Exploration of Autoencoder as Feature Extractor for Face Recognition System

Irfan Dwiki Bhaswara
Faculty of EEMCS, University of Twente
Supervised by Raymond Veldhuis and Dan Zeng

***Abstract*—Face recognition has been a challenging research problem due to many variations, for example occlusions, illuminations, poses, and expressions. In this paper, we review one of the unsupervised learning methods called autoencoder to be used as feature extractor for face recognition system. We explore several types of autoencoder, including regular and generative model, and take quantitative measurements on reconstruction and recognition of face images. Experimental results on Face Recognition Grand Challenge dataset show that there is a potential ability in using autoencoder as feature extractor for face recognition. Furthermore, apart from the latent variable dimensions, the encoder and decoder network of the autoencoder have an important role in the reconstruction and recognition performance. We also found that generative autoencoder model gives better clustering against identity of a subject. In addition, we apply residual network in the generative autoencoder model. We called this Resnet-WAE. It performs better in reconstruction and recognition and achieves area under the curve score of 0.8763 using likelihood ratio classifier. In the end, Resnet-WAE demonstrates promising results of using generative model as feature extractor in face recognition system.**

***Index Terms*—Unsupervised Learning; Autoencoder; Face Recognition**

I. Introduction

Supervised learning is widely applied in face recognition applications. The reason is because it is easier to adapt the features in a good representative for the whole training data. Hence, it performs very well on classifying subjects either on a small subset or on a large subset of a training example. Although state of the art performance of face recognition can be achieved by supervised learning, it is by the expense of needs of large-scale training data.

Apart from that, unsupervised learning method could be used in face recognition systems. Normally, unsupervised learning is used as a feature extractor and it will be tied up with some machine learning methods, such as Support Vector Machines (SVM), nearest neighbour, and neural network. The reason why unsupervised learning is choosen because it does not depend on labeled data during training.

One common used of unsupervised learning methods is autoencoder. Some studies show a lot of advantages in using autoencoder as a feature extractor. One of the research compares between regular autoencoder and Principle Component Analysis (PCA) in face recognition [1]. It points out that autoencoder is superior in feature extractor compared to PCA because it is able to learn complex representation of the data. Another usage of autoencoder is proposed in [2], where it uses the features from autoencoder in facial emotion recognition. It proves that nonlinear dimension reduction in autoencoder is more effective compared to linear dimension reduction in PCA. In addition, the result of the autoencoder achieves recognition rate of 99.60%, while PCA only reaches 96.44% on CK+ dataset [3].

In this research, we focus on understanding several types of autoencoders as a feature extractor in face recognition system, including regular autoencoder and generative autoencoder, such as Variational Autoencoder (VAE) [4], Adversarial Autoencoder (AAE) [5], and Wasserstein Autoencoder (WAE) [6]. Some investigations need to be analyzed under several autoencoder models to see which types of autoencoder is promising as feature extractor in face recognition system. Therefore, some research questions are proposed:

Research Question 1: Is it possible that autoencoder to be used as a feature extractor in face recognition system?

Research Question 2: What factors that have an impact on the reconstruction and recognition performance?

Research Question 3: Does the autoencoder based generative model can bring better recognition performance compared to the regular autoencoder?

The rest of the paper is organized as follows. Section II outlines an overview of related works of this research. Section III explains the methods that are used in this research. Section IV describes and discusses the experiment setups and results of the research. Finally, Section V gives conclusions of the research works.

II. Related Work

A. Existing face recognition systems

A lot of methods have been proposed to develop face recognition systems. Those methods are divided into three approaches, that are holistic, local handcraft, and deep learning. Holistic (appearance-based) approach uses distributional concept to build low-dimensional representation, for example Eigenface [7] and Fisherface [8]. This approach has many limitations on the uncontrollable variations in the facial appearance such as lightning condition, expression, and pose. It needs another face recognition system which is robust on environment changes. Therefore, local handcraft methods replace the holistic approach by extracting some geometric shapes and locations from the face, such as eyes, nose, mouth and use it as features for face recognition system. The examples of

local handcraft are Local Binary Pattern (LBP) [9] and Gabor wavelets [10].

However, since the methods are based on the extracted features of the dataset, there is a lack of difficulties in designing the optimal size of the codebook for extracting features. On the other hand, many variations of non-linearity also appear on the face dataset where the local handcraft methods cannot achieve an optimal performance. Therefore, deep learning approaches fix it by introducing nonlinearity in each layer of the system such as Rectified Linear Unit (ReLU) [11]. It is able to handle some problems from the previous approaches, such as illumination, expression, occlusion, and pose [12]. DeepFace [13] and Facenet [14] are the examples of deep learning approaches. DeepFace uses 9 layers of CNN with 3D alignment for face processing. It employs a Siamese network [15] to extract features from pair of faces and compares the features using Euclidean distance. DeepFace is trained to minimize the distance between two images with same identity and maximize the distance of different identity. This network achieves state of the art with accuracy of 97.35% on the Labeled Faces in the Wild (LFW) dataset [16]. Meanwhile, Facenet takes GoogleNet [17] model as its backbone and trains with triplet loss function. The Facenet network needs face triplets which contain anchor, positive sample of same identity, and negative sample of different identity. Basically, the triplet loss calculates the distance between anchor, positive sample, and negative sample. During training, the network tries to minimize the distance between anchor and positive sample of same identity and maximize the distance between anchor and negative sample of different identity. Facenet achieves good performance with accuracy of 99.63% on LFW which is better than DeepFace.

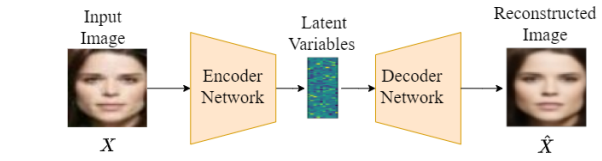
B. Autoencoder

The domain of autoencoder is divided into two models: regular and generative autoencoder model. The regular autoencoder model is only able to reconstruct its input. Meanwhile, the generative autoencoder does not only reconstruct its input but also generate a new sample. Both of them consist of encoder and decoder network. However, in some types of generative autoencoder model, it has a discriminator network which is similar to Generative Adversarial Network (GAN) [18]. AAE and WAE are the example of this model.

