HyperDomainNet: Universal Domain Adaptation for Generative Adversarial Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Domain adaptation framework of GANs has achieved great progress in recent years as a main successful approach of training contemporary GANs in the case of very limited training data. In this work, we significantly improve this framework by proposing an extremely compact parameter space for fine-tuning the generator. We introduce a novel domain-modulation technique that allows to optimize only 6 thousand-dimensional vector instead of 30 million weights of StyleGAN2 to adapt to a target domain. We apply this parameterization to the state-of-art domain adaptation methods and show that it has almost the same expressiveness as the full parameter space. Additionally, we propose a new regularization loss that considerably enhances the diversity of the fine-tuned generator. Inspired by the reduction in the size of the optimizing parameter space we consider the problem of multi-domain adaptation of GANs, i.e. setting when the same model can adapt to several domains depending on the input query. We propose the HyperDomainNet that is a hypernetwork that predicts our parameterization given the target domain. We empirically confirm that it can successfully learn a number of domains at once and may even generalize to unseen domains.

1 Introduction

Contemporary generative adversarial networks (GANs) [7, 13, 14, 12, 2] show remarkable performance in modeling image distributions and have applications in a wide range of computer vision tasks (image enhancement [16, 39], editing [8, 28], image-to-image translation [11, 43, 44], etc.). However, the training of modern GANs requires thousands of samples that limits its applicability only to domains that are represented by a large set of images. The mainstream approach to sidestep this limitation is transfer learning (TL), i.e. fine-tuning the generative model to a domain with few samples starting with a pretrained source model.

The standard approach of GAN TL methods is to fine-tune almost *all* weights of the pretrained model [17, 19, 35, 34, 12, 41, 21, 5, 45]. It can be reasonable in the case when the target domain is very far from the source one, e.g. when we adapt the generator pretrained on human faces to the domain of animals or buildings. However, there is a wide range of cases when the distance between data domains is not so far. In particular, the majority of target domains used in works [17, 34, 21, 5, 45] are similar to the source one and differ mainly in texture, style, geometry while keep the same content like faces or outdoor scenes. For such cases it seems redundant to fine-tune all weights of the source generator. It was shown in the paper [37] that after transfer learning of the StyleGAN2 [14] to similar domains some parts of the network almost do not change. This observation motivates us to find a more efficient and compact parameter space for domain adaptation of GANs.

In this paper, we propose a novel *domain-modulation* operation that reduces the parameter space for fine-tuning the StyleGAN2. The idea is to optimize for each target domain only a single vector

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

d. We incorporate this vector into the StyleGAN2 architecture through the modulation operation at each convolution layer. The dimension of the vector d equals 6 thousand that is *5 thousand times* less than the original weights space of the StyleGAN2. We apply this parameterization for the state-of-the-art domain adaptation methods StyleGAN-NADA [5] and MindTheGAP [45]. We show that it has almost the same expressiveness as the full parameterization while being more lightweight. To further advance the domain adaptation framework of GANs we propose a new regularization loss that improves the diversity of the fine-tuned generator.

Such considerable reduction in the size of the proposed parameterization motivates us to consider the problem of multi-domain adaptation of GANs, i.e. when the same model can adapt to multiple domains depending on the input query. Typically, this problem is tackled by previous methods just by fine-tuning separate generators for each target domain independently. In contrast, we propose to train a hyper-network that predicts the vector d for the StyleGAN2 depending on the target domain. We call this network as HyperDomainNet. Such hyper-network would be impossible to train if we needed to predict all weights of StyleGAN2. The immediate benefits of multi-domain framework consist of reducing the training time and the number of trainable parameters because instead of fine-tuning n separate generators we train one HyperDomainNet to adapt to n domains simultaneously. Another advantage of this method is that it can generalize to *unseen* domains if n is sufficiently large and we empirically observe this effect.

We provide extensive experiments to empirically confirm the effectiveness of the proposed parameterization and the regularization loss on a wide range of domains. We illustrate that our parameterization can achieve quality comparable with the full parameterization (i.e. when we optimize all weights). The proposed regularization loss significantly improves the diversity of the fine-tuned generator that is validated qualitatively and quantitatively. Further, we conduct experiments with the HyperDomainNet and show that it can be successfully trained on a number of target domains simultaneously. Also we show that it can generalize to a number of diverse unseen domains.

To sum up, our main contributions are

- We reduce the number of trainable parameters for domain adaptation of StyleGAN2 [14] generator by proposing the domain-modulation technique. Instead of fune-tuning all 30 millions weights of StyleGAN2 for each new domain now we can train only 6 thousand-dimensional vector.
- We introduce a novel regularization loss that considerably improves the diversity of the adapted generator.
- We propose a HyperDomainNet that predicts the parameterization vector for the input domain and allows multi-domain adaptation of GANs. It shows inspiring generalization results on unseen domains.

2 Related Work

Domain Adaptation of GANs The aim of few-shot domain adaptation of GANs is to learn accurate and diverse distribution of the data represented by only a few images. The standard approach is to utilize a generator pretrained on source domain and fine-tune it to a target domain. There are generally two different regimes of this task. The first one is when we adapt the generator to completely new data (e.g. faces $\rightarrow$ landscapes, churches, etc.), and the second regime is when the target domain relates to the source one (e.g. faces $\rightarrow$ sketches, artistic portraits, etc.).

Methods that tackle the first regime typically require several hundreds or thousands samples to adapt successfully. Such setting assumes that the weights of the generator should be changed significantly because the target domain can be very far from the source. The paper [12] shows that for distant domains training from scratch gives comparable results to transfer learning. It also confirms that for such regime there is no point to reduce the parameter space. Typcally such approaches utilize data augmentations [12, 30, 41, 42], or use auxiliary tasks for the discriminator to more accurately fit the available data [18, 38], or freeze lower layers of the discriminator to avoid overfitting [19].

In the second regime the transfer learning is especially crucial because the pretrained generator already contains many information about the target domain. In this setting the required number of available data can be significantly smaller and range from hundreds to several images. The main challenges in the case of such limited data are to avoid over-fitting of the generator and leverage its

diversity learned from the source domain. To tackle these challenges existing methods introduce restrictions on the parameter space [26, 20], mix the weights of the adapted and the source generators [23], utilize a small network to force sampling in special regions of the latent space [34], propose new regularization terms [17, 31], or apply constrastive learning techniques to enhance cross-domain consisitency [21]. The state-of-the-art methods [5, 45] leverage supervision from vision-language CLIP model [24]. StyleGAN-NADA [5] applies it for text-based domain adaptation when we have no access to images but only to the textual description. MindTheGap [45] employs CLIP model to further significantly improve the quality of one-shot domain adaptation.

Constraining Parameter Space for GAN's Adaptation In the second regime of GAN's adaptation it is especially important for the generator to leverage the information from the source domain during adapting to the target one. The common approach is to introduce some restrictions on the trainable weights to regularize them during fine-tuning. For example, the work [26] proposes to optimize only the singular values of the pretrained weights and apply it for few shot domain adaptation, however the reported results show the limited expressiveness of such parameterization [26]. Another method [20] constrains the parameter space for models with batch normalization (BN) layers such as BigGAN [2] by optimizing only BN statistics during fine-tuning. While it allows to decrease the number of trainable parameters, it also considerably reduces the expressiveness of the generator [26, 21]. Other approach is to adaptively choose a subset of layers during optimization at each step as in StyleGAN-NADA [5]. It helps to stabilize the training, however it does not reduce the parameter space because each layer can potentially be fine-tuned. In contrast, the size of our parameterization is less by orders of magnitude than the size of the full parameter space while having the comparable expressiveness.

3 Preliminaries

In this work, we focus on StyleGAN generators in the context of domain adaptation. We consider StyleGAN2 [14] as a base model. As the state-of-the-art domain adaptation methods we use StyleGAN-NADA [5] and MindTheGAP [45].

StyleGAN2 The StyleGAN2 generation process consists of several components. The first part is a mapping network $M(z)$ that takes as an input random vectors $z \in \mathcal{Z}$ from the initial latent space, $\mathcal{Z}$ that is typically normally distributed. It transforms these vectors z into the intermediate latent space $\mathcal{W}$. Each vector $w \in \mathcal{W}$ is further fed into different affine transformations $A(w)$ for each layer of the generator. The output of this part forms StyleSpace $\mathcal{S}$ [36] that consists of channel-wise style parameters $s = A(w)$. The next part of the generation process is the synthesis network G_{sys} that takes as an input the constant tensor c and style parameters s at the corresponding layers and produces the final feature maps at different resolutions $F = G_{sys}(c, s)$. These feature maps move on to the last part which consists of toRGB layers G_{tRGB} that generate the output image $I = G_{tRGB}(F)$.

