LETTER **Communicated by Charles Stevens**

Replicating Neuroscience Observations on ML/MF and AM Face Patches by Deep Generative Model

Tian Han
hantian@ucla.edu
Department of Statistics, University of California, Los Angeles, Los Angeles, CA 90095, U.S.A.

Xianglei Xing*
xingxl@hrbeu.edu.cn
College of Automation, Harbin Engineering University, Heilongjiang, China 150001

Jiawen Wu
jiawenwu@usc.edu
Department of Computer Science, University of Southern California, Los Angeles, CA 90089, U.S.A.

Ying Nian Wu
ywu@stat.ucla.edu
Department of Statistics, University of California, Los Angeles, Los Angeles, CA 90095, U.S.A.

A recent *Cell* paper (Chang & Tsao, 2017) reports an interesting discovery. For the face stimuli generated by a pretrained active appearance model (AAM), the responses of neurons in the areas of the primate brain that are responsible for face recognition exhibit a strong linear relationship with the shape variables and appearance variables of the AAM that generates the face stimuli. In this letter, we show that this behavior can be replicated by a deep generative model, the generator network, that 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 pretrained AAM model using a variational autoencoder, and we show that the inferred latent variables of the learned generator network have a strong linear relationship with the shape and appearance variables of the AAM model that generates the face images. Unlike the AAM model, which 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 it can learn the shape knowledge in the sense that some of

*Corresponding author.

Neural Computation **31, 2348–2367** (2019)
https://doi.org/10.1162/neco_a_01236

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* (Chang & Tsao, 2017) 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, this letter is concerned with how these neurons respond to and encode the face stimuli generated by a pretrained active appearance model (AAM) (Cootes, Edwards, & Taylor, 2001; Cootes, Roberts, Babalola, & Taylor, 2015). In AAM, explicit shape variables and appearance variables 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. Chang and Tsao (2017) discovered that the responses of the neurons to the face image generated by the AAM exhibit a strong linear relationship with the shape and appearance variables of the AAM that generate 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 letter, we investigate whether this phenomenon can be replicated by deep generative models. In particular, we focus on a popular deep generative model, the generator network (Goodfellow et al., 2014), which can be considered a nonlinear 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 nonlinear 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 and appearance variables that 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 nonlinear but known warping function of the control points and the nominal image. In contrast, the generator network is more generic; 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 pretrained 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 nonlinear mapping from the latent variables to the output image in the generator network accounts for the highly nonlinear 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 show that such a linear relationship indeed exists, thus qualitatively reproducing the behavior of the neuron responses (or neural code) observed by Chang and Tsao (2017).

The generator network can be trained by various methods, including the wake-sleep algorithm (Hinton, Dayan, Frey, & Neal, 1995), variational autoencoder (VAE; Kingma & Welling, 2014; Rezende, Mohamed, & Wierstra, 2014; Salimans, Kingma, & Welling, 2015), generative adversarial networks (GAN; Goodfellow et al., 2014; Radford, Metz, & Chintala, 2016; Denton, Chintala, Szlam, & Fergus, 2015), moment matching networks (Li, Swersky, & Zemel, 2015), alternating backpropagation (ABP; Han, Lu, Zhu, & Wu, 2017), and other related methods (Oord, Kalchbrenner, & Kavukcuoglu, 2016; Dinh, Sohl-Dickstein, & Bengio, 2016). They have led to impressive results in a wide range of applications, such as image/video synthesis (Dosovitskiy, Springenberg, & Brox, 2015), disentangled feature learning (Chen et al., 2016; Higgins et al., 2017), and pattern completion (Han et al., 2017).

In this letter, we 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. That network and the generator network form an autoencoder, 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 decodes 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 (Blei, Kucukelbir, & McAuliffe, 2017). 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.

This letter is phenomenological in nature. It is our hope that this work is of interest to both the neuroscience community and the deep learning community. This letter makes the following contributions:

- 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 behavior similar to that of the primate neural system.

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

2 Active Appearance Model

The active appearance model (AAA; Cootes et al., 2001, 2015) 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).

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

$$x = \bar{x} + P_s b_s, \tag{2.1}$$

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.

