TinyStarGAN v2: Distilling StarGAN v2 for Efficient Diverse Image Synthesis for Multiple Domains

Paras Kapoor
p_apoor@encs.concordia.ca
Tien D. Bui
http://explore.concordia.ca/tien-d-bui

Gina Cody School of Engineering
and Computer Science,
Concordia University,
Montreal, Canada

Abstract

Image-to-image translation models, such as StarGAN v2, have enabled the translation of diverse images over multiple domains in a single framework. However, StarGAN v2 is computationally expensive making it challenging to execute on resource-constrained environments. To reduce the computation requirement of StarGAN v2 while maintaining accuracy, we propose a novel cross distillation method that is specially designed for knowledge distillation (KD) of multiple networks in a single framework. By leveraging this new KD method, the knowledge of a multi-network large teacher StarGAN v2 can be effectively transferred to a small student TinyStarGAN v2 framework. Without losing the quality and diversity of generated images, we reduce the size of the original framework by more than 20× and the computation requirement by more than 5×. Experiments on CelebA-HQ and AFHQ datasets show the effectiveness of the proposed method.

1 Introduction

Recently, the image-to-image translation problem has received considerable attention under rapid improvement in the resolution and quality of images produced by generative adversarial networks (GANs) [7]. Multi-modal multi-domain image-to-image translation [4, 5, 19] refers to the process of generating diverse images across multiple domains given a single input image. Generally, the output images look similar to the original image in some attributes (e.g. pose, identity) yet reflecting certain attributes of the target domain (e.g. gender, hair color). The current state-of-the-art method for multi-modal multi-domain image-to-image translation is StarGAN v2 [5], which generates images with rich styles across multiple domains. However, StarGAN v2 has a computation bottleneck at a parameter count of more than 50M and consumes more than 60G MACs (Multiply-Accumulate Operations to quantify computation cost, 1 MAC = 2 FLOPs) to produce one 256×256 image, preventing its widespread adoption.

Immense efforts have been made recently to compress and speed-up GANs. For example, GAN Compression [14] utilized neural architecture search (NAS) [20] via weight sharing and knowledge transfer of intermediate representations of original model to obtain a compressed model. Autogan-distiller [6] performed differentiable NAS under a target compression ratio, which preserves the original GAN generation quality via the guidance of knowledge distillation [9]. GAN Slimming [16] integrated model distillation, channel pruning and quantization together with the GAN minimax objective that can be efficiently optimized from end to end. Distilling portable GANs for image translation [3] developed on the idea of minimizing the pixel-wise and perceptual difference between images generated by student and teacher networks. Co-evolutionary compression for unpaired image translation [15] obtained compact and effective generators by iteratively pruning least important convolution filters in a dual generator setting. Compressing GANs based on knowledge distillation [1] used mean square error between the images generated from the student and the teacher along with regular GAN training as a joint student training loss.

Although these approaches can provide very high compression and speed-up ratios with slight degradation in performance, they are not straightforwardly applicable to StarGAN v2, because of the following two major reasons:

1. StarGAN v2 is a multi-network framework [5] and existing techniques [1, 3, 6, 14, 15, 16] can only compress a single generator network.

2. Most methods [3, 6, 14, 15, 16] work on deterministic generator networks whereas StarGAN v2 generates diverse multi-modal images from a single source image using random Gaussian noise.

As a solution to this problem, we propose Tiny-StarGAN v2, which achieves similar performance to StarGAN v2 while requiring a smaller computation budget. To the best of our knowledge, we are the first to compress a multi-network non-deterministic framework using knowledge distillation. Our contributions are three-fold:

1. We design efficient networks composed of Depthwise separable convolution [10] layers with reduced channels.

2. We develop a combination of different cross distillation losses operating on different modules of StarGAN v2 to be used along with the original objective in a GAN minimax optimization setting.

3. We achieve outstanding results on benchmark datasets.

2 BACKGROUND

2.1 StarGAN v2

StarGAN v2 [5] generates diverse images across multiple domains. The framework of StarGAN v2 comprises of four modules.

- **Generator** transforms an input image to an output image reflecting the input style code of a specific domain.