Problem Formulation of Domain Adaptation The problem of domain adaptation of StyleGAN2 can be formulated as follows. We are given a trained generator G^A for the source domain A, and the target domain B that is represented by the one image I_B (one-shot adaptation) or by the text description t_B (text-guided adaptation). The aim is to fine-tune the weights θ of a new generator G^B_θ for the domain B starting from the weights of G^A. The optimization process in the general form is

$$\mathcal{L}_B(\theta) = \mathcal{L}(\{G^B_\theta(w_i)\}_{i=1}^n, \{G^A(w_i)\}_{i=1}^n, G^B_\theta, B, A) \;\rightarrow\; \min_\theta, \tag{1}$$

where $\mathcal{L}$ is some loss function, n is a batch size, $w_1, \dots, w_n$ are random latent codes, $\{G^B_\theta(w_i)\}_{i=1}^n$ and $\{G^A(w_i)\}_{i=1}^n$ are batches of images sampled by G^B_θ and G^A generators, respectively, and B, A are domains that are represented by images or text descriptions.

CLIP model CLIP [24] is a vision-language model that is composed of text and image encoders E_T, E_I, respectively, that maps their inputs into a joint, multi-modal space of vectors with a unit norm (this space is often called as CLIP space). In this space the cosine distance between embeddings reflects the semantic similarity of the corresponding objects.

StyleGAN-NADA StyleGAN-NADA [5] is a pioneering work that utilizes the CLIP model [24] for text-guided domain adaptation of StyleGAN. The proposed loss function is

$$\Delta T(B, A) = E_T(t_B) - E_T(t_A),$$
$$\Delta I(G^B_\theta(w), G^A(w)) = E_I(G^B_\theta(w)) - E_I(G^A(w)),$$
$$\mathcal{L}_{direction}(G^B_\theta(w), G^A(w), B, A) = 1 - \frac{\Delta I(G^B_\theta(w), G^A(w)) \cdot \Delta T(B, A)}{|\Delta I(G^B_\theta(w), G^A(w))||\Delta T(B, A)|}. \tag{2}$$

The idea is to align the CLIP-space direction between the source and target images $\Delta I(G^B_\theta(w), G^A(w))$ with the direction between a pair of source and target text descriptions $\Delta T(B, A)$. So, the overall optimization process has the form

$$\mathcal{L}_B(\theta) = \sum_{i=1}^{n} \mathcal{L}_{direction}(G^B_\theta(w_i), G^A(w_i), B, A) \ \rightarrow \ \min_\theta . \tag{3}$$

In StyleGAN-NADA method the $\mathcal{L}_B(\theta)$ loss is optimized only with respect to the weights θ of the synthesis network G^B_{sys} which has 24 million weights.

MindTheGap The MindTheGap method [45] is proposed for a one-shot domain adaptation of StyleGAN, i.e. the domain B is represented by the single image I_B. In principle StyleGAN-NADA method can solve this problem just by replacing the text direction $\Delta T(B, A)$ from Equation (2) to an image one

$$\Delta I'(B, A) = E_I(I_B) - \frac{1}{|A|} \sum_{I_A \in A} [E_I(I_A)], \tag{4}$$

where $\frac{1}{|A|} \sum_{I_A \in A} [E_I(I_A)]$ is the mean embedding of the images from domain A. However, as stated in [45] this leads to an undesirable effect that transferred images lose the initial diversity of domain A and become too close to the I_B image. So, the key idea of the MindTheGap is to replace the mean embedding from Equation (4) by the embedding of projection I^*_A of I_B image to A domain obtained by the GAN inversion method II2S [46]:

$$\Delta I''(B, A) = E_I(I_B) - E_I(I^*_A), \tag{5}$$

So, the MindTheGap uses the modified $\mathcal{L}'_{direction}$ loss that is renamed to $\mathcal{L}_{clip_accross}$

$$\mathcal{L}_{clip_accross}(G^B_\theta(w), G^A(w), B, A) = 1 - \frac{\Delta I(G^B_\theta(w), G^A(w)) \cdot \Delta I''(B, A)}{|\Delta I(G^B_\theta(w), G^A(w))||\Delta I''(B, A)|}. \tag{6}$$

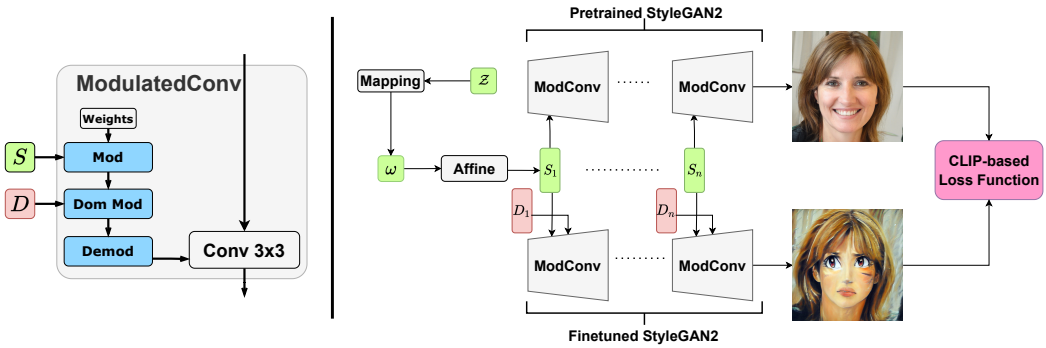
In addition to this idea several new regularizers are introduced that force the generator G^B_θ to reconstruct the I_B image from its projection I^*_A. It further stabilizes and improves the quality of domain adaption. Overall, the MindTheGAP loss function $\mathcal{L}_{MTG}$ has four terms to optimize G^B_θ. For more details about each loss please refer to the original paper [45].

4 Approach

4.1 Domain-Modulation Technique for Domain Adaptation

Our primary goal is to improve the domain adaptation of StyleGAN by exploring an effective and compact parameter space to use it for fine-tuning G^B_θ. As we described in Section 3 StyleGAN has four components: the mapping network $M(\cdot)$, affine transformations $A(\cdot)$, the synthesis network $G_{sys}(\cdot, \cdot)$, and toRGB layers $G_{tRGB}(\cdot)$. It is observed in the paper [37] that the main part of StyleGAN that is mostly changed during fine-tuning to a target domain is the synthesis network $G_{sys}(\cdot, \cdot)$. It is also confirmed by StyleGAN-NADA [5] and MindTheGap [45] methods as they adapt only the weights of $G_{sys}(\cdot, \cdot)$ for the target domain.

So, we aim to find an effective way to fine-tune the weights of feature convolutions of $G_{sys}(\cdot, \cdot)$. In StyleGAN2 [14] these convolutions utilize modulation/demodulation operations to process the input



(a) Domain-modulation technique (b) Fine-tuning StyleGAN2 by optimizing the domain vector D

Figure 1: Detailed diagram of proposed method. (a) Revised ModulatedConv block with introduced domain-modulation operation. (b) Fully detailed training process of the domain adaptation with the proposed domain-modulation technique.

tensor and the corresponding style parameters s. Let us revisit the mechanism of these operations:

$$\text{modulation: } w'_{ijk} = s_i \cdot w_{ijk}, \tag{7}$$

$$\text{demodulation: } w''_{ijk} = \frac{w'_{ijk}}{\sqrt{\sum_{i,k} {w'_{ijk}}^2 + \varepsilon}}, \tag{8}$$

where w, w' and w'' are the original, modulated and demodulated weights, respectively, s_i is the component of the style parameters s, i and j enumerate input and output channels, respectively. The idea behind modulation/demodulation is to replace the standard adaptive instance normalization (AdaIN) [32, 4] to a normalization that is based on the expected statistics of the input feature maps rather than forcing them explicitly [14]. So, the modulation part is basically an adaptive scaling operation as in AdaIN that is controled by the style parameters s. This observation inspires us to use this technique for the domain adaptation.

