

Supplementary Materials for Polymorphic-GAN: Generating Aligned Samples across Multiple Domains with Learned Morph Maps

1. Model Architecture

We provide additional descriptions of the architecture of PMGAN in this section.

1.1. Pre-trained StyleGAN

PMGAN is composed of the pre-trained StyleGAN2’s generator G , domain-specific morph layers M^1, \dots, M^N and rendering layers R^1, \dots, R^N . We first sample a noise vector $z \sim p(z)$ from the standard Normal prior distribution and feed it through G , which produces the output image I^P and also the intermediate features u_1, \dots, u_L for L features in G . In this work, all experiments are carried out at 256×256 RGB image resolution. Thus, we store the generator features for each spatial resolution from 4×4 to 256×256 before the final features are transformed via a 1×1 convolution layer (*i.e.* tRGB) that produces the output RGB values. The features are shaped as $(4 \times 4 \times 512)$, $(8 \times 8 \times 512)$, $(16 \times 16 \times 512)$, $(32 \times 32 \times 512)$, $(64 \times 64 \times 512)$, $(128 \times 128 \times 256)$ and $(256 \times 256 \times 128)$, where the first two dimensions correspond to the height and width, and the last dimension is for the number of channels.

1.2. MorphNet

Features u_1, \dots, u_L contain valuable information, including semantic content as well as fine-grained edge information. We use these features to produce domain-specific morph maps that can modify the geometry embedded in the features to be suitable for each target domain. The MorphNet component of PMGAN first reduces each feature map’s channel dimension to be smaller through a 1×1 convolution layer and then upsamples all features to match the largest spatial resolution 256×256 . Each 1×1 convolution layers reduce the number of channels to 128 followed by a leaky ReLU [9] activation function.

The upsampled features are concatenated channel-wise, resulting in a $(256 \times 256 \times 896)$ tensor. It goes through two 3×3 convolution layers whose output channel dimensions are 512, followed by leaky ReLU. These layers are shared across domains and the spatial dimension is preserved (with stride=1 and padding=1). The conv layers and upsampling operations are represented as MergeFeatures in Algorithm 1 in the main text.

We add a sinusoidal positional encoding [15] for 2D to the merged features to inject grid position information which can be useful for learning geometric biases in a dataset. We define the positional encoding as

$$\begin{aligned} PE(x, y, 4c) &= \sin(x/10000^{8c/512}) \\ PE(x, y, 4c + 1) &= \cos(x/10000^{8c/512}) \\ PE(x, y, 4c + 2) &= \sin(y/10000^{(8c+4)/512}) \\ PE(x, y, 4c + 3) &= \cos(y/10000^{(8c+4)/512}) \end{aligned}$$

where $x \in [0, 255]$, $y \in [0, 255]$ for spatial dimensions, and $c \in [0, 127]$ for the channel dimension.

Finally, this summed tensor is processed by domain-specific convolution layers M^d for each domain d . M^d is composed of two convolution layers. The first layer is spatial-dimension preserving 3×3 conv layer that outputs 512 channels, followed by a leaky ReLU activation function. The second layer is spatial-dimension preserving 3×3 conv layer that outputs 2 channels, followed by a Tanh activation function and a scalar division by η which is a hyperparameter that controls the maximum displacement we allow the morphing operation to produce. We use $\eta = 3$ for all experiments in this paper. Thus, M^d produces a $H \times W \times 2$ morph map \mathcal{M}_{Δ}^d , normalized between $[-1/\eta, 1/\eta]$. \mathcal{M}_{Δ}^d represents the relative horizontal and vertical direction that each pixel would get its value from (a pixel here is (p, q) position in a 3-dim spatial tensor).

1.3. Feature Morphing

We follow Spatial Transformer Networks (SPN) [5] to differentially morph features with \mathcal{M}_{Δ}^d . We initialize a 2D sampling grid from an identity transformation matrix, normalized between $[-1, 1]$. The sampling grid has the same shape as \mathcal{M}_{Δ}^d , and each pixel (p, q) in the sampling grid contains the absolute position (x, y) of the source pixel that will be morphed into (p, q) . For example, if pixel (p, q) has value $(-1, -1)$, the vector at the top left corner of the source feature map will be morphed into (p, q) . The morph map \mathcal{M}_{Δ}^d is added to the grid, and we denote the resulting grid as $\Gamma \in \mathbb{R}^{H \times W \times 2}$. Unlike SPN that produces an affine transformation matrix with six parameters for sampling grid, we learn pixel-wise morphing maps, which gives us precise control for fine-detailed morphing. For each layer l of generator features $\{u_1, \dots, u_L\}_d$ from Section 1.1, we perform the following Morph operation that bilinearly interpolates features:

$$\tilde{u}_l^{pq} = \sum_n \sum_m^{H_l, W_l} u_l^{nm} \max(0, 1 - |x^{pq} - m|) \max(0, 1 - |y^{pq} - n|) \quad (1)$$

where $\tilde{u}_l^{pq} \in \mathbb{R}^c$ is the morphed feature vector with c channels at pixel (p, q) for layer l , $u_l^{nm} \in \mathbb{R}^c$ is the source feature vector prior to Morph at pixel (n, m) of $u_l \in \mathbb{R}^{H_l \times W_l \times c}$, and (x^{pq}, y^{pq}) is the sample point in Γ for pixel (p, q) , assuming unnormalized grid coordinates for ease of presentation. Note that Γ is also bilinearly interpolated to match the spatial dimension of each layer (H_l, W_l) .

The morphed features $\{\tilde{u}_1, \dots, \tilde{u}_L\}_d$ are now geometrically transformed to be suitable for domain d . Each of these features is then processed via further convolution layers R^d to produce RGB images. Each R^d is composed of L output heads for each morphed features in $\{\tilde{u}_1, \dots, \tilde{u}_L\}_d$. Each head is implemented as three-layer modulated convolution layers from StyleGAN2 [8] which takes the feature \tilde{u}_l as input. It also takes the latent code $w = \text{mapping}(z)$ as an additional input for the modulation process, where mapping is the mapping layer of the core generator in G . The first two layers output 512 channels, followed by leaky ReLU activation, and the last layer outputs 3 RGB channels. The RGB outputs from L layers are summed together using skip connections as in StyleGAN2 [8]. Importantly, the R^d layers can correct small unnatural distortions caused by the feature morphing process, in contrast to previous works that directly warp output images [13].

2. Datasets

We construct two multi-domain datasets for evaluation:

Cars dataset consists of five classes of cars from the LSUN-Car dataset [16]. We use an object classifier by Ridnik *et al.* [11] that can output fine-grained object classes to divide the dataset into the following domains: Sedan (149K), SUV (52K), Sports car (58K), Van (25K), and Truck (22K), with the number of images in parentheses. LSUN dataset is distributed from <https://www.yf.io/p/lsun>.

Faces dataset consists of Flickr-Faces-HQ [7] (70K), MetFaces [6] (1.3K), as well as Cat (5.6K), Dog (5.2K) and Wild life (5.2K) from the AFHQ dataset [3].

FFHQ dataset is distributed from <https://github.com/NVlabs/ffhq-dataset>. The images are published in Flickr by their uploaders under either Creative Commons BY 2.0, Creative Commons BY-NC 2.0, Public Domain Mark 1.0, Public Domain CC0 1.0, or U.S. Government Works. The dataset itself is licensed under Creative Commons BY-NC-SA 4.0 license by NVIDIA Corporation.

MetFaces dataset is distributed from <https://github.com/NVlabs/metfaces-dataset>. The images are distributed under Creative Commons Zero (CC0) license by the Metropolitan Museum of Art. The dataset itself is licensed under Creative Commons BY-NC 2.0 license by NVIDIA Corporation.

AFHQ dataset is distributed from <https://github.com/clovaai/stargan-v2>. We use the original version of the dataset. The dataset is licensed under Creative Commons BY-NC 4.0 license by NAVER Corporation.

