Replicating neuroscience observations on ML/MF and AM face patches by deep generative model

Tian Han\(^1\), Xianglei Xing\(^2\), Jiawen Wu\(^3\) and Ying Nian Wu\(^1\)

\(^1\)University of California, Los Angeles.
\(^2\)Harbin Engineering University.
\(^3\)University of Southern California.

**Keywords:** Active Appearance Model, Generator Network, Variational Auto-encoder

**Abstract**

A recent Cell paper [2] reports an interesting discovery. For the face stimuli generated by a pre-trained active appearance model (AAM), the responses of neurons in the areas of the primate brain that are responsible for face recognition exhibit strong linear relationship with the shape variables and appearance variables of the AAM that generates the face stimuli. In this paper, we show that this behavior can be replicated by a deep generative model called the generator network, which assumes that the observed
signals are generated by latent random variables via a top-down convolutional neural network. Specifically, we learn the generator network from the face images generated by a pre-trained AAM model using variational auto-encoder, and we show that the inferred latent variables of the learned generator network have strong linear relationship with the shape and appearance variables of the AAM model that generates the face images. Unlike the AAM model that has an explicit shape model where the shape variables generate the control points or landmarks, the generator network has no such shape model and shape variables. Yet the generator network can learn the shape knowledge in the sense that some of the latent variables of the learned generator network capture the shape variations in the face images generated by AAM.

1 Introduction

Recently, a paper published in Cell [2] reports an interesting discovery about the neurons in the middle lateral (ML)/middle fundus (MF) and anterior medial (AM) areas of the primate brain that are responsible for face recognition. Specifically, the paper is concerned with how these neurons respond to and encode the face stimuli generated by a pre-trained Active Appearance Model (AAM) [5, 4]. In AAM, there are explicit shape variables and appearance variables that generate the positions of the control points and the nominal face image respectively, and the output image is then generated by wrapping the nominal face image using the control points. [2] discovers that the responses of the aforementioned neurons to the face image generated by the AAM exhibits strong linear relationship with the shape and appearance variables of the AAM that generates
the face image. In fact, the shape and appearance variables of the AAM can be recovered from the neuron responses so that the face image can be reconstructed by the AAM using the recovered shape and appearance variables.

In this paper, we investigate whether the above phenomenon can be replicated by deep generative models. In particular, we focus on a popular deep generative model called the generator network [10], which can be considered a non-linear generalization of the factor analysis model. Recall that in the factor analysis model, the signal is generated by latent factors that are assumed to be independent Gaussian random variables, and the signal is a linear transformation of the latent variables (plus observational noises). In the generator network, the latent variables still follow a simple known prior distribution such as independent Gaussian or uniform distribution, but the mapping from the latent variables to the observed signal is modeled by a convolutional neural network (ConvNet), which has proven to be an exceedingly powerful approximator of high-dimensional non-linear mappings.

Both the AAM and the generator network are latent variable models where the signal is obtained by transforming the latent variables. In the AAM, the latent variables consist of explicit shape variables and appearance variables, which generate the control points and the appearance image by linear mappings learned by principal component analysis (PCA). The output image is generated by a highly non-linear but known warping function of the control points and the nominal image. In contrast, the generator network is more generic, in that it does not assume any prior knowledge about shape and deformation, and it does not have any explicit shape variables and shape model. We are interested in whether the generator model can replicate the AAM in the sense that
whether the generator network can learn from the images generated by a pre-trained AAM, so that the latent variables of the learned generator network are closely related to the latent shape and appearance variables of the AAM, and the non-linear mapping from the latent variables to the output image in the generator network accounts for the highly non-linear warping function of the AAM. As it is impossible for the latent variables of the learned generator network to be the same as the latent variables of the AAM, a strong linear relationship between the two sets of latent variables (or codes) is the best we can hope for. We shall show that such a linear relationship indeed exists, thus qualitatively reproducing the behavior of the neuron responses (or neural code) observed by [2].