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 gray-level image. The appearance model is

$$g = \bar{g} + P_a b_a, \tag{2.2}$$

where $\bar{g}$ is the mean normalized gray-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 gray-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: $Z_{AAM} = [b_s, b_a]$. 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 nonlinear in g and x.

3 Generator Network

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

$$Z \sim \mathrm{N}(0, I_d), \tag{3.1}$$

$$Y = f_\theta(Z) + \epsilon_i, \tag{3.2}$$

where Z is the vector of latent variables (or latent code) and d is the dimension of Z, that is, 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 nonlinear mapping f_θ, which is modeled by a top-down convolutional neural network (ConvNet), where θ collects all the weight and bias parameters of the top-down ConvNet. ϵ 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 complex distribution and rich patterns because of the expressiveness of f_θ. The generator model, equation 3.1 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 equation 2.1 with control points x and shape variables b_s, nor does it have the explicit nonlinear warping function $Y = h(g, x)$. The generator network relies on the highly expressive ConvNet f_θ to account for the linear shape model and the nonlinear warping function. Although no prior knowledge of shape and warping is built into the generator network, it can learn such knowledge by itself.

Specifically, we use a pretrained 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 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 Autoencoder

Given a set of N training images $\{Y_i, i = 1, \ldots, N\}$ generated by AAM, we train the generator network by variational autoencoder (VAE; Kingma & Welling, 2014; Rezende et al., 2014; Salimans et al., 2015). 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, θ can be estimated by maximizing the log likelihood. However, this is intractable because $p_\theta(Z)$ involves an intractable integral. The EM algorithm (Dempster, Laird, & Rubin, 1977) 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 ϕ, such as a gaussian distribution with independent components $N(\mu_\phi(Y), \sigma_\phi^2(Y))$, where $\mu_\phi(Y)$ is the vector of means of the components of Z and $\sigma_\phi^2(Y)$ 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 (θ, ϕ) can be learned by jointly maximizing the variational lower bound of the log likelihood,

$$L(\theta, \phi) = \sum_{i=1}^{N} \Big[\log p_\theta(Y_i) - \mathrm{KL}(q_\phi(Z_i|Y_i)||p_\theta(Z_i|Y_i)) \Big], \tag{4.1}$$

where $\mathrm{KL}(q||p)$ 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 Kingma & Welling, 2013, for more details.) $q_\phi(Z|Y)$ is the encoder, and $p_\theta(Y|Z)$ is the decoder. After learning (θ, ϕ), 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 $\mathrm{KL}(q_{\text{data}}(Y)||p_\theta(Y))$ over θ. VAE is equivalent to minimizing

$$\begin{aligned} &\mathrm{KL}(q_{\text{data}}(Y)||p_\theta(Y)) + \mathrm{KL}(q_\phi(Z|Y)||p_\theta(Z|Y)) \\ &= \mathrm{KL}(q_{\text{data}}(Y)q_\phi(Z|Y)||p(Z)p_\theta(Y|Z)) \qquad (4.2) \\ &= \mathrm{KL}(q_\phi(Z, Y)||p_\theta(Z, Y)) \qquad (4.3) \end{aligned}$$

over both θ and ϕ, 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 $\mathrm{KL}(q_{\text{data}}(Y)||p_\theta(Y))$, which is the KL divergence between the marginal distributions, the variational objective function $\mathrm{KL}(q_\phi(Z, Y)||p_\theta(Z, Y))$ 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.

Let $Q = \{q_\phi(Z, Y), \forall\phi\}$ 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 $\mathrm{KL}(q||p)$ over Q and P. Such joint minimization can be accomplished by alternating projection, as illustrated in 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 $\mathrm{KL}(q||p)$ over $q \in Q$ to obtain q_1. Then we project $q = q_1$ onto P by minimizing $\mathrm{KL}(q||p)$ over $p \in P$ to obtain p_1, and so on. This process will converge to a local minimum of $\mathrm{KL}(q||p)$. In Figure 1, the two projections are illustrated by 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 (Hinton et al., 1995), which amounts to replacing minimizing $\mathrm{KL}(q||p)$ over $q \in Q$ by minimizing $\mathrm{KL}(p||q)$ over q by switching the order of p and q. The minimization

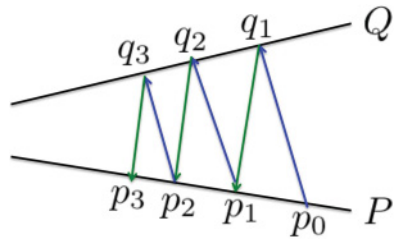

Figure 1: VAE as the joint minimization of KL($q||p$) over Q and P via alternating projection.

of KL($p||q$) can be accomplished by generating data from p in the sleep phase and learn q_ϕ from the generated data. Because of the switched order, the wake-sleep algorithm does not have a single objective function. However, in the wake-sleep algorithm, both projections are of the model-fitting type.