The problem of fine-tuning GANs to a new domain is very related to the task of style transfer where the goal is also to translate images from the source domain to a new domain with the specified style. The contemporary approach to solve this task is to train an image-to-image network which takes the target style as an input condition. The essential ingredient of such methods is the AdaIN that provides an efficient conditioning mechanism. In particular, it allows to train arbitrary style transfer models [10]. So, it motivates us to apply the AdaIN technique for adapting GANs to new domains.

We introduce a new *domain-modulation* operation that reduces the parameter space for fine-tuning StyleGAN2. The idea is to optimize only a vector d with the same dimension as the style parameters s. We incorporate this vector into StyleGAN architecture by the additional modulation operation after the standard one from Equation (7):

$$\text{domain-modulation: } w'_{ijk} = d_i \cdot w_{ijk}, \tag{9}$$

where d_i is the component of the introduced domain parameters d (see Figure 1a). So, instead of optimizing all weights θ of the G_{sys} part we train only the vector d.

We apply these new parameterization to StyleGAN-NADA and MindTheGAP methods, i.e. instead of optimizing its loss functions wrt θ we optimize it wrt d vector (see Figure 1b) The dimension of the vector d equals 6 thousand that is 4 thousand times less than the original weights space θ of $G_{sys}(\cdot, \cdot)$ part. While the proposed parameter space is radically more constrained we observe that it has the expressiveness comparable with the whole weight space.

4.2 Improving Diversity of CLIP-Guided Domain Adaptation

The CLIP-based domain adaptation methods StyleGAN-NADA and MindTheGap use $\mathcal{L}_{direction}$ (or $\mathcal{L}_{clip_accross}$) loss (see Equations (2) and (6)) that was initially introduced to deal with the mode

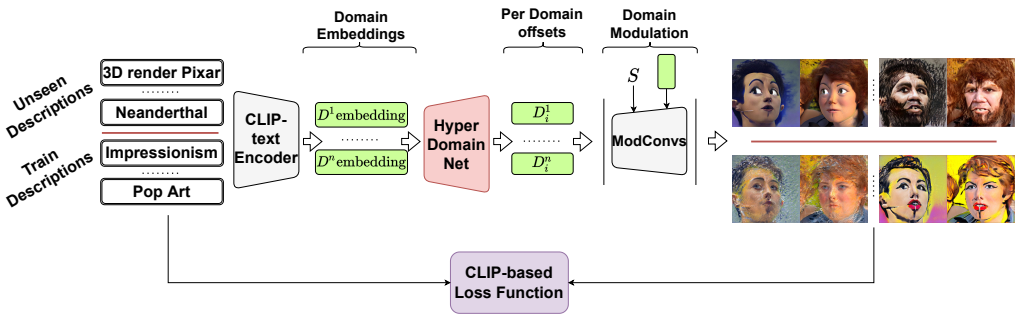


Figure 2: Detailed training process of the HyperDomainNet. On the training phase only reference descriptions are included into CLIP-guided training.

collapsing problem of the fine-tuned generator [5]. However, we empirically observe that it solves the issue only partially. In particular, it preserves the diversity only at the begining of the fine-tuning process and starts collapsing after several hundred iterations. It is a significant problem because for some domains we need much more iterations to obtain the acceptable quality.

The main cause of such undesirable behaviour of the $\mathcal{L}_{direction}$ (the same for $\mathcal{L}_{clip_accross}$) loss is that it calculates the CLIP cosine distance between embeddings that do not lie in the CLIP space. Indeed, the cosine distance is a natural distance for objects that lie on a CLIP sphere but becomes less evident for vectors $\Delta T, \Delta I$ that represent the difference between clip embeddings that no longer lie on a unit sphere. Therefore, the idea behind the $\mathcal{L}_{direction}$ loss may be misleading and in practice we can observe that it still suffers from mode collapse.

We introduce a new regularizer for improving diversity that calculates the CLIP cosine distance only between clip embeddings. We called it *indomain angle consistency* loss and we define it as follows

$$\mathcal{L}_{indomain-angle}(\{G_d^B(w_i)\}_{i=1}^n, \{G^A(w_i)\}_{i=1}^n, B, A) = \tag{10}$$

$$= \sum_{i,j}^{n} (\langle E_I(G^A(w_i)), E_I(G^A(w_j))\rangle - \langle E_I(G_d^B(w_i)), E_I(G_d^B(w_j))\rangle)^2, \tag{11}$$

The idea of $\mathcal{L}_{indomain-angle}$ loss is to preserve the CLIP pairwise cosine distances between images before and after domain adaptation. We observe that this loss significantly improves the diversity of the generator G_d^B compared to the original $L_{direction}$ or $\mathcal{L}_{clip_accross}$ losses.

4.3 Designing the HyperDomainNet for Universal Domain Adaptation

The proposed domain-modulation technique allows us to reduce the number of trainable parameters which motivates us to tackle the problem of multi-domain adaption of StyleGAN2. Our aim is to train the HyperDomainNet that predicts the domain parameters given the input target domain. This problem can be formulated as follows. We are given a trained generator G^A for a source domain A and a number of target domains $B_1, \dots, B_m$ that can be represented by the single image or the text description. The aim is to learn the HyperDomainNet $D_\varphi(\cdot)$ that can predict the domain parameters $d_{B_i} = D_\varphi(B_i)$ which will be used to obtain the fine-tuned generator $G_{d_{B_i}}^{B_i}$ by the domain-modulation operation (see Section 4.1).

In this work, we focus on the setting when the target domains $B_1, \dots, B_m$ are represented by text descriptions $t_{B_1}, \dots, t_{B_m}$. The HyperDomainNet $D_\varphi(\cdot)$ takes as an input the embedding of the text obtained by the CLIP encoder $E_T(\cdot)$ and outputs the domain parameters $d_{B_i} = D_\varphi(E_T(t_{B_i}))$. The training process is described in the Figure 2.

To train the HyperDomainNet $D_\varphi(\cdot)$ we use the sum of $\mathcal{L}_{direction}$ losses for each target domains. In addition, we introduce $\mathcal{L}_{tt-direction}$ loss ("tt" stands for target-target) that is the same as $\mathcal{L}_{direction}$, but we compute it between two target domains instead of target and source. The idea is to keep away the images from different target domains in the CLIP space. We observe that without $\mathcal{L}_{tt-direction}$ loss the HyperDomainNet tends to learn the mixture of domains.

Figure 3: Comparison with the original StyleGAN-NADA [5] method (left) and its version with our parameterization.

In multi-domain adaptation setting, the regularizer $\mathcal{L}_{indomain-angle}$ becomes inefficient because during training batch consists of samples from different domains and the number of images from one domain can be very small. Therefore, we introduce an alternative regularization $\mathcal{L}_{domain-norm}$ for the HyperDomainNet that constrains the norm of the predicted domain parameters. To be exact it equals to $\|D_\varphi(E_T(t_{B_i})) - 1\|^2$.

So, the objective function of the HyperDomainNet consists of $\mathcal{L}_{direction}$, $\mathcal{L}_{tt-direction}$ and $\mathcal{L}_{domain-norm}$ losses. For more detailed description of these losses the overall optimization process, please refer to Appendix A.2.

5 Experiments

In this section, we provide qualitative and quantitative results of the proposed approaches. At first, we consider the text-based domain adaptation and show that our parameterization has comparable quality with the full one. Next, we tackle one-shot domain adaptation and confirm the same quantitatively and also show the importance of the $\mathcal{L}_{indomain-angle}$ loss. Finally, we solve the multi-domain adaptation problem by the proposed HyperDomainNet, show its generalization ability on unseen domains. For the detailed information about setup of the experiments please refer to Appendix A.1.

Text-Based Domain Adaptation We compare the StyleGAN-NADA [5] method with the proposed parameterization and the original version on a number of diverse domains. In Figure 3, we see that the expressiveness of our parameterization is on par with the original StyleGAN-NADA. We observe that the domain-modulation technique allows to adapt the generator to various style and texture changes. For results on more domains please refer to Appendix A.3.

One-Shot Domain Adaptation In this part, we examine our parameterization and the indomain angle consistency loss by applying them to the MindTheGap [45] method. We show qualitative and quantitative results and compare them with other few-shot domain adaptation methods such as

Figure 4: Comparison with one-shot domain adaptation methods. Left block is MindTheGap+indomain and right block is StyleGAN-NADA [45]. The middle block is the MindTheGap+indomain with our parameterization.

