HumanGAN for Human Faces

Lanston Hau Man Chu*
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, WI 53706
hchu34@wisc.edu

Abstract

In this paper, we applied the HumanGAN to synthesize human facial images. Therefore, human participants would replace the discriminator in GAN after a regular GAN training of a specific number of iterations. The techniques of **Uneven NES** and **Iterations-Forking Analysis** would be used to enhance the process of **after-human gradient descent (AHGD)**.

1 Introduction

Since the launch of Generative Adversarial Nets (GAN) by Goodfellow et. al. in 2014 [3], the architecture succeeds in data synthesis and data reconstruction. GAN has been used in different domains, from synthesizing low dimensional data such as speech, photo, painting, to reconstructing high dimensional data such as 3D IKEA furniture [11].

Different variations of GAN are created for different usages and areas, such as domain swapping of photos and paintings by CycleGAN [13], or generating malware to attack a targeted black-box neural network by MalGAN [4].

The two main components of GAN are generator G and discriminator D, which are also neural networks. In a study of data synthesis, Fujii et. al. [2] have replaced the discriminator in GAN by human participants and call the new architecture HumanGAN. They succeed to train the generator to synthesize more realistic syllables, which reside in the formants space of $\mathbb{R}^2$ and can be considered as low dimensional data.

In the meantime, for high dimensional data, there is a bottleneck in synthesizing more realistic output. For example, a recent project of 3D reconstruction of human chest by 2D X-ray images via neural network [6] finds that the reconstruction output is not realistic enough.

In view of this, Fujii et. al.'s approach of human involvement would be a way to improve the synthesis/reconstruction of high dimensional data. The problem becomes how effective the approach would be and how the setting and implementation should be when the dimensions of data increase. Human faces is a domain that can be considered as either $\mathbb{R}^{H\times W}$ (when considered as pixels of a grey image) or $\mathbb{R}^q$ (when considered as a composition of q eigenfaces). This project would extend the method of HumanGAN to the domain of human facial images, so as to pave the way for future studies of high dimensional reconstruction.

In other words, this is an exploratory project on how to enhance the human-involvement-in-GAN approach on synthesizing high dimensional data. To achieve the goal, this paper suggests the technique of **Uneven NES** and **Iterations-Forking Analysis**. We call the process of doing gradient descent in this manner the **after-human gradient descent (AHGD)**.

*Source Code: https://github.com/lanstonchu/HumanGAN-Faces

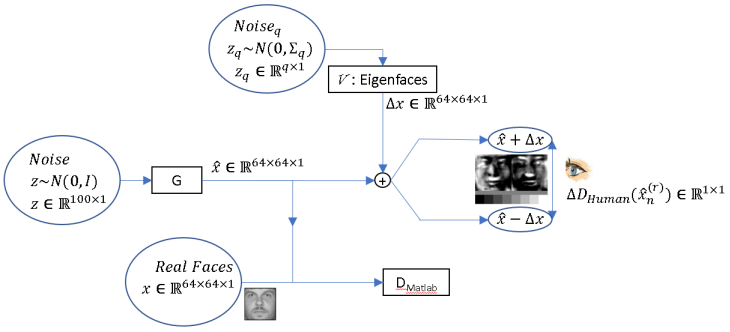

Figure 1: Architecture of HumanGAN.

1.1 Related Work

While Fujii et. al added stochastic noises to the generator's outputs and query human on a Monte Carlo manner, there are some alternative approach that query human directly on the generator's output. Periyasamy (2018) [7] extends the GAN by replacing the generative model with a human in the domain of the rectangle concept.

In some deep learning architectures other than GAN, some attempts were made to replace certain components of the network by humans. Wallace et. al. [9] ask human authors to write Quizbowl questions with interaction in RNN/DAN and successfully generate adversarial questions to beat the latest QA system.

2 Model Architecture and set up

2.1 Dataset

Human faces were acquired from the database LFWcrop [8]. In view of the scale of this exploratory project, we randomly picked a small portion of 587 images from the entire data set. The faces of these 587 images face squarely at the camera and those that are not would be excluded in the sampling process. We would not pick more than 1 image of a single person.

The faces from LFWcrop are grey scale images of 64×64 pixels, and thus we would regard an image x to belong to $X = \mathbb{R}^{64\times64\times1}$.