- **Discriminator** recognizes input image as genuine or fake for multiple domains.

- **Style encoder** extracts the style code from an input image and its domain label, allowing the generator to perform reference-guided image synthesis.

- **Mapping network** converts latent code sampled from Gaussian noise into style codes for multiple domains, allowing the generator to perform latent-guided image synthesis.

All the modules but the generator are multi-headed, with number of heads equal to the number of domains.

2.2 Knowledge Distillation

Knowledge distillation [9] is a powerful knowledge transfer method that enables a single network with a relatively low number of parameters to learn from an ensemble of networks or from a network with large number of parameters. The bigger model is referred as the teacher and the smaller model as the student. Several methods leverage response-based knowledge distillation for compressing GANs [3, 6, 14, 15, 16]. The main idea of response-based knowledge distillation is to directly mimic the final response of the last output layer of the teacher model.

3 METHOD

Proposed methodology has following three stages:

1. Developing a style discriminator.
2. Designing efficient architectures for generator, style encoder and mapping network.
3. Training the networks using a combination of cross knowledge distillation losses and the original objective.

3.1 Style Discriminator

We introduce the style discriminator in order to transfer knowledge from teacher to student mapping network via adversarial learning [2]. It is a multi-task network which contains multiple linear output branches, one for each domain as seen in Figure 1. As shown in Figure 1, each branch of the style discriminator learns a binary classification that determines if the style code is created by adversarial loss by the teacher mapping network or the student mapping network of its domain. The style discriminator capacity is close to the student mapping network. For non-linear activation Leaky-ReLU layers are used in intermediate layers.

3.2 Efficient Architecture Design

Knowledge distillation is most successful when there is a high degree of correlation between teacher and student model architectures. Therefore, we use the original generator, discriminator, style encoder, and mapping networks as our baseline architectures. Since our primary aim is to obtain memory efficient networks for resource constrained environments, we focus on altering the baseline architectures in the following ways:

- **Mobile Residual Blocks.** Residual blocks (ResBlk) are used extensively in StarGAN-v2 framework for both encoding and decoding in the generator and the style encoder networks. In mobile residual blocks (MobileResBlk) we replace the full convolution

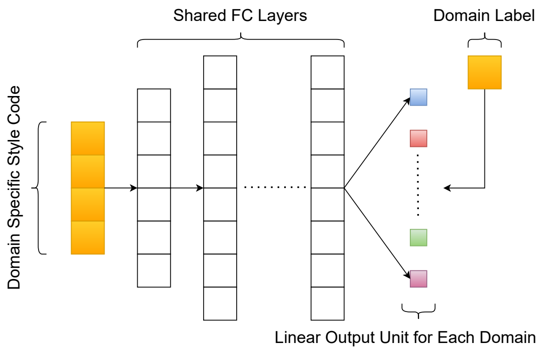

Shared FC Layers
Domain Label
Domain Specific Style Code
Linear Output Unit for Each Domain

Figure 1: Style Discriminator. The input domain specific style code is classified as originated from teacher or student mapping network.

layers of the residual blocks with depthwise separable convolution layers on account of their better performance-computation trade-off [6, 10, 14]. We use mobile residual blocks in place of all the residual blocks for the encoding parts of all the modules of StarGAN-v2.

- **Reduced channels.** In order to control the capacity of the whole framework we restrict the maximum number of channels in convolution layers and the maximum number of hidden nodes in linear layers for all the networks. The restriction is parametric named as a single global parameter $\alpha \in [64, 96, 128, 256]$. The value of α is determined from empirical results on standard benchmark datasets.

Table 1 shows the alterations done on the baseline Generator architecture to obtain an efficient Generator network. Similar changes are done to the Style Encoder and the Mapping Network.

3.3 Cross Distillation Losses

Inspired by the success of knowledge distillation in recent works [3, 6, 15, 16], we add multiple model distillation losses to enforce efficient student modules to mimic the behaviour of original networks of StarGAN-v2 framework. We do not use a pre-trained teacher discriminator in our training. Denoting our pre-trained teacher generator as G_T, style encoder as E_T, mapping network as F_T and similarly, student generator as G_S, style encoder as E_S, mapping network as F_S, image discriminator as D, and style discriminator as D^S.