StyleGAN-NADA, TargetCLIP [3] and Cross-correspondence [21] method. To assess the domain adaptation quality we use the standard metrics FID [9], precision and recall [15]. As a target domain we take the common benchmark dataset of face sketches [33] that has approximately 300 samples. We consider the one-shot adaptation setting. We provide the results in Table 1. At fisrt, we see that the MindTheGap with our parameterization shows comparable results with the original version while having less trainable parameters by three orders of magnitude. Secondly, we examine the effectiveness of the indomain angle consistency. We show that it considerably improves FID and precision metrics for both the original MindTheGap and the one with our parameterization.

The qualitative results are provided in Figure 4 for MindTheGap+indomain, MindTheGap+indomain with our parameterization ("Ours") and StyleGAN-NADA. For other methods please see Appendix A.4. We observe that MindTheGap+indomain and our version shows comparable visual quality and outperform StyleGAN-NADA in terms of diversity and maintaining the similarity to the source image.

Overall, we demonstrate that our parameterization is applicable to the state-of-the-art methods StyleGAN-NADA and MindTheGap and it can be further improved by the indomain angle consistency loss.

Multi-Domain Adaptation Now we consider the multi-domain adaptation problem. We apply the HyperDomainNet in two different scenarios: (i) training on fixed number of domains, (ii) training on potentially arbitrary number of domains. The first scenario is simple, we train the HyperDomainNet on 20 different domains such as "Anime Painting", "Pixar", etc. (for the full list of domains please

Table 1: Evaluation of one-shot adaptation methods. Results for TargetCLIP, Cross-correspondence and StyleGAN-NADA methods are taken from [45].

	Model quality			**Model complexity**
Model	FID	Precision	Recall	# trainable parameters
TargetCLIP [3]	199.33	0.000	0.293	30M
Cross-correspondence [21]	158.86	0.001	0	30M
StyleGAN-NADA [5]	124.55	0.118	0	24M
MindTheGap [45]	78.35	0.326	0.017	24M
MindTheGap (our param.)	79.83	0.452	0.017	6k
MindTheGap+indomain	71.46	0.503	0.014	24M
MindTheGap+indomain (our param.)	72.71	0.472	0.028	6k

Figure 5: Comparison of training setups. The top row represents the real images embedded into StyleGAN2 latent space which latents are then used for HyperDomainNet inference. The left block represents results obtained from text-descriptions presented in the train list. The right block represents results of HyperDomainNet inference on unseen text-descriptions.

refer to Appendix). The second scheme is more complicated. We fix large number of domains (several hundreds) and calculate its CLIP embeddings. During training we sample new embeddings from the convex hull of the initial ones and use them in the optimization process (see Figure 2). This technique allows us to generalize to unseen domains. For more details about both scenarios please refer to Appendix A.2.

The results of the HyperDomainNet for both scenarios are provided in Figure 5. The left part is results for the first setting, the right one is results for the unseen domains in the second scheme. For more domains and generated images please refer to Appendix A.2. We see that in the first scenario the HyperDomainNet shows results comparable to the case when we train separate models for each domain (see Figure 3). It shows that the proposed optimization process for the HyperDomainNet is effective. The results for the second scenario looks promising. We can observe that the HyperDomainNet has learnt very diverse domains and shows sensible adaptation results for unseen ones.

6 Conclusion

We propose a novel domain-modulation technique that allows us to considerably reduce the number of trainable parameters during domain adaptation of StyleGAN2. We successfully apply it to the state-of-the-art methods. In addition, we introduce a new indomain angle consistency loss that improves the diversity of the fine-tuned generator. Finally, equipped with the efficient parameterization we propose the HyperDomainNet for multi-domain adaptation of StyleGAN2. We empirically show that it can be trained to several domains successfully. And when the number of training domains is large it can generalize to unseen domains.

The main limitation of our approach is that it is not applicable for the cases when target domains are very far from the source one. In such setting, we cannot limit the parameter space, so we should use the full parameterization.

The potential negative societal impacts of domain adaptation of GANs and generally training of GANs include different forms of disinformation, e.g. deepfakes of celebrities or senior officials, fake avatars in social platforms. However, it is the issue of the whole field and this work does not amplify this impact.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.1
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] The reported results of the baselines we compare with have no error bars
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.1
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5
   (b) Did you mention the license of the assets? [Yes] See Appendix A.1
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

References

[1] Yuval Alaluf, Or Patashnik, and Daniel Cohen-Or. Restyle: A residual-based stylegan encoder via iterative refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6711–6720, 2021.

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[3] Hila Chefer, Sagie Benaim, Roni Paiss, and Lior Wolf. Image-based clip-guided essence transfer. *arXiv preprint arXiv:2110.12427*, 2021.

[4] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016.

[5] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *arXiv preprint arXiv:2108.00946*, 2021.

[6] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[8] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. *Advances in Neural Information Processing Systems*, 33:9841–9850, 2020.

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[10] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[12] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114, 2020.

[13] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[14] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[15] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

[16] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[17] Yijun Li, Richard Zhang, Jingwan Lu, and Eli Shechtman. Few-shot image generation with elastic weight consolidation. *arXiv preprint arXiv:2012.02780*, 2020.

[18] Bingchen Liu, Yizhe Zhu, Kunpeng Song, and Ahmed Elgammal. Towards faster and stabilized gan training for high-fidelity few-shot image synthesis. In *International Conference on Learning Representations*, 2020.

[19] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Freeze the discriminator: a simple baseline for fine-tuning gans. *arXiv preprint arXiv:2002.10964*, 2020.

[20] Atsuhiro Noguchi and Tatsuya Harada. Image generation from small datasets via batch statistics adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2750–2758, 2019.

[21] Utkarsh Ojha, Yijun Li, Jingwan Lu, Alexei A Efros, Yong Jae Lee, Eli Shechtman, and Richard Zhang. Few-shot image generation via cross-domain correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10743–10752, 2021.

[22] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2085–2094, 2021.

[23] Justin NM Pinkney and Doron Adler. Resolution dependent gan interpolation for controllable image synthesis between domains. *arXiv preprint arXiv:2010.05334*, 2020.

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[25] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2287–2296, 2021.

[26] Esther Robb, Wen-Sheng Chu, Abhishek Kumar, and Jia-Bin Huang. Few-shot adaptation of generative adversarial networks. *arXiv preprint arXiv:2010.11943*, 2020.

[27] Kim Seonghyeon. Stylegan2-pytorch. *https://github.com/rosinality/stylegan2-pytorch*, 2020.

[28] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9243–9252, 2020.

[29] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. *ACM Transactions on Graphics (TOG)*, 40(4):1–14, 2021.

[30] Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, and Ngai-Man Cheung. On data augmentation for gan training. *IEEE Transactions on Image Processing*, 30:1882–1897, 2021.

[31] Hung-Yu Tseng, Lu Jiang, Ce Liu, Ming-Hsuan Yang, and Weilong Yang. Regularizing generative adversarial networks under limited data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7921–7931, 2021.

[32] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[33] Xiaogang Wang and Xiaoou Tang. Face photo-sketch synthesis and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(11):1955–1967, 2008.

[34] Yaxing Wang, Abel Gonzalez-Garcia, David Berga, Luis Herranz, Fahad Shahbaz Khan, and Joost van de Weijer. Minegan: effective knowledge transfer from gans to target domains with few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9332–9341, 2020.

[35] Yaxing Wang, Chenshen Wu, Luis Herranz, Joost van de Weijer, Abel Gonzalez-Garcia, and Bogdan Raducanu. Transferring gans: generating images from limited data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 218–234, 2018.

[36] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12863–12872, 2021.

[37] Zongze Wu, Yotam Nitzan, Eli Shechtman, and Dani Lischinski. Stylealign: Analysis and applications of aligned stylegan models. *arXiv preprint arXiv:2110.11323*, 2021.

[38] Ceyuan Yang, Yujun Shen, Yinghao Xu, and Bolei Zhou. Data-efficient instance generation from instance discrimination. *Advances in Neural Information Processing Systems*, 34, 2021.

[39] Tao Yang, Peiran Ren, Xuansong Xie, and Lei Zhang. Gan prior embedded network for blind face restoration in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 672–681, 2021.

[40] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[41] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems*, 33:7559–7570, 2020.

