Width-Wise Parameter Sharing for Multi-Domain GAN Learning

Ryan Webster, Julien Rabin, Loïc Simon, Frédéric Jurie

WIDTH-WISE PARAMETER SHARING FOR MULTI-DOMAIN GAN LEARNING

Ryan Webster⋆ *Julien Rabin*⋆ *Loïc Simon*⋆ *Frédéric Jurie*⋆

⋆ Normandie Univ., UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France
ryan.webster@unicaen.fr, julien.rabin@unicaen.fr, loic.simon@ensicaen.fr, frederic.jurie@unicaen.fr

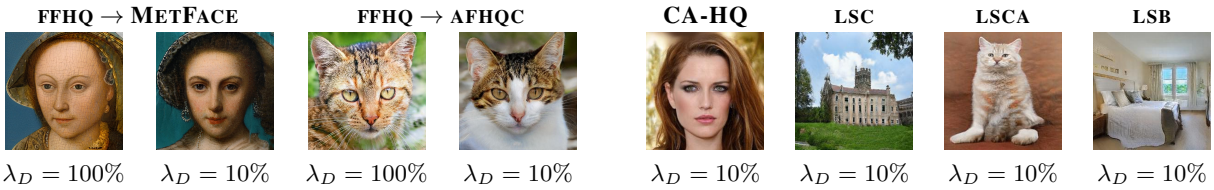

Fig. 1: *Near state of the art generation quality can be obtained on transfer learning with ten time less trainable parameters (left two examples). Our parameter sharing procedure can also learn multiple domains simultaneously with higher quality than the state of the art (right 4 images). Here, λ_D refers to the ratio of trainable parameters per domain, see Sec. 3*

ABSTRACT

In this work, we propose a new parameter efficient sharing method for the training of GAN generators. While there has been recent progress in transfer learning for generative models with limited data, they are either limited to domains close to the original one, or adapt a large part of the parameters. This is somewhat redundant, as the goal of transfer learning should be to re-use old features. In this way, we propose width wise parameter sharing, which can learn a new domain with ten times fewer trainable parameters without a significant drop in quality. Previous approaches are less flexible than our method and also fail to preserve image quality for challenging transfers. Finally, as our goal is ultimately parameter re-use, we show that our method performs well in the multi-domain setting, wherein several domains are learned simultaneously with higher visual quality than the state of the art StarGAN-V2.

Index Terms— Image generation, GANs, Distributed learning, Network compression

1. INTRODUCTION

Recent years has seen substantial progress on improving image generation quality. Notably, the StyleGAN2 network substantially improved the state of the art for image generation and can fool even human observers at high resolutions [1]. Even so, StyleGAN2 is a resource intensive network with millions of parameters even if only for a single domain or with few images. As such, the research focus for image generation has shifted to more specialized tasks aiming at increasing the efficiency or usability of these generators. For instance several works have addressed transfer learning between datasets [2, 3, 4] or learning with limited data [4, 5]. Indeed, the *status quo* for GANs is to re-train from scratch millions of parameters for every new image dataset. Clearly this is a poor approach, as even disparate image classes can contain commonalities, such as color and texture distributions. Several methods have employed explicitly freezing discriminator layers when training a new generator on similar data [4, 3], which helps train on limited data by regularizing the learning problem and converging faster as less parameters need to be learned. Similarly, MineGAN [2] appends layers to a generator and freezes the original generator parameters.

In this work we address to what extent image features can transfer across datasets. Unlike previous work, we consider capacity needed to perform transfer learning and multi domain learning in terms of trainable parameters per domain. This is useful because after training, one only needs to store and distribute the domain specific parameters. For multi domain learning, one also has a shared representation which contains features from many distributions. As an additional result, we demonstrate intra layer redundancy of Stylegan2 parameters. In summary, we provide the following contributions :

— We demonstrate a parameter sharing method for GAN generators, which we call *width-wise* sharing, summarized in Fig. 2 and detailed in Sec. 3. Naive sharing strategies used in previous works reported in Sec. 2, such as sharing by layer, are unable to generate realistic samples with small parameter budgets. In Sec. 4, we show our sharing procedure can generate new images with ten times less trainable parameters without significant drop in image quality, as illustrated

in Fig. 1.

