On the Versatile Uses of Partial Distance Correlation in Deep Learning (Supplementary Material)

Xingjian Zhen[1], Zihang Meng[1], Rudrasis Chakraborty[2], and Vikas Singh[1]

[1] University of Wisconsin-Madison
{xzhen3, zmeng29}@wisc.edu, vsingh@biostat.wisc.edu
[2] Butlr
rudrasischa@gmail.com

1 Technical Analysis Using Block Stochastic Gradient

In this section, we describe results (which were briefly mentioned in the main paper) showing the viability of a stochastic scheme for using distance correlation within the loss when training our neural network models.

In the paper, we noted the existence of an algorithm where the convergence rate of the stochastic version of distance correlation in the deep neural network setting is $O(\frac{1}{\sqrt{T}})$. Here, we will describe it more formally.

We note that SGD works well for the distance correlation objective – and so a majority of users will revert to such mature implementations anyway. Despite desirable practical behavior, its theoretical analysis of the form included here is more involved. Therefore, the analysis shown below, a modified version of [5], is reassuring in the sense that we know that stochastic updates (carried out in a specific way) can provably optimize our loss.

1.1 Notation in the Proof

We use Θ_X, Θ_Y to denote the parameters of neural networks, and X, Y as features extracted by the respective neural networks. Let the minibatch size be m, and the dataset $\mathcal{D} = (\mathcal{D}_\mathcal{X}, \mathcal{D}_\mathcal{Y})$ be of size n. Let, $X \in \mathbb{R}^{m\times p}, Y \in \mathbb{R}^{m\times q}$, with p, q be the dimension of features. We use $(x_t, y_t)_{t=1}^T, x_t \subset \mathcal{D}_\mathcal{X}, y_t \subset \mathcal{D}_\mathcal{Y}$ to represent the data samples at step t, T is the total number of training steps. The distance matrices A_t, B_t are computed when given X_t, Y_t using (1), which is of dimension $m \times m$ for each minibatch. Further, we use $(X_t)_k$ to represent the k^{th} element in X_t. Also, $(A_t)_{k,l}$ is the k^{th} row and l^{th} column element in the matrix A_t. The inner-product between two matrices A, B is defined as $\langle A, B\rangle = \sum_{i,j}^m (A)_{i,j}(B)_{i,j}$.

$$
\begin{aligned}
&a_{k,l} = \|X_k - X_l\|, \quad \bar{a}_{k,\cdot} = \frac{1}{n}\sum_{l=1}^n a_{k,l}, \quad \bar{a}_{\cdot,l} = \frac{1}{n}a_{k,l}, \\
&\bar{a}_{\cdot,\cdot} = \frac{1}{n^2}\sum_{k,l=1}^n a_{k,l}, \quad A_{k,l} = a_{k,l} - \bar{a}_{k,\cdot} - \bar{a}_{\cdot,l} + \bar{a}_{\cdot,\cdot}
\end{aligned} \tag{1}
$$

1.2 Objective Function

Consider the case where we minimize DC between two networks Θ_X, Θ_Y. Since the parameters between Θ_X, Θ_Y are separable, we can use block stochastic gradient iteration in [5] with some modification.

To minimize the distance correlation, we need to solve the following problem

$$\min_{\Theta_X, \Theta_Y} \quad \frac{\langle A(\Theta_X; x), B(\Theta_Y; y)\rangle}{\sqrt{\langle A(\Theta_X; x), A(\Theta_X; x)\rangle \langle B(\Theta_Y; y), B(\Theta_Y; y)\rangle}} \tag{2}$$

$$\text{s.t.} \quad (A)_{k,l} = ||(X)_k - (X)_l||_2,\ X = \Theta_X(x) \tag{3}$$

$$(B)_{k,l} = ||(Y)_k - (Y)_l||_2,\ Y = \Theta_Y(y)$$

We slightly abuse the notation of $\Theta_X(x)$ to correspond to applying the network Θ_X on the data x, and reuse A to simplify the notation $A(\Theta_X; x)$ and the distance matrix. We can rewrite the expression (with A, B defined above) using:

$$\min_{\Theta_X, \Theta_Y} \langle A, B\rangle \text{ s.t. } \max_{x \subset \mathcal{D}_\mathcal{X}} \langle A, A\rangle \leq m;\ \max_{y \subset \mathcal{D}_\mathcal{Y}} \langle B, B\rangle \leq m \tag{4}$$

where (x, y) are the minibatch of samples from the data space $(\mathcal{D}_\mathcal{X}, \mathcal{D}_\mathcal{Y})$.