[42] Zhengli Zhao, Zizhao Zhang, Ting Chen, Sameer Singh, and Han Zhang. Image augmentations for gan training. *arXiv preprint arXiv:2006.02595*, 2020.

[43] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[44] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. *Advances in neural information processing systems*, 30, 2017.

[45] Peihao Zhu, Rameen Abdal, John Femiani, and Peter Wonka. Mind the gap: Domain gap control for single shot domain adaptation for generative adversarial networks. *arXiv preprint arXiv:2110.08398*, 2021.

[46] Peihao Zhu, Rameen Abdal, Yipeng Qin, John Femiani, and Peter Wonka. Improved stylegan embedding: Where are the good latents? *arXiv preprint arXiv:2012.09036*, 2020.

A Appendix

A.1 Setup of the Experiments

A.1.1 Implementation Details

We implement our experiments using PyTorch[1] deep learning framework. For StyleGAN2 [14] architecture we use the popular PyTorch implementation [2]. We attach all source code that reproduces our experiments as a part of the supplementary material. We also provide configuration files to run each experiment.

A.1.2 Datasets

We use source StyleGAN2 models pretrained on the following datasets: (i) Flickr-Faces-HQ (FFHQ) [13], (ii) LSUN Church, (iii) LSUN Cars, and (iv) LSUN Cats [40]. As target domains we mainly use the text descriptions from [5] and style images from [45]. For quantitative comparison with other methods we use face sketches [33] as the standard dataset for domain adaptation.

A.1.3 Licenses and Data Privacy

Tables 2, 3 provide sources and licenses of the models and datasets we used in our work.

Table 2: Models used in our work, their sources and licenses.

Model	Source	License
StyleGAN2	[14]	Nvidia Source Code License-NC
pSp	[25]	MIT License
e4e	[29]	MIT License
ReStyle	[1]	MIT License
StyleCLIP	[22]	MIT License
CLIP	[24]	MIT License
StyleGAN2-pytorch	[27]	MIT License
StyleGAN-ADA	[12]	Nvidia Source Code License
Cross-correspondenc	[21]	Adobe Research License

Table 3: Datasets used in our work, their sources and licenses.

Dataset	Source	License
FFHQ	[13]	CC BY-NC-SA 4.0[3]
LSUN	[40]	No License
Sketches	[21]	Adobe Research License

A.1.4 Total Amount of Compute Resources

We run our experiments on Tesla A100 GPUs. We used approximately 12000 GPU hours to obtain the reported results and for intermediate experiments.

[1] https://pytorch.org

[2] https://github.com/rosinality/stylegan2- pytorch

A.2 Training of the HyperDomainNet (HDN)

A.2.1 Training Losses

As we describe in Section 4.3 we train HDN $D_\varphi(\cdot)$ using three losses $\mathcal{L}_{direction}$, $\mathcal{L}_{tt-direction}$, and $\mathcal{L}_{domain-norm}$. Each loss is defined as follows:

$$\mathcal{L}_{direction}(G^{B_i}_{d_{B_i}}(w), G^A(w), B_i, A) = 1 - \frac{\Delta I(G^{B_i}_{d_{B_i}}(w), G^A(w)) \cdot \Delta T(B_i, A)}{|\Delta I(G^{B_i}_{d_{B_i}}(w), G^A(w))||\Delta T(B_i, A)|}, \tag{12}$$

$$\mathcal{L}_{tt-direction}(G^{B_i}_{d_{B_i}}(w), G^{B_j}_{d_{B_j}}(w), B_i, B_j) = 1 - \frac{\Delta I(G^{B_i}_{d_{B_i}}(w), G^{B_j}_{d_{B_j}}(w)) \cdot \Delta T(B_i, B_j)}{|\Delta I(G^{B_i}_{d_{B_i}}(w), G^{B_j}_{d_{B_j}}(w))||\Delta T(B_i, B_j)|}, \tag{13}$$

$$\mathcal{L}_{domain-norm}(D_\varphi, B_i) = \|D_\varphi(E_T(t_{B_i})) - 1\|^2 \tag{14}$$

Then the overall training loss for the HDN $D_\varphi(\cdot)$ is

$$\mathcal{L}(\varphi) = \lambda_{direction} \sum_{i=1}^{m} \mathcal{L}_{direction}(G^{B_i}_{D_\varphi(E_T(t_{B_i}))}(w), G^A(w), B_i, A) + \tag{15}$$

$$+ \lambda_{tt-direction} \sum_{i \neq j}^{m} \mathcal{L}_{tt-direction}(G^{B_i}_{D_\varphi(E_T(t_{B_i}))}(w), G^{B_j}_{D_\varphi(E_T(t_{B_j}))}(w), B_i, B_j) +$$

$$+ \lambda_{domain-norm} \sum_{i=1}^{m} \mathcal{L}_{domain-norm}(D_\varphi, B_i) \tag{16}$$

A.2.2 Architecture of the HDN

We use the standard ResNet-like architecture for the HDN. It has the backbone part which has 10 ResBlocks and the part that consists of 17 heads. The number of heads equals the number of StyleGAN2 layers in the synthesis network G_{sys}. Each head has 5 ResBlocks and outputs the domain vector for the corresponding StyleGAN2 layer. We illustrate the overal architecture of the HDN in Figure 6. It has 43M parameters. We use the same architecture for all experiments.

A.2.3 Inference Time

The inference time of the HDN network on 1 Tesla A100 GPU is almost the same as the one forward pass through StyleGAN2 generator which works in 0.02 seconds.

A.2.4 Training on Fixed Number of Domains

For training the HDN on fixed number of domains we use the loss function from Equation (16). As training target domains we take the following 20 domains (we provide in the format "the target domain - the corresponding source domain"):

1. Anime Painting - Photo
2. Impressionism Painting - Photo
3. Mona Lisa Painting - Photo
4. 3D Render in the Style of Pixar - Photo
5. Painting in the Style of Edvard Munch - Photo
6. Cubism Painting - Photo
7. Sketch - Photo
8. Dali Painting - Photo
9. Fernando Botero Painting - Photo
10. A painting in Ukiyo-e style - Photo

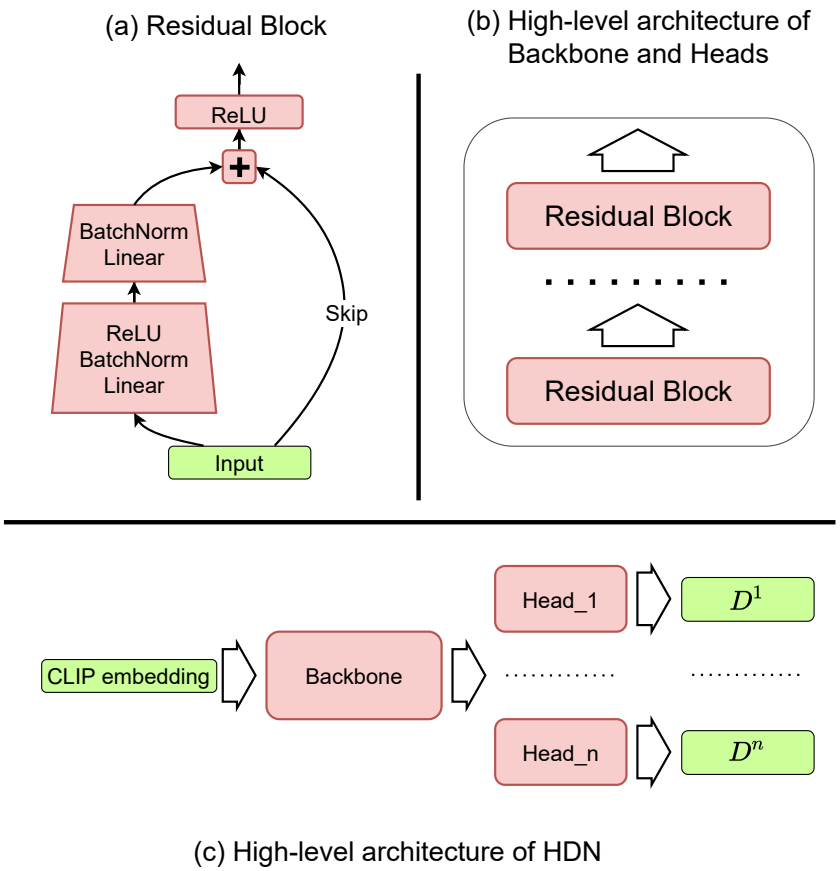


Figure 6: Detailed HDN architecture diagram. (a) - base residual block which is included into backbone and head parts of the HDN. (b) - the detailed backbone and head architecture, each module use the same sequence of ResBlocks. (c) - the detailed architecture of the HDN with data flow.