Let $\mathcal{X}$, $\mathcal{Y}$ be the sets of images and possible domains and $\mathcal{Z}$ be the set of all possible latent codes. During training, we randomly sample an image $x \in \mathcal{X}$ with its original domain label $y \in \mathcal{Y}$, two latent codes $z_1 \in \mathcal{Z}$, $z_2 \in \mathcal{Z}$, and a target reference image $\tilde{x} \in \mathcal{X}$ with its target domain label $\tilde{y} \in \mathcal{Y}$. We generate target style codes using teacher networks:

$$s_{z_1} = F_T(z_1, \tilde{y}) \qquad s_{z_2} = F_T(z_2, y)$$
$$s_{\tilde{x}} = E_T(\tilde{x}, \tilde{y}) \qquad s_x = E_T(x, y)$$

Cross Image Adversarial Losses ($\mathcal{L}^1_{G_S,D}$, $\mathcal{L}^2_{G_S,D}$). We generate images from G_T and G_S using s_{z_1}, $s_{\tilde{x}}$, and x. The image discriminator D learns to assess the divergence between

Original Generator (maximum channel width 512)			**Efficient Generator (maximum channel width 128)**	
Output Shape	**Layer**	**Layer Type**	**Layer**	**Output Shape**
$256 \times 256 \times 3$	RGB Image	Input	RGB Image	$256 \times 256 \times 3$
$256 \times 256 \times 64$	Conv1×1	-	Conv1×1	$256 \times 256 \times 64$
$128 \times 128 \times 128$	ResBlk	Encoding	Conv1×1	$128 \times 128 \times 128$
$64 \times 64 \times 256$	ResBlk	Encoding	MobileResBlk	$64 \times 64 \times 128$
$32 \times 32 \times 512$	ResBlk	Encoding	MobileResBlk	$32 \times 32 \times 128$
$16 \times 16 \times 512$	ResBlk	Encoding	MobileResBlk	$16 \times 16 \times 128$
$16 \times 16 \times 512$	ResBlk	Intermediate	MobileResBlk	$16 \times 16 \times 128$
$16 \times 16 \times 512$	ResBlk	Intermediate	MobileResBlk	$16 \times 16 \times 128$
$16 \times 16 \times 512$	ResBlk	Intermediate	ResBlk	$16 \times 16 \times 128$
$16 \times 16 \times 512$	ResBlk	Intermediate	ResBlk	$16 \times 16 \times 128$
$32 \times 32 \times 512$	ResBlk	Decoding	ResBlk	$32 \times 32 \times 128$
$64 \times 64 \times 256$	ResBlk	Decoding	ResBlk	$64 \times 64 \times 128$
$128 \times 128 \times 128$	ResBlk	Decoding	ResBlk	$128 \times 128 \times 128$
$256 \times 256 \times 64$	ResBlk	Decoding	ResBlk	$256 \times 256 \times 64$
$256 \times 256 \times 3$	Conv1×1	-	Conv1×1	$256 \times 256 \times 3$

Table 1: Architecture comparison of the Generator from StarGAN v2 and TinyStarGAN v2 frameworks.

images generated by G_T and G_S. D maximizes the divergence, while G_S minimizes it. In this way, G_S learns to mimic G_T implicitly.

$$\mathcal{L}^1_{G_S,D} = \mathbb{E}_{x,z_1,\tilde{y}}[\log(D_{\tilde{y}}(G_T(x,s_{z_1})) + log(1 - D_{\tilde{y}}(G_S(x,s_{z_1})))] \tag{1}$$

$$\mathcal{L}^2_{G_S,D} = \mathbb{E}_{x,\tilde{x},\tilde{y}}[\log(D_{\tilde{y}}(G_T(x,s_{\tilde{x}})) + log(1 - D_{\tilde{y}}(G_S(x,s_{\tilde{x}})))] \tag{2}$$

Cross Style Adversarial Losses ($\mathcal{L}^1_{F_S,D^S}$, $\mathcal{L}^2_{F_S,D^S}$). We generate style codes from F_S using z_1, z_2, y, and $\tilde{y}$. The style discriminator D^S learns to assess the divergence between style codes generated by F_T and F_S. D^S maximizes the divergence, while F_S minimizes it. In this way, F_S learns to mimic F_T implicitly.