— We train a model on multiple image generation benchmarks simultaneously. Our method outperforms the baseline multi domain image generation method of StarGAN-V2 in terms of FID.

2. PREVIOUS WORK

We start by reviewing several recent works related to transfer learning with GANs and parameter sharing in generative networks.

2.1. Generative Transfer Learning

Recently, transfer learning has been shown to be effective in the generative setting to address the problems of limited data training and slow training time [2, 4, 5]. The quality of state of the art GANs, such as StyleGAN2 or BigGAN [6], degrades significantly with "small" amounts of training data, which can even be as large as 1k-5k samples. This significantly limits the applicability of such methods in real world scenarios.

Various approaches have addressed this problem via transfer learning. In [3], several input layers of the *discriminator* are frozen and the generator is learned from scratch. In MineGAN [2], knowledge is transferred from a previously learned generator to a new domain by either freezing all layers of the generator and relearning a compact MLP layer appended to the input, or by finally allowing fine tuning after an initial learning step. In this work, we consider only the setting where the source generator parameters are frozen, as we consider parameter efficiency and finetuning corresponds to fully relearning the generator. MineGAN is attractive in that only a very compact set of parameters needs to be stored, alongside the original generator to generate new data. However, we will demonstrate that MineGAN is unable to tackle difficult transfer problems, when the new data distribution is far from the original distribution. Finally, we note the work of [7], which only retrains layers near the input of the source generator, and then only relearns filter statistics of subsequent frozen layers, which they dub "AdaFM," due to it's similarity to the AdaIN layers in Stylegan [1]. While this is in some ways similar to our layerwise strategy (see next section), we do not consider this method directly as they re-learn a substantial portion of the generator parameters.

2.2. Learning GANs on Multiple Domains

Many problems seek to train GANs to generate data from many different domains. In conditional image generation, one seeks to generate potentially thousands of classes, each containing a small amount of samples [6, 8]. In BigGAN for example, class specific generation is controlled by an embedding layer at the input, and all other parameters are shared amongst every class. In domain translation, one seeks to translate samples from domain to another with a generative model, typically with a GAN training loss [9, 10, 11]. For instance in StarGAN-V2 [10], each image domain shares a common representation with one another, and each domain has specific style layers. Then samples from each domain can be translated simply by replacing the style. These methods have a similar objective to this work in that they promote models with parameter re-use across different domains. Finally, also related is the task of lifelong learning, wherein data is learned in an online fashion and may be subject to domain shift [12, 13].

In this work, we consider two settings where models share parameters to generate data from different domains. The first is transfer learning where most parameters are simply re-used as is (frozen) while the remaining ones are adapted to generate a new domain. We also consider a use-case wherein multiple domains are learned simultaneously with the majority of parameters shared. However, we believe our parameter sharing procedure, introduced in the next section, can be applied to many of the multi-domain settings introduced herein.

3. SHARING STRATEGIES FOR TRANSFER LEARNING

We consider transferring knowledge from a GAN generator/discriminator pair (G_1, D_1) trained on dataset $\mathcal{S}_1$ to a GAN (G_2, D_2) trained on dataset $\mathcal{S}_2$ (denoting henceforth the transfer as $\mathcal{S}_1 \rightarrow \mathcal{S}_2$). We initialize training of (G_2, D_2) with the parameters of (G_1, D_1). Then, a subset of parameters are frozen to force the new generator to use knowledge from the old one. Shared (frozen) parameters are referred to as θ_S and domain (trainable) parameters are referred to as θ_D. As ultimately we wish to train G_2 with as few parameters as possible, we measure the number of new parameters needed to train G_2 by simply measuring the ratio of trainable parameters, which we refer to as the learn ratio λ_D, *i.e.* $\lambda_D = |\theta_D|/(|\theta_D| + |\theta_S|)$. Note that λ_D is also the ratio of memory needed to store G_2 ; if one has already stored G_1, G_2 can be constructed by loading θ_S from G_1. We detail several strategies for splitting capacity of G_2 amongst θ_D and θ_S given λ_D in the following paragraphs.