We can rewrite as the following expression similar to Eq. (1) in [5].

$$\min_{\Theta_X, \Theta_Y} \Phi(\Theta_X, \Theta_Y) = \mathbb{E}_{x,y} f(\Theta_X, \Theta_Y; x, y) + \gamma(\Theta_X) + \gamma(\Theta_Y) \tag{5}$$

where $f(\Theta_X, \Theta_Y; x, y)$ is $\langle A, B\rangle$ and $\gamma(\Theta_X)$ encodes the convex constraint on the network Θ_X, i.e., $\max_{x \subset \mathcal{D}_\mathcal{X}} \langle A, A\rangle \leq m$. Similarly, $\gamma(\Theta_Y)$ encodes $\max_{y \subset \mathcal{D}_\mathcal{Y}} \langle B, B\rangle \leq m$. Further, $\Phi(\Theta_X, \Theta_Y)$ is the constrained objective function to be optimized.

1.3 Block Stochastic Gradient Iteration

We adapt Alg. 1 from [5] to our case in Alg. 1. Since we will need the entire minibatch (x_t, y_t) to compute the objective function, there will be no mean term when computing the sample gradient $\tilde{\mathbf{g}}_X^t$. Further, since both blocks (Θ_X, Θ_Y) are constrained, line 3, 5 will use (5) from [5]. The detailed algorithm is presented in Alg. 1.

Proposition 1. *After T iterations of Algorithm 1 with step size $\eta_X = \eta_Y = \frac{\eta}{\sqrt{T}} < \frac{1}{L}$, for some positive constant $\eta < \frac{1}{L}$, where L is the Lipschitz constant of the partial gradient of f, by Theorem. 6 in [5], we know there exists an index subsequence $\mathcal{T}$ such that:*

$$\lim_{t \to \infty, t \in \mathcal{T}} \mathbb{E}[dist(\mathbf{0}, \nabla\Phi(\Theta_X^t, \Theta_Y^t))] = 0 \tag{6}$$

where $dist(\mathbf{y}, \mathcal{X}) = \min_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - \mathbf{y}\|$.

Algorithm 1 Block Stochastic Gradient for Updating Distance Correlation

Input: Two neural network with starting point Θ_X^1, Θ_Y^1. Training data $\{(x_t, y_t)\}_{t=1}^T$, step size η_X, η_Y, and batch size m.
Output: $\tilde{\Theta}_X^T, \tilde{\Theta}_Y^T$

1: **for** $t = 1, \cdots, T$ **do**
2: Compute sample gradient for Θ_X
$\tilde{\mathbf{g}}_X^t = \nabla_{\Theta_X} f(\Theta_X^t, \Theta_Y^t; x_t, y_t)$
3: $\Theta_X^{t+1} = \arg\min_{\Theta_X} \langle \tilde{\mathbf{g}}_X^t + \tilde{\nabla}\gamma_X(\Theta_X^t), \Theta_X - \Theta_X^t \rangle + \frac{1}{2\eta_X}\|\Theta_X - \Theta_X^t\|^2$
4: Compute sample gradient for Θ_Y
$\tilde{\mathbf{g}}_Y^t = \nabla_{\Theta_Y} f(\Theta_X^{t+1}, \Theta_Y^t; x_t, y_t)$
5: $\Theta_Y^{t+1} = \arg\min_{\Theta_Y} \langle \tilde{\mathbf{g}}_Y^t + \tilde{\nabla}\gamma_Y(\Theta_Y^t), \Theta_Y - \Theta_Y^t \rangle + \frac{1}{2\eta_Y}\|\Theta_Y - \Theta_Y^t\|^2$
6: **end for**
7: $\tilde{\Theta}_X^T = \frac{1}{T}\sum_{t=1}^T \Theta_X^t$
8: $\tilde{\Theta}_Y^T = \frac{1}{T}\sum_{t=1}^T \Theta_Y^t$

Further, in the special case where $\mathbb{E}_{x,y} f(\Theta_X, \Theta_Y; x, y)$ is convex, by Theorem. 1 in [5], the following statement holds:

$$\mathbb{E}[\Phi(\tilde{\Theta}_X^T, \tilde{\Theta}_Y^T) - \Phi(\Theta_X^*, \Theta_Y^*)] \leq D\eta\frac{1+\log T}{\sqrt{1+T}} + \frac{\|\Theta_X - \Theta_X^1\|^2 + \|\Theta_Y - \Theta_Y^1\|^2}{2\eta\sqrt{1+T}} \tag{7}$$

where $\tilde{\Theta}_X, \tilde{\Theta}_Y$ is computed in Algorithm 1, Θ_X^*, Θ_Y^* are the optimum of the desired function, and D is a constant depending on $\|(\Theta_X^*; \Theta_Y^*)\|$.