The generator network can also be trained by GAN (Goodfellow et al., 2014). 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 has access only to the images generated by the AAM. It does not have access to the shape and appearance variables (latent code) that the AAM uses 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.

5.1.1 Data Generation. We consider models on both gray-scaled and colored face images to show the generalizability of our work. For gray-scaled images, we pretrain 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 equation 2.1) are retained. The landmarks of each training image are then smoothly morphed into the average shape, so that the resulting image carries only shape-free appearance information. Another PCA is then performed on the shape-normalized training images, where the first 10 PCs for appearance (see equation 2.2) are retained. This

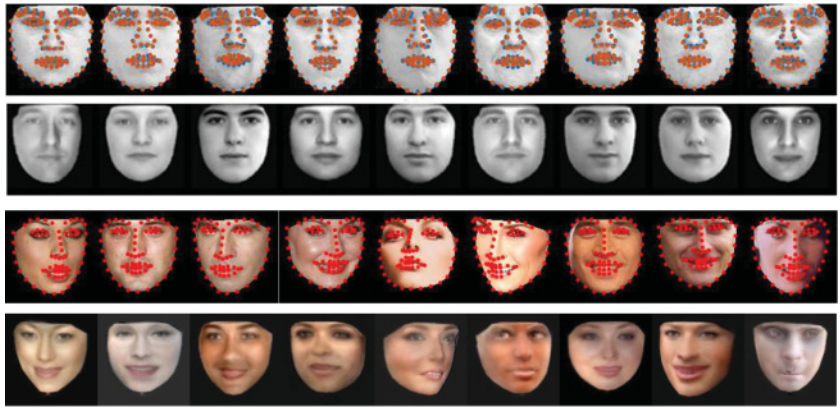

Figure 2: Top and third row: Training images with landmarks labeled for AAM. Second and bottom row: Synthesized AAM images for training the generator network.

results in a 20-dimensional latent face space. Similarly, for color images, we pretrain the AAM on 800 images from multiple views with given landmarks. Since the color face data set contains more data and more view information than the gray face data set, 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 dimensions for gray-scaled images and 100 dimensions for colored ones.

To generate face stimuli for our experiments, we randomly generate 20,000 gray-scaled face images of size 256×256 and 10,000 colored ones of size 128×128 from the above pretrained 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 pretrain the AAM and the synthesized face images generated by the trained AAM.

5.1.2 VAE Training. The training images obtained in Figure 2 are scaled so that the intensities are within the range $[-1, 1]$. No further preprocessing steps are needed.

For the generator network, we adopt the structure similar to Radford et al. (2016) and Dosovitskiy et al. (2015). The network consists of multiple deconvolution (a.k.a convolution-transpose) layers interleaved with ReLU

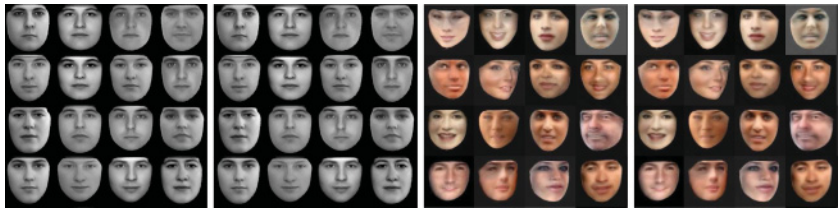

Figure 3: Reconstruction of the VAE training. In each pair, left: training images; right: reconstructed images after VAE training.

nonlinearity and batch normalization (Ioffe & Szegedy, 2015). For gray-scaled images, we learn a seven-layer top-down convNet. The first deconvolutional layer has 512 filters with kernel size 4×4 and stride 1. There are 512, 384, 256, 128, 64, and 1 filters with kernel size 4×4 and stride 2 for the following deconvolution layers, respectively. For colored images, we learn a six-layer top-down convNet, where the first layer has 512 filters with kernel size 4×4 of stride 1, and the rest of the 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 nonlinearity and batch normalization except the last deconvolution layer, which is instead followed by the tanh nonlinearity.