MineGAN. In MineGAN [2], MLP layers are appended to the input of the generator and all other parameters are frozen. Therefore the trainable parameters in this layer are θ_D. Furthermore, θ_S =θ is simply all original parameters frozen at the start of training. While in [2], the generator may also be fine tuned after the MLP layers are learned, we consider only the case where no finetuning takes place, as we are investigating the parameter efficiency of different sharing procedures. Thus, MineGAN will only refer to the learned MLP layers with every other layer frozen.

Layer wise sharing. As a simple baseline, we explore simply choosing entire layers of StyleGAN2 to be within θ_D

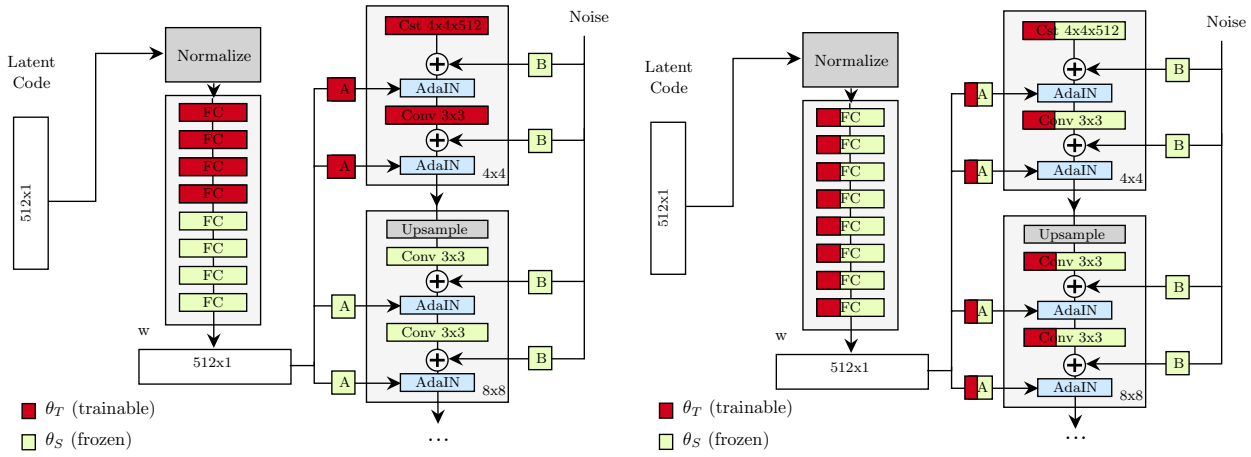

Fig. 2: Sharing strategies illustrated on the StyleGAN network. Left : layerwise sharing, a fixed ratio of the style / convolutional layers are domain specific (red) and the rest are shared between models (green). Right : widthwise sharing, in each style and convolutional layer a fixed ratio of the parameters are specific (red) and the remainder are shared (green).

or not. We consider two configurations for $\lambda_D = 1\%$ and λ_D=10%. For $\lambda_D = 1\%$, we choose only the first two "style" MLP layers in the mapping network and for $\lambda_D = 10\%$ we choose the first four style layers and first convolutional block (see Fig. 2). We chose to favor early layers as they have more of a global influence on generation, however the choice of layers to share is not trivial and is part of the drawback of a layer wise approach. Note that several methods in the literature use the early layers of Stylegan2 as the domain specific layers [10, 14]. Note that both StarGAN-V2 [10] and [14] are layer wise procedures, which learn the MLP layers of Stylegan2 as the domain specific parameters and then share the generator network. For our multi domain learning experiments, discussed in Sec. 4.5, we train StarGAN-V2 using the official code, and then only use the networks for generating random samples. Note as well that unconditional generation from each domain is one of the objectives for StarGAN-V2, so it serves as a decent comparison for our setting.