1.4 Modification from BSG [5] to DC

The statement of Prop. 1 is similar to the statement of Theorem 1 and 6 in [5]. So, we can use the statement in [5] with some modification of our setup. We define

$$F(\Theta_X, \Theta_Y) = \mathbb{E}_{x,y} f(\Theta_X, \Theta_Y; x, y), \;\; \Gamma(\Theta_X, \Theta_Y) = \gamma(\Theta_X) + \gamma(\Theta_Y)$$

Then, for the gradient w.r.t. X, we have the following expression (similar for Y):

$$\begin{aligned}
\tilde{\mathbf{g}}_X^t &= \nabla_{\Theta_X} f(\Theta_X^t, \Theta_Y^t; x_t, y_t) \\
\mathbf{g}_X^t &= \nabla_{\Theta_X} F(\Theta_X^t, \Theta_Y^t) \\
\boldsymbol{\delta}_X^t &= \tilde{\mathbf{g}}_X^t - \mathbf{g}_X^t
\end{aligned}$$

We first restate four assumptions from [5].

Assumption 1. *There exist a constant c and a sequence $\{\sigma_k\}$ such that for any t,*

$$\begin{aligned}
&\|\mathbb{E}[\boldsymbol{\delta}_X^t | x_t, y_t]\| \leq c \cdot \max(\eta_X, \eta_Y), \\
&\mathbb{E}\|\boldsymbol{\delta}_X^t\|^2 \leq \sigma_t^2
\end{aligned}$$

Assumption 2. *The objective function is lower bounded, i.e.,* $\Phi(\Theta_X, \Theta_Y) > -\infty$. *And there is a uniform Lipschitz constant* $L > 0$ *such that:*

$$\|\nabla_X F(\Theta_X, \Theta_Y) - \nabla_X F(\Theta'_X, \Theta'_Y)\| \leq L\|(\Theta_X; \Theta_Y) - (\Theta'_X; \Theta'_Y)\|, \\ \forall (\Theta_X, \Theta_Y), (\Theta'_X, \Theta'_Y)$$

Assumption 3. *There exists a constant* ρ *such that* $\mathbb{E}\|(\Theta_X^t; \Theta_Y^t)\|^2 \leq \rho^2$ *for all* t.

Assumption 4. *The constraint function* γ *is Lipschitz continuous. There is a constant* L_γ, *such that:*

$$\|\gamma(\Theta_X) - \gamma(\Theta'_X)\| \leq L_\gamma \|\Theta_X - \Theta'_X\|, \forall \Theta_X, \Theta'_X$$

Theorem 6. *(from [5]) Let* $\{\Theta^t\}$ *be generated from Algorithm 1 with* η_X^t, η_Y^t, *being constained as,*

$$0 < \inf_t \eta_X^t \leq \sup_t \eta_X^t < \frac{1}{L}$$
$$0 < \inf_t \eta_Y^t \leq \sup_t \eta_Y^t < \frac{1}{L}$$

Under Assumptions 1 through 4, if either $\mathcal{X} = \mathbb{R}^{n_X}, \mathcal{Y} = \mathbb{R}^{n_Y}$ *or* $\gamma = 0$, *and*

$$\sum_{t=1}^{\infty} \sigma_t^2 < \infty$$

then there exists an index subsequence $\mathcal{T}$ *such that*

$$\lim_{t \to \infty, t \in \mathcal{T}} \mathbb{E}[dist(\mathbf{0}, \nabla \Phi(\Theta_X^t, \Theta_Y^t))] = 0,$$

where $dist(\mathbf{y}, \mathcal{X}) = \min_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - \mathbf{y}\|$.