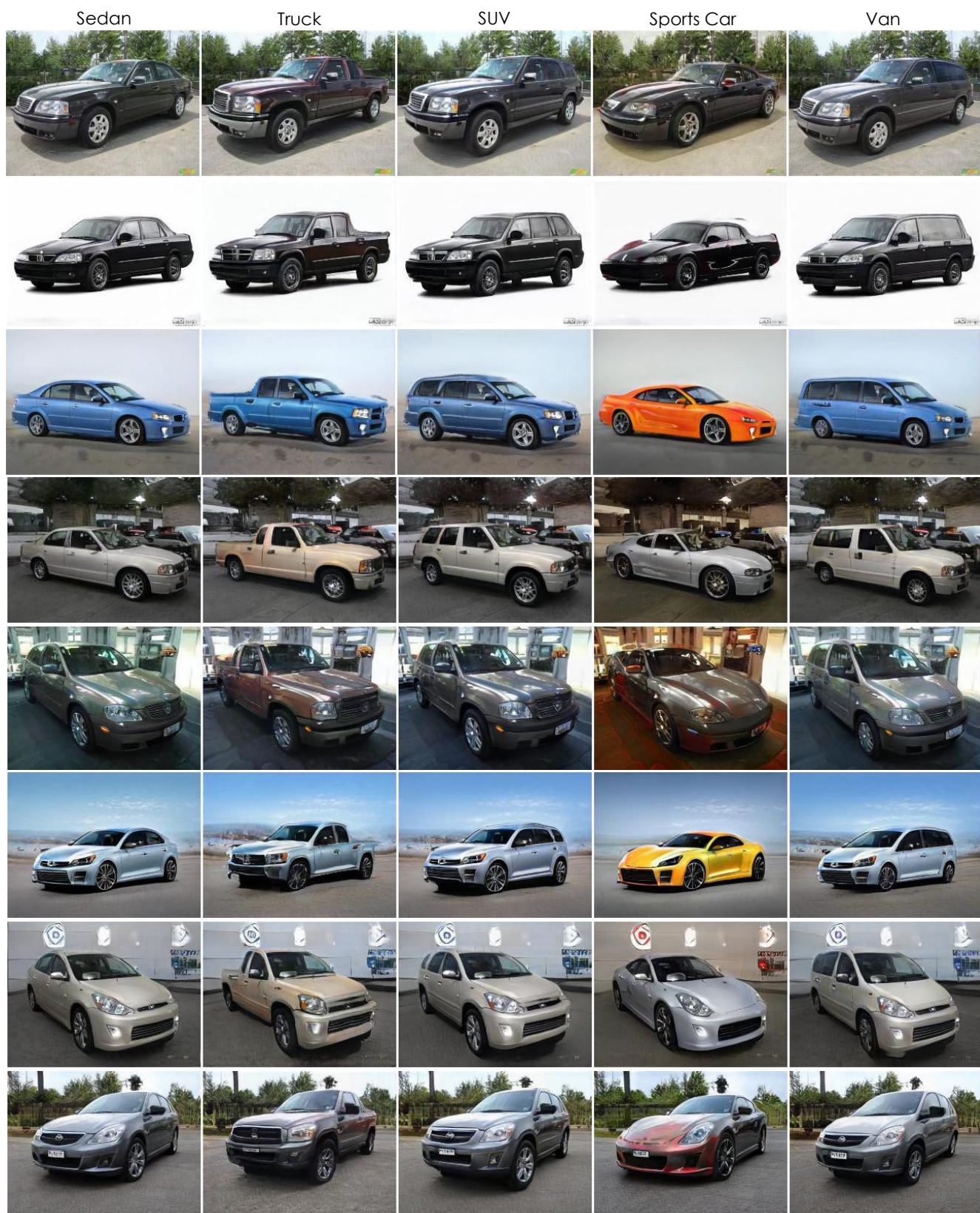


Figure 1. Aligned Samples from PMGAN trained on Cars dataset.

