}^{(2)}, \theta_g^{(2)}) \tag{10}$$

Train more DAEs in a similar way until the desired number of layers is achieved. After pre-training, the weights and biases of stacked DAE are fine-tuned by back-propagation as ordinary neural networks.

3 Pose Normalization using Stacked DAEs

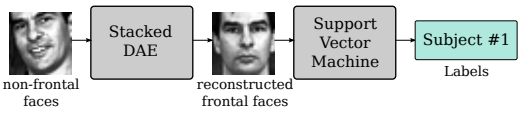


Fig. 4. The schematics of the proposed pose normalization and face recognition system composed of stacked DAEs and SVMs.

The number of face images collected from users is often insufficient to cover various changes in poses. On the other hand, it is relatively easy to obtain large amount of face images with pose variations from the existing databases. Therefore, it is necessary to utilize the face databases to improve the performance of classifiers that are trained with relatively small number of images provided by users. Pose normalization methods can assist classifiers with the following procedure.

1. Train a pose normalization algorithm that learns a general mapping from non-frontal faces to frontal faces, using the existing face image databases.
2. Collect face images from users and train classifier with the collected images.
3. Given a non-frontal face image as a query, run the pose normalization on the query, then feed the pose-normalized image to classifier for recognition.

As mapping from non-frontal faces to frontal faces is highly complex, deep models like stacked DAEs are ideal choice for learning such mappings. Moreover, the fact that the learning procedure of DAEs is not affected by the type of corruption applied to inputs suggests that one can use more sophisticated procedures to corrupt inputs for DAEs, instead of just adding random noises to samples. Therefore, we consider non-frontal face images as corrupted versions of a frontal face, and learn mapping between non-frontal and frontal face images using stacked DAEs for pose normalization and face recognition.

As images take real-values, affine decoding function is used for the bottom layer of stacked DAE, and sigmoid function for the rest of layers. Sigmoid function was used for encoding function for all layers. As described above, Corrupted inputs for the bottom layer was given as non-frontal images, and inputs for higher layers were corrupted by Gaussian noises with standard deviation α (i.e. $q_{\mathcal{D}}(\tilde{\boldsymbol{y}}^{(1)}|\boldsymbol{y}^{(1)}) = \mathcal{N}(\boldsymbol{y}^{(1)}, \alpha^2\boldsymbol{I})$).

4 Face Recognition on Georgia Tech Face Database

4.1 Data Pre-processing

Georgia Tech face database [8] consists of pictures 50 subjects in 15 different poses. One of the 15 poses is frontal, and the variations among poses are relatively

Fig. 5. Frontal and 10 non-frontal faces of 10 subjects from Georgia Tech face database.

high (Fig. 5). We consider the frontal faces as uncorrupted inputs for stacked DAEs, and the remaining 14 non-frontal faces as corrupted inputs.

The resulting dataset is still too small to train a large deep network. Therefore, we applied additional corruptions as below on every frontal and non-frontal image to generate more corrupted samples:

- Translate horizontally and vertically by up to 2 pixels.
- Rotate in -30 to 30 degrees.
- Flip horizontally.

By applying these corruption procedures, we expanded the original dataset with 750 images into a larger dataset with 26,550 images. All corrupted images were converted into grayscale, and histogram-normalized.

4.2 Experimental settings

Two different experiment settings were tested to measure the performance of the proposed method.

1. Setting #1: To simulate the situation of having multiple training samples for each subject, whole dataset was randomly partitioned into training set and test set that contains 80% and 20% of the samples. Training set was used to train both stacked DAE and SVMs, and test set was used to test the proposed face recognition system.
2. Setting #2: To simulate an extreme case of having only single image for each subject, we used the face images of 80% of *subjects* for training stacked DAE, and used the remaining 20% as the training set for SVMs and test set for the proposed face recognition system.