Remark 1. In our case, we have that Θ_X, Θ_Y are the parameters of neural networks. During training, we have no constraint on the weights and biases, so the space of Θ_X, which is $\mathcal{X}$, is the Euclidean space. Also, we can have $\eta_X^t = \eta_Y^t = \frac{\eta}{\sqrt{t}} < \frac{1}{L}$. All the other assumptions are similar to [5]. Thus, we have the same result in Prop. 1.

In the convex case, we use the Theorem 1 from [5].

Theorem 1. *[5] (Ergodic convergence for non-smooth convex case). Let $\{\Theta^t\}$ be generated from Algorithm 1 with $\eta_X^t = \eta_Y^t = \eta_t = \frac{\eta}{\sqrt{t}} < \frac{1}{L}, \forall t$, for some positive constant $\eta < \frac{1}{L}$. Under Assumptions 1 through 4, if F and γ are both convex, Θ_X^*, Θ_Y^* is a solution of (5), and $\sigma = \sup_t \sigma_t < \infty$, then*

$$\mathbb{E}[\Phi(\Theta_X^t, \Theta_Y^t) - \Phi(\Theta_X^*, \Theta_Y^*)] \leq D\eta \frac{1 + \log T}{\sqrt{1+T}} + \frac{\|(\Theta_X^*; \Theta_Y^*) - (\Theta_X^1; \Theta_Y^1)\|^2}{2\eta\sqrt{1+T}}$$

where $\tilde{\Theta}_X^T = \frac{\eta_t \Theta_X^{t+1}}{\sum_{t=1}^T \eta_t}$, $\tilde{\Theta}_Y^T = \frac{\eta_t \Theta_Y^{t+1}}{\sum_{t=1}^T \eta_t}$, and

$$D = \frac{s(\sigma^2 + 4L_\gamma^2)}{1 - L\eta} + \sqrt{s}(\|(\Theta_X^*; \Theta_Y^*)\| + \rho)(c + L\sqrt{8M_\rho^2 + 8\sigma_t^2 + 4L_\gamma^2})$$

where $M_\rho = \sqrt{4L^2\rho^2 + 2\max(\|\nabla_{\Theta_X} F(\mathbf{0})\|^2, \|\nabla_{\Theta_Y} F(\mathbf{0})\|^2)}$

Remark 2. Our case is a special case of the Block Stochastic Gradient problem with $s = 2$ (s is the number of blocks). So the above theorem can be directly applied to our analysis when F, γ are both convex. However, this may not be true in most deep neural networks, we obtain the $O(T^{1/2})$ convergence rate using Algorithm 1.

2 Experimental Details in Section 4: Independent Features Help Robustness

When we train f_1 using cross entropy loss and f_2 using cross entropy loss plus our distance correlation loss (to learn independent features with f_1), we first train f_1 for one epoch, and then train f_2 for one epoch given the current f_1, and we repeat this process for the total number of epochs (200 for CIFAR10 and 40 for ImageNet). Our hyperparameter α controls the tradeoff between the cross entropy loss and the distance correlation loss. In practice, we could increase α to emphasize (or weight) learning independent features more, and decrease α if we want to keep the classification accuracy of f_2 in standard setting (non adversarial) even closer to that of f_1. During training, we do not utilize data augmentation for all experiments. The training is done on Nvidia A100 GPUs. Our distance correlation adds approximtely 20% cost to the training time compared with training only using cross entropy loss.

We also include some more visualization of feature spaces in addition to those shown in our main paper in Fig. 1. Our method shows more independence than the baseline model. This implies training with Distance Correlation (DC) can help independence, thus improve robustness to transferred samples.

3 Experimental Details in Section 5: Informative Comparison Between Networks

3.1 Measure similarity between neural networks

We take the pretrained neural network from [4]. The features are reshaped to a 1D vector and we compute the Euclidean distance between samples from the