$$\mathcal{L}^1_{F_S,D^S} = \mathbb{E}_{z_1,\tilde{y}}[\log(D^S_{\tilde{y}}(s_{z_1})) + log(1 - D^S_{\tilde{y}}(F_S(z_1,\tilde{y})))] \tag{3}$$

$$\mathcal{L}^2_{F_S,D^S} = \mathbb{E}_{z_2,y}[\log(D^S_{y}(s_{z_2})) + log(1 - D^S_{y}(F_S(z_2,y)))] \tag{4}$$

Cross Style Utilization Losses ($\mathcal{L}^1_{G_S,E_S}$, $\mathcal{L}^2_{G_S,E_S}$). The student style encoder E_S learns to extract style code similar to target style codes over the images generated by G_S using s_{z_1}, $s_{\tilde{x}}$, and x. Meanwhile, G_S learns to utilize target style codes. This objective is similar to previous approaches [5, 11, 19].

$$\mathcal{L}^1_{G_S,E_S} = \mathbb{E}_{x,z_1,\tilde{y}}||s_{z_1} - E_S(G_S(x,s_{z_1}),\tilde{y})||_1 \tag{5}$$

$$\mathcal{L}^2_{G_S,E_S} = \mathbb{E}_{x,\tilde{x},\tilde{y}}||s_{\tilde{x}} - E_S(G_S(x,s_{\tilde{x}}),\tilde{y})||_1 \tag{6}$$

Cross Source Attributes Preservation Losses ($\mathcal{L}^1_{G_S}$, $\mathcal{L}^2_{G_S}$). The student generator learns to preserve the domain invariant characteristics of input image x while generating images using s_{z_1} and $s_{\tilde{x}}$. We employ the cycle consistency loss [4, 13, 18] by reconstructing the source image from generated images using s_x.

$$\mathcal{L}^1_{G_S} = \mathbb{E}_{x,z_1,\tilde{y}}||x - G_S(G_S(x, s_{z_1}), s_x)||_1 \tag{7}$$

$$\mathcal{L}^2_{G_S} = \mathbb{E}_{x,\tilde{x},\tilde{y}}||x - G_S(G_S(x, s_{\tilde{x}}), s_x)||_1 \tag{8}$$

3.4 Full Objective

The full objective utilizes the original training losses of StarGAN v2 ($\mathcal{L}_{org}$) [5] and our distillation losses combined into minimax optimization setting.

$$\begin{aligned}\min_{G_S,E_S,F_S} \quad \max_{D^S,D}[\mathcal{L}_{org} + \mathcal{L}^1_{G_S,D} + \mathcal{L}^2_{G_S,D} + \mathcal{L}^1_{F_S,D^S} + \\ \mathcal{L}^2_{F_S,D^S} + \mathcal{L}^1_{G_S,E_S} + \mathcal{L}^2_{G_S,E_S} + \mathcal{L}^1_{G_S} + \mathcal{L}^2_{G_S}]\end{aligned} \tag{9}$$

3.5 Evaluation metrics

We use Frechét Inception Distance (FID) [8] and Learned Perceptual Image Patch Similarity (LPIPS) [17] to measure the visual quality and the diversity of generated images on test data as done previously in [5]. We follow the original evaluation protocols from StarGAN v2 to compute FID and LPIPS on validation set. We calculate the metrics for every pair of image domains within a dataset and report their average values. We use the total number of Multiply-Accumulate Operations (MAC) required in a single forward pass and the total number of parameters in generator, style encoder, and mapping network combined to quantify the computation cost and size of each framework.

3.6 Implementation Details

Details regarding the training parameters, data augmentation and pre-trained teacher networks is given below:

Training parameters for our method are same as the StarGAN v2 training framework. We use a batch size of 8 and train for 100K iterations. We store a checkpoint after each 10K iterations. All the networks are trained using Adam optimizer with $\beta_1 = 0$ and $\beta_2 = 0.99$. We set the learning rate for all the networks to 10^{-4}. We evaluate our method with different values of $\alpha \in [64, 96, 128, 160]$. For each value of α we evaluate all the checkpoint networks on test images and select the best scoring checkpoint. We keep the random seed fixed to 777 for all our experiments.