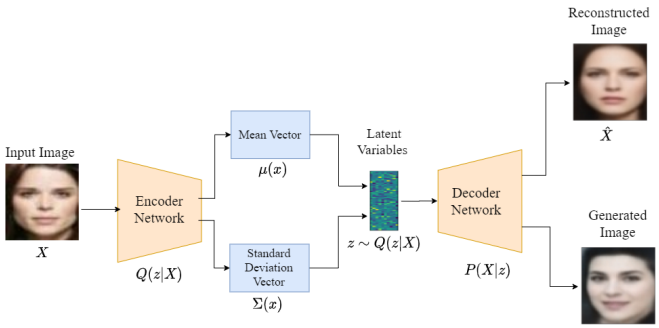
The concept of how a new sample is generated in generative autoencoder model and GAN is totally different. In the generative autoencoder model the encoder output distribution (latent variables) tries to be as close as possible to the true prior distribution and use its decoder to generate a new sample based on its latent variables. Meanwhile, in GAN, the generator network generates fake data from noise and it tries to fool the discriminator network by improving its generated data as close as possible to the real data.

Some details about regular autoencoder and generative autoencoder models are explained as follows:

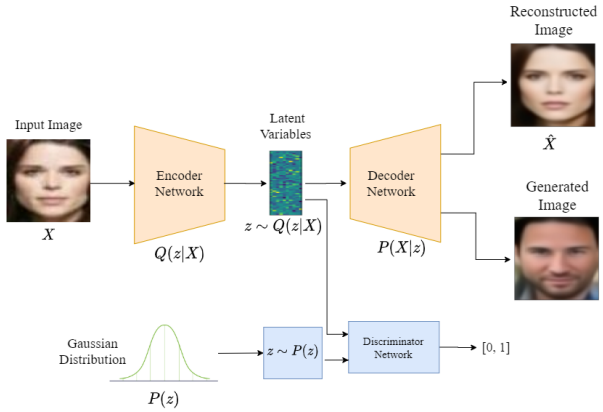
1) ***Regular Autoencoder****:* A Regular autoencoder (AE) has an objective function that is to reconstruct the input data to



(a) Regular AE model



(b) VAE model



(c) AAE and WAE

Figure 1: Autoencoder diagram for image representation. Zoom in for a better resolution.

the output with the minimum possible error. The regular AE network consists of two parts: an encoder part which compress the input of the network into a lower dimensional variables, called codes or latent variables; and a decoder part that reconstructs the latent variables back into its representation (such as image, text, or speech) at the output of the network. This network is shown in Figure 1a.

The regular AE is not designed to be perfectly able to copy its input, otherwise the network will not learn any meaningful representations. Instead, there is a constraint in the regular AE network. The constraint is located at the dimension of the latent variables, where it has a low dimension than the input dimension. This makes the encoder learns to extract some important features from the input, and the decoder tries to learn to use this features to reconstruct back at the output. Together, they are usually trained by using mean-squared error (MSE) loss between the reconstruction and the input as shown

in Equation 1, where $\mathcal{L}$ is the MSE loss function, $\hat{X}$ is the reconstructed image, X is the input image, and N is the total amount of the training data. The loss penalizes the network if it makes a different between the output and the input. Therefore, the reconstructed image is blur than the original image because of the pixel-wise MSE loss. An example of this regular AE network is called undercomplete autoencoder [19].

$$\mathcal{L} = \frac{1}{N}\sum_{i=0}^{N}(\hat{X} - X)^2 \tag{1}$$

*2) **Variational Autoencoder**:* Variational Autoencoder (VAE) is a generative autoencoder model where it has a goal to learn the probability distribution $P(X)$ over its training data X[4, 20]. The reason why we want to model the distribution because we want VAE to be able to create new plausible values of X based on the samples from the distribution. The VAE network is shown in Figure 1b.

From the previous description about VAE, it can be written mathematically as:

$$P(X) = \int P(X|z)P(z)dz \tag{2}$$

Where $P(z)$ is the probability distribution of latent variables z.

The idea of VAE is to sample latent variables of z so that z will be likely to produce data X [20]. Or mathematically, we say: infer $P(z)$ from $P(z|X)$. $P(z|X)$ is the probability distribution that projects the distribution data into latent variables. However, the Equation 2 is intractable or it has no closed-form solution. Moreover, the function $P(z|X)$ is unknown. Fortunately, $P(z|X)$ can be computed using variational inference by approaching $P(z|X)$ on a simple distribution $Q(z|X)$. The distribution that is used here can be any type of distribution, such as in semi-supervised [21] and unsupervised [22] task where they use Gaussian mixture model. However, most of the VAE use standard Gaussian distribution because it is easy to calculate.

In order to approach $P(z|X)$ using $Q(z|X)$, Kullback-Leibler (KL) divergence is used here as in Equation 3.

$$D_{KL}[Q(z|X)||P(z|X)] = \sum_{z} Q(z|X)\log\frac{Q(z|X)}{P(z|X)} \tag{3}$$

From the KL divergence equation above, it can be expanded to find the relationship between $P(X)$, $P(X|z)$, and $P(z)$ by taking Bayes rule. Hence, the result will be VAE objective function:

$$\log P(X) - D_{KL}[Q(z|X)||P(z|X)] = E[\log P(X|z)] + D_{KL}[Q(z|X)||P(z)] \tag{4}$$

The objective function inteprets VAE to find the lower bound of $\log P(X)$, so it can find the distribution of the data by maximizing $\log P(X)$ and simultaneously minimizing $D_{KL}[Q(z|X)||P(z|X)]$. Therefore, Equation 4 is rewritten into Equation 5.

$$\mathcal{L} = E[\log P(X|z)] + D_{KL}[Q(z|X)||P(z)] \tag{5}$$

Where $\mathcal{L}$ is the VAE loss with $E[\log P(X|z)]$ acts as reconstruction loss and $D_{KL}[Q(z|X)||P(z)]$ acts as regularization loss.

*3) **Adversarial Autoencoder**:* Adversarial Autoencoder (AAE) has the same aim as VAE but with different method. It imposes the encoder output distribution into some known prior distribution, such as Gaussian, by using discriminator network instead of taking KL divergence as in VAE. Figure 1c shows the network of AAE.

During training AAE model, there are two stage happens. The first part is reconstruction stage. In this stage, the reconstruction part works the same as regular AE where the model minimizes the reconstruction error between the output and input. The second part is regularization stage. At this stage, the model trains the discriminator network to differentiate between the true prior distributions with the posterior generated samples. The generator (which is the encoder of the autoencoder) is updated to confuse the discriminator network so that the posterior of the generated samples are as close as possible to the true prior distributions. The loss function of AAE is shown in Equation 6

$$\mathcal{L} = E[\log P(X|z)] + [\log D(z) + \log(1 - D(\tilde{z})] \tag{6}$$

Where $D(z)$ is the discriminator model takes on true prior distribution and $D(\tilde{z})$ is the discriminator model that takes the generated sample of posterior distribution $Q(z|X)$. The function $E[\log P(X|z)]$ is the reconstruction loss where it usually uses MSE loss function.