For the inference model or the encoder network of VAE, we use the mirror structure of the generator network (which is the decoder network) where we use convolutional layers instead of deconvolutional ones. We also use the ReLU with leaky factor 0.2 as our nonlinearity. 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 gray-scaled images as well as 100, 200 dimensions for colored ones. We used the Adam optimizer (Kingma & Ba, 2014) 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 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.)

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.

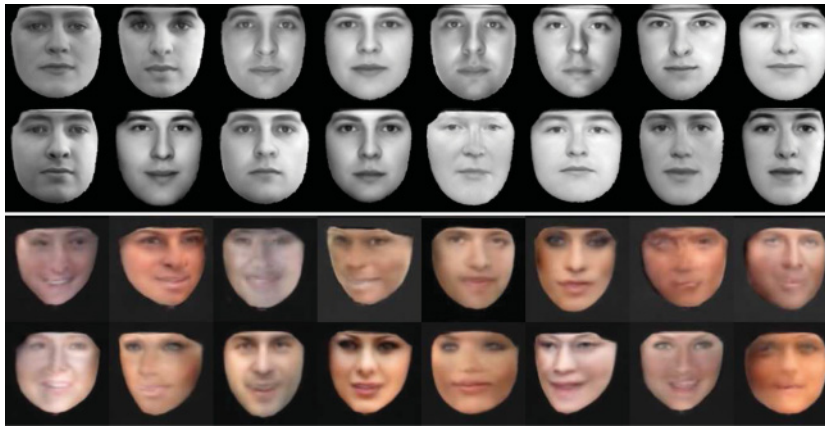

Figure 4: Synthesized images generated by the trained generator network.

5.2 Linear Relationship. Chang and Tsao (2017)) 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 a 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: Z_G, and vice versa. Specifically, we fit linear model A and linear model B, respectively:

$$Z_G \approx AZ_{AAM}, \tag{5.1}$$

$$Z_{AAM} \approx BZ_G. \tag{5.2}$$

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—the so-called R^2,

$$R^2 = 1 - \frac{\sum_i \|Z_i - \hat{Z}_i\|^2}{\sum_i \|Z_i - \bar{Z}\|^2}, \tag{5.3}$$

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 Tables 1 and 2. The models show strong linear relations on both gray-scaled and colored images. This is nontrivial and surprising, because when presented

Table 1: Strength of Linearity (R^2) for Models A and B on Gray-Scaled Face Images.

Dimension d for Z_G	$d = 20$	$d = 100$	$d = 200$
R^2 (A)	0.9749	0.9332	0.9641
R^2 (B)	0.9728	0.9926	0.9951

Table 2: Strength of Linearity (R^2) for Models A and B on Colored Face Images.

Dimension d for Z_G	$d = 100$	$d = 200$
R^2 (A)	0.6578	0.6627
R^2 (B)	0.7604	0.7983

with only synthesized face stimuli, the VAE training of the highly nonlinear generator network (Montufar, Pascanu, Cho, & Bengio, 2014) 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 behavior similar to that of the face patch systems ML/MF and AM in the primate brain.

5.3 Decoding. As Chang and Tsao (2017) argued, 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. Denote the estimated coefficients as $B^\star$. To test the decoding quality, we carry out the following two steps. We first randomly sample a new set of AAM-generated face images, which are used as the testing set. We then use the trained encoder network (i.e., mean network) of VAE to get a point estimate of the latent code of the generator network (i.e., Z_G^{test}). Second, we use the optimal $B^\star$ to get the predicted AAM code:

$$\hat{Z}_{AAM}^{test} = B^\star Z_G^{test}. \tag{5.4}$$

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

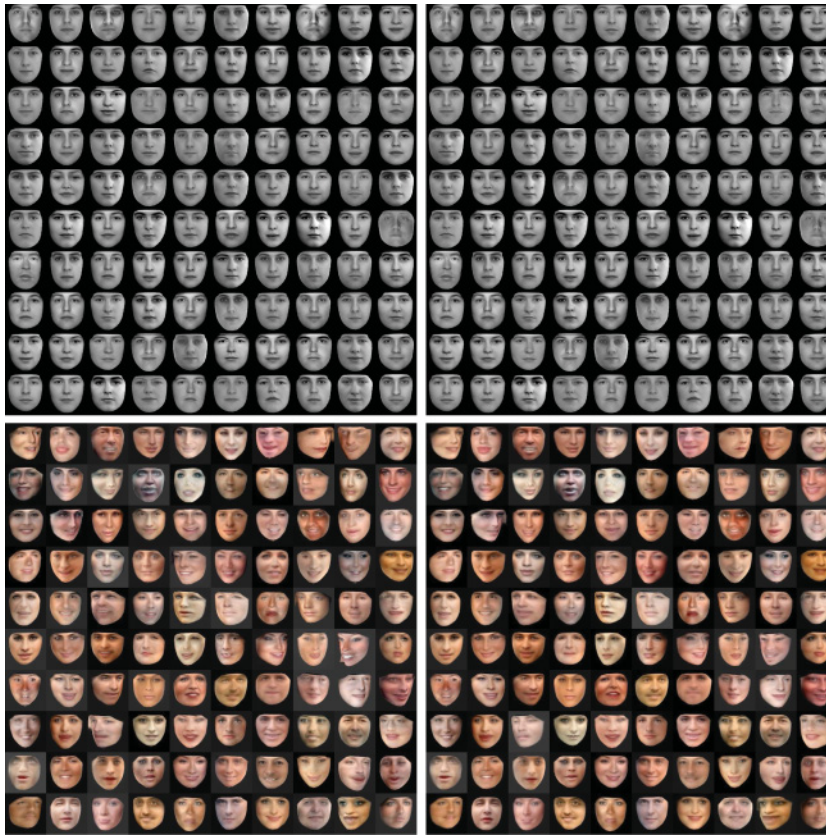

Figure 5: (Left) Test faces. (Right) Reconstructed faces using linear decoding. (Top) Results for gray-scaled face images. (Bottom) Results for colored images.

For model fitting for $B^\star$, we use the obtained 20,000 Z_{AAM} during training for gray-scaled images and use 10,000 Z_{AAM} for colored ones. Two thousand 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 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.

5.4 Shape and 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

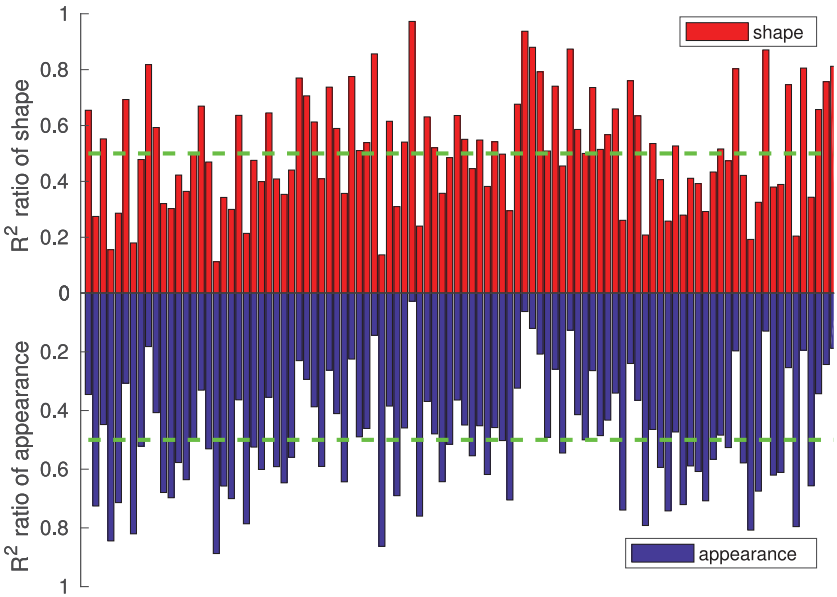

Figure 6: Normalized bar plot for R^2 values of shape and appearance. Each vertical bar corresponds to a dimension of the latent code.