Data Augmentations are applied on the images of both source and target domains. Images are horizontally flipped with a probability of 0.5. A random crop of Size $\in [0.8, 1.0]$ and Aspect Ratio $\in [0.9, 1.1]$ is extracted from original image and resized to 256×256.

Pre-trained teacher networks are obtained from the StarGAN v2 training framework by using default parameters. The learning rate for generator, discriminator, style encoder is 10^{-4}, while that of mapping network is set to 10^{-6}. The dimensions of the latent code, the maximum channels of hidden layer, and the style code are 16, 512, and 64 respectively.

4 EXPERIMENTS

4.1 Datasets

We use the CelebA-HQ [12] and the AFHQ [5] datasets for our evaluations. The CelebA-HQ dataset is divided into two domains of male and female while the AFHQ dataset contains three domains of cat, dog, and wildlife. We follow the train and validation splits as shown in Table 2. All the images are resized to 256×256 as originally used in StarGAN v2.

	CelebA-HQ		**AFHQ**		
	Male	Female	Cat	Dog	Wildlife
Train	10057	17943	5153	4739	4738
Val	1000	1000	500	500	500

Table 2: Train-Validation splits for CelebA-HQ and AFHQ datasets.

4.2 Analysis of Cross Distillation Losses

We evaluate each cross distillation loss proposed in our method by adding them to the original StarGAN v2 losses and training each configuration to completion. The best FID and LPIPS scores of each experiment are compared for both Latent-Guided synthesis and Reference-Guided synthesis. Efficient architectures with network capacity at $\alpha = 128$ are trained on the AFHQ dataset in each experiment from scratch with all the training parameters kept the same. The baseline configuration trained with the original StarGAN v2 losses without any distillation loss produced high FID scores, which is expected to be the case due to the limited capacity of the networks. We first improve the baseline by adding cross image adversarial losses to let the efficient student generator and the image discriminator learn the distribution of images produced by the original heavy teacher StarGAN v2 framework. We further enhance the training stability by applying style adversarial losses which helps the student mapping network mimic the original latent style code distribution. To enable the student style encoder network to produce the latent style codes similar to the original style encoder we added cross style utilization losses. The fourth component of the cross distillation forces the student generator to preserve the domain invariant characteristics of input image while utilizing the original latent style codes.

	Loss Components	**Latent Guided Synthesis**		**Reference Guided Synthesis**	
		FID	**LPIPS**	**FID**	**LPIPS**
A	Original StarGAN v2 Losses	24.06	0.514	33.13	0.484
B	(A) + Cross Image Adversarial Losses	23.59	0.478	25.12	0.433
C	(B) + Cross Style Adversarial Losses	21.76	0.459	23.00	0.419
D	(C) + Cross Style Utilization Losses	20.81	0.456	21.78	0.416
E	(A) + All Cross Distillation Losses	20.69	0.437	21.60	0.415

Table 3: Quantitative Performance Evaluation of various cross distillation losses configurations on **AFHQ** dataset at $\boldsymbol{\alpha = 128}$. FID indicates the distance between two distributions of real and generated images (lower is better), while LPIPS measures the diversity of generated images (higher is better).

As can be observed in Table 3 with the addition of each loss component the FID performance improves however, the LPIPS performance degrades. Since LPIPS measures how diverse images are in a set, the more sporadic the artifacts are in the images the more diverse they become and that may show better LPIPS performance. We speculate that the high LPIPS scores in the baseline configuration is due to poor translation of input images causing spurious artifacts in the output images.