Width wise sharing. In an effort to give a more flexible sharing strategy w.r.t. the parameter budget and also spread capacity through the various resolutions of the network, we propose splitting *each layer* into trainable and shared parameters. Note that in the related domain of continual learning for classification, existing models are augmented by appending new filters

To do this, we split weight tensors in each layer according to λ_D, taking the first dimensions to be θ_D and the second part to be θ_S. The style blocks in StyleGAN2 are based off adaptive instance normalization [15] with an affine layer taking as input style codes and applying a gain to convolution channels. The affine layer contains a $I \times I$ matrix, we take the first $\lfloor \lambda_D I \rfloor$ columns to be θ_D. Similarly for convolution layers we take the first $\lfloor \lambda_D I \rfloor$ input filters to be θ_D. In practice, these weight tensors are kept separate where θ_D and θ_S are concatenated on the forward pass. In this way, our method can be added in place to any generator, without changing output. See the rightmost diagram of Fig. 2 for more details.

3.1. Sharing for multi-domain learning

In the previous section, we discussed sharing in a transfer learning setting where parameters θ_S are merely frozen. In this work, we also explore a distributed setting where multiple domains $\mathcal{S}_1$,$\mathcal{S}_2$,$\mathcal{S}_3$,... are learned *simultaneously*. In this case θ_S is learned along with θ_D, however θ_S is trained in a distributed fashion wherein gradients are synchronized at each optimizer update. Thus, the downstream distribution and storage of the network still depends on λ_D, as it pertains the number of unique parameters needed for each generator.

4. EXPERIMENTAL RESULTS

In this section, we view generation results for the various training strategies detailed in Sec. 3, namely MineGAN, layerwise sharing and finally our width-wise sharing method for the transfer learning problem.

We indicate by $\mathcal{S}_1 \rightarrow \mathcal{S}_2$ the transfer learning from a domain $\mathcal{S}_1$ to a new domain $\mathcal{S}_2$, where only a fraction λ_D of the generative network parameters are retrained on the new domain. Various settings are tested for FFHQ $\rightarrow$ METFACE and FFHQ $\rightarrow$ LSC for different methods : MineGAN (where λ_D is close to 1%), the naive layer-wise approach and the proposed width-wise sharing for $\lambda_D = 10\%$, and finally full training the generator ($\lambda_D = 100\%$). FID values for corresponding experiments are also reported in Table 1.

One could expect the transfer of FFHQ $\rightarrow$ METFACE to be easier than the transfer of FFHQ $\rightarrow$ LSC, because while the color and textures within METFACE are different from FFHQ,

the datasets still comprises registered faces, and contains similar attributes to FFHQ. Still, the results show that MineGAN is unable to produce plausible samples for METFACE, while layerwise and width-wise training are both able to handle this transfer. On the other hand, FFHQ → LSC requires the generator to learn completely new large scale structures. In this case, while layerwise training can produce plausible backgrounds and colors, the objects (churches) are blurry and unrecognizable. Surprisingly, even with ten times less learnable parameters, the width-wise sharing method is able to handle this difficult transfer, both in terms of the visual quality of samples (see Fig. 1) and the FID values provided in Table 1.

	FFHQ→METFACE		FFHQ→LSC	
strategy	λ_D (%)	FID	λ_D (%)	FID
fully trainable	*100*	*22.1*	*100*	*10.7*
MineGAN [1]	≈1	59.5	≈1	254
layer-wise	10	50	10	191
	1	52.1	1	136
width-wise	10	**27.8**	10	**22.5**
	1	**39.2**	1	**31.1**

Table 1: FID values for different sharing strategies under various parameter budgets λ_D. Both MineGan [2] and layer-wise strategies fail for the difficult transfer of faces to churches FFHQ → LSC, as indicated by the high FID's over 100, contrary to the proposed width-wise approach. See the supplementary material for visual comparison.

4.1. Sharing strategies for Multi-domain distributed learning

In the previous section, we explored transfer learning from a single source domain to a target domain and froze some source generator parameters to enforce parameter sharing. In this section, we'll explore learning multiple domains simultaneously. In some generative applications, parameter sharing is built in to the learning procedure. For example, in domain translation, a shared representation is used to translate samples from one domain to another [10, 15, 9]. Here, we'll compare to the state of the art in image translation network StarGAN-V2 [10] and only use the final network for unconditional image generation. As was discussed in Sec 4, StarGAN-V2 takes θ_D to be the MLP mapping layers in the Stylegan2 network, and θ_S to be all other parameters, which is equivalent to a layerwise strategy. Four our sharing strategies, we'll learn several domains $\mathcal{S}_1, \mathcal{S}_2, ...\mathcal{S}_n$ whilst learning both θ_S (shared amongst every domain) along with θ_D. Note that in this setting, the overall memory needed to store all parameters is $n|\theta_D| + |\theta_S|$, and thus λ_D is an important parameter effecting the bandwidth needed for parameter updates (if training is done in a distributed parameter-server paradigm) and distribution of the network after training.