*4) **Wasserstein Autoencoder**:* Wasserstein Autoencoder (WAE) is a generalization form of AAE. WAE has a goal to minimize the distance between data distribution P_X and the model distribution P_G by using Wasserstein distance [6]. Figure 1c shows the WAE model.

In [6], WAE is divided into two different regularizers. First is GAN-based regularizer which uses adversarial loss during training and the training procedure is similar to AAE. It uses discriminator network to distinguish between true prior distribution $P(z)$ and generated posterior distribution $Q(z|X)$. Second is maximum mean discrepancy (MMD) regularizer where it calculates the distance between two means distribution $P(z)$ and $Q(z|X)$.

In this research paper, we focus on WAE-GAN based which uses the adversarial loss. This loss is defined in Equation 7.

$$\mathcal{L} = \inf_{Q(z|X)} E_{P_X} E_{Q(z|X)}[c(X, G(z))] + \lambda D(Q(z|X), P(z)) \tag{7}$$

Here, $c(X, G(z))$ is the reconstruction loss, $D(Q(z|X), P(z))$ is the regularization loss where it encourages posterior distribution $Q(z|X)$ to match with prior

Table I: Autoencoder Architecture

Encoder Network

Layer Name	Input Channels	Output Channels	Kernel Size	Stride	Padding	Activation Function
Conv + BN	3	128	4	2	1	ReLU
Conv + BN	128	256	4	2	1	ReLU
Conv + BN	256	512	4	2	1	ReLU
Conv + BN	512	1024	4	2	1	ReLU
Flatten			$1024 \times 4 \times 4$			
Fully Connected	$1024 \times 4 \times 4$	z_{dim}	-	-	-	-

Decoder Network

Layer Name	Input Channels	Output Channels	Kernel Size	Stride	Padding	Activation Function
Fully Connected	z_{dim}	$1024 \times 8 \times 8$	-	-	-	-
Reshape			$1024 \times 8 \times 8$			
ConvTranspose + BN	1024	512	4	2	1	ReLU
ConvTranspose + BN	512	256	4	2	1	ReLU
ConvTranspose + BN	256	128	4	2	1	ReLU
ConvTranspose + BN	128	3	1	1	0	-

Discriminator Network

Layer Name	Input Channels	Output Channels	Kernel Size	Stride	Padding	Activation Function
Fully Connected	z_dim	512	-	-	-	ReLU
Fully Connected	512	512	-	-	-	ReLU
Fully Connected	512	512	-	-	-	ReLU
Fully Connected	512	512	-	-	-	ReLU
Fully Connected	512	1	-	-	-	-

Notes:
[a] z_{dim} = The dimension of latent variables
[b] Input image and output image size are 64×64
[c] Regular AE learning rate = 1×10^{-3}
[d] VAE learning rate = 2×10^{-4}
[e] AAE and WAE learning rate = 3×10^{-4}
[f] λ constant for WAE = 1

distribution $P(z)$, and λ acts as regularization coefficient where it is used as a trade-off between regularizer and reconstruction loss [23]. The reconstruction loss can use any type of loss such as L_1 loss function or binary cross-entropy loss function. However, if the loss used here is MSE loss function, then it will be equivalent to AAE. Therefore, WAE generalize the AAE model into two parts. First it can use any reconstruction loss functions. Second, it can replace the adversarial loss function into any discrepancy measurement loss function, such as MMD [6].

III. Methodology

A. Architecture of autoencoder

In this following sections, we describe the architecture of existing autoencoder models on our experiments, that are regular AE, VAE, AAE, and WAE. We also explain the details about the model that we modify from WAE, i.e. Resnet-WAE.

The encoder and decoder for regular AE, VAE, AAE, and WAE have the same architecture. The loss functions for regular AE, VAE, AAE, and WAE have been explained in the section II.

In the encoder network, we use 4×4 convolutional filters (Conv) with stride of 2 followed by Batch Normalization (BN) [24] and ReLU. As for the decoder, we use transposed convolution (ConvTranspose) with 4×4 convolutional filters followed by BN and ReLU as well. All the architectures and the hyperparameters are shown in Table I.

As for Resnet-WAE model, we apply a residual network (resnet) [25] as the encoder and decoder of WAE model. The idea of using resnet is because of the residual layer, which can be stacked together into deeper network, hence it will capture a lot of important features. Therefore, It is expected that the autoencoder is not only able to generate a good reasonable image but also to have a good feature extractor.

We build two types of Resnet-WAE. First it contains 18 layers, which we call it Resnet-WAE18. Second it contains 34 layers, which we call it Resnet-WAE34. We do this to see whether the deeper layer will have an effect on the reconstruction and recognition performance. The loss function of Resnet-WAE is same with WAE model.

The encoder consists of 3×3 convolution with fix feature size of 64, 128, 256, and 512. For the decoder, we use some combinations of resize-convolution [26, 27] to reshape the latent variables into output image of 64×64. This is also used to avoid checkerboard artifacts at the reconstructed and generated images [26, 27]. Last, for the discriminator network, we use the same discriminator network as in WAE model. The architecture and hyperparameters are the same as listed in Table II.

B. Classifier

We use two classifiers for measuring the recognition performance of autoencoder. The likelihood ratio classifier and Euclidean distance.

Table II: Resnet-WAE Architecture

Encoder Network

Layer Name	Resnet_WAE18	Resnet_WAE34
Conv1 + BN	3×3, 64, stride 2	3×3, 64, stride 2
Conv2_x + BN	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$
Conv3_x + BN	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$
Conv4_x + BN	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$
Conv5_x + BN	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$
Average Pooling		
Flatten		
Fully Connected	in = 512, out = z_{dim}	in = 512, out = z_{dim}

Decoder Network

Layer Name	Resnet_WAE18	Resnet_WAE34
Fully Connected	in = z_{dim}, out = 512	in = z_{dim}, out = 512
Reshape	$512 \times 4 \times 4$	
ResizeConv2_x + BN	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 3$
ResizeConv3_x + BN	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 6$
ResizeConv4_x + BN	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 4$
ResizeConv5_x + BN	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$

Notes:
[a] z_{dim} = The dimension of latent variables
[b] Input image and output image size are 64×64
[c] Resnet-WAE learning rate = 3×10^{-4}

1) Likelihood ratio classifier [28]: The likelihood ratio classifier takes any two features and compare it to get the likelihood ratio score. From this score, we can measure the ROC curve to get the recognition performance. To obtain the features, we just simply extract it using the encoder of autoencoder.