5 RESULTS

5.1 Quantitative Results

Dataset	**Method**	**Latent Guided Synthesis**		**Reference Guided Synthesis**		**Parameter Count (M) (E+G+F)**	**MACs (G) (E+G+F)**
		FID	**LPIPS**	**FID**	**LPIPS**		
AFHQ	Original	16.09	0.451	19.73	0.431	58.10	65.30
	Proposed						
	$\alpha = 64$	26.65	0.492	30.49	0.448	0.81 (71×)	5.37 (12×)
	$\alpha = 96$	23.52	0.446	24.21	0.425	1.62 (35×)	8.54 (7×)
	$\alpha = \mathbf{128}$	**20.69**	**0.437**	**21.60**	**0.415**	2.71 (**21**×)	11.53 (**5**×)
	$\alpha = 160$	20.95	0.421	21.32	0.408	3.94 (14×)	13.47 (4×)
CelebA HQ	Original	13.76	0.451	23.88	0.388	66.82	62.03
	Proposed						
	$\alpha = 64$	20.69	0.423	27.17	0.381	0.89 (75×)	5.35 (11×)
	$\alpha = 96$	19.30	0.425	26.23	0.382	1.79 (37×)	8.09 (7×)
	$\alpha = \mathbf{128}$	**18.41**	**0.417**	**25.18**	**0.385**	3.00 (**22**×)	10.92 (**5**×)
	$\alpha = 160$	18.91	0.419	26.24	0.380	4.40 (15×)	12.49 (4×)

Table 4: Quantitative Evaluation of the proposed method at different α values on validation set. We use FID (the lower the better) and LPIPS (the higher the better) for both latent and reference guided synthesis. We accumulate the individual size and MACs of generator (G), style encoder (E), and mapping network (F). At α = **128**, our method can compress original StarGAN v2 framework by more than **20**× in size and by more than **5**× in MACs, with minor performance degradation.

Table 4 shows a summary of evaluation scores on test images by using our proposed method with different values of α on both CelebA-HQ and AFHQ datasets. At $\alpha = 64$ and 96 we can see huge increase in FID scores, whereas at $\alpha = 128$ and $\alpha = 160$ the scores are close to the original method. As α increases the images produced by our approach look more realistic which is reflected in FID scores. However, the LPIPS performance degrades with increasing α. We speculate that the high LPIPS scores in lower α values are due to spurious artifacts in the output images. LPIPS measures how diverse images are in a set, the more sporadic the artifacts are in the images the more diverse they become and that may show better LPIPS performance. Since we care about the diversity of realistic images, therefore, we need to compare both FID and LPIPS together against the original method for a holistic view. We obtain a higher compression rate at $\alpha = 128$ compared to $\alpha = 160$, thus achieving a better balance of compression and performance.