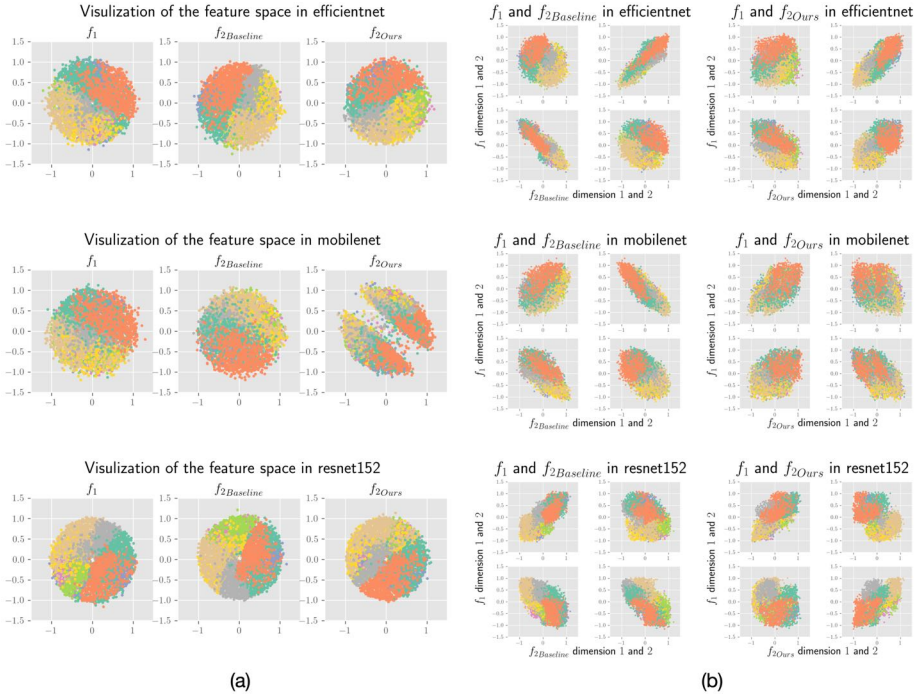

Fig. 1: Picasso visualization of features space and the correlation between different models for all three models. **(a)** Feature space distribution. **(b)** Cross-correlation between the feature space of f_1 and f_2 trained with/without DC. We get better independence.

official validation set of ImageNet [2]. No finetuning was used in this experiment. The results are shown in the main paper.

3.2 What remains when "taking out" (aka controlling for) Y from X

We will first discuss the details of the heatmap that we plotted using Grad-CAM [3]. In the original implementation of Grad-CAM, the model uses one layer as the target and uses both gradients (from the loss function) and the activation in that layer for the visualization.

In the original Grad-CAM, the loss is extracted as the intensity before the softmax layer of one given class. For example, assume "dog" is the 6^{th} class in the dataset. If we want to see which location in the image is related to "dog", we will use $f(x)[6]$ as the loss function, where $\text{Softmax}(f(x))$ is the final output of the model when given image x.

In our case, the activation remains the same. But the loss function is different. We use the distance correlation between the features extracted by the neural network and the ground truth linguistics embedding, i.e., $\text{Loss} = \mathcal{R}^2(X, GT)$, where X is the feature of input image extracted by the neural network.

After showing the Grad-CAM results for each individual network, we want to check if the partial distance correlation can help the network focus on a different location. Thus, we finetune the network X with an extra loss term Loss_{PDC}.

We take ViT-B/16 as our model X and Resnet 18 as our model Y. We first load the pretrained weights from [4] and finetune model X with model Y being fixed. α in our case is set as 1 in the loss term

$$\text{Loss}_{\text{CE}}(f_1(x), y) - \alpha \cdot \text{Loss}_{\text{PDC}}\left((g_1(x)|g_2(x)), gt\right)$$

Learning rate is set as $1e^{-5}$ and batch size is set as 64. We train 15 epochs in total. The finetuning on two RTX 2080Ti takes 2 days.

3.3 Extra Results

We show several additional heat maps using Grad-CAM in addition to those in our main paper in Fig. 2 and 3.