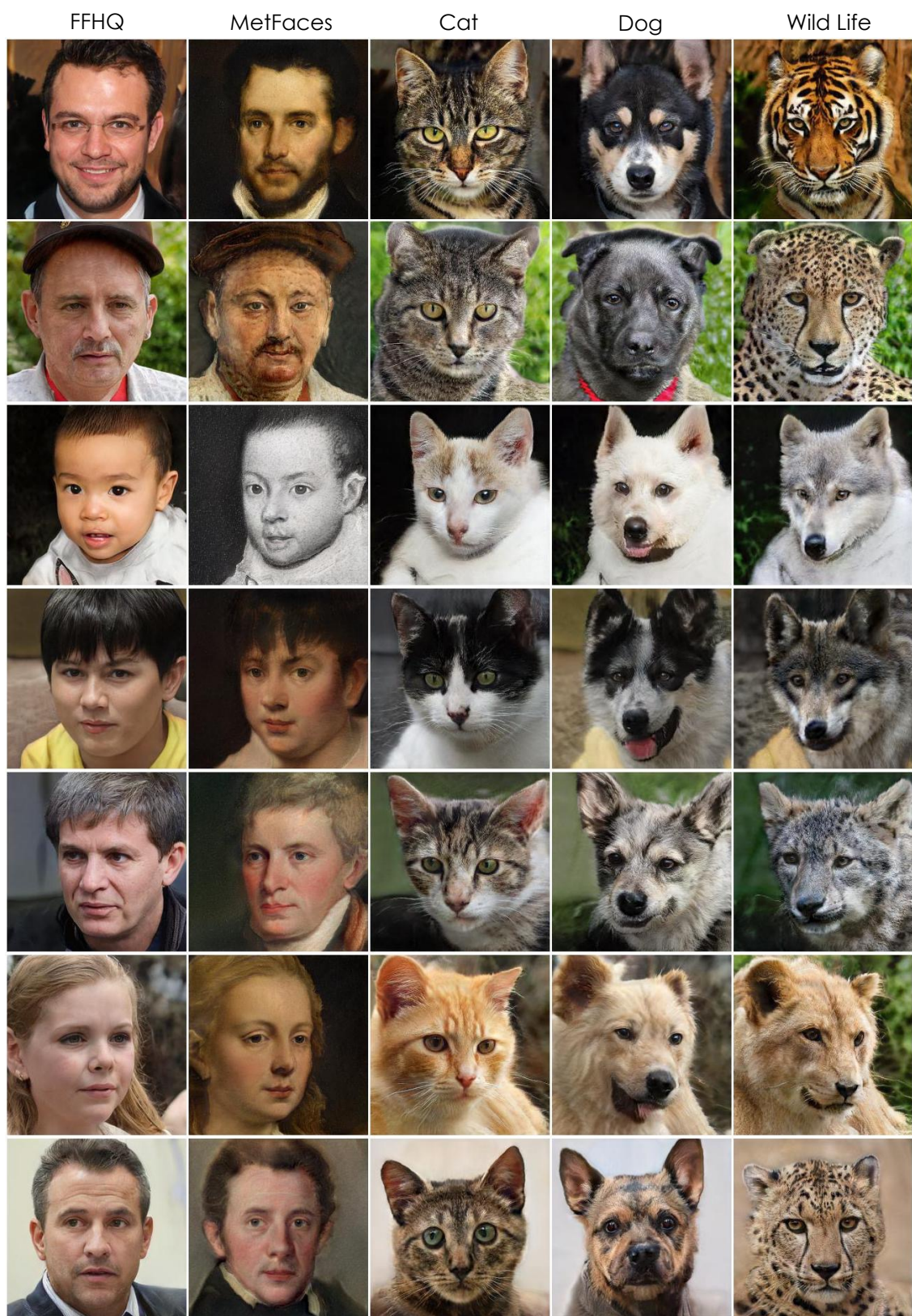


Figure 2. Aligned Samples from PMGAN trained on Faces dataset.

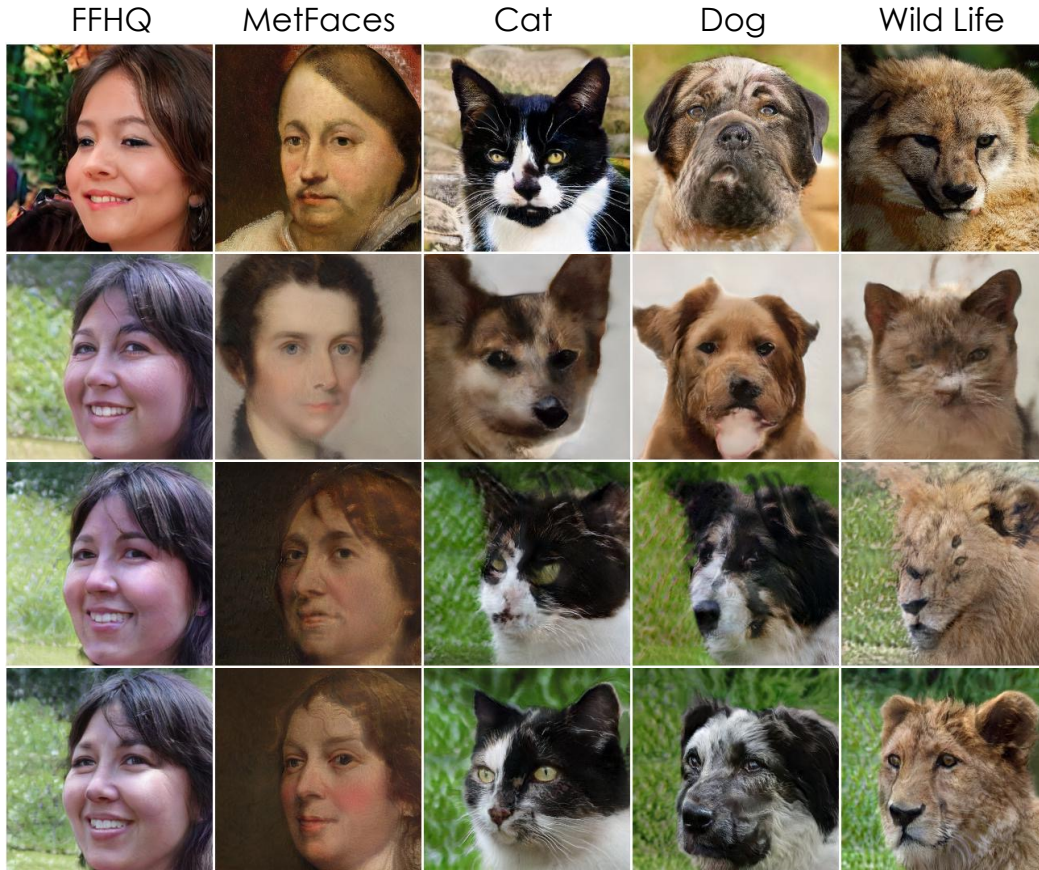


Figure 3. *Top row: DC-StyleGAN2, Second row: *DC-StyleGAN2, Third row: Ours without MorphNet, Last row: Ours.*

3. Experiments

In this section, we provide additional details on each of the models and algorithms used in the experiments section.