The generator network can be trained by various methods, including the wake-sleep algorithm [13], variational autoencoder (VAE) [17, 22, 23], generative adversarial networks (GAN) [10, 21, 7], moment matching networks [18], alternating back-propagation (ABP) [11], and other related methods [20, 8]. They have led to impressive results in a wide range of applications, such as image/video synthesis [9], disentangled feature learning [3, 12] and pattern completion [11] etc.

In this paper, we shall adopt the VAE method to train the generator network. Unlike GAN, the VAE complements the generator network with an inference network that transforms the observed image to the latent variables. The inference network seeks to approximate the posterior distribution of the latent variables given the observed image. The inference network and the generator network form an auto-encoder, where the inference network plays the role of the encoder that encodes the signal into the latent variables (or latent code), and the generator network plays the role of the decoder that
decode the latent variables (or latent code) back to the signal. The parameters of the two networks can be learned by maximizing a variational lower bound of the log-likelihood [1]. We show that the latent variables computed by the inference network from the observed face image are highly correlated with the latent variables of the AAM that generates the face image.

**Contributions.** This paper is phenomenological in nature. It is our hope that the paper is of interest to both the neuroscience community and the deep learning community. The followings are the contributions of this paper:

- We study the linear relationship between the latent code learned by the generator network and the AAM code that generates the face stimuli. Our experiments suggest that the deep generative model exhibits similar behavior as the primate neural system.

- We show that the latent variables learned by the generator network can be separated into shape-related part and appearance-related part, and the generator network is expressive enough to replicate the AAM model.

2 **Active Appearance Model (AAM)**

The active appearance model [5, 4] is a generative model for representing face images. It has a shape model and an appearance model. Both models are learned by principal component analysis (PCA).

*Shape model:* The shape model is based on a set of landmarks or control points. In the training stage, the control points are given for each training image. Let $x$ denote the
coordinates of all the control points. The shape model is

\[ x = \bar{x} + P_sb_s, \]

where \( \bar{x} \) is the average shape, \( P_s \) is the matrix of eigenvectors, and \( b_s \) is the vector of shape variables. \( (\bar{x}, P_s) \) are shared across all the training examples, while \( x \) and \( b_s \) are different for different examples. The model can be learned from the given control points of the training images by PCA, where the number of eigenvectors is determined empirically.

**Appearance model:** The appearance model generates the nominal image before shape deformation. To learn the model, we can wrap each training image to the shape-normalized image so that its control points match those of the mean shape \( \bar{x} \). Then a PCA is performed on the shape-normalized training images. Let \( g \) denote the vector of the grey-level image. The appearance model is

\[ g = \bar{g} + P_ab_a, \]

where \( \bar{g} \) is the mean normalized grey-level image, \( P_a \) is the matrix of eigenvectors, and \( b_a \) is the vector of appearance variables. \( (\bar{g}, P_a) \) are shared by all the training examples, while \( g \) and \( b_a \) are different for different examples. For colored images with RGB channels, we then concatenate three channels into a single vector, then perform the PCA as in grey-level images.

We can learn \( (P_s, P_a) \) from the training images with given control points. We concatenate the shape and appearance variables to form the face representation or the latent code, i.e., \( Z_{AAM} = [b_s, b_a] \). Given \( Z_{AAM} \), we can generate face image \( Y \) by generating \( x \) and \( g \) first, and then warping \( g \) according to \( x \) using a warping function to output the
image $Y = h(g, x)$. The warping function $h$ is given and is highly non-linear in $g$ and $x$.