11. Tolkien Elf - Human
12. Neanderthal - Human
13. The Shrek - Human
14. Zombie - Human
15. The Hulk - Human
16. The Thanos - Human
17. Werewolf - Human
18. Nicolas Cage - Human
19. The Joker - Human
20. Mark Zuckerberg - Human

Hyperparameters For 20 descriptions in traning list following setup is considered. The HDN trained for 1000 number of iterations. Batch size 24 is used. Weights of the terms from Equation (16) as follows: $\lambda_{direction} = 1.0$, $\lambda_{tt-direction} = 0.4$, $\lambda_{domain-norm} = 0.8$. We use two Vision-Transformer based CLIP models, "ViT-B/32" and "ViT-B/16". To optimize HDN we use an ADAM Optimizer with betas$= (0.9, 0.999)$, learning rate$= 5e-5$, weight decay$= 0$.

A.2.5 Training Time

The training time of the HDN on 20 domains for 1000 iterations on single Tesla A100 GPUs takes about 2 hours.

A.2.6 Ablation Study on the Loss Terms

We perform both the quantitative and qualitative ablation study on the domain-norm and tt-direction loss terms that are defined in Appendix A.2.1.

For the qualitative analysis we consider three domains (Anime Painting, Mona Lisa Painting, A painting in Ukiyo-e style) for the HyperDomainNet that was trained on 20 different domains (see the full list in Appendix A.2.4). We provide the visual comparison for these domains with respect to the using loss terms in the training loss of the HyperDomainNet (see Figure 7). We can see that without additional loss terms the model considerably collapses within each domain. After adding domain-norm it solves the problem of collapsing within domains but it starts mix domains with each other, so we obtain the same style for different text descriptions. And after using tt-direction loss eventially allows us to train the HyperDomainNet efficiently on these domains without collapsing.

For the quantitative results we use the metrics Quality and Diversity that were introduced in Appendix A.3.3. The results are provided in Table 4. We see that the initial model without loss terms obtains good Quality but very low Diversity. The domain-norm significantly improves the diversity in the cost of degrading the Quality. The tt-direction provides a good balance between these two metrics which we also we qualitatively in Figure 7.

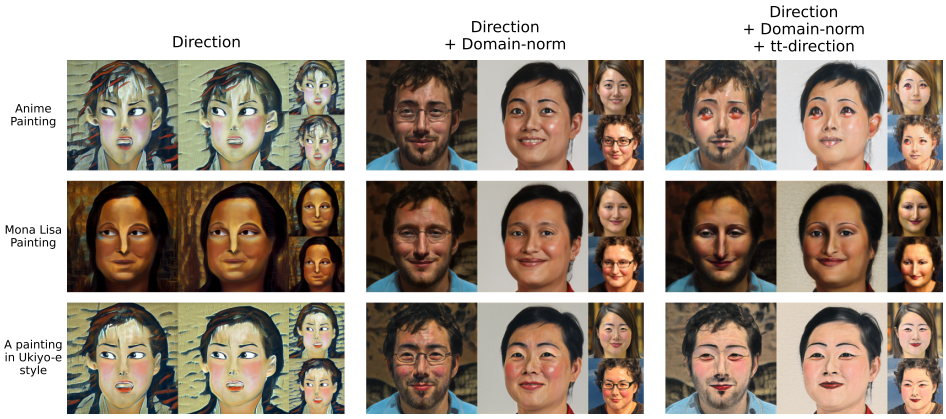


Figure 7: Ablation study on the loss terms of the HyperDomainNet.

Additional Samples We show results for the first 10 domains in Figure 5. The next 10 domains we provide in Figure 8.

A.2.7 Training on Potentially Arbitrary Number of Domains

Improving the generalization ability of the HDN is challenging because it tends to overfit on training domains and for unseen ones it predicts domains that are very close to train ones. One way to tackle this problem is to considerably extend the training set. For this purpose, we use three techniques: (i) generating many training domains by taking combinations of different ones; (ii) sampling CLIP embeddings from convex hull of the initial training embeddings; (iii) resample initial CLIP embeddings given cosine similarity. We discuss each technique further.

Generating Training Domains by Taking Combinations We can describe domains by indicating different properties of the image such as image style (e.g. Impressionism, Pop Art), image type (e.g. Photo, Painting), artist style (e.g. Modigliani). Also we can construct new domains by taking combinations of these properties (e.g. Modigliani Painting, Impressionism Photo). So, we take 32

image styles, 13 image types and 7 artists and by taking all combinations of these properties we come up with 1040 domains. We provide the full list of properties we use in our training:

- image styles: 'Pop Art', 'Impressionism', 'Renaissance', 'Abstract', 'Vintage', 'Antiquity', 'Cubism', 'Disney', 'Chinese', 'Japanese', 'Spanish', 'Italian', 'Dutch', 'German', 'Surreal', 'WaltDisney', 'DreamWorks', 'Modern', 'Realism', 'Starry Night', 'Old-timey', 'Pencil', 'Gouache', 'Acrylic', 'Watercolor', 'Oil', 'Black', 'Blue', 'Charcoal', 'Manga', 'Kodomo';
- image types: 'Portrait', 'Image', 'Photo', 'Painting', 'Graffiti', 'Photograph', 'Cartoon', 'Stereo View', 'Drawing', 'Graphics', 'Mosaic', 'Caricature', 'Animation';
- artists: 'Raphael', 'Salvaror Dali', 'Edvard Munch', 'Modigliani', 'Van Gogh', 'Claude Monet', 'Leonardo Da Vinci'

The algorithm of generating combinations is

Table 4: Ablation study on the loss terms of the HyperDomainNet.

Model	Quality	Diversity
Anime Painting		
Multi-Domain	0.271	0.128
Multi-Domain+domain_norm	0.210	0.338
Multi-Domain+domain_norm+tt_direction	0.260	0.256
Zombie		
Multi-Domain	0.254	0.079
Multi-Domain+domain_norm	0.246	0.203
Multi-Domain+domain_norm+tt_direction	0.258	0.191
Across ten domains		
Multi-Domain	0.275 ± 0.035	0.099 ± 0.026
Multi-Domain+domain_norm	0.218 ± 0.026	0.306 ± 0.040
Multi-Domain+domain_norm+tt_direction	0.247 ± 0.026	0.250 ± 0.041

Figure 8: Other domains which were included into train description list from left block of Figure 5.

```python
def get_train_descriptions(
    base_image_types: list,
    base_image_styles: list,
    artists: list
):
    result_target_domains = []

    for stylization, image_type in product(base_image_styles + artists, base_image_types):
        result_target_domains.append(f"{stylization} {image_type}")

    return result_target_domains
```

Figure 9: The algorithm of generating combinations implemented on **Python**.

Generating CLIP Embeddings from Convex Hull In the usual training of the HDN as in Appendix A.2.4 we use the CLIP embeddings of the target domains $t_{B_1} = E_T(B_1), \ldots, t_{B_m} = E_T(B_m)$. To cover more CLIP space we propose to use new embeddings $t'_{B_1}, \ldots, t'_{B_m}$ from the

convex hull of the initial ones:

$$t'_{B_i} = \sum_{j=1}^{m} \alpha_j t_{B_j},\ i = 1, \dots, m, \tag{17}$$

$$\text{where } \alpha_1, \dots, \alpha_m \sim \text{Dir}(\beta) \text{ (Dirichlet distribution)},\ \sum_{i=1}^{m} \alpha_i = 1,\ \alpha_i \geqslant 0, i = 1, \dots, m. \tag{18}$$

We use $\beta = \dfrac{1}{\textit{batch size}}$.