Suppose we have biometric feature vectors $\mathbf{x} \in \mathbb{R}^M$ and $\mathbf{y} \in \mathbb{R}^N$, where M and N are the dimension of the feature vectors with $M \geq N$. These features should support the hypothesis H_s where the samples come from the same individual versus H_d where the samples come from different individual. In addition, the features $\mathbf{x}$ and $\mathbf{y}$ are feature vectors from random individual which are specified by its feature mean. Hence, we can write mathematical assumption of our features $\mathbf{x}$ and $\mathbf{y}$ as:

$$\mathbf{x} = \mu_\omega + \mathbf{w}_\omega \quad \text{and} \quad \mathbf{y} = \mu'_\theta + \mathbf{w}'_\theta \tag{8}$$

Where $\mu_\omega = E\{\mathbf{x}|\omega\} \in \mathbb{R}^M$ and $\mu'_\theta = E\{\mathbf{y}|\theta\} \in \mathbb{R}^N$ are the subject-specific mean that model the between-subject variations, $\mathbf{w}_\omega$ and $\mathbf{w}'_\theta$ are the stastically independent which have zero-mean within-subject variations, and ω and θ represent the identity of $\mathbf{x}$ and $\mathbf{y}$. The covariance and cross-covariance matrices of $\mathbf{x}$ and $\mathbf{y}$ are

$$\begin{aligned} \Sigma_{xx} &= E\{\mathbf{x}\mathbf{x}^T\} \in \mathbb{R}^{M \times M} \quad \text{and} \quad \Sigma_{yy} = E\{\mathbf{y}\mathbf{y}^T\} \in \mathbb{R}^{N \times N} \\ \Sigma_{xy} &= E\{\mathbf{x}\mathbf{y}^T\} \in \mathbb{R}^{M \times N} \quad \text{and} \quad \Sigma_{yx} = \Sigma_{xy} \end{aligned} \tag{9}$$

In the cross-covariance Σ_{xy} above, if $\omega = \theta$, then the cross-covariance will become $\Sigma_{xy} = E\{\mu_\omega \mu'^T_\theta\}$. Otherwise, the cross-covariance is $\Sigma_{xy} = 0$. During training, this covariance and cross-covariance above need to be estimated. The estimated cross-covariance $\hat{\Sigma}_{xy} = \frac{1}{K} \sum_{i=1}^{K} \hat{\mu}_i \hat{\mu'}_i^T$, with K is the number of individuals in training, $\hat{\mu}_i$ and $\hat{\mu'}_i$ are the estimated sample means of subject i.

From the previous hypotheses H_s and H_d, we can write it into mathematical form that quantified by likelihood ratio as:

$$l(\mathbf{x},\mathbf{y}) = \frac{p\left(\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix}|H_s\right)}{p\left(\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix}|H_d\right)} \tag{10}$$

Next, we assume that the probability density of the pairs of feature vectors for μ_ω, μ'_θ, $\mathbf{w}_\omega$, $\mathbf{w}'_\theta$ with unknown ω and θ have Normal and zero-mean. Subsequently, we use the previous assumption to Equation 10 and take log as well as ignoring some constants, we will get similarity score of $\mathbf{x}$ and $\mathbf{y}$

$$s(\mathbf{x},\mathbf{y}) = \\ \left(\mathbf{x}^T\mathbf{y}^T\right)\left(\begin{pmatrix}\Sigma_{xx} & 0\\ 0 & \Sigma_{yy}\end{pmatrix}^{-1} - \begin{pmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_{yy}\end{pmatrix}^{-1}\right)\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} \tag{11}$$

The Equation 11 can be simplified by reducing the dimensionality and applying whitening transforms to $\mathbf{x}$ and $\mathbf{y}$ as in Equation 12. This will also make sure that the estimated covariance matrix can be inverted and have full rank.

$$s(\mathbf{x}_w,\mathbf{y}_w) = \\ \left(\mathbf{x}_w^T\mathbf{y}_w^T\right)\left(\begin{pmatrix}\mathbf{I} & 0\\ 0 & \mathbf{I}\end{pmatrix}^{-1} - \begin{pmatrix}\mathbf{I} & \Sigma^w_{xy}\\ \Sigma^w_{yx} & \mathbf{I}\end{pmatrix}^{-1}\right)\begin{pmatrix}\mathbf{x}_w\\ \mathbf{y}_w\end{pmatrix} \tag{12}$$

With $\mathbf{I}$ is the identity matrix, $\mathbf{x}_w = \mathbf{W}_H\mathbf{x} \in \mathbb{R}^{M_w}$ and $\mathbf{y}_w = \mathbf{W}_L\mathbf{y} \in \mathbb{R}^{N_w}$. M_w and N_w are the dimension of $\mathbf{x}_w$ and $\mathbf{y}_w$ with $M_w < M$, $N_w < N$, and $M_w \geq N_w$. Additionally, the identity matrix from Equation 12 are obtained from $\Sigma^w_{xx} = E\{\mathbf{x}_w\mathbf{x}_w^T\} = \mathbf{I}$, $\Sigma^w_{yy} = E\{\mathbf{y}_w\mathbf{y}_w^T\} = \mathbf{I}$, and $\Sigma^w_{xy} = \mathbf{W}_H\Sigma_{xy}\mathbf{W}_L^T$.

The Σ^w_{xy} can be decomposed using singular value decomposition (SVD). Hence $\Sigma^w_{xy} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ with $\mathbf{U} \in \mathbb{R}^{M_w\times M_w}$, and orthonormal, $\mathbf{V} \in \mathbb{R}^{N_w\times N_w}$, and orthonormal, and $\mathbf{D} \in \mathbb{R}^{M_w\times N_w}$. From $\mathbf{D}$ matrix, the first N_w rows form a diagonal matrix that contains singular values in decreasing order $v_i = 1,\ldots,N_w$. In addition, from $\mathbf{D}$ matrix, the last $M_w - N_w$ are all 0 matrix. Moreover, the rank of $\mathbf{D}$ can be determined from $D = min(N_w, K-1)$ in a trained classifier, where K is the number of individuals in training. We can transform the feature vectors again using the SVD components above, resulting in

$$\mathbf{x}_c = (\mathbf{U}_{*,1:D})^T\mathbf{x}_w \in \mathbb{R}^D \quad \text{and} \quad \mathbf{y}_c = (\mathbf{V}_{*,1:D})^T\mathbf{y}_w \in \mathbb{R}^D \tag{13}$$

The subscript *,$1:D$ above indicates that the only first D columns of matrix $\mathbf{U}$ and $\mathbf{V}$ are taken.