3.1. Ablation Studies

Domain-Conditional StyleGAN2 (DC-StyleGAN2) is a modified StyleGAN2 model that takes a one-hot encoded domain vector as an input. The 5-dimensional one-hot vector is embedded through a linear layer that outputs a 512-dimensional embedding vector. Then, the embedding is concatenated with w latent from the mapping network $w = \text{mapping}(z)$, and then is fed through a linear layer that finally produces a 512-dimensional vector that goes through the generator. The discriminator is the same as StyleGAN2’s discriminator except that it is also conditioned on domain similar to the discriminator architecture of class-conditional BigGAN [1]. We add the dot product of the penultimate layer’s output and domain embedding to the unconditioned output of the discriminator. We provide additional aligned samples in Figure 1 and Figure 2. Figure 3 provides an additional comparison with baselines. Except for DC-StyleGAN2 which does not share the same parent model, other models show samples from the same latent code.

We use domain classifiers to measure the domain classification accuracy indicating if models produce corresponding samples for each domain. They are implemented as ResNet-18 [4] for the 5-way classification task, achieving 90.0% and 99.9% accuracy for Cars and Faces, respectively. We note that classification is much easier for Faces because of their distinct texture.

3.2. Morph Map and Edit Vector Transfer

We provide additional examples on cross domain interpolation in Figure 4. Figure 5 shows translation between a source and target domain where we fix the morph map of the source domain and use target domain’s rendering layers. Figure 6 also shows translations between two domains, but this time, the rendering layers of the source domain are kept fixed while the morph map from the target domain is used. They show how PMGAN is able to produce novel outputs by disentangling the