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

$$Z_G \approx A_s b_s, \tag{5.5}$$

$$Z_G \approx A_a b_a. \tag{5.6}$$

Figures 6 and 7 show the R^2 values for each dimension of Z_G. We show only the plots for gray-scaled images here to illustrate the idea; similar plots can be obtained for colored images as well. Specifically, Figure 6 indicates the normalized R^2 for shape and appearance parts, $\frac{R_s^2}{R_s^2+R_a^2}$ versus $\frac{R_a^2}{R_s^2+R_a^2}$, where R_s^2 and R_a^2 are R^2 values for shape and appearance, respectively. It shows that each dimension of Z_G responds differently to shape and appearance. To further verify and visualize our analysis, we first choose four

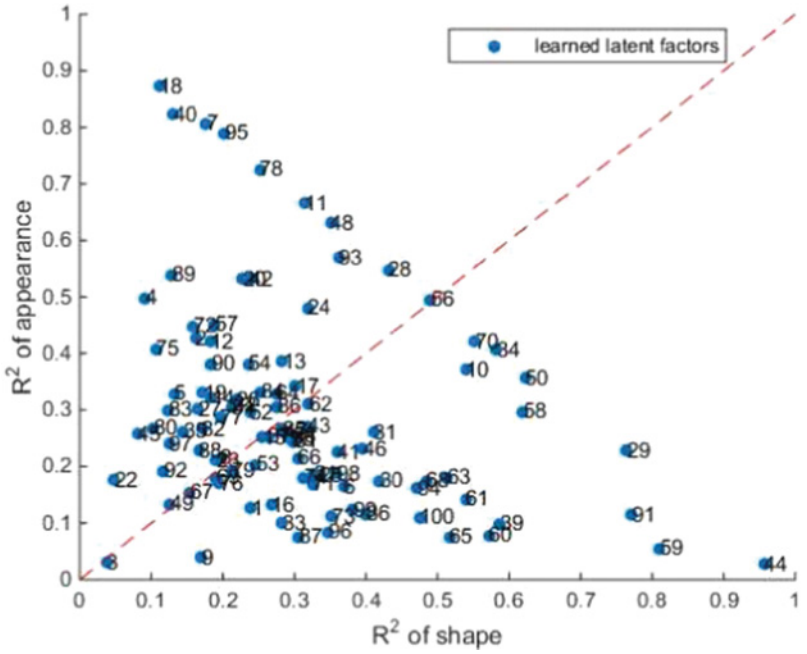

Figure 7: Scatter plot for R^2 values of shape and appearance. Each point corresponds to a dimension of the latent code. The red dashed line indicates the equal R^2 values for shape and appearance.

dimensions with the top R^2 for shape and four dimensions with the top R^2 for appearance. Then we simultaneously traverse the selected four shape dimensions from $[-3, 3]$ (± 3 SD) in an equally spaced manner, denoted as $[v_1^s, \ldots, v_{10}^s]$. Similarly, for the selected four appearance dimensions, we have $[v_1^a, \ldots, v_{10}^a]$. For unselected dimensions, we use randomly but fixed values drawn from $N(0, 1)$ and denote them as v_{fix}. In order to show the shape change (e.g., horizontal tth row in Figure 8), we traverse four shape dimensions from v_1^s to v_{10}^s, fix the four selected appearance dimensions to be v_t^a, and keep the other dimensions to be v_{fix}. The appearance change is similar, except that we traverse the four 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. If we vary only 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 estimate the corresponding shape and appearance strength for each dimension of their latent factors Z_G. We select the dimensions that have a relatively high

Figure 8: Vertical: Appearance variation. Horizontal: Shape variation.

R^2 value for appearance and a low R^2 for shape and vary the dimension by ± 3 SD while keeping other dimensions fixed. The first two rows of Figure 9 show that the dimension mainly controls the appearance variations. Dimensions that have a high R^2 value for shape but a low R^2 for appearance mainly control the shape variations The last two rows of Figure 9 show examples.

5.5 Replicating AAM by Supervised Learning. So far the generator network learns from the AAM in an unsupervised manner, where the generator network has access only to the training images, 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.