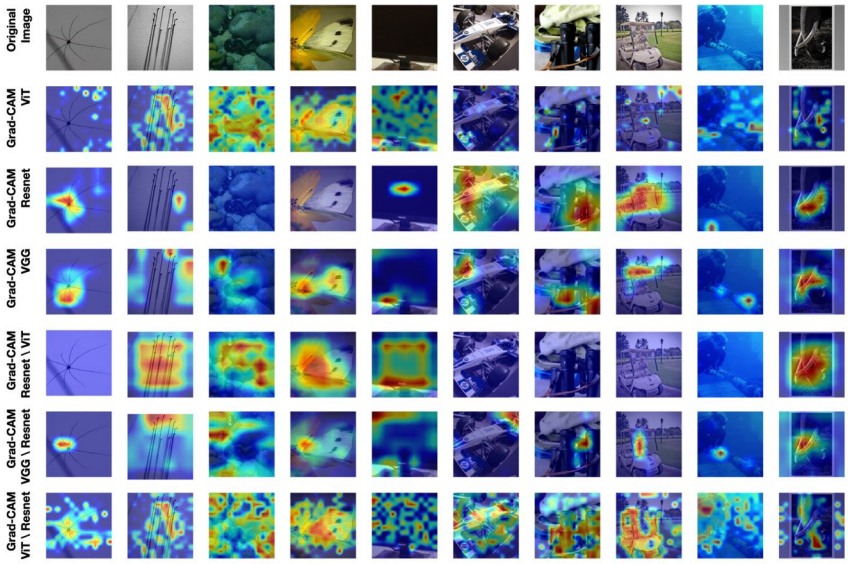

Fig. 2: Extra Grad-CAM results on ImageNet using ViT, Resnet18 and VGG16. After using Partial DC to remove the information learned by another network, ViT can focus on detail places and Resnet can only look in major spots. Similar issue happens to VGG.

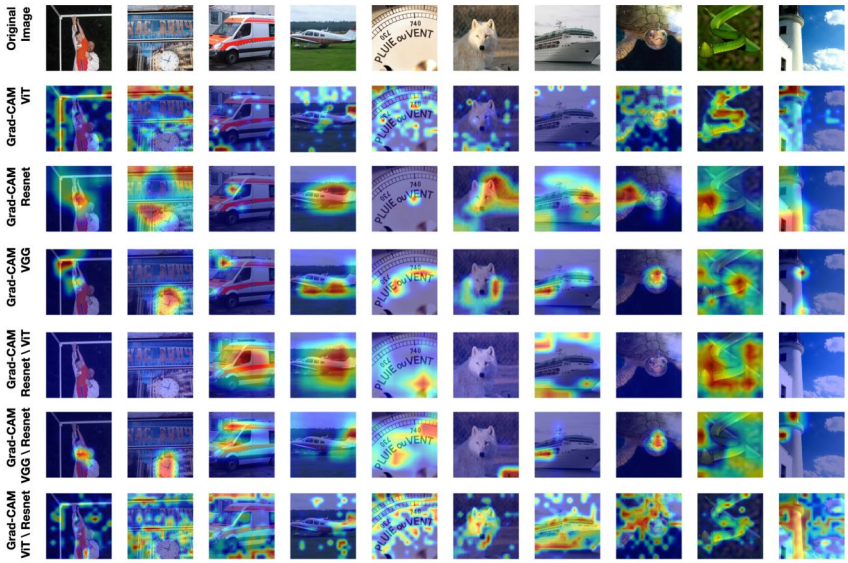

Fig. 3: Extra Grad-CAM results on ImageNet using ViT, Resnet18 and VGG16. After using Partial DC to remove the information learned by another network, ViT can focus on detail places and Resnet can only look in major spots. Similar issue happens to VGG.

4 Experimental Details in Section 6: Disentanglement

We follow the setup in [1] where the dataset contains both labeled and unlabeled data. The provided labels are indicated by the function ℓ:

$$\ell(i,j) = \begin{cases} 1, f_i^j \text{exists (attribute } j \text{ of image } i \text{ is labeled)} \\ 0, \text{otherwise} \end{cases}$$

For each attributes, we train k classifiers of the form $C^j : \mathcal{X} \to [m^j]$ where m^j denotes the number of values of attribute j. The gender attribute here contains male and female, and age attribute contains kid, teenage, adult, and old person. Details are shown in in Table. 1

Table 1: Values that we use in the disentangle experiment on FFHQ dataset.

Attribute	Values
age	kid, teenage, adult, old person
gender	male, female
ethnicity	African person, white person, Asian person
hair	brunette, blond, red, white, black, bald
beard	beard, mustache, goatee, shaved
glasses	glasses, shades, without glasses

For the classifiers when given the true label, we use the cross-entropy loss

$$\mathrm{L}_{cls} = \sum_{i=1}^{n}\sum_{j=1}^{n} \ell(i,j) \cdot H(\mathrm{Softmax}(C^j(x_i)), f_i^j) \tag{8}$$

For the classifiers without the true label, we use the entropy so that the information is not leaking

$$\mathrm{L}_{ent} = \sum_{i=1}^{n}\sum_{j=1}^{n} (1-\ell(i,j)) \cdot H(\mathrm{Softmax}(C^j(x_i))) \tag{9}$$