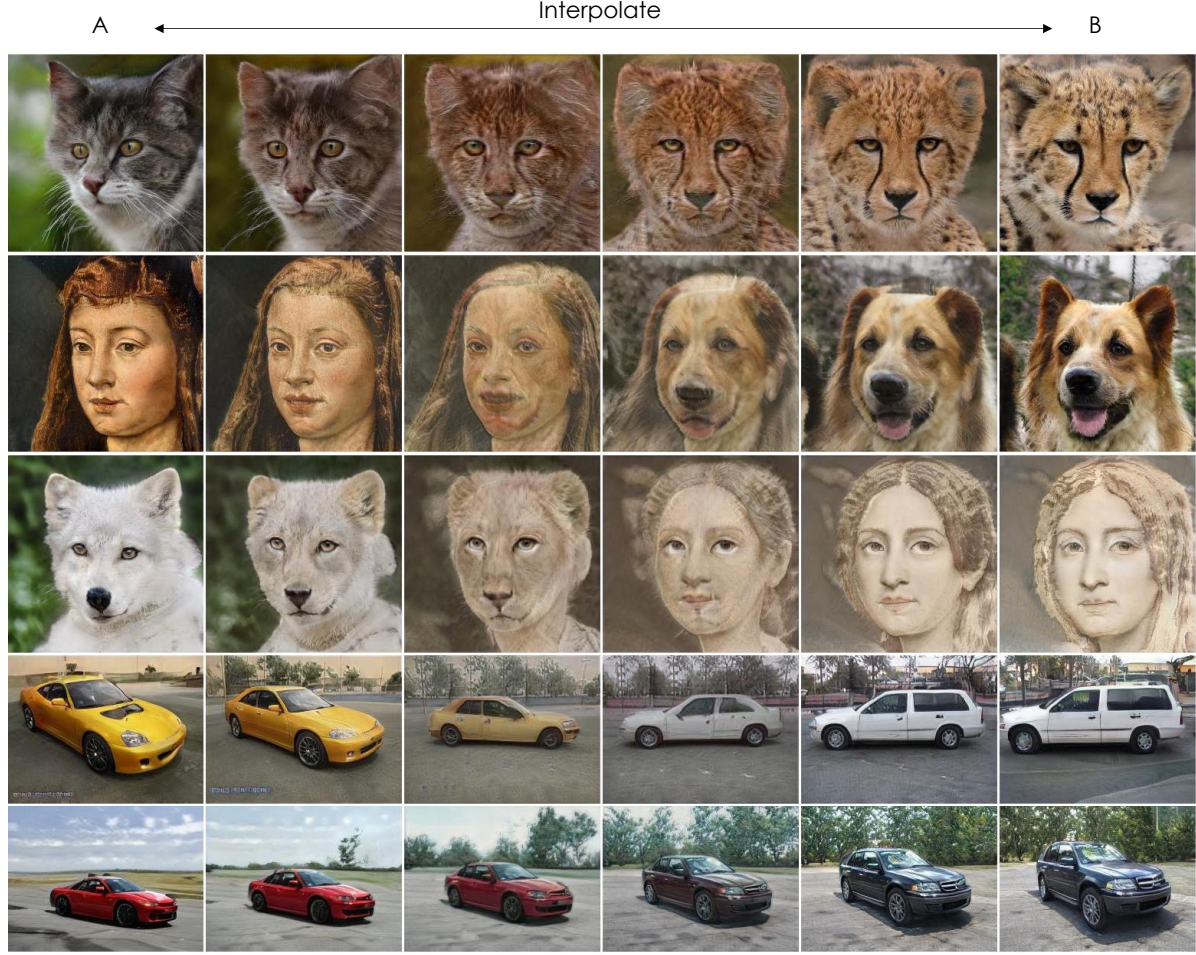


Figure 4. Cross-Domain Interpolation: we interpolate both the weights of domain-specific layers of two domains and their latent vectors A&B.

shape and texture with morph maps.

For edit transfer, we use SeFa [12] for its simplicity to find edit vectors in PMGAN. SeFa produces edit directions in an unsupervised way by finding the eigenvectors of $A^T A$ where A is the weight matrix of style layers in the core generator. Therefore, it is data independent and takes less than one second to find the edit vectors. We use the official implementation from <https://github.com/genforce/sefa>. We find meaningful vectors such as rotation, zoom, lighting and elevation. Figure 7 contains additional examples of how edit vectors can be transferred across all domains for Faces and Cars datasets.

3.3. Zero-shot Segmentation Transfer

Assuming there exists a method that can output a segmentation map for images from the parent domain, it is possible to zero-shot transfer the segmentation mask to all other domains using PMGAN’s learned morph map. We directly use the Morph operation on the segmentation map with \mathcal{M}_Δ after bilinearly interpolating the morph map to match the size of the mask. As the morph map \mathcal{M}_Δ captures the geometric differences between domains, we can successfully use \mathcal{M}_Δ to transfer the parent’s segmentation masks across domains. We use pre-trained deeplab segmentation networks [2] from DatasetGAN [18] for Sedan and FFHQ domains for Cars and Faces datasets, respectively. We then transfer them to other domains. Specifically, we use the 20-part car model and 34-part face model trained with synthesized labels from DatasetGAN. Figure 9 and 10 show additional segmentation transfer results. For Faces, we note that the noses of animals are always registered at the same location as the mouths of human domains. PMGAN’s 2D morph maps are interpretable and easy to edit as we know exactly what each pixel of the morph maps represent - the position of the source pixel that will be morphed into the pixel. Therefore, to compensate for the fixed difference between the nose locations, we add a gaussian shaped