Resampling Initial CLIP Embeddings Given Cosine Similarity To further extend the CLIP space we cover during training of the HDN we resample initial CLIP embeddings of the target domains $t_{B_1}, \dots, t_{B_m}$ constrained to the cosine similarity. So, before generating from convex hull we replace the initial embeddings by new ones $\hat{t}_{B_1}, \dots, \hat{t}_{B_m}$ such that $\cos(t_{B_1}, \hat{t}_{B_1}) = \gamma$. To obtain these embeddings we use the following operation:

$$\hat{t}_{B_i} = \text{resample}(t_{B_i}),\ i = 1, \dots, m, \tag{19}$$

$$\text{where resample}(t_{B_i}) = t_{B_i} \cdot \cos\gamma + \text{norm}(v - \textit{proj}_{t_{B_i}} v) \cdot \sin\gamma, \tag{20}$$

$$v \sim \mathcal{N}(v|0, \mathbf{I}), \quad \text{norm}(u) = \frac{u}{||v||_2} \tag{21}$$

It allows us to cover the part of the CLIP space outside of the initial convex hull. We observe that it improves the generalization ability of the HDN.

Hyperparameters We train the HDN for 10000 number of iterations. We use batch size of 96. We set weights of the terms from Equation (16) as follows: $\lambda_{direction} = 1.0$, $\lambda_{tt-direction} = 0.4$, $\lambda_{domain-norm} = 0.8$. We use two Vision-Transformer based CLIP models, "ViT-B/32" and "ViT-B/16". To optimize HDN we use an ADAM Optimizer with betas= $(0.9, 0.999)$, learning rate= $5e{-}5$, weight decay= 0.

Training Time The training time of the HDN for 10000 iterations on 4 Tesla A100 GPUs takes about 50 hours.

Additional Samples Additional samples of unseen domains for the HDN is demonstrated in Figure 10.

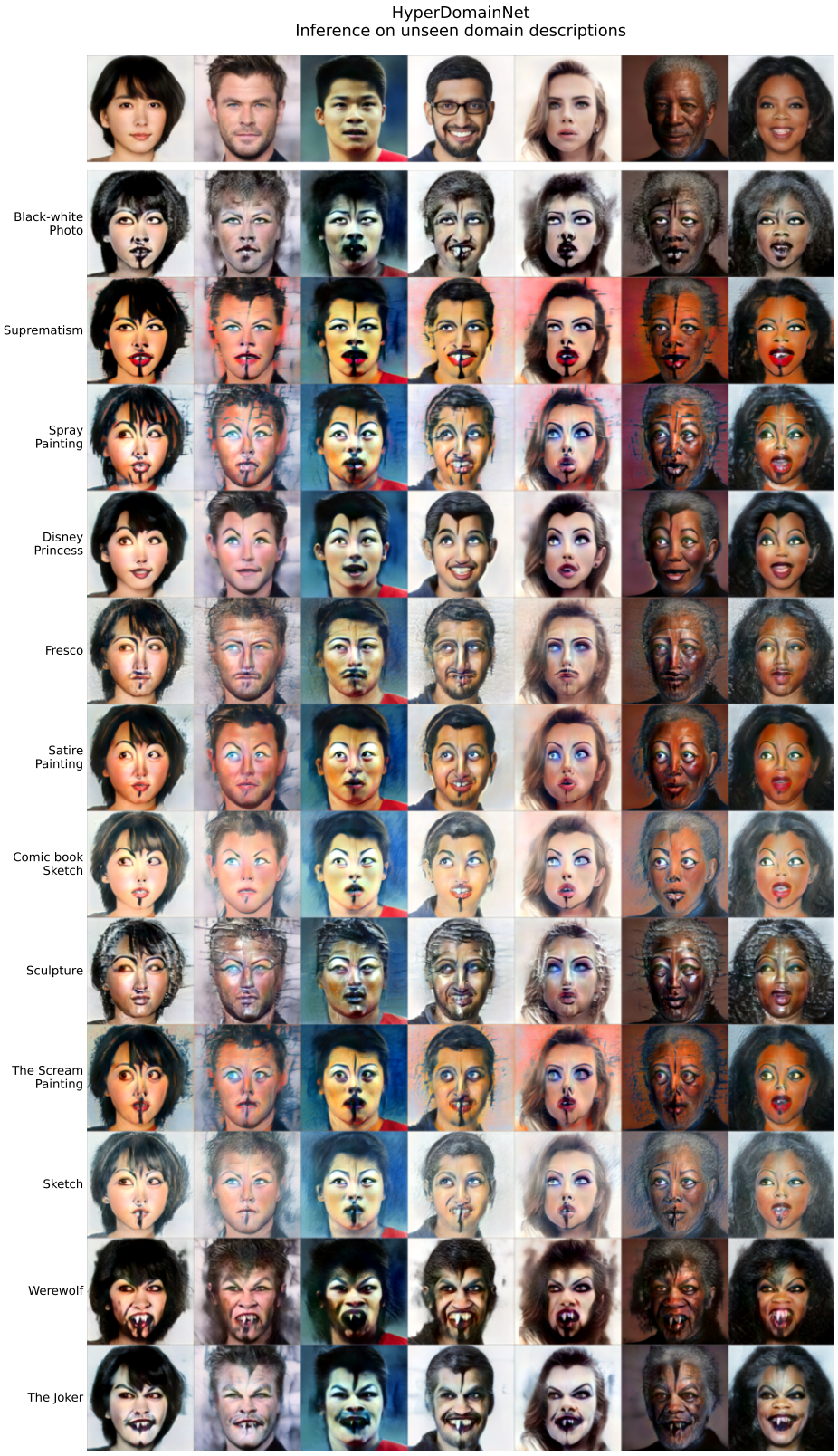


Figure 10: Other visual results for descriptions which were not included into training list during HDN training.

A.3 Results for Text-Based Domain Adaptation

A.3.1 Hyperparameters

StyleGAN-NADA with our parameterization trained for 600 iterations with batch size of 4. Style mixing probability is set to 0.9, the weight of the $\mathcal{L}_{direction}$ is 1.0 and $\mathcal{L}_{indomain-angle}$ is 0.5 and ADAM optimizer with betas$= (0., 0.999)$, learning rate$= 0.002$, weight decay$= 0$.

For the original StyleGAN-NADA [5] number of iterations is decreased to 200 because for more iterations it starts to collapse.

"ViT-B/32" and "ViT-B/16" CLIP Vision-Transformed models used in all setups.

A.3.2 Training and Inference Time

The training of the one target domain for 600 iterations on a single Tesla A100 GPU takes about 15 minutes on batch size 4.

The inference time consists of two parts. The first one is the embedding process of the real image which takes 0.23 seconds using ReStyle [1]. The second part is the forward pass through adapted GAN generator which works in 0.02 seconds.

A.3.3 Quantitative Results

We provide the quantitative comparison for the text-based domain adaptation by evaluating the "Quality" and "Diversity" metrics in a straightforward way.

As the “Quality” metric we estimate how close the adapted images to the text description of the target domain. That is we compute the mean cosine similarity between image CLIP embeddings and the embedding of the text description:

$$\text{Quality} = \frac{1}{n}\sum_{i=1}^{n}\langle E_T(\text{target_text}), E_I(I_i)\rangle, \text{ where} \tag{22}$$

$$n \text{ - number of the generated adapted images (we use 1000),}$$
$$E_T \text{ - text CLIP encoder,}$$
$$E_I \text{ - image CLIP encoder,}$$
$$I_1, \ldots, I_n \text{ - generated adapted images.}$$

As E_I encoder we use only ViT-L/14 image encoder that is not applied during training (in the training we use ViT-B/16, ViT-B/32 image encoders).

As the “Diversity” metric we estimate the mean pairwise cosine distance between all adapted images:

$$\text{Diversity} = \frac{2}{n(n-1)}\sum_{i<j}^{n}(1 - \langle E_I(I_i), E_I(I_j)\rangle), \text{ where} \tag{23}$$

$$n \text{ - number of the generated adapted images (we use 1000),}$$
$$E_I \text{ - image CLIP encoder,}$$
$$I_1, \ldots, I_n \text{ - generated adapted images.}$$

We compute these two metrics for the ten text domains: Anime Painting, Mona Lisa Painting, 3D Render Pixar, Sketch, Ukiyo-e Painting, Fernando Botero Painting, Werewolf, Zombie, The Joker, Neanderthal. We separately report metrics for two domains Anime Painting and Zombie to better reflect the metrics behaviour. Also we report the overall metrics across all nine domains. The results are provided in Table 5.

From these results we see that our model performs comparably with the StyleGAN-NADA with respect to Quality while having better Diversity. Also we can observe that the indomain angle loss significantly improves the Diversity for both models StyleGAN-NADA and Ours while lightly decreases the Quality.