Next, we can rewrite 12 as

$$s(\mathbf{x}_c,\mathbf{y}_c) = \left(\mathbf{x}_c^T\mathbf{y}_c^T\right)\left(\begin{pmatrix}\mathbf{I} & 0\\ 0 & \mathbf{I}\end{pmatrix}^{-1} - \begin{pmatrix}\mathbf{I} & \mathbf{D}\\ \mathbf{D} & \mathbf{I}\end{pmatrix}^{-1}\right)\begin{pmatrix}\mathbf{x}_c\\ \mathbf{y}_c\end{pmatrix} \tag{14}$$

Where $\mathbf{D} \in \mathbb{R}^{D\times D}$ is a diagonal matrix from Σ^w_{xy} with D largest singular values v_i on the diagonal. Equation 14 can be simplified into

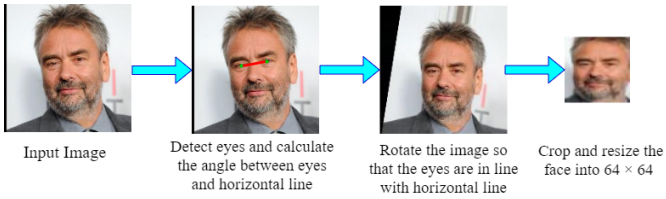


Figure 2: Alignment process on the dataset.

$$s(\mathbf{x}_c,\mathbf{y}_c) = \\ -\sum_{i=1}^{D}\frac{v_i}{1-v_i}(x_{c,i}-y_{c,i})^2 + \sum_{i=1}^{D}\frac{v_i}{1+v_i}(x_{c,i}+y_{c,i})^2 \tag{15}$$

The log-likelihood ratio classifier is then defined as

$$\log(l(\mathbf{x}_c,\mathbf{y}_c)) = -\frac{1}{2}\sum_{i=1}^{D}\log(1-v_i^2) + \frac{1}{4}s(\mathbf{x}_c,\mathbf{y}_c) \tag{16}$$

2) Euclidean distance: To measure the recognition performance based on Euclidean distance, first we extract the reconstructed image features by using Facenet. Next, we use Euclidean distance to calculate the distance between two features. The mathematical form of Euclidean distance is shown as follows

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=0}^{n}(p_i-q_i)^2} \tag{17}$$

Where $\mathbf{p}$ and $\mathbf{q}$ are the feature vectors 1 and 2, respectively.

IV. Results and Discussions

All training and testing were done using Pytorch [29]. We used batch of size 100 and trained all models for 50 epochs. The experiments conducted are comparing all model performances on reconstruction and recognition.

A. Experiment

1) Dataset: We use dataset from CASIA-WebFace [30] to train all autoencoder models. This dataset contains about 10,000 subjects with 500,000 images. All the images are cropped and resized into 64×64.

Before we train all autoencoder models, we have to make sure that the location of the face for all images are the same in the center. In order to do so, we apply an affine transformation to align the face image dataset according to the angle between the center of the eyes and the horizontal level. For finding the center of the eyes, we use dlib [31] facial landmark. The aligment process diagram is shown in Figure 2.

Regarding to test the autoencoder models, we use dataset from Face Recognition Grand Challenge (FRGC) [32]. This dataset contains 50,000 recordings which are divided into training and validation. The FRGC dataset has 3 parts: 3D images, high resolution still images, and multi-images of each person [32]. As for our experiment, we only take about 5000

Table III: Some dimension settings for likelihood ratio classifier

Name	Latent variables of 64		Latent variables of 96		Latent variables of 128	
	N_{pc}	N_{dc}	N_{pc}	N_{dc}	N_{pc}	N_{dc}
Regular AE	64	57	96	71	128	89
VAE	64	63	96	13	128	7
AAE	64	52	96	92	128	119
WAE	64	17	96	87	128	125
RWAE18	64	8	96	28	128	68
RWAE34	64	9	96	58	128	34

Table IV: Measurement of reconstructed results on all autoencoder models

Name	Latent variables of 64		Latent variables of 96		Latent variables of 128	
	SSIM	FaceQNet	SSIM	FaceQNet	SSIM	FaceQNet
Regular AE	0.81	0.576	0.83	0.584	0.84	0.590
VAE	0.76	0.541	0.76	0.541	0.76	0.544
AAE	0.78	0.553	0.81	0.575	0.82	0.569
WAE	0.8	0.573	0.82	0.575	0.83	0.578
Resnet-WAE18	0.82	0.575	0.83	0.596	0.84	0.598
Resnet-WAE34	0.82	0.583	0.84	0.593	0.83	0.600

(a) CASIA-WebFace dataset

(b) FRGC dataset

Figure 3: Some sample images for training and testing.

high resolution still images with 270 subjects. Apart from that, we also create a small dataset containing genuine and impostor images to test the recognition performance by following the same procedure as in [28]. From FRGC dataset, 4 images in each subjects are randomly selected. Next, those images are generated into 6 genuine pairs and 6 impostor pairs. In the end, we will have 270 subjects with 3000 images for genuine pairs and the same amount for impostor pairs. Figure 3 shows some sample images from CASIA-WebFace and FRGC dataset.

2) Experimental Settings: During the experiment, we do some settings to measure the reconstruction and recognition performance. As for the reconstruction performance, we set the dimension of latent variables into three sizes i.e. 64, 96, 128. We also use some learning rate values as it is explained in the notes of Table I and II.

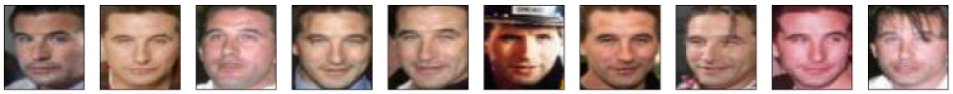
As for recognition performance, we apply dimensionality reduction in the likelihood ratio so it can get the optimal performance. The reduction of these dimensions are shown in Table III where N_{pc} comes from the dimension of the latent variables and N_{dc} is the dimensionality reduction that we apply.

3) Reconstruction Results: To compare the models performance, we need to check how good the reconstructions are for each autoencoder models. We test all autoencoder models on 3 different latent variable dimensions: 64, 96, and 128. We do this to see the effect of increasing the latent variable dimensions with the reconstruction results. Moreover, to check the reconstructed images, we take 1000 sample images from FRGC dataset and use common reconstruction metric that is Structural Similarity Index Measure (SSIM) [33]. SSIM is a method to measure the similarity between two images. In our research, we use it to calculate the similarity between reconstruction and input images. SSIM itself measures 3 aspects from images: luminance, contrast, and structure. The equation of SSIM is shown in Equation 18.