### 3 Generator Network

The generator network is a deep generative model of the following form:

$$Z \sim N(0, I_d), \quad (3)$$

$$Y = f_\theta(Z) + \epsilon_i, \quad (4)$$

where $Z$ is the vector of latent variables (or latent code), $d$ is the dimension of $Z$, i.e., the number of latent variables. $Z$ is assumed to follow a simple prior distribution where each component is a Gaussian $N(0, 1)$ random variable ($I_d$ is the $d$-dimensional identity matrix). The latent vector $Z$ generates the output image $Y$ by a non-linear mapping $f_\theta$, which is modeled by a top-down convolutional neural network (ConvNet), where $\theta$ collects all the weight and bias parameters of the top-down ConvNet. $\epsilon$ is the noise vector whose elements are independent $N(0, \sigma^2)$ random variables. Even though $Z$ follows a simple distribution, the model can generate $Y$ with very complex distribution and with very rich patterns because of the expressiveness of $f_\theta$. The generator model (3) is a generalization of the factor analysis model, where the mapping from $Z$ to $Y$ is assumed linear.

Compared to the AAM, the generator network has no explicit shape model such as (1) with control points $x$ and shape variables $b_s$, nor does it have the explicit non-linear warping function $Y = h(g, x)$. The generator network relies on the highly expressive ConvNet $f_\theta$ to account for the linear shape model and the non-linear warping func-
tion. Even though no prior knowledge of shape and warping is built into the generator network, it can learn such knowledge by itself.

Specifically, we shall use a pre-trained AAM as a teacher model, and we let the generator network be the student model. The AAM generates training images, and the generator network learns from the training images. We shall show that the inferred $Z$ from the face image $Y$ has a strong linear relationship with the corresponding $Z_{AAM}$ that the AAM uses to generate $Y$.

4 Variational auto-encoder (VAE)

Given a set of $N$ training images $\{Y_i, i = 1, ..., N\}$ generated by AAM, we train the generator network by variational auto-encoder (VAE) [17, 22, 23]. Let $p(Z)$ be the prior distribution of $Z$. Let $p_\theta(Y|Z)$ be the conditional distribution of the image $Y$ given the latent vector $Z$. Then the marginal distribution of $Y$ is $p_\theta(Y) = \int p_\theta(Z,Y)dZ = \int p(Z)p_\theta(Y|Z)dZ$. The log-likelihood is $\sum_{i=1}^{N} \log p_\theta(Y_i)$, and in principle $\theta$ can be estimated by maximizing the log-likelihood. However, this is intractable because $p_\theta(Z)$ involves intractable integral. The EM algorithm [6] is also impractical because the posterior distribution $p_\theta(Z|Y) = p(Z)p_\theta(Y|Z)/p_\theta(Y)$ is intractable. The basic idea of VAE is to approximate the posterior distribution $p_\theta(Z|Y)$ by a tractable inference model $q_\phi(Z|Y)$ with a separate set of parameters $\phi$, such as a Gaussian distribution with independent components $N(\mu_\phi(Y), \sigma^2_\phi(Y))$, where $\mu_\phi(Y)$ is the vector of means of the components of $Z$, and $\sigma^2_\phi(Y)$ is the vector of variances of the components of $Z$. Both $\mu_\phi(Y)$ and $\sigma_\phi(Y)$ can be modeled by bottom-up ConvNets.
The parameters \((\theta, \phi)\) can be learned by jointly maximizing the variational lower bound of the log-likelihood
\[
L(\theta, \phi) = \sum_{i=1}^{N} \left[ \log p_{\theta}(Y_i) - \text{KL}(q_{\phi}(Z_i|Y_i) || p_{\theta}(Z_i|Y_i)) \right],
\] (5)
where \(\text{KL}(q||p)\) denotes the Kullback-Leibler divergence from \(q\) to \(p\). \(L(\theta, \phi)\) is computationally tractable as long as the inference model \(q_{\phi}(Z|Y)\) is tractable. See [16] for more details. \(q_{\phi}(Z|Y)\) is the encoder, and \(p_{\theta}(Y|Z)\) is the decoder. After learning \((\theta, \phi)\), we can estimate \(Z\) from \(Y\) by the learned posterior mean vector \(Z_G = \mu_{\phi}(Y)\). In our work, we use \(Z_G\) as the code of \(Y\).

We can understand VAE as follows. Let \(q_{\text{data}}(Y)\) be the data distribution. Then the maximum likelihood is equivalent to minimizing \(\text{KL}(q_{\text{data}}(Y)||p_{\theta}(Y))\) over \(\theta\). VAE is equivalent to minimizing
\[
\text{KL}(q_{\text{data}}(Y)||p_{\theta}(Y)) + \text{KL}(q_{\phi}(Z|Y)||p_{\theta}(Z|Y))
\] (6)
\[
= \text{KL}(q_{\text{data}}(Y)q_{\phi}(Z|Y)||p(Z)p_{\theta}(Y|Z))
\] (6)
\[
= \text{KL}(q_{\phi}(Z, Y)||p_{\theta}(Z, Y))
\] (7)
over both \(\theta\) and \(\phi\), where \(q_{\phi}(Z, Y) = q_{\text{data}}(Y)q_{\phi}(Z|Y)\) and \(p_{\theta}(Z, Y) = p(Z)p_{\theta}(Y|Z)\).

Unlike the maximum likelihood objective function \(\text{KL}(q_{\text{data}}(Y)||p_{\theta}(Y))\), which is the KL divergence between the marginal distributions, the variational objective function \(\text{KL}(q_{\phi}(Z, Y)||p_{\theta}(Z, Y))\) is the KL divergence between the joint distributions. While the marginal distribution \(p_{\theta}(Y)\) is intractable, the joint distribution \(p_{\theta}(Z, Y)\) is tractable.

Define \(Q = \{q_{\phi}(Z, Y), \forall \phi\}\) and \(P = \{p_{\theta}(Z, Y), \forall \theta\}\) be the two families of joint distributions, where \(q_{\phi}(Z, Y) = q_{\text{data}}(Y)q_{\phi}(Z|Y)\) and \(p_{\theta}(Z, Y) = p(Z)p_{\theta}(Y|Z)\). We
can view VAE as the joint minimization of \( KL(q||p) \) over \( Q \) and \( P \). Such joint minimization can be accomplished by alternating projection as illustrated by Figure 1, where \( Q \) and \( P \) are illustrated by two lines, and each distribution in \( Q \) and \( P \) is illustrated by a point. Starting from \( p = p_0 \in P \), we project \( p_0 \) onto \( Q \) by minimizing \( KL(q||p) \) over \( q \in Q \) to obtain \( q_1 \). Then we project \( q = q_1 \) onto \( P \) by minimizing \( KL(q||p) \) over \( p \in P \) to obtain \( p_1 \), and so on. This process will converge to a local minimum of \( KL(q||p) \). In Figure 1, the two projections are illustrated by two different colors, because they are of different natures. \( \min_{q \in Q} (q|p) \) is a variational projection that minimizes over the first argument, while \( \min_{p \in P} (q|p) \) is a model fitting projection that minimizes over the second argument. As is commonly known, the former has mode seeking behavior while the latter has moment matching behavior.

A precursor to VAE is the wake-sleep algorithm [13], which amounts to replacing minimizing \( KL(q||p) \) over \( q \in Q \) by minimizing \( KL(p||q) \) over \( q \) by switching the order of \( p \) and \( q \). The minimization of \( KL(p||q) \) can be accomplished by generating data from
p in the sleep phase and learn $q_\phi$ from the generated data. Because of the switched order, the wake-sleep algorithm does not have a single objective function. However, in wake-sleep algorithm, both projections are of the model fitting type.

The generator network can also be trained by GAN [10]. However, GAN does not have an inference model or an encoder, which is crucial for our work.

5 Experiments

We conduct experiments to investigate whether the generator network can replicate or imitate the AAM, where the AAM serves as the teacher model and the generator network plays the role of the student model. In the learning stage, the generator network only has access to the images generated by the AAM. It does not have access to the shape and appearance variables (latent code) used by the AAM to generate the images. After learning the generator network, we investigate the relationship between the latent code of the learned generator network and the latent code of the AAM.

5.1 Experiment Setting

Data Generation. We consider models on both grey-scaled and colored face images to show the generalizability of our work. For grey-scaled images, we pre-train the AAM using approximately 200 frontal face images with given landmarks or control points. Coordinates of the landmarks are first averaged, then PCA is performed where the first 10 principal components (PCs) for shape (see Eqn 1) are retained. The landmarks of each training image are then smoothly morphed into the average shape, so
that the resulting image only carries shape-free appearance information. Another PCA is then performed on the shape-normalized training images, where the first 10 PCs for appearance (see Eqn 2) are retained. This results in a 20-dimensional latent face space. Similarly, for color images, we pre-train the AAM on 800 images from multiple views with given landmarks. Since the color face dataset contains more data, and more view information than the gray face dataset, we retain 50 PCs for the shape and another 50 PCs for the appearance which results in a 100-dimensional latent space. Every face has a corresponding AAM code denoted as $Z_{AAM}$, which encodes its shape and appearance variables. $Z_{AAM}$ has 20 dimension for grey-scaled images and 100 dimension for colored ones.

To generate face stimuli for our experiments, we randomly generate 20,000 grey-scaled face images of size $256 \times 256$ and 10,000 colored ones of size $128 \times 128$ from the above pre-trained AAMs. Specifically, for each dimension of the latent code, we record the standard deviation of the training responses of that dimension, and sample the variable from the Gaussian distribution with the same standard deviation as the real training faces. After that, these sampled variables are combined with the learned eigenvectors $P_s$ and $P_a$ to generate the synthesized images. The obtained images are then used as our training data for the generator network. Figure 2 shows some examples of training images to pre-train the AAM, and the synthesized face images generated by the trained AAM.

**VAE Training.** The training images obtained above are scaled so that the intensities are within the range $[-1, 1]$. No further pre-processing steps are needed.

For the generator network, we adopt the structure similar to [21, 9]. The network
Figure 2: Top and third row: training images with landmarks labeled for AAM. Second and bottom row: synthesized AAM images for training the generator network.

...consists of multiple deconvolution (a.k.a convolution-transpose) layers interleaved with ReLU non-linearity and batch normalization [14]. For grey-scaled images, we learn a 7 layer top-down convNet. The first deconvolutional layer has 512 filters with kernel size $4 \times 4$ and stride 1. There are 512, 384, 256, 128, 64, 1 filters with kernel size $4 \times 4$ and stride 2 for the following deconvolution layers respectively. For colored images, we learn a 6 layer top-down convNet where the first layer has 512 filters with kernel size $4 \times 4$ of stride 1, and the rest layers have 512, 256, 128, 64, 3 filters which have the same kernel size as the first layer but with stride 2. Each deconvolution layer is followed by ReLU non-linearity and batch normalization except the last deconvolution layer which is instead followed by the tanh non-linearity.

For the inference model or the encoder network of VAE, we utilize the mirror structure of the generator network (which is the decoder network) where we use convolutional layers instead of deconvolutional ones. Besides, we use the ReLU with leaky
factor 0.2 as our non-linearity. The mean and variance networks of the inference model share the same network structure except the top fully-connected layer. We also adopt the batch normalization in the inference model as in the generator network.

We tried different dimensionalities for the latent code $Z$, including 20, 100 and 200 dimensions for grey-scaled images as well as 100, 200 dimensions for colored ones. We used Adam optimizer [15] with initial learning rate 0.0002 for 500 iterations. The outputs of the mean network of the inference model are used as the learned latent code and are denoted as $Z_G$. Both the generator model and inference model can be well learned by VAE training which can be indicated by the accurate and faithful reconstruction (see Figure 3). The per-pixel $l_1$ reconstruction error is 0.005 for 20,000 gray scale images and 0.011 for 10,000 color images. The range of the pixel intensities is [0, 1]. Realistic synthesized images can also be generated by the trained generator network. See Figure 4 for some examples.

![Figure 3: Reconstruction of the VAE training. In each pair, left: training images. Right: reconstructed images after VAE training.](image)

We design four experiments to examine the relationship between the AAM code $Z_{AAM}$ for generating the face images and the code learned by the generator network, $Z_G$. 

14
5.2 Linear Relationship

[2] discovered that if a neuron has ramp-shaped tuning to different facial features, then its neural response can be approximated by a linear combination of the facial features. That is, the neural code for face patches ML/MF and AM has linear relationship with the AAM code of the presented face stimuli. In our first experiment, we check the strength of linearity between the code learned by the generator network and the underlying AAM code.

The codes for AAM, i.e., $Z_{AAM}$, are used to predict the corresponding codes learned by generator network, i.e., $Z_G$, and vice versa. Specifically, we fit linear model A and linear model B respectively:

\[ Z_G \approx AZ_{AAM}, \tag{8} \]

\[ Z_{AAM} \approx BZ_{G}. \tag{9} \]

We also include interception terms in both models. The goodness of fit of the model is
determined by the percentage of variance in data that is explained by the fitted linear model, i.e., the so-called R-square ($R^2$):

$$R^2 = 1 - \frac{\sum_i \|Z_i - \hat{Z}_i\|^2}{\sum_i \|Z_i - \bar{Z}\|^2},$$  \hspace{1cm} (10)

where $Z_i$ is the given code for image $i$, $\hat{Z}_i$ denotes the fitted value, and $\bar{Z}$ is the average of the code. Higher $R^2$ indicates stronger linear strength.

The $R^2$ values for different dimensionalities of $Z_G$ are shown in Table 1 and Table 2. It can be seen that models show strong linear relations on both grey-scaled and colored images. This is non-trivial and surprising, because when presented with only synthesized face stimuli, the VAE training of the highly non-linear generator network [19] can automatically learn the code that is linearly related to the underlying AAM code that generates the given face stimuli. That is, the learned generator network shares similar behavior as the face patch systems ML/MF and AM in the primate brain.

<table>
<thead>
<tr>
<th>dimension d for $Z_G$</th>
<th>d=20</th>
<th>d=100</th>
<th>d=200</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R^2$ (A)</td>
<td>0.9749</td>
<td>0.9332</td>
<td>0.9641</td>
</tr>
<tr>
<td>$R^2$ (B)</td>
<td>0.9728</td>
<td>0.9926</td>
<td>0.9951</td>
</tr>
</tbody>
</table>

Table 1: Strength of linearity ($R^2$) for models A and B on grey-scaled face images.

<table>
<thead>
<tr>
<th>dimension d for $Z_G$</th>
<th>d=100</th>
<th>d=200</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R^2$ (A)</td>
<td>0.6578</td>
<td>0.6627</td>
</tr>
<tr>
<td>$R^2$ (B)</td>
<td>0.7604</td>
<td>0.7983</td>
</tr>
</tbody>
</table>

Table 2: Strength of linearity ($R^2$) for models A and B on colored face images.
5.3 Decoding

As argued in [2], we should be able to linearly decode the facial features from the neural responses if there is a linear relationship between them. If so, we can accurately predict what the primate brain sees by knowing only the neural responses of the brain. Knowing that our learned code of the generator network $Z_G$ shows strong linear relationship with the facial features $Z_{AAM}$ from the above experiment, we expect that our automatically learned code $Z_G$ can accurately predict the facial features $Z_{AAM}$, which can then be used to reconstruct the input face image via the AAM. Therefore we further examine the decoding quality in this section.

To proceed, for training, we use $Z_{AAM}$ and $Z_G$ obtained during the learning process to fit model B as described above. Denote the estimated coefficients as $B^*$. To test the decoding quality, we carry out the following two steps: (1) randomly sample a new set of AAM generated face images, which are used as the testing set. Then use the trained encoder network, i.e., mean network, of VAE to get point estimate of latent code of the generator network, i.e., $Z_{G_{test}}$. (2) Use the optimal $B^*$ to get the predicted AAM code:

$$\hat{Z}^{test}_{AAM} = B^*Z_{G_{test}}. \quad (11)$$

The predicted AAM code is then projected onto the previously learned AAM eigenvectors $P_s$ and $P_a$ to get the reconstructed image.

For model fitting for $B^*$, we use the obtained 20,000 $Z_{AAM}$ during training for grey-scaled images and use 10,000 $Z_{AAM}$ for colored ones. 2,000 newly generated testing images are used for both cases. Figure 5 shows some testing images and the reconstructed ones. It can be seen that the linear model between the learned code by VAE
Figure 5: Left: test faces. Right: reconstructed faces using linear decoding. Top: results for grey-scaled face images. Bottom: results for colored images.

and the AAM code gives us high decoding quality. In this experiment as well as the subsequent experiments, we set the dimensionality of $Z_G$ to be 100. Other dimensionalities give similar results.
Figure 6: Normalized bar plot for $R^2$ values of shape and appearance. Each vertical bar corresponds to a dimension of the latent code.

5.4 Shape/Appearance Separation

The latent code $Z_G$ learned by the deep generative model is mixed with shape and appearance information. It would be useful to separate the shape and appearance parts of $Z_G$. In this experiment, we further identify the strengths of shape and appearance parts of the learned code $Z_G$.

From the first two experiments, we show that $Z_G$ is linearly related to the AAM code $Z_{AAM}$, which contains both the shape and appearance parts. We can identify these two parts by projecting each dimension of $Z_G$ onto the shape code $b_s$ and the appearance code $b_a$. We can then obtain the relative $R^2$ for each part. A higher $R^2$ for one part indicates the stronger response for this part. Recall that $Z_{AAM} = [b_s, b_a]$. We fit the
Figure 7: Scatter plot for $R^2$ values of shape and appearance. Each point corresponds to a dimension of the latent code. Red dashed line indicates the equal $R^2$ values for shape and appearance.

linear models on the shape part $b_s$ and the appearance part $b_a$ respectively:

$$Z_G \approx A_s b_s,$$

$$Z_G \approx A_a b_a.$$  \hspace{1cm} (12)

Figure 6 and 7 show the $R^2$ values for each dimension of $Z_G$. We only show plots for grey-scaled images here to illustrate the idea, note that the similar plots can be obtained for colored images as well. Specifically, Figure 6 indicates the normalized $R^2$ for shape and appearance parts, i.e., $\frac{R^2_s}{R^2_s + R^2_a}$ vs $\frac{R^2_a}{R^2_s + R^2_a}$, where $R^2_s$ and $R^2_a$ are $R^2$ values for shape and appearance respectively. It shows that each dimension of $Z_G$ responds
Figure 8: Vertical: appearance variation. Horizontal: shape variation.

differently to shape and appearance. To further verify and visualize our analysis, we first choose four dimensions with the top $R^2$ for shape and four dimensions with the top $R^2$ for appearance. Then we simultaneously traverse the selected 4 shape dimensions from $[-3, 3]$ ($\pm 3 \text{ sd}$) in equally spaced manner, denoted as $[v_1, ..., v_{10}]$. Similarly, for selected 4 appearance dimensions, we have $[v_1^a, ..., v_{10}^a]$. For un-selected dimensions, we use randomly but fixed values draw from $N(0, 1)$, denote them as $v_{fix}$. In order to show the shape change (e.g., horizontal $t$-th row in Figure 8), we traverse 4 shape dimensions from $v_1^s$ to $v_{10}^s$, fix the 4 selected appearance dimensions to be $v_t^a$, and keep the other dimensions to be $v_{fix}$. Appearance change is similar, except that we traverse
the 4 selected appearance dimensions but fix the shape dimensions to be corresponding values. It is clear that if we only vary the shape dimensions of the code (horizontally in the figure), the generated images mainly change their shapes while the appearances tend to remain similar. On the other hand, if we only vary the appearance dimensions of the code (vertically in the figure), the generated images mainly change their appearances instead of shapes. Similarly, for colored images, we also estimate the corresponding shape and appearance strength for each dimension of their latent factors $Z_G$. We select the dimensions which have relatively high $R^2$ value for appearance and low $R^2$ for shape, and vary such dimension by $\pm 3\, sd$ while keeping other dimensions fixed. The first two rows of Figure 9 show that such dimension mainly controls the appearance variations. For dimensions that have high $R^2$ value for shape but low $R^2$ for appearance mainly controls the shape variations. The last two rows of Figure 9 show such examples.

5.5 Replicating AAM by Supervised Learning

So far the generator network learns from the AAM in the unsupervised manner, where the generator network only has access to the training images but not the latent code of the AAM. We now examine whether the generator network has enough expressive power to replicate the AAM in the supervised setting where we also provide the latent code of the AAM to the generator.

In this experiment, the synthesized face images and their AAM codes are given, and we use these pairs to learn the generator network. To be more specific, we also consider two cases which include the grey-scaled images and more realistic colored images. We first train the generator network using the 20,000 grey-scaled images (or 10,000 colored
images) and their codes. Let us denote the trained generator as $G^\star$. Then, we prepare a new set of 2,000 synthesized images $Y_{test}$ and their AAM code $Z_{test}^{AAM}$ as our testing set. $Z_{test}^{AAM}$ is then fed into $G^\star$ to get $\hat{Y}_{test}$. If the generator network is capable of replicating the AAM, then $\hat{Y}_{test}$ should be close to $Y_{test}$, that is, the generated images by the trained generator network should be similar to the testing face images.

We use the same generator network structure as in the VAE training. We use Adam optimizer to train the generator network for supervised learning. The learning rate is 0.0002 with 800 epochs. Figure 10 shows the ground-truth testing images generated by the AAM and the reconstructed images generated by the trained generator network for both grey-scaled and colored face images. We also calculate the per-pixel $\ell_1$ reconstruction error, which is 0.0068 for grey-scaled images and 0.0420 for colored faces.
Figure 10: Left: test face images generated by AAM. Right: reconstructed face images by the generator network trained by supervised learning.

The range of the pixel intensities is [0, 1].

6 Conclusion

The recent work in neuroscience [2] shows that the face images can be reconstructed using the cell responses from face patches ML/MF and AM. To investigate whether
the widely used generator network has the similar property, we design and conduct experiments to examine the relationship between the AAM code that generates the face stimuli and the automatically learned code by the generator network. Through the linearity analysis and the decoding quality analysis, we find that the biological observations made in [2] can be qualitatively reproduced by the generator network, i.e., the learned code shows a strong linear relationship with the AAM code. Additionally, we can also use this relationship to further separate the shape and appearance parts of the learned code. Again this is similar to the neural system as it is found that ML/MF and AM carry complementary information about shape and appearance. Furthermore, we show that the generator network is capable of replicating AAM and we demonstrate this through supervised learning. In our current work, we consider neurons in ML/MF and AM as a whole and do not distinguish them. As pointed in [24], the AM face patches specifically have AM neuron rates that are exponential distribution and have the same mean for all face stimuli, as one of our future direction, we would like to study the relation between the learned latent codes and the specific areas of face patches to gain a deeper understanding of the generator model.

In this paper, we distill the knowledge of a pre-trained AAM to the generator network. It will also be interesting to distill the knowledge of a learned generator network to an AAM in order to interpret the generator network. The learned deep generator model, despite its powerful modeling ability, is in general a black-box and hard to interpret. While the AAM model is explainable in the sense that it has explicit shape and appearance variables. It would be interesting to learn a generator model with explicit variables for shape and appearance explicitly. A recent paper has explored this idea
[25] to disentangle the appearance and geometric knowledge by a deformable generator network in an unsupervised way. The basic rational is to use simple and explainable models to represent the complex and un-interpretable models based on which more reliable decisions and predictions can be made. We leave it to future work.

Acknowledgments

The work is supported by DARPA SIMPLEX N66001-15-C-4035, ONR MURI N00014-16-1-2007, DARPA ARO W911NF-16-1-0579, DARPA N66001-17-2-4029, Natural Science Foundation of China No. 61703119 and Natural Science Fund of Heilongjiang Province of China No. QC2017070.

References