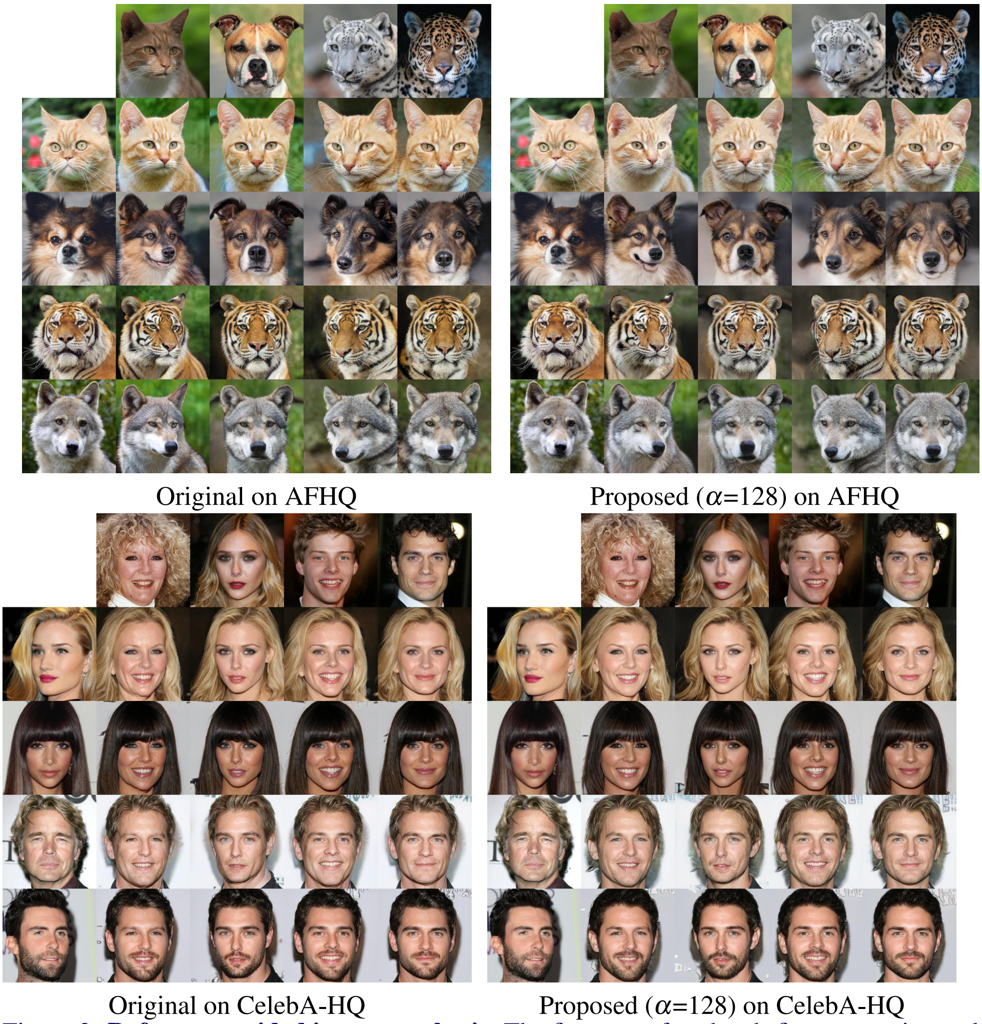
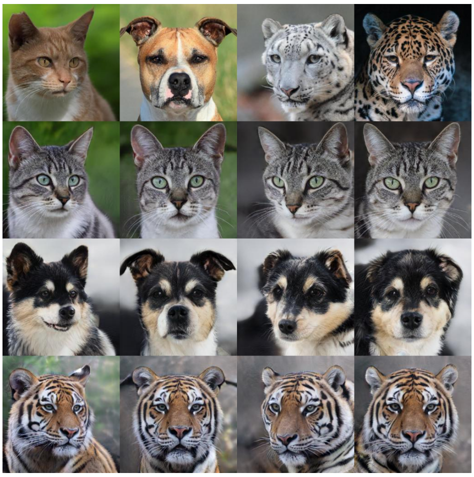
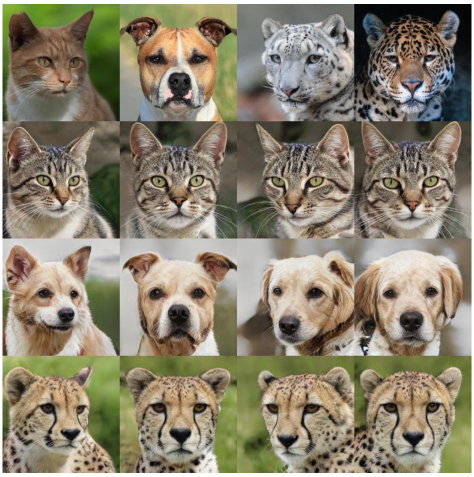

Original on AFHQ
Proposed (α=128) on AFHQ
Original on CelebA-HQ
Proposed (α=128) on CelebA-HQ

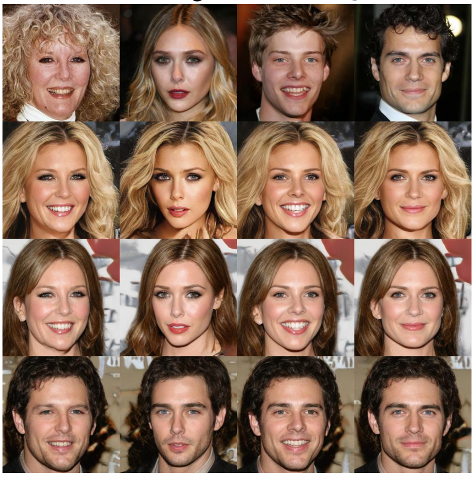
Figure 2: **Reference-guided image synthesis.** The first row of each sub-figure contains real source images and the first column of each sub-figure contains real reference images. The rest are fake images generated using generator and style codes produced using style encoder.