For the multi-domain adaptation model we see that it has lower diversity than StyleGAN-NADA and Ours and comparable Quality while being adapted to all these domains simultaneously.

Also we report samples for the StyleGAN-NADA and our model with and without indomain angle loss in Figures 11 and 12. We see that qualitatively indomain angle loss also significantly improves the diversity of the domain adaptation methods.

A.3.4 Additional Samples

We show additional domains for FFHQ dataset in Figure 13. Also we demonstrate how our method works on another datasets such as LSUN Church in Figure 14, LSUN Cats in Figure 15, and LSUN Cars in Figure 16.

Table 5: Evaluation of text-based adaptation methods.

Model	Quality	Diversity
Anime Painting		
StyleGAN-NADA [5]	0.289	0.244
Ours	0.284	0.305
StyleGAN-NADA+indomain	0.256	0.401
Ours+indomain	0.251	0.404
Multi-Domain+domain_norm+tt_direction	0.260	0.256
Zombie		
StyleGAN-NADA [5]	0.257	0.153
Ours	0.264	0.275
StyleGAN-NADA+indomain	0.261	0.354
Ours+indomain	0.247	0.372
Multi-Domain+domain_norm+tt_direction	0.258	0.191
Across ten domains		
StyleGAN-NADA [5]	0.270 ± 0.032	0.196 ± 0.034
Ours	0.256 ± 0.019	0.306 ± 0.030
StyleGAN-NADA+indomain	0.249 ± 0.018	0.394 ± 0.026
Ours+indomain	0.240 ± 0.018	0.398 ± 0.026
Multi-Domain+domain_norm+tt_direction	0.247 ± 0.026	0.250 ± 0.041

Figure 11: Comparison of text-based domain adaptation methods without indomain angle loss. Left column represents StyleGAN-NADA [5], right column represents our model.

Figure 12: Comparison of text-based domain adaptation methods with indomain angle loss. Left column represents StyleGAN-NADA [5], right column represents our model.

Figure 13: Comparison for other domains in single domain adaptation setup on real images. Left column represents StyleGAN-NADA [5], right column represents results with same approach patched with ours parameterization.

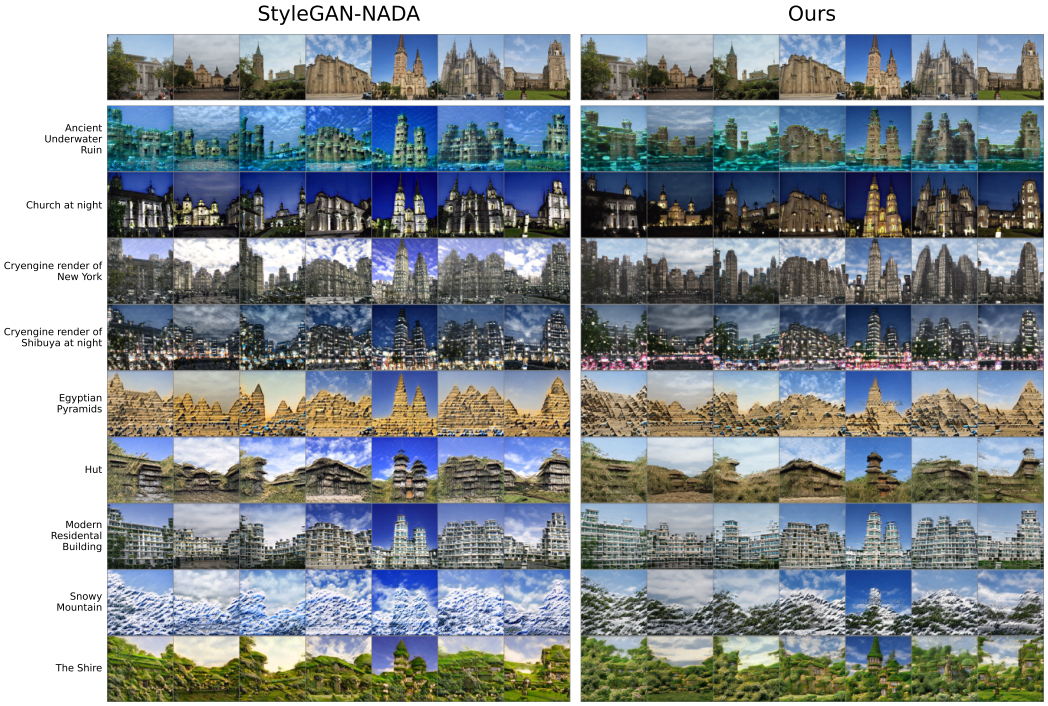


Figure 14: Single domain adaptation comparison for LSUN Church dataset.

Figure 15: Single domain adaptation comparison for LSUN Cats dataset.

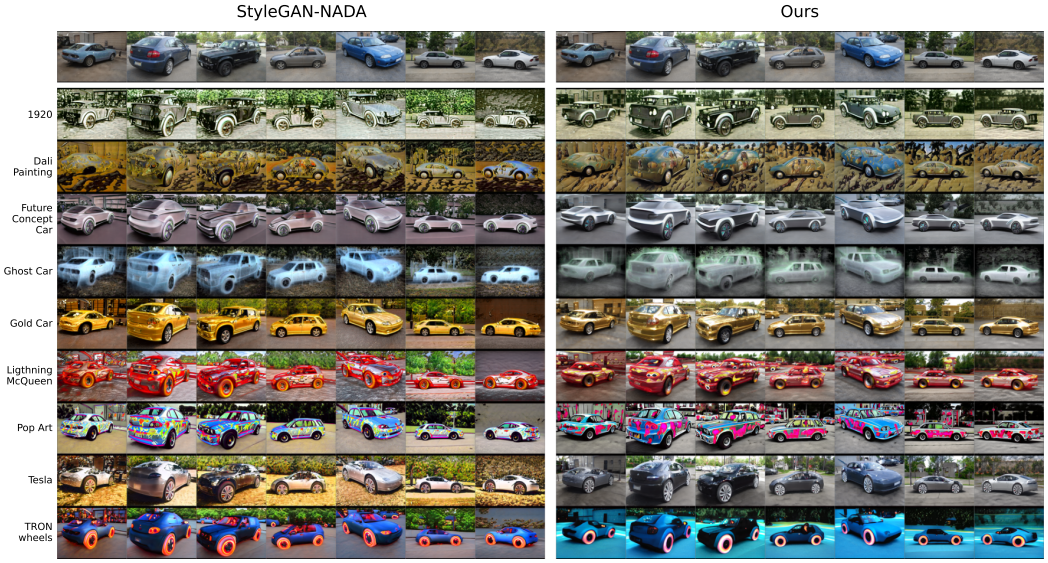


Figure 16: Single domain adaptation comparison for LSUN Cars dataset.

A.4 Results for One-Shot Domain Adaptation

A.4.1 Hyperparameters

For each target style image we adapt the generator for 600 iterations as in [45]. We use batch size of 4, fine-tune all layers of the StyleGAN2, set the mixing probability to 0.9. We use all loss terms as in [45] with the same weights and add the $\mathcal{L}_{indomain-angle}$ term with weight 2. For all experiments, we use an ADAM Optimizer with a learning rate of 0.002.

A.4.2 Training and Inference Time

The training of the one target style image for 600 iterations on a single Tesla A100 GPU takes about 20 minutes. The same as for the text-based adaptation the inference time consists of two parts: embedding process and the forward pass through the generator. The embedding process takes 0.36 seconds for e4e [29] and two minutes for II2S [46]. The second part is the forward pass through adapted GAN generator which works in 0.02 seconds.

A.4.3 Additional Samples

We provide additional samples in Figures 17 and 18. Also we provide results for other baseline methods in Figure 19.

Figure 17: Comparison of one-shot domain adaptation methods: original MindTheGap [45] (left), MindTheGap + indomain (center) and MindTheGap with our parameterization (right).

Figure 18: Comparison of one-shot domain adaptation methods: original MindTheGap [45] (left), Multi-Domain model (center) and StyleGAN-NADA (right).

Figure 19: Additional comparisons with other baseline methods including TargetCLIP [3], Gatys et al. [6], and AdaIN [10]. Compare these results to our method in **??**. We can see that both the original MindTheGAP and with our parameterization has fewer artifacts.