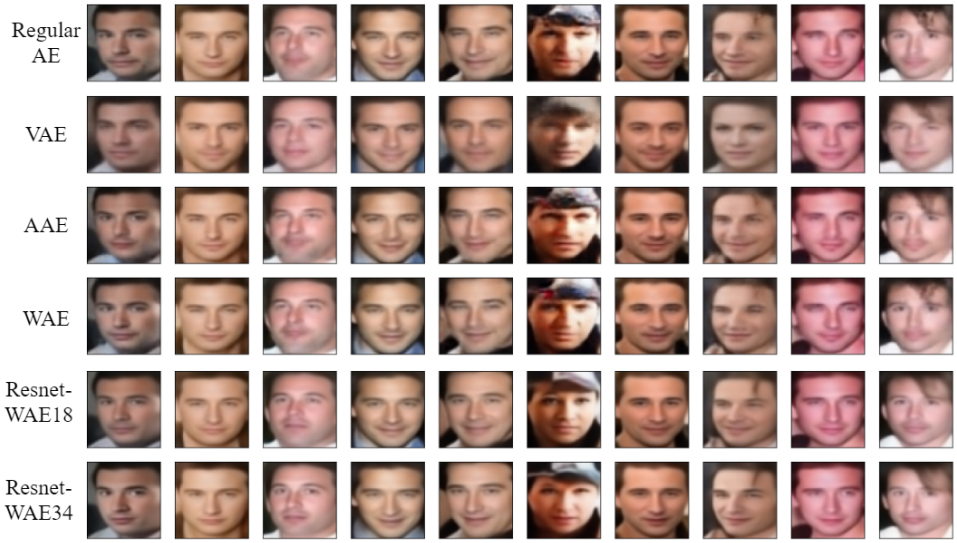
$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{18}$$

Where x and y represent the small Gaussian window on image 1 and 2, μ_x and μ_y are the mean of the window x and y, σ_x and σ_y are the variance, $C_1 = (K_1L)^2$, $C_2 = (K_2L)^2$. K_1 and K_2 are a constant which have a value K_1, $K_2 << 1$, and L is dynamic range of pixel values (255 for grayscale images).

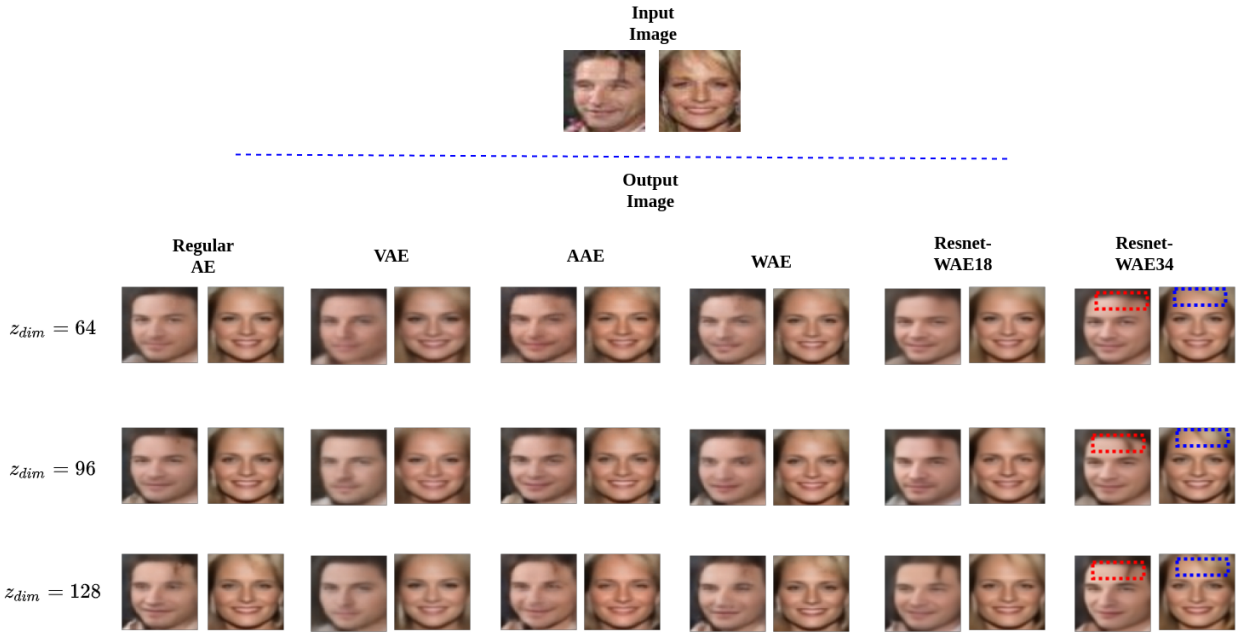
We also measure the quality of the images by using FaceQnet [34, 35]. FaceQnet is useful to examine how good the quality of the images for face recognition. It is trained on Resnet-50 with the dataset from VGGFace2 [36] and