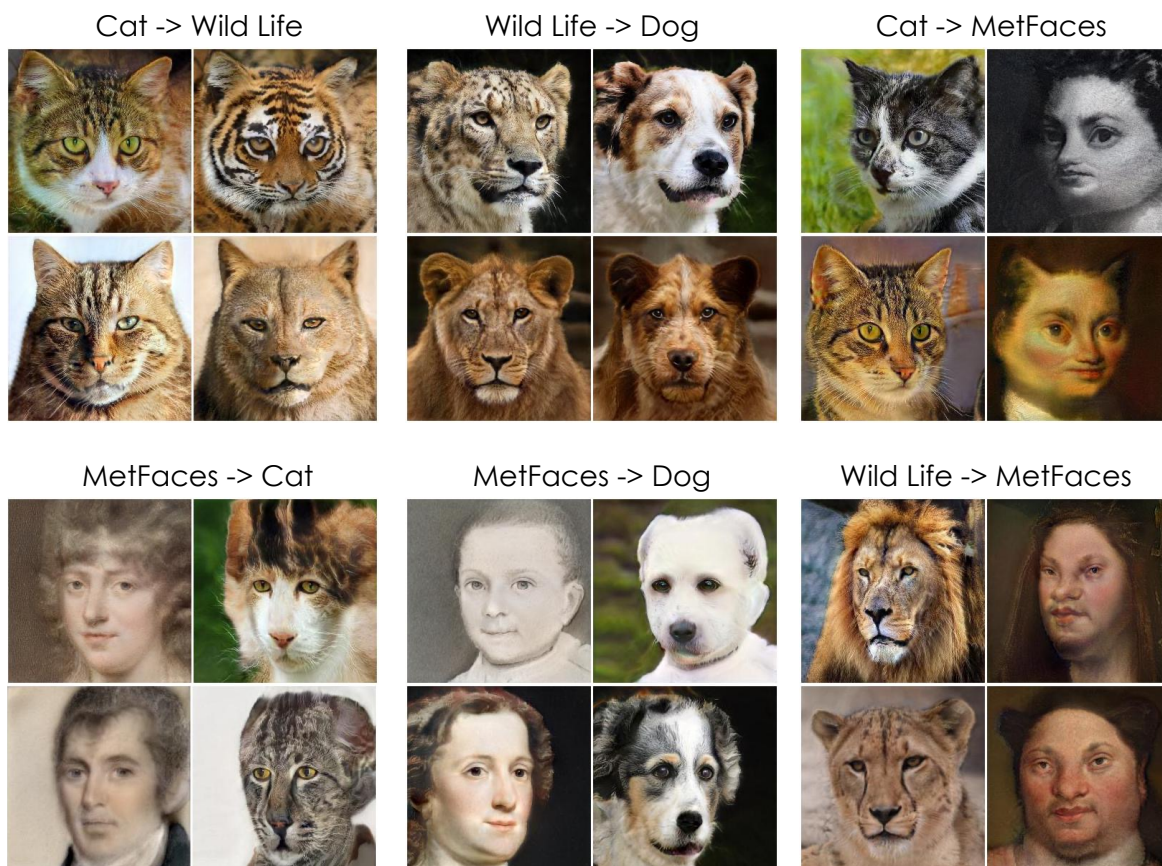


Figure 5. Rendering with the target domain's rendering layers while using the source domain's morph maps.

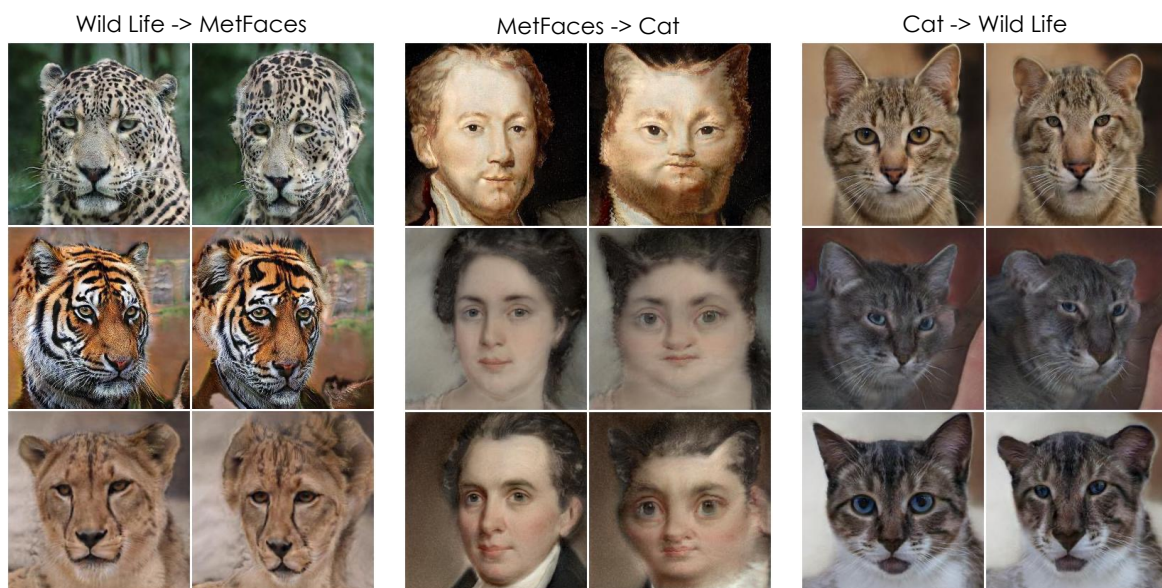


Figure 6. Rendering with the source domain's rendering layers while using the target domains morph maps. Note how only the shape changes according to the target domain indicating the disentanglement between shape and rendering.