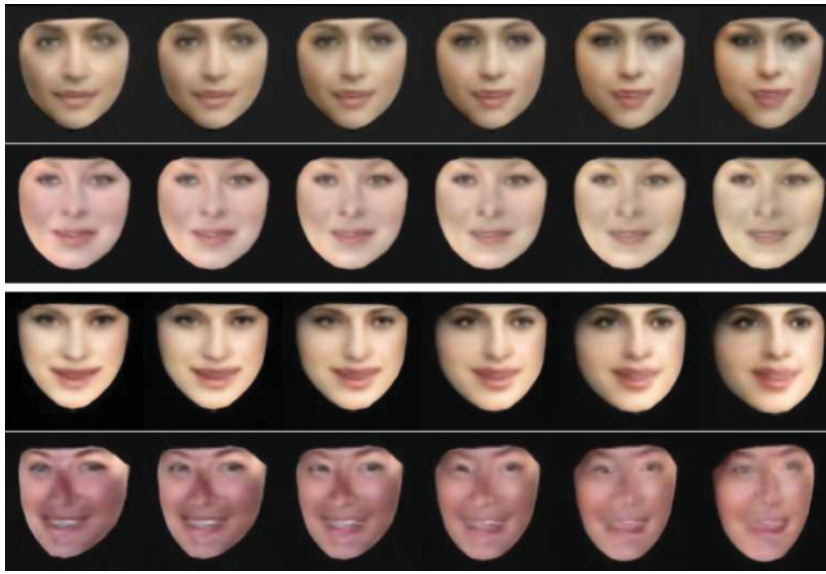

Figure 9: Top two rows: Appearance variations (first row: identity and expression changes; second row: color changes). Bottom two rows: shape variations (third row: the width of the face changes; fourth row: view points change.)

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 that include the gray-scaled images and more realistic colored images. We first train the generator network using the 20,000 gray-scaled images (or 10,000 colored images) and their codes. We 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 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 the 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 gray-scaled and colored face images. We also calculate the per pixel ℓ_1 reconstruction error, which is 0.0068 for gray-scaled images and 0.0420 for colored faces. The range of the pixel intensities is [0, 1].

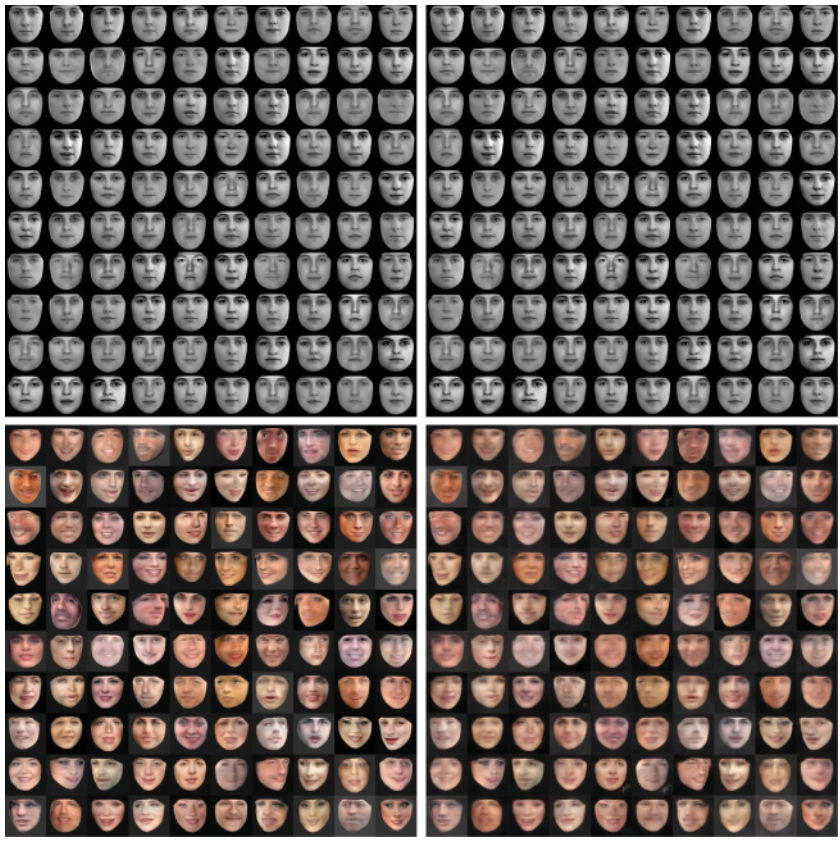

Figure 10: Left: Test face images generated by AAM. Right: Reconstructed face images by the generator network trained by supervised learning.