(a) Input images

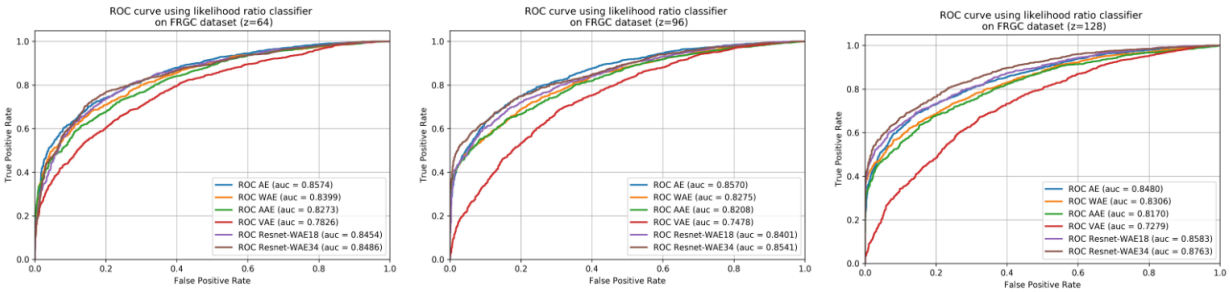


(b) Reconstructed images with latent variable dimensions of **128**

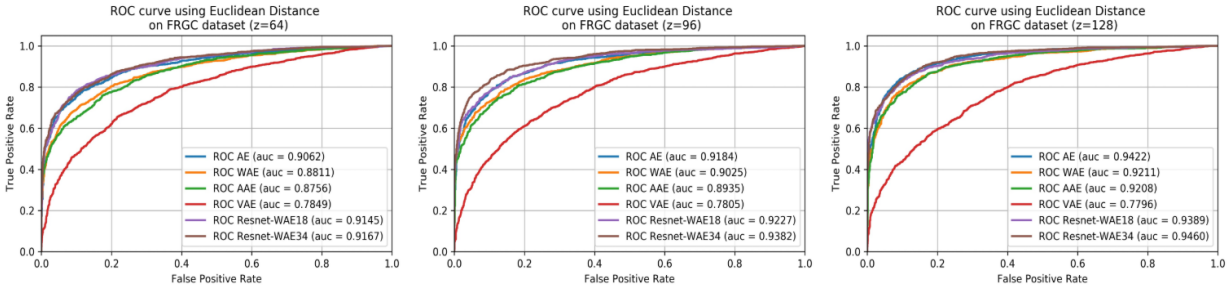


(c) Variation of latent variable dimensions

Figure 4: Some results of reconstructed images. Zoom in for a better resolution.

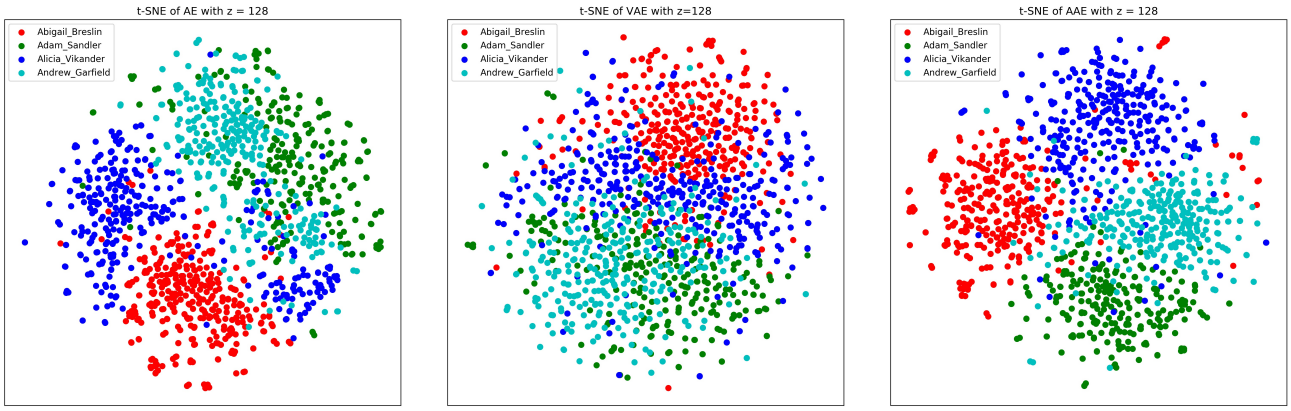


(a) ROC curve results from latent variables features using likelihood ratio classifier



(b) ROC curve results from reconstructed images using Euclidean distance

Figure 5: ROC curve results for all autoencoder models. Zoom in for a better resolution.



(a) TSNE of Regular AE **(b)** TSNE of VAE **(c)** TSNE of AAE

(d) TSNE of WAE **(e)** TSNE of Resnet-WAE34

Figure 6: TSNE results for all autoencoder models. Zoom in for a better resolution.

BioSecure Multimodal Database [37]. The output of FaceQnet shows a quality score of an image which has range from 0 to 1. A good quality image gives a score of 1 and 0 for a bad quality image. The result of the measurements and the reconstructed images are shown in Table IV and Figure 4.

4) Recognition Results: The recognition performance is measured by using likelihood ratio classifier and Euclidean distance measurement. The likelihood ratio classifier measurement uses the genuine and impostor features that are extracted in the latent variables, while the Euclidean distance measurement uses Facenet to take the features from the reconstructed images of genuine and impostor. As for training the likelihood ratio classifier, we apply dimensional reduction as it can be seen from Table III. We manually change the value of N_{dc} until we get the optimal Area Under the Curve (AUC) score. Figure 5 shows the Receiver Operating Characteristic (ROC) curve results of all models.

B. Discussion

In this discussion, we divide into two parts. First, we discuss our reconstruction results. From Table IV, VAE has very low SSIM value compared to other models for all latent variable dimensions. In addition, the scores of FaceQNet are small. Those happen because the reconstructed images from VAE are very blurry which makes the FaceQNet scores are very low and the similarity between the input and output images have many differences.

We see that increasing the dimension of the latent variables for all autoencoder models, it will increase the sharpness of the images. This is because many features extracted from encoder network are captured in the latent variables. As an example, the hair in Figure 4c gives significant difference between each other. In Resnet-WAE, we can see that the reconstructed images of hair are sharper than the other.

Second, we talk about our recognition performance results. We observe that the recognition results using autoencoders features for regular AE, VAE, AAE, and WAE are not as good as using the reconstructed images. There is a drawback in the relationship between reconstruction and recognition task in the autoencoder where increasing the dimension of latent variables does not guarantee that it will also increase the recognition performance. Additionally, [38] explains that good reconstructed images do not depend only the quality of the features but also depend on the decoder network. This is the reason why the result of the Euclidean distance measurement achieves better recognition results, since it is based on the quality of the reconstructed images, which also depends on the decoder network. However, we analyze that the encoder network also gives an important role in the recognition performance, where it is used to extract and select good features. From Figure 5, it shows that increasing Resnet-WAE layer from 18 to 34 layer, it will increase the recognition performance. This is because many important and meaningful features can be captured and selected throughout the encoder network, hence the latent variables will contain essential features for reconstruction and recognition.

Another observation we made is the effect of using adversarial network in the generative autoencoder model. We gather 4 random subjects from the internet which contain 2 male and 2 female identities. From these subjects, we use autoencoder to extract the features and plot it using TSNE [39]. Figure 6 shows the TSNE of regular AE, VAE, AAE, WAE, and Resnet-WAE34 on latent variable dimensions of 128.

From the TSNE results, the regular AE could separate between male (light blue and green circle) and female (dark blue and red) identity. However, among male subjects, regular AE are still mixed up with each other. Meanwhile, for the VAE results, we see that all of the identities are mixed together. Consequently, the VAE model is not quite good to be used for recognition task. Apart from that, the TSNE results from AAE, WAE, and Resnet-WAE34 could separate the class of each subjects. The results prove that the discriminator network helps in imposing and classifying the distribution over the latent variables. Hence, the subjects are clustered well.