For the residual, we use the distance correlation loss in our paper

$$\mathrm{L}_{res} = dCor([f^1; f^2; ...; f^k], r) \tag{10}$$

Let the value for each of the attributes of interest j to be $\tilde{f}_i^j$

$$\tilde{f}_i^j = \begin{cases} f_i^j & , \ell(i,j) = 1 \\ \mathrm{Softmax}(C^j(x_i)) & , \mathrm{otherwise} \end{cases}$$

Also, we include the reconstruction loss to generate the target image

$$\mathrm{L}_{rec} = \sum_{i=1}^{n} \phi(G(\tilde{f}_i^1, ..., \tilde{f}_i^k, r_i'), x_i) \tag{11}$$

The final loss will be the linear combination of all the loss above.

$$\mathrm{L}_{disentangle} = \mathrm{L}_{rec} + \lambda_{cls}\mathrm{L}_{cls} + \lambda_{ent}\mathrm{L}_{ent} + \lambda_{res}\mathrm{L}_{res} \tag{12}$$

In our implementation, $\lambda_{cls} = 0.1$, $\lambda_{ent} = 0.01$, $\lambda_{res} = 1e^{-5}$.

4.1 Additional examples of generated images

Some more generated images of the same individual are shown in Fig. 4, 5, 6, 7, 8, and 9. We can see that our model can maintain most features in the image (keeps unchanged) and changes the attributes of interest separately. The results here are mostly qualitative.

4.2 Quantitative results

We also performed quantitative checks to assess if our model can disentangle the attributes of interest and the remaining attributes well. As shown in Tab. 2 below, we measure the distance correlation (as it can handle mismatched dimensions easily) between the residual attributes (R) and the attributes of interest. For the supervised label (using the pretrained CLIP model), we use BERT to embed label descriptions into vector space $d = 768$. For the unsupervised samples, we use the in-model classifiers to embed to $d = 32$ space.

Fig. 4: Generated images pertaining to the different ages for the same individual. While the results are qualitative, perceptually the generated results appear meaningful.

age vs R.	gender vs R.	ethnicity vs R.	hair color vs R.	beard vs R.	glasses vs R.
0.0329	0.0180	0.0222	0.0242	0.0219	0.0255
age vs R.	gender vs R.	ethnicity vs R.	hair color vs R.	beard vs R.	glasses vs R.
0.0430	0.0124	0.0376	0.0259	0.0490	0.0188

Table 2: DC between residual attributes (R) and attributes of interest. (Top) We use the ground truth CLIP labeled data to measure the attribute of interest. (Bottom) We use in-model classifier to classify the attribute of interest (smaller is better).

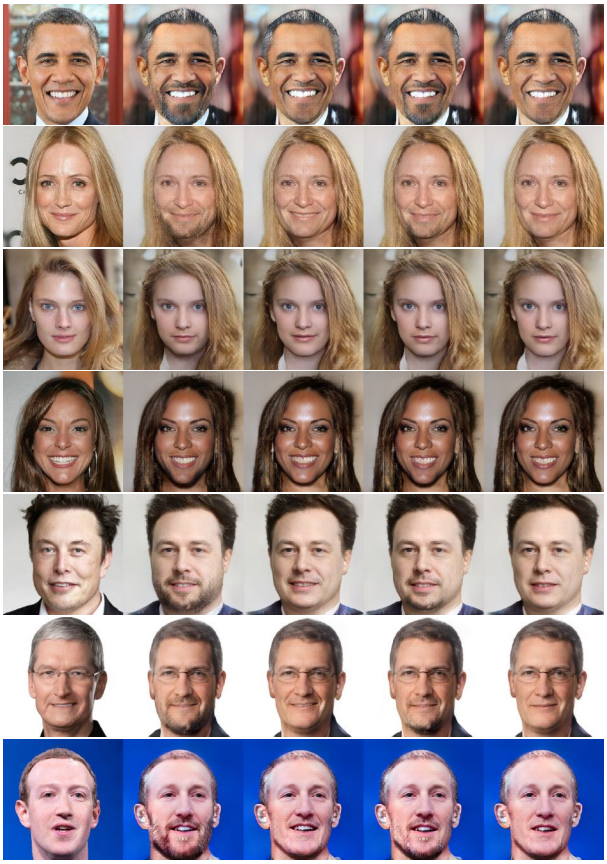

Fig. 5: Generated images pertaining to different beard levels for the same individual. The first two rows appear perceptually meaningful.

Fig. 6: Generated images pertaining to different ethnicity for the same individual.