For pose normalization, we trained 3-layer stacked DAE with 2000 and 1000 latent dimensions for each hidden layers (resulting into a network with 1024-2000-1000-2000-1024 nodes), and ran the stochastic gradient updates for 1,200

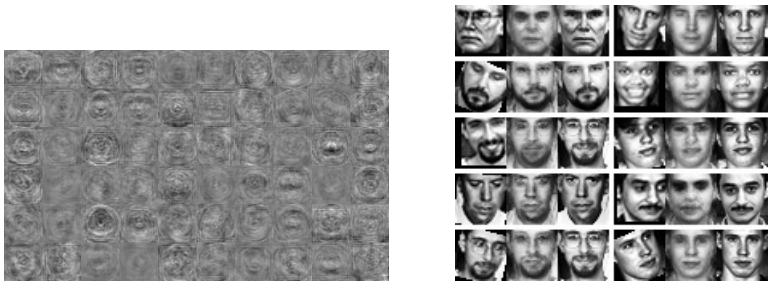
Table 1. Face recognition accuracies on Georgia Tech face database.

settings	baseline(linear)	baseline(RBF)	proposed(linear)	proposed(RBF)
setting #1	0.108	0.098	**0.843**	0.806
setting #2	0.235	0.289	**0.454**	0.409

epochs with batch size 100. The noise level α was set to 0.2. We used linear support vector machine (SVM) and kernel SVM with RBF kernel to classify the faces. We also ran SVMs on the the raw non-frontal face images (without passing them through stacked DAE) as a baseline.

4.3 Experimental Results

Before quantitatively analyzing the face recognition results, we first visualized the weights of stacked DAE learned from the pairs of corrupted non-frontal faces and frontal faces. Most of the weights contained circular filters which captures the rotations of the faces (Fig. 6(a)).



(a) (b)

Fig. 6. (a) Subset of weights learned by the first layer of stacked DAE trained on Georgia Tech face database. Each small square corresponds to weight values between a hidden node and all input pixels. Higher pixel intensity indicates larger weight value. (b) 10 examples of corrupted face images from Georgia tech face database (left), their pose-normalized version obtained by the proposed method (middle), and ground-truth frontal face images (c).

The reconstructed frontal images obtained by processing non-frontal corrupted face images using stacked DAEs were a bit blurry, but still retained important characteristics of the ground-truth frontal face images (Fig. 6(b)).

The quantitative comparison reveals the effectiveness of the proposed approach more clearly. When relatively large number of training samples were provided (setting #1), the improvement of accuracy over naive baseline method was dramatic (84.3% vs. 10.8%). Even when only one training image was given (setting #2), the proposed method significantly improved the classification accuracy (45.4% vs. 23.5%).

5 Conclusion

In this paper, we introduced a pose normalization and face recognition system that uses stacked DAEs. By learning a general mapping from non-frontal faces to frontal faces using stacked DAEs, the proposed method performs pose normalization without any sophisticated appearence models. Experiments on Georgia Tech face database evaluated the effectiveness of the proposed system in terms of face recognition accuracy in usual settings and extreme settings with only single training sample per subject.

Acknowledgments. This work was supported by POSTECH & KT Open R & D Program.

References

1. Du, S., Ward, R.: Component-wise pose normalization for pose-invariant face recognition. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). (2009) 873–876
2. Asthana, A., Marks, T.K., Jones, M.J., Tieu, K.H., MV, R.: Fully automatic pose-invariant face recognition via 3d pose normalization. In: Proceedings of the International Conference on Computer Vision (ICCV). (2011) 937–944
3. Cootes, T., Walker, K., Talyor, C.J.: View-based active appearance models. In: Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition. (2000) 227–232
4. Becker, S.: Unsupervised learning procedures for neural networks. The International Journal of Neural Systems **1 & 2** (1991) 17–33
5. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the International Conference on Machine Learning (ICML). (2008) 1096–1103
6. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research **11** (2010) 3371–3408
7. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation **18**(7) (2006) 1527–1554
8. Nefian, A.V.: Georgia tech face database. `http://www.anefian.com/research/face_reco.htm` (1999)