5.2 Qualitative Results

In Fig. 2 and 3, we do a visual comparison of images generated by the original and our proposed method for reference-guided and latent-guided image synthesis on test images. Our method produces images with high quality and diverse styles across all domains on both CelebA-HQ and AFHQ datasets as good as the original StarGAN v2. On the AFHQ dataset the proposed method renders distinctive styles (e.g. breeds) of each domain as well as its fur pattern and eye color. Similarly, on the CelebA-HQ dataset diverse hair colors, hair styles and skin tones are produced effectively using the proposed method. However, there are a few differences in the quality of generated images. The images of the original method, have more contrast whereas, the images produced by the proposed method are a little smooth. Also, in a few cases the shape of the body parts, are not developed properly by the proposed method.


Original on AFHQ


Proposed (α=128) on AFHQ


Original on CelebA-HQ

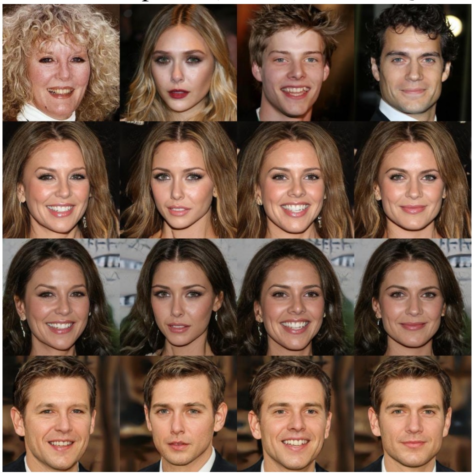

Proposed (α=128) on CelebA-HQ

Figure 3: **Latent-guided image synthesis.** The first row of each sub-figure contains real source images. The rest are fake images generated using generator and style codes produced using mapping network from randomly sampled latent codes.

6 CONCLUSION

We proposed Tiny-StarGAN v2, a method to address the computation bottleneck problem of StarGAN v2. The core contribution of our work is in defining an end-to-end training algorithm to distill the knowledge from a multi-network framework. We trained efficient architectures for generator, style encoder, and mapping network under the guidance of knowledge distillation. Extensive experiments showed that our method preserved the visual quality and style diversity of generated images while compressing StarGAN v2 framework massively. Additionally, we examined the performance of networks at different network capacities and observed that the FID score improvements reduces upon increasing the capacity, a potential drawback to the proposed approach.

References

[1] Angeline Aguinaldo, Ping-Yeh Chiang, Alex Gain, Ameya Patil, Kolten Pearson, and Soheil Feizi. Compressing gans using knowledge distillation. *arXiv preprint arXiv:1902.00159*, 2019.

[2] Vasileios Belagiannis, Azade Farshad, and Fabio Galasso. Adversarial network compression. In *ECCV Workshops*, 2018.

[3] Hanting Chen, Yunhe Wang, Han Shu, Changyuan Wen, Chunjing Xu, Boxin Shi, Chao Xu, and Chang Xu. Distilling portable generative adversarial networks for image translation. *arXiv preprint arXiv:2003.03519*, 2020.

[4] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018.

[5] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *CVPR*, 2020.

[6] Yonggan Fu, Wuyang Chen, Haotao Wang, Haoran Li, Yingyan Lin, and Zhangyang Wang. AutoGAN-distiller: Searching to compress generative adversarial networks. In *ICML*, 2020.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[8] 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. *NIPS*, 2017.

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[10] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[11] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.

[12] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[13] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192*, 2017.

[14] Muyang Li, Ji Lin, Yaoyao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. Gan compression: Efficient architectures for interactive conditional gans. In *CVPR*, 2020.

[15] Han Shu, Yunhe Wang, Xu Jia, Kai Han, Hanting Chen, Chunjing Xu, Qi Tian, and Chang Xu. Co-evolutionary compression for unpaired image translation. In *ICCV*, 2019.

[16] Haotao Wang, Shupeng Gui, Haichuan Yang, Ji Liu, and Zhangyang Wang. Gan slimming: All-in-one gan compression by a unified optimization framework. In *ECCV*, 2020.

[17] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[18] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[19] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NIPS*, 2017.

[20] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.