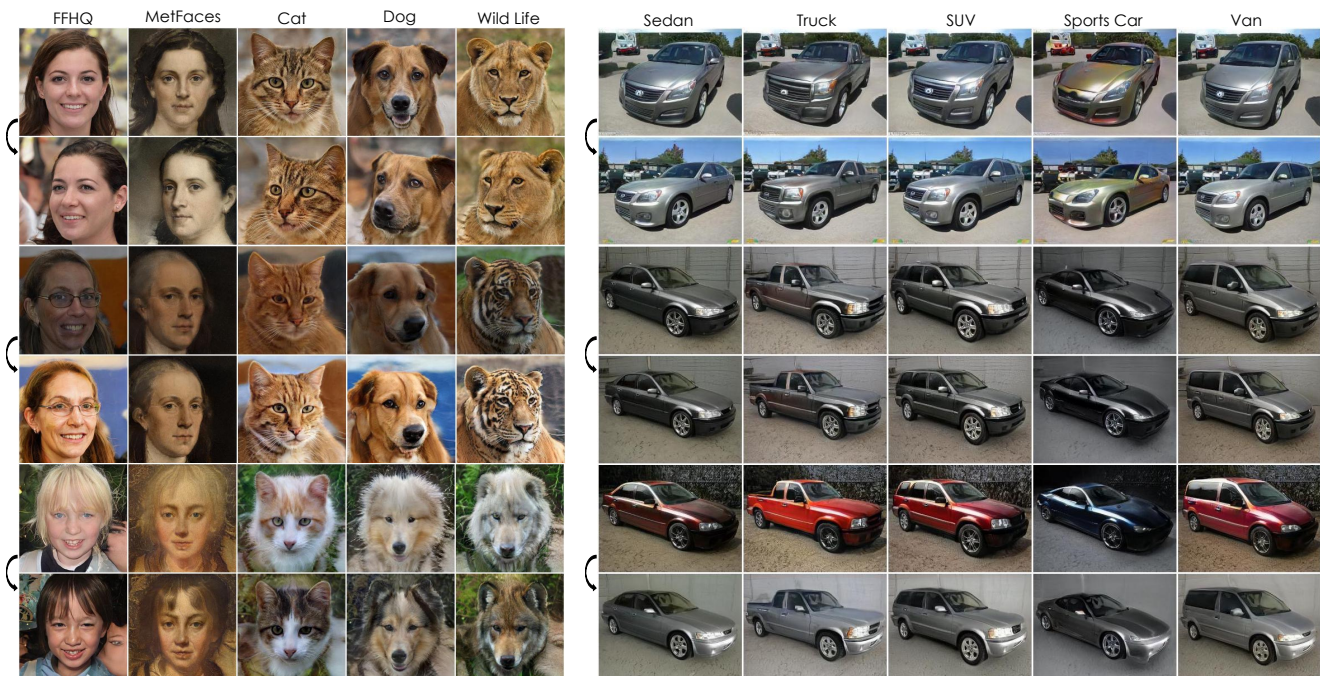


Figure 7. Edit transfer. Edit directions discovered through PMGAN’s core generator can be transferred across all domains. **Faces:** *top-rotation, middle-brightness, bottom-color*. **Cars:** *top-rotation, middle-zoom, bottom-color*.



Figure 8. Random samples produced by StyleGAN2 trained on 5% of MetFaces dataset from Table 7 in the main paper.

downward offsets to the location of human face’s nose before transferring its segmentation. We set the peak of offset to be -0.15 in the vertical direction (0 for the horizontal direction) which corresponds to moving nose downward by 7.5% of the image size. The offset only needs to be calculated once. We emphasize that this still promotes feature sharing, as the rendering layers need to render the same features (*i.e.* features representing *mouth* from human domain) according to their domain. This also shows editing shapes directly using morph map is an interesting direction, which we leave for future work.

3.4. Image-to-Image Translation

Once an image is inverted in the latent space, PMGAN can naturally be used for image-to-image translation (I2I) tasks by synthesizing every other domain with the same latent code. For both datasets, we use the w -latent space of StyleGAN which is the output space of the mapping network of the core generator. On Faces dataset, we use encoder4editing [14] (official code from <https://github.com/omertov/encoder4editing>) with an additional latent space loss we found to be helpful. The latent space loss is defined as $\sum_d \mathbb{E}_{w \sim p(w)} \|E_d(G_d(w)) - w\|$ where E_d is the encoder for domain d to be learned, G_d is the fixed pre-trained PMGAN for domain d and w is the sampled latent vector. As we can synthesize as many sampled image and latent pair $(G(w), w)$ as we want, this loss helps stabilizing the encoder training. We add the latent space loss to the original loss function of encoder4editing.

On Cars dataset, we use latent optimization [7] to encode input images, which we found to be encoding better than encoder4editing. We suspect the diversity of Cars dataset makes learning a generic encoder for the dataset challenging. We use 250 optimization steps with learning rate of 0.1, reducing the LPIPS [17] distance between the output and input images.

Figure 11 and 12 contain additional image-to-image translation results from PMGAN. We also provide more translation results from StarGANv2 in Figure 13.

3.5. Low-Data Regime

As indicated in Section 3.5 of the main text, for low-data regime training, we weigh losses by $|\pi^d|/\max_l |\pi^l|$ where $|\pi^d|$ is the number of training examples in domain d . The intuition is that we want the generator features to be mostly learned from data-rich domains while domains with significantly less data leverage the rich representation with domain-specific layers. To demonstrate how FID was not able to capture the mode-collapse phenomenon, we include Figure 8 showing random samples produced by StyleGAN2 trained on 5identity, indicating the model produces high-quality faces by memorizing them, but this mode-collapse was not reflected well in FID as shown in Table 7. We will add more examples for different models in the supplementary.

3.6. Comparison to Plain Fine-