2.2 Generator and Discriminator

The generator G and discriminator D are the main components of GAN. They are both artificial neural networks in Training Phase, and the discriminator D_{Matlab} will be replaced by human participants D_{Human} in the Human Participants Phase. In this paper, D mostly refers to D_{Human}, but sometimes it refers to either D_{Matlab} or D_{Human}, if specifically mentioned.

The generator is a neural network with 5 layers of transpose convolutional layer which will expand 1 pixel into a 64×64 image. Thus the input $z \in \mathbb{R}^{100\times1}$ (i.e. 100 "one-pixel") will be expanded to $\hat{x} = G(z) \in X = \mathbb{R}^{64\times64\times1}$.

In the Training Phase, the discriminator D_{Matlab} is a 5-layer CNN (convolutional neural network) which would squeeze the 64×64 image x into a scalar $D_{Matlab}(x) \in [0, 1]$. Note that $D(x) \approx 1$ means that D (which can be D_{Matlab} or D_{Human}) classifies x as "very likely to be a real image", while $D(x) \approx 1$ would be at the opposite.

Same as the objective function of the original GAN in [3], the objective functions of G and D are as below:

$$D : \arg\max_D \mathbb{E}_x[\,logD(x)] + \mathbb{E}_z[\,log(1 - D(G(z)))] \tag{1}$$

$$G : \arg\min_G \mathbb{E}_z[\, log(1 - D(G(z)))] \tag{2}$$

Please note that D in (1) refers to D_{Matlab}, while D in (2) can refer to either D_{Matlab} and D_{Human}.

In the Human Participants Phase, the discriminator D will be the human participants D_{Human}. This will be explained further in the Human Participants Phase section.

2.3 Eigenfaces

The 587 real faces will be used in two places: The training process in the Training Phase, and to create the eigenfaces being used in the Human Participants Phase.

To create the eigenfaces, we will apply PCA (i.e. Principal component analysis) to the faces in the manner of [12] to achieve Eigenfaces $\mathcal{V}$ and Eigenvalues $\mathcal{D}$. By picking the top q eigenfaces of the PCA, $\mathcal{V}$ and $\mathcal{D}$ would be a $64^2 \times q$ and $q \times q$ matrix.

$$PCA(587 \text{ Real Faces}) \xrightarrow{\text{dim: } q=100} \mathcal{V} \in \mathbb{R}^{64^2 \times q} \text{ and } \mathcal{D} \in \mathbb{R}^{q \times q} \tag{3}$$

Multiplying $\mathcal{V}$ with a noise z_q (as the weight of the eigenfaces) will produce a "liminal face" as below

$$\Delta x = \Delta = \mathcal{V} z_q \tag{4}$$

For simplicity we also write Δx as Δ. For the value of q, we picked $q = 100$ which is same as the size of the noise z that would be input into the generator G to produce face images.

Note that mathematically it is not necessary to use the same size for z and z_q but we picked the same size as 100 dimensions, as philosophically it is more reasonable to embed same type of data into spaces of the same dimension.

3 Theory

3.1 Objective function with human involvement

When the discriminator are human participants in the Human Participants Phase, N images would be generated from N noises z, and the objective function for G would be set as below, while G would maximize V:

$$V = \sum_{n=1}^{N} D(G(z_n)) \tag{5}$$

Therefore, the gradient descent for θ_G (i.e. parameters of G) would be implemented by computing the gradient $\frac{\partial V}{\partial \theta_G}$, which can be decomposed as:

$$\frac{\partial V}{\partial \theta_G} = \sum_{n=1}^{N} (\frac{\partial V}{\partial \hat{x}_n})^T \cdot \frac{\partial \hat{x}_n}{\partial \theta_G} \tag{6}$$

As we will see, the term $\frac{\partial V}{\partial \hat{x}_n}$ will be obtained by implementing NES with the data collected from the human responses, while $\frac{\partial \hat{x}_n}{\partial \theta_G}$ will be obtained from backpropagation of G.

3.2 NES

NES (Natural Evolutionary Strategies) [1][10] would be used to estimate $\frac{\partial V}{\partial \hat{x}_n}$.

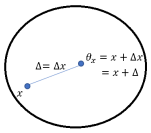

Figure 2: x, Δx and θ_X of the NES algorithm in the space X.

In the space X where $x \in X$, for a function $F : X \to \mathbb{R}$ we can estimated the gradient of $F(X)$ by using the NES algorithm in a Monte Carlo manner:

$$\nabla_x F(x) = \nabla_x \mathbb{E}_{\pi(\theta_X|x)}[F(\theta_X)] = \mathbb{E}_{\pi(\theta_X|x)}[F(\theta_X)\nabla_x log(\pi(\theta_X|x))] \tag{7}$$

where we can sample Δx from a specific distribution and obtain $\theta_X = x + \Delta x$, and we write the pdf (probability density function) of θ_X as $\pi(\theta_X|x)$. For simplicity, we also write Δx as Δ and thus we have $\theta_X = x + \Delta$.

Fujii et. al. [2] picked $\Delta \sim N(0, \sigma^2 I)$, in which case each dimension are independent to each other and have even weight. We called this approach **Even NES**.

In this paper, we would not use Even NES, since by doing so Δ will most probably to be white noise and thus $X + \Delta$ and $X - \Delta$ may not have significant difference in human sense. We will therefore use the technique of **Uneven NES** by using eigenfaces, so as to create effective image pairs and to speed up the human training process.

3.3 Uneven NES

Uneven NES with eigenfaces is the technique put forth by this paper. In (4), we have mentioned that $\Delta = \Delta x = \mathcal{V} z_q$. Now we pick a Gaussian distribution with a covariance matrix $\Sigma_q = \sigma^2 \mathcal{D} \in \mathbb{R}^{q \times q}$ for the distribution of z_q, where $\sigma \in \mathbb{R}$ is a scalar:

$$z_q \sim N(0, \Sigma_q) = N(0, \sigma^2 \mathcal{D}) \tag{8}$$

Note that $\mathcal{D}$ instead of I is used as the covariance matrix for z_q, since we hope we can have a more significant fluctuation for a more significant eigenface. Therefore $\Delta = \mathcal{V} z_q$ will give us the followings:

$$\Delta \sim N(0, \mathcal{V}\Sigma_q \mathcal{V}^T) = N(0, \sigma^2 \mathcal{V}\mathcal{D}\mathcal{V}^T) \tag{9}$$

We write $\Sigma = \mathcal{V}\Sigma_q \mathcal{V}^T$ (with size $64^2 \times 64^2$). Therefore we have:

$$\theta_X = x + \Delta \sim N(x, \Sigma) \tag{10}$$

Generally, Σ would not be a multiple of an identity matrix, and thus we called this approach **Uneven NES**. It is also worth to point out that it is not necessary to pick $\Sigma = \mathcal{V}\Sigma_q \mathcal{V}^T$ to make (12) below hold. We can pick any Σ to make the equation hold but we expect a more effective convergence by selecting $\Delta = \mathcal{V} z_q$.

From (10), for the multivariate Gaussian variable θ_X we have its pdf (probability density function) as below:

$$\pi(\theta_X|x) = \frac{1}{(2\pi)^{\frac{64^2}{2}} \sqrt{det(\Sigma)}} exp\Big(\frac{-1}{2}(\theta_X - x)^T \Sigma^{-1} (\theta_X - x)\Big) \tag{11}$$

The above leads to (12) (please refer to the Appendix for the proof), which now repeats as below:

$$\frac{\partial V}{\partial \hat{x}_n} \approx \frac{\Sigma^{-1}}{2R} \sum_{r=1}^{R} \big(\Delta D^{(r)}(\hat{x}_n)\big) \Delta_n^{(r)} \tag{12}$$

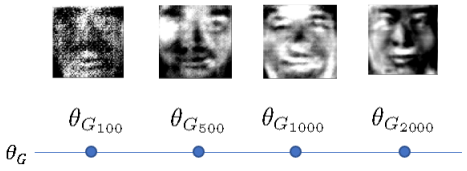

Figure 3: Output images of G_i after training of i iterations.

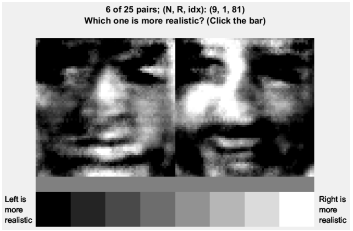

Figure 4: Participants interface for a face pair at iteration 1000. The images pair are produced by the **Uneven NES** technique to ensure significant difference in appearance.

In the Human Participant Phase we would collect data for $\Delta D^{(r)}(\hat{x}_n)$. From (9) we already have Σ, in the stochastic setting we have R, and in the sampling process we obtained $\Delta_n^{(r)}$. Therefore we have all necessary ingredients and will obtain $\frac{\partial V}{\partial \hat{x}_n}$.

As the model architecture of G is already defined in the setting, we can simply do backpropagation to obtain $\frac{\partial \hat{x}_n}{\partial \theta_G}$. Therefore we already have all ingredients of (6) to compute the gradient $\frac{\partial V}{\partial \theta_G}$ and therefore we can proceed AHGD by $\theta_G \leftarrow\!\shortmid \theta_G - \lambda \frac{\partial V}{\partial \theta_G}$.

4 Training Process and Human Participation

Since human responses are expensive and the output $\hat{x}$ would look like white noises in the early iterations, this project would undergo two phases, namely the **Training Phase** and **Human Participants Phase**, instead of doing Human Participants Phase only as what Fujii et. al. did.

4.1 Phase I: Training Phase

In the Training Phase (i.e. Phase I), we started training the GAN for 4000 iterations in regular GAN manner (computation would be done in *Matlab*) with her two components G and D_{Matlab} without human involvement, with respect to the objective functions (1) and (2).

The generator G will synthesize more realistic human faces $\hat{x}$ during the training. For the respective output images of G_i after the training of i iterations, please refer to Figure 3.

4.2 Phase II: Human Participants Phase

In the Human Participants Phase (i.e. Phase II), twenty five (i.e. $N = 25$) noises z will be generated. The values of these twenty five random z would be fixed for different G_i to produce $\hat{x}$, where i is the i-th iteration.

Again, human responses are expensive and a large number of iterations are required for the generator training to synthesize significant faces. We would not train the generator G back and forth as what Fujii et. al. did (they got significant convergence in the formants space $\mathbb{R}^2$ by using only 5 iterations). Instead, we picked some specific iteration $i = 100, 500, 1000$ and 2000 and the respective generator G_i would be the main objects for us to study.

Twenty five $\hat{x} = G_i(z)$ would be obtained, and each $\hat{x}_n$ (i.e. $n = 1, ..., N$) will be added to and subtracted from ten (i.e. $R = 10$) different perturbations $\Delta x_n^{(r)}$ (i.e. $r = 1, ..., R$) to achieve $\hat{x}_n \pm \Delta x_n^{(r)}$, which form two facial images. The participants would determine which facial image is more realistic by clicking the 8-box color bar, which will return a value in $[-1, 1]$. This value refers to $\Delta D^{(r)}(\hat{x}_n) = D(\hat{x}_n + \Delta x_n^{(r)}) - D(\hat{x}_n - \Delta x_n^{(r)})$. There are ten participants in total, and they need to answer $\frac{NR}{10}$ shuffled queries for a specific iteration i, and then move to next $\frac{NR}{10}$ queries for another iteration i', until all chosen iterations are covered. The participants' responses are collected via the *ginput()* function of *Matlab*.

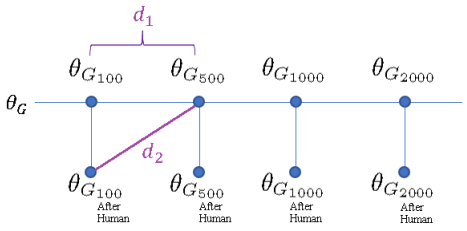

Figure 5: Iterations-Forking Analysis

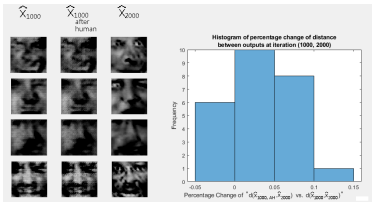

Figure 6: Analysis for $G_{1000,\,\mathrm{AH}}$. Each row of image is generated by the same z

As mentioned in the NES section, the perturbation $\Delta x_n^{(r)}$ is the main ingredient to implement the NES [2] algorithm, which is a Monte Carlo approach to estimate the gradient $\frac{\partial V}{\partial \theta_G}$ for doing AHGD for G (or the parameters θ_G).

4.3 Iterations-Forking Analysis

As mentioned above, human responses are expensive and we would pick the generators being trained after a specific number of iterations for analysis, instead of doing back and forth training for all iterations.

For each iteration i, say $i = 100$ and G_{100}, we would do **after-human gradient descent (AHGD)** $\theta_G \leftarrow\!\shortmid\, \theta_G - \lambda \frac{\partial V}{\partial \theta_G}$ for 1 iteration, and achieved $G_{100,\,\mathrm{AH}}$ (i.e. AH stands for "After Human"). Therefore we would obtain the "AH" version of G for the iterations 100, 500, 1000 and 2000. Therefore we have created four "forks" for specific iteration i.

We would like to know whether the "distance" d_2 between $\theta_{G_{100,\,\mathrm{AH}}}$ and $\theta_{G_{500}}$ would be smaller than the "distance" d_1 between $\theta_{G_{100}}$ and $\theta_{G_{500}}$.

The distance here can be evaluated by a different approach. As a preliminary result of the project, the distance d_1 and d_2 in this paper would be computed based on the pixelwise Euclidean distance between output of the generators given the same set of noises z.

5 Result

Currently, the preliminary result is not as expected. We expect d_2 to generally be smaller than d_1, which is not the case in the preliminary result. Taking $G_i = G_{1000,\,\mathrm{AH}}$ as example, for each z we have $d_1 = d(\hat{x}_{1000}, \hat{x}_{2000})$ and $d_2 = d(\hat{x}_{1000,\,\mathrm{AH}}, \hat{x}_{2000})$, while $d(\cdot)$ is the pixelwise Euclidean distance between two images.

Within the $N = 25$ noises, we noted that most (i.e. 76% of the twenty five noises) of the ratio $\frac{d_2 - d_1}{d_1}$ are positive (i.e. Figure 6), while we expect them to be generally negative.

If we looked into the images, it seems that $\hat{x}_{1000,\,\mathrm{AH}}$ are adding regular pattern to the image $\hat{x}_{1000}$, instead of doing changes with semantic meaning as what $\hat{x}_{2000}$ does (e.g. the "eye-openers" of $\hat{x}_{1000}$ in Figure 6).

The results are similar for other iterations, namely $G_{100,\,\mathrm{AH}}$, $G_{500,\,\mathrm{AH}}$ and $G_{2000,\,\mathrm{AH}}$.

6 Conclusion

We expect d_2 to be generally smaller than d_1, but currently in the preliminary result it is not the case. There are several possible reasons for not getting the expected result. Further investigation needs to be made to determine the significance of the possible reasons that this paper will discuss below.

6.1 Implementation Limitation

The choice of $N = 25$ and $R = 10$ seem to be too small. While the dimension of an image $\hat{x}$ is $64^2 = 4096$, we only employ $25 \times 10 = 250$ queries to perform the AHGD for each iteration. Even if we use the main eigenspace instead of pixelwise space, typically the space size would still be around several hundreds dimensions. 250 seems a small number to have an effective gradient descent. Regarding how to adjust the numbers N and R, further statistical study, especially the 1-step iteration-forking analysis (e.g. G_{1000} vs. $G_{1000,\,\mathrm{AH}}$ vs. G_{1001}), needs to be carried out.

Besides, the matrix Σ^{-1} may be numerically unstable to compute as $q = 100$ we picked is much smaller than 64^2 and thus Σ is numerically closed to a singular matrix. We have computed Σ^{-1} given Σ in three different softwares, namely *Matlab*, *Mathematica* and *Numpy* of Python. We found that the Σ^{-1} of *Mathematica* and *Numpy* match with each other, but that of *Matlab* would reach some unreasonable magnitude. Currently the numerical values of the Σ^{-1} being used in this project is computed by *Numpy*. In view of the above, we cannot ignore the possibility that the unstable status of Σ^{-1} would lead to some unreliable result of (12). Further numerical tests need to be carried out on the current Σ, as well as some other Σ with different value of q.

Due to the exploratory feature of this project, we had only used 587 real faces in the Training Phase and to get the eigenfaces $\mathcal{V}$. There are totally 5749 persons in the LFWcrop data set, and therefore by picking one facial image from each of them we can use ten times more facial images for our study to enhance better result.

6.2 Theoretical Enhancement

It is also possible that we have not gotten the expected result due to the current design of the objective function.

First, as some images $\hat{x}$ may seem less meaningful than the other images, e.g. much darker than the other images, we may achieve more effective AHGD by replacing the objective function "$V = \sum_{n=1}^{N} D(G(z_n))$" by "$V = \sum_{n=1}^{N} w_n \cdot D(G(z_n))$", where $w_n = w_n(G(z_n))$ is the weight for an image with specific features.

Second, $D(G(z_n))$ was designed as 8-box color bar as we want to force the participants to choose a value, but some participants feedback that in many cases they could not make a definite decision on which box to click. Perhaps the design of eight boxes are too "discrete" and a continuous color bar may be a better design for this purpose.

We can even consider to loosen the fixed choice of Σ in (9). We can allow some more interaction for participants instead of just feed-backing us the values of $D(G(z_n))$. One enhancement at the participants' interface being considered is to add extra sliders for participants to play with until they find significant discrepancies between the images pair for them to click the color bar. By doing so, Σ will no longer be a fixed matrix in (9) as the participant's interactions need to be included.

7 Future Work

While there are some improvement suggestions in the previous section regarding how to obtain the theoretically expected result, this section refers to possible research avenues given that the current project can finally achieve the expected result.

First of all, the relationship between generator's architecture and the learning performance of AHGD can be further explored. We can also explore the empirical equivalence of learning performance (in terms of number of iterations in the Training Phase) with 1 iteration in the Human Participant Phase across different generators' architecture (e.g. different number of layers, different filters' sizes etc.). We can even look at the sentiment and facial expression of the facial images (similar to the study of [5]) instead of looking at the faces/eigenfaces per se, as the former would provide additional psychological meaning. Regarding the network structure, we can also consider to add a "human gate" between the machinery G and D in a similar manner of the "black box gate" of MalGAN [4] so as to guide the generator(s) further to synthesize more realistic facial images effectively.

Notation	Space	Meaning
x	$X = \mathbb{R}^{64\times64\times1}$	Real Faces
$\hat{x}$	X	Generated faces
$\Delta = \Delta x$	X	Perturbation
θ_X	X	Perturbated faces
$\pi(\theta_X\|x)$	$X \to \mathbb{R}$	p.d.f. of θ_X
z	$Z = \mathbb{R}^{100\times1}$	Noises to generate faces
z_q	$Z_q = \mathbb{R}^{q\times1}$	Noises to generate faces by eigenfaces
$G(z)$	$Z \to X$	Generator
$D(\cdot)$	$X \to \mathbb{R}$	Discriminator by human
$D_{Matlab}(\cdot)$	$X \to \mathbb{R}$	Discriminator by *Matlab*
$\mathcal{V}$	$\mathbb{R}^{64^2\times q}$	Eigenfaces
$\mathcal{D}$	$\mathbb{R}^{q\times q}$	Eigenvalues
$\Sigma = \Sigma_{\theta_X}$	$\mathbb{R}^{64^2\times64^2}$	Covariance matrix of θ_X
Σ_q	$\mathbb{R}^{q\times q}$	Covariance matrix of z_q
V_{train}	$\mathbb{R}$	Objective function for G in Phase I
V	$\mathbb{R}$	Objective function for G in Phase II
θ_G or $\{\theta_G\}$	Θ_G	Parameters of G
$F(\theta_X)$	$X \to \mathbb{R}$	Any function of θ_X
ΔD	$\mathbb{R}$	$\Delta D = D(x^+) - D(x^-)$
N	$\mathbb{R}$	Number of data points
R	$\mathbb{R}$	Number of perturbations
n or r	$\mathbb{R}$	Dummy index for N or R
q	$\mathbb{R}$	Number of eigenfaces being chosen.
σ	$\mathbb{R}$	Scaling factor for Σ_q

As mentioned before, this project is application driven and we would like to pave the way for synthesis/reconstruction of high dimensional data. We hope that the techniques of **Uneven NES** and **Iterations-Forking Analysis** can be further enhanced and practically applied to high dimensional data. For example, the AHGD approach can be used to synthesize realistic 3D human-chest reconstruction (e.g. To replace the auto-encoder structure of [6] by a mixture of 3D-GAN [11] and HumanGAN, and then invite physicians to be the participants of D_{Human}).

8 Appendix

This appendix section will prove the followings:

$$\frac{\partial V}{\partial \hat{x}_n} \approx \frac{\Sigma^{-1}}{2R} \sum_{r=1}^{R} \big(\Delta D^{(r)}(\hat{x}_n)\big)\Delta_n^{(r)} \tag{13}$$

From (11), we have:

$$\nabla_x log(\pi(\theta_X|x)) = 0 + \frac{-1}{2}\nabla_x\Big((\theta_X - x)^T\Sigma^{-1}(\theta_X - x)\Big) = \Sigma^{-1}(\theta_X - x) = \Sigma^{-1}\Delta \tag{14}$$

The second equality is achieved by $\nabla_x(x^TAx) = (A + A^T)x$ for any square matrix A. By plugging in (14) into (7), we have:

$$\nabla_x F(x) = \mathbb{E}_{\pi(\theta_X|x)}[\,F(\theta_X)\nabla_x log(\pi(\theta_X|x))] \;= \mathbb{E}_{\pi(\theta_X|x)}[\,F(\theta_X)\Sigma^{-1}\Delta]$$

$$\approx \frac{1}{2R}\sum_{r=1}^{R}\Big(F(x + \Delta^{(r)}) - F(x - \Delta^{(r)})\Big)\Sigma^{-1}\Delta^{(r)}$$

Now we pick $F(x) = D(x)$ and $x = \hat{x}_n$, we have

$$\frac{\partial D(\hat{x}_n)}{\partial \hat{x}_n} \approx \frac{1}{2R}\sum_{r=1}^{R}\Big(D(\hat{x}_n + \Delta_n^{(r)}) - D(\hat{x}_n - \Delta_n^{(r)})\Big)\Sigma^{-1}\Delta_n^{(r)}$$

$$= \frac{1}{2R}\sum_{r=1}^{R}\Delta D^{(r)}(\hat{x}_n)\Sigma^{-1}\Delta_n^{(r)} = \frac{\Sigma^{-1}}{2R}\sum_{r=1}^{R}\Delta D^{(r)}(\hat{x}_n)\Delta_n^{(r)}$$

By $V = \sum_{n=1}^{N} D(G(z_n))$ (i.e. (5)) and above, we have:

$$\frac{\partial V}{\partial \hat{x}_n} = \frac{\partial}{\partial \hat{x}_n}\Big(\sum_{i=1}^{N} D(G(z_i))\Big) = \frac{\partial D(G(z_n))}{\partial \hat{x}_n}$$

$$\approx \frac{\Sigma^{-1}}{2R}\sum_{r=1}^{R}\big(\Delta D^{(r)}(\hat{x}_n)\big)\Delta_n^{(r)}$$

This proved (13).

References

[1] A. Eyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. *35th International Conference on Machine Learning, ICML 2018*, 5:3392–3401, 2018.

[2] K. Fujii, Y. Saito, S. Takamichi, Y. Baba, and H. Saruwatari. HumanGAN: generative adversarial network with human-based discriminator and its evaluation in speech perception modeling. 2019.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. Technical report, 2014.

[4] W. Hu and Y. Tan. Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN. Technical report, 2017.

[5] L. L. Kontsevich and C. W. Tyler. What makes Mona Lisa smile? *Vision Research*, 44(13):1493–1498, 2004. ISSN 00426989. doi: 10.1016/j.visres.2003.11.027.

[6] K. Park, L. H. M. Chu, Y. J. Chuang, and M. F. Demirel. 3D Reconstruction of Chest X-Rays, 2019. URL http://sites.google.com/view/cxr2ct.

[7] V. Periyasamy. Imitating Generative Adversarial Networks with Humans. 2018. URL http://pages.cs.wisc.edu/~viswesh/projects/cs841.pdf.

[8] C. Sanderson. LFWcrop Face Dataset. URL http://conradsanderson.id.au/lfwcrop/. last accessed on 11/03/2019.

[9] E. Wallace, P. Rodriguez, S. Feng, I. Yamada, and J. Boyd-Graber. Trick Me If You Can: Human-in-the-Loop Generation of Adversarial Examples for Question Answering. *Transactions of the Association for Computational Linguistics*, 7(Section 6):387–401, 2019. doi: 10.1162/tacl_a_00279. URL https://github.com/Eric-Wallace/trickme-interface/.

[10] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural Evolution Strategies. Technical report, 2014.

[11] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. Technical report.

[12] J. Zhang, Y. Yan, and M. Lades. Face recognition: Eigenface, elastic matching, and neural nets. *Proceedings of the IEEE*, 85:1423–1435, 1997. ISSN 00189219. doi: 10.1109/5.628712.

[13] J.-Y. Zhu, T. Park, P. Isola, A. A. Efros, and B. A. Research. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks Monet Photos. Technical report.