V. Conclusion

In this work, we have analyzed several types of autoencoder from the regular autoencoder to generative autoencoder model. We also investigate using deeper layer such as residual network to see the performance of the system.

The regular AE, VAE, AAE, WAE, and Resnet-WAE models can be used as feature extractor. However, the VAE model is not good enough to be used as feature extractor because the reconstructed and the recognition quality of VAE are the lowest compared to other models. Moreover, the identities in the latent variables are still mixed up with each other. Other than that, Resnet-WAE performs better for feature extractor due to the usage of deeper network, hence it gives good reconstructed and recognition score.

Contrary to our expectation, the shallow layer which only has 4 convolutional layers such as in regular AE, VAE, AAE, and WAE have a drawback between reconstruction and recognition performance. Increasing the latent variables will increase the sharpness in the reconstructed images, but it does not guarantee that it will increase the recognition performance, since the latent variables do not contain rich features from the encoder network. Nevertheless, taking on a deeper layer and increasing the latent variable dimensions, such as in Resnet-WAE, will balance both reconstruction and recognition performance, because the encoder of the network contains rich features.

Comparing to the regular autoencoder, the generative autoencoder model based on discriminator network helps the model to force the distribution in better generalization over latent variables. Therefore, as the results, the identity of the data distribution could be clustered well.

References

[1] Krzysztof Siwek and Stanislaw Osowski. Autoencoder versus pca in face recognition. pages 1–4, 09 2017.

[2] Muhammad Usman, Siddique Latif, and Junaid Qadir. Using Deep Autoencoders for Facial Expression Recognition. *arXiv e-prints*, page arXiv:1801.08329, January 2018.

[3] Patrick Lucey, Jeffrey Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. pages 94 – 101, 07 2010.

[4] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv e-prints*, page arXiv:1312.6114, December 2013.

[5] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial Autoencoders. *arXiv e-prints*, page arXiv:1511.05644, November 2015.

[6] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein Auto-Encoders. *arXiv e-prints*, page arXiv:1711.01558, November 2017.

[7] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3(1):71–86, January 1991.

[8] Peter Belhumeur, Joao Hespanha, and David Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19:711–720, 07 1997.

[9] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(12):2037–2041, December 2006.

[10] Linlin Shen and Li Bai. A review on gabor wavelets for face recognition. *Pattern Anal. Appl.*, 9:273–292, 09 2006.

[11] Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines vinod nair. volume 27, pages 807–814, 06 2010.

[12] Andrew Jason Shepley. Deep Learning For Face Recognition: A Critical Analysis. *arXiv e-prints*, page arXiv:1907.12739, July 2019.

[13] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. 09 2014.

[14] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. *arXiv e-prints*, page arXiv:1503.03832, March 2015.

[15] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. 2015.

[16] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. *arXiv e-prints*, page arXiv:1409.4842, September 2014.

[18] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1406.2661, June 2014.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[20] Carl Doersch. Tutorial on Variational Autoencoders. *arXiv e-prints*, page arXiv:1606.05908, June 2016.

[21] Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. Semi-Supervised Learning with Deep Generative Models. *arXiv e-prints*, page arXiv:1406.5298, June 2014.

[22] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders. *arXiv e-prints*, page arXiv:1611.02648, November 2016.

[23] Hongteng Xu, Dixin Luo, Ricardo Henao, Svati Shah, and Lawrence Carin. Learning Autoencoders with Relational Regularization. *arXiv e-prints*, page arXiv:2002.02913, February 2020.

[24] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv e-prints*, page arXiv:1502.03167, February 2015.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv e-prints*, page arXiv:1512.03385, December 2015.

[26] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[27] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image Super-Resolution Using Deep Convolutional Networks. *arXiv e-prints*, page arXiv:1501.00092, December 2014.

[28] Y. Peng, Luuk J. Spreeuwers, and Raymond N. J. Veldhuis. Likelihood ratio based mixed resolution facial comparison. *3rd International Workshop on Biometrics and Forensics (IWBF 2015)*, pages 1–5, 2015.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv e-prints*, page arXiv:1912.01703, December 2019.

[30] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li. Learning Face Representation from Scratch. *arXiv e-prints*, page arXiv:1411.7923, November 2014.

[31] Davis E. King. Dlib-ml: A machine learning toolkit.

Journal of Machine Learning Research, 10:1755–1758, 2009.

[32] P. Jonathon Phillips, P.J. Flynn, T. Scruggs, Kevin Bowyer, Jin (Kyong) Chang, K. Hoffman, J. Marques, Jaesik Min, and W. Worek. Overview of the face recognition grand challenge. volume 1, pages 947– 954 vol. 1, 07 2005.

[33] Zhou Wang, Alan Bovik, Hamid Sheikh, and Eero Simoncelli. Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13:600 – 612, 05 2004.

[34] Javier Hernandez-Ortega, Javier Galbally, Julian Fierrez, Rudolf Haraksim, and Laurent Beslay. FaceQnet: Quality Assessment for Face Recognition based on Deep Learning. *arXiv e-prints*, page arXiv:1904.01740, April 2019.

[35] Javier Hernandez-Ortega, Javier Galbally, Julian Fierrez, and Laurent Beslay. Biometric Quality: Review and Application to Face Recognition with FaceQnet. *arXiv e-prints*, page arXiv:2006.03298, June 2020.

[36] Qiong Cao, Li Shen, Weidi Xie, Omkar M. Parkhi, and Andrew Zisserman. VGGFace2: A dataset for recognising faces across pose and age. *arXiv e-prints*, page arXiv:1710.08092, October 2017.

[37] Javier Ortega-Garcia, Julian Fierrez, Fernando Alonso-Fernandez, Javier Galbally, Manuel Freire, Joaquin Gonzalez-Rodriguez, Carmen García-Mateo, Jose Luis Alba-Castro, Elisardo González Agulla, Enrique Otero-Muras, Sonia Garcia-Salicetti, Lorene Allano, Bao Ly-Van, Bernadette Dorizzi, Josef Kittler, Thirimachos Bourlai, Norman Poh, Farzin Deravi, M. Ng, and Arman Savran. The multiscenario multienvironment biosecure multimodal database (bmdb). *IEEE Transactions on Pattern Analysis Machine Intelligence*, 32, 01 2010.

[38] Michele Alberti, Mathias Seuret, Rolf Ingold, and Marcus Liwicki. A Pitfall of Unsupervised Pre-Training. *arXiv e-prints*, page arXiv:1712.01655, November 2017.

[39] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.