Table 2 compares FID scores for StarGAN-V2, layerwise and width-wise sharing strategies when training on 4 datasets (CA-HQ, LSC, LSCA, LSB). In the case of $\lambda_D = 1\%$, width-wise sharing beats layerwise by a large margin for every dataset. When $\lambda_D = 10\%$, the two sharing methods are more comparable. This is somewhat to be expected ; a particular choice of layers for layerwise sharing might perform better for a specific set of datasets, as style layers each control different image semantics and particular layers may represent commonalities better than others (see Sec. 3). On the other hand, when λ_D is too small, the choice of layers is limited and we chose input layers as it reflects the sharing in BigGAN or MineGAN. StarGAN-V2 performs comparably on 3 datasets and has generator collapse on CelebA-HQ. This is likely because it is the most different of the three datasets, and the extra constrain of domain translation has forced the shared representation to favor similar domains. Width-wise is clearly a better approach in general as it spreads domain specific capacity throughout the network. We note that good performance with low λ_D desirable because as with the previous setting it effects the downstream storage and distribution of parameters.

Strategy	Layer-Wise		StarGAN-V2	Width-Wise	
λ_D	1	10	10	1	10
CHQ	23.71	18.84	254.7	**17.06**	**15.08**
LSC	17.29	15.49	21.31	**14.45**	**14.46**
LSCA	39.40	43.43	47.24	**36.44**	**32.05**
LSB	19.14	18.67	19.15	**17.29**	**13.56**

Table 2: FID values for different sharing strategies under various parameter budgets λ_D for multi-domain distributed learning (here 4 datasets simultaneously). Note that width-wise has the highest quality samples for each domain, across both parameter budgets. See the supplementary material for visual comparison.

5. CONCLUSION

In this work, we addressed the problems of transfer learning and multi domain learning with a GAN generator. We proposed the simple yet effective weight sharing method of width wise sharing. This method was more efficient at transfer learning than currently the currently proposed method of MineGAN when constrained to a set parameter budget. Finally, we used our width wise sharing method to learn multiple domains simultaneously, and again showed that our method was more effective than a state of the art method StarGAN-V2, when simultaneously learning several challenging datasets.

Acknowledgment This work is partially supported by the project ANR-19-CHIA-0017.

6. REFERENCES

[1] 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*, 2020, pp. 8110–8119.

[2] 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 (CVPR)*, June 2020.

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

[4] 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*, vol. 33, 2020.

[5] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han, "Differentiable augmentation for data-efficient gan training," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan, "Large scale gan training for high fidelity natural image synthesis," in *International Conference on Learning Representations*, 2018.

[7] 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*, vol. 33, 2020.

[8] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*. PMLR, 2019, pp. 7354–7363.

[9] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 172–189.

[10] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha, "Stargan v2 : Diverse image synthesis for multiple domains," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[11] 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*, 2017, pp. 2223–2232.

[12] Chenshen Wu, Luis Herranz, Xialei Liu, Joost van de Weijer, Bogdan Raducanu, et al., "Memory replay gans : Learning to generate new categories without forgetting," *Advances in Neural Information Processing Systems*, vol. 31, pp. 5962–5972, 2018.

[13] Yulai Cong, Miaoyun Zhao, Jianqiao Li, Sijia Wang, and Lawrence Carin, "Gan memory with no forgetting," *arXiv preprint arXiv :2006.07543*, 2020.

[14] Miaoyun Zhao, Yulai Cong, and Lawrence Carin, "On leveraging pretrained gans for generation with limited data," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11340–11351.

[15] 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*, 2017, pp. 1501–1510.