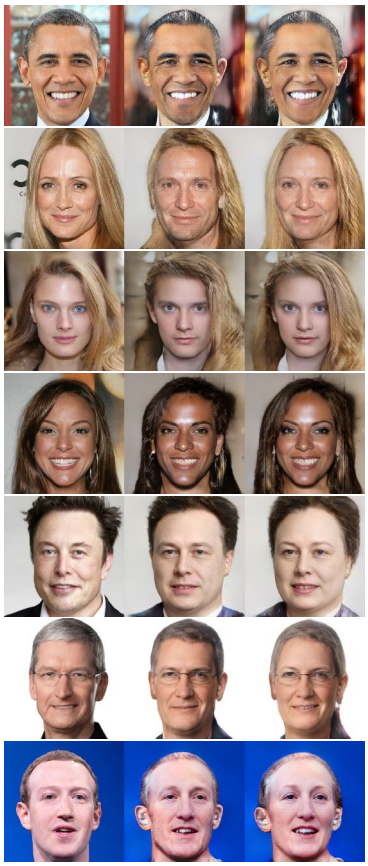

Fig. 7: Generated images pertaining to different gender for the same individual.

Fig. 8: Generated images pertaining to different level of “glasses” attribute for the same individual.

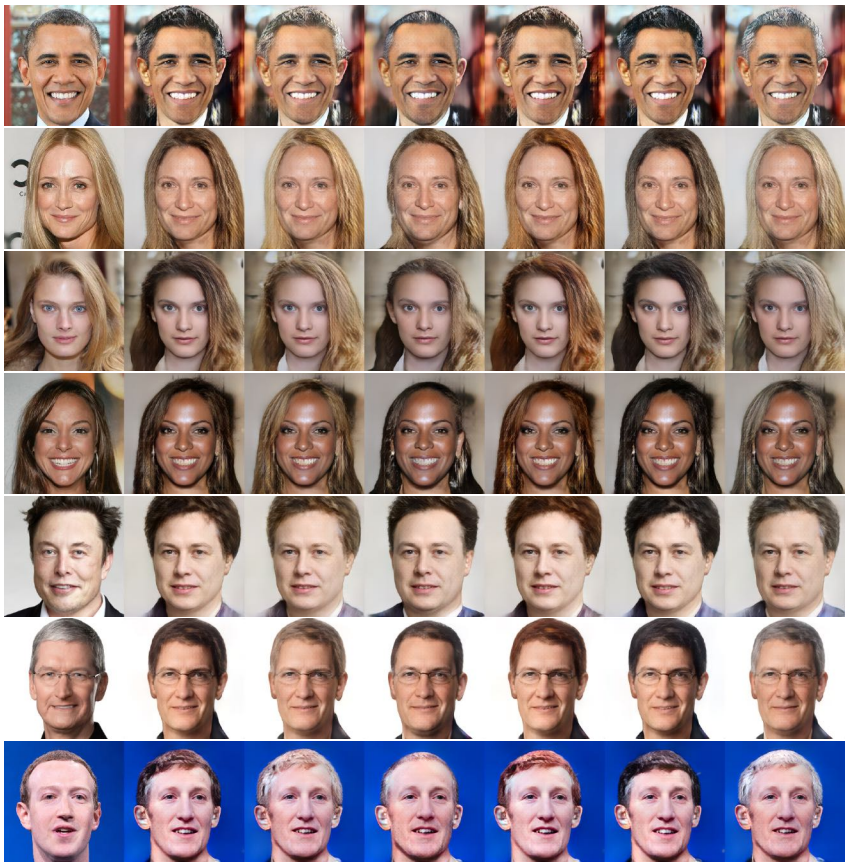

Fig. 9: Generated images pertaining to different hair color for the same individual.

References

1. Gabbay, A., Cohen, N., Hoshen, Y.: An image is worth more than a thousand words: Towards disentanglement in the wild. Advances in Neural Information Processing Systems **34**, 9216–9228 (2021)
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25**, 1097–1105 (2012)
3. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. pp. 618–626 (2017)
4. Wightman, R.: Pytorch image models. `https://github.com/rwightman/pytorch-image-models` (2019). https://doi.org/10.5281/zenodo.4414861
5. Xu, Y., Yin, W.: Block stochastic gradient iteration for convex and nonconvex optimization. SIAM Journal on Optimization **25**(3), 1686–1716 (2015)