6 Conclusion

The recent work in neuroscience by Chang and Tsao (2017) shows that 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 a 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 of Chang and Tsao (2017) can be qualitatively reproduced by the generator network; the learned code shows a strong linear relationship with the AAM code. 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 Stevens (2018) pointed out, 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 directions, 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 letter, we distill the knowledge of a pretrained 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 (Xing, Han, Gao, Zhu, & Wu, 2019) to disentangle the appearance and geometric knowledge by a deformable generator network in an unsupervised way. The basic rationale is to use simple and explainable models to represent the complex and uninterpretable 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 61703119, Natural Science Fund of Heilongjiang Province of China QC2017070, and Fundamental Research Funds for the Central Universities 3072019CFT0402.

References

Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, *112*(518), 859–877.

Chang, L., & Tsao, D. Y. (2017). The code for facial identity in the primate brain. *Cell*, *169*(6), 1013–1028.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems, 29* (pp. 2172–2180). Red Hook, NY: Curran.

Cootes, T. F., Edwards, G. J., & Taylor, C. J. (2001). Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(6), 681–685.

Cootes, T., Roberts, M., Babalola, K., & Taylor, C. (2015). Active shape and appearance models. In N. Paragios, D. Nikos, & N. Ayache (Eds.), *Handbook of biomedical imaging* (pp. 105–122). Berlin: Springer.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: B*, *39*, 1–38.

Denton, E. L., Chintala, S., Szlam, A., & Fergus, R. (2015). Deep generative image models using a Laplacian pyramid of adversarial networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems*, *28* (pp. 1486–1494). Red Hook, NY: Curran.

Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2016). Density estimation using real NVP. *CoRR*, abs/1605.08803.

Dosovitskiy, E., Springenberg, J. T., & Brox, T. (2015). Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *27* (pp. 2672–2680). Red Hook, NY: Curran.

Han, T., Lu, Y., Zhu, S.-C., & Wu, Y. N. (2017). Alternating back-propagation for generator network. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. Palo Alto: AAAT Press.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., . . . Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, *2*(5), 6.

Hinton, G. E., Dayan, P., Frey, B. J., & Neal, R. M. (1995). The "wake-sleep" algorithm for unsupervised neural networks. *Science*, *268*(5214), 1158–1161.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*.

Kingma, D., & Ba, J. (2014). *Adam: A method for stochastic optimization*. arXiv:1412.6980.

Kingma, D. P., & Welling, M. (2013). *Auto-encoding variational Bayes*. arXiv:1312.6114.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In *ICLR*.

Li, Y., Swersky, K., & Zemel, R. (2015). Generative moment matching networks. In *Proceedings of the 32nd International Conference on Machine Learning* (pp. 1718–1727).

Montufar, G. F., Pascanu, R., Cho, K., & Bengio, Y. (2014). On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Weinberger (Eds.), *Advances in neural information processing systems*, *27* (pp. 2924–2932). Red Hook, NY: Curran.

Oord, A. V. d., Kalchbrenner, N., & Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *Proceedings of the 33rd International Conference on Machine Learning* (pp. 1747–1756).

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*.

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In T. Jebara & E. P. Xing (Eds.), *Proceedings of the International Conference on Machine Learning* (pp. 1278–1286).

Salimans, T., Kingma, D., & Welling, M. (2015). Markov chain Monte Carlo and variational inference: Bridging the gap. In *Proceedings of the 32nd International Conference on Machine Learning* (pp. 1218–1226).

Stevens, C. F. (2018). Conserved features of the primate face code. *Proceedings of the National Academy of Sciences*, *115*(3), 584–588.

Xing, X., Han, T., Gao, R., Zhu, S.-C., & Wu, Y. N. (2019). Unsupervised disentangling of appearance and geometry by deformable generator network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 10354–10363). Piscataway, NJ: IEEE.

Received February 14, 2019; accepted July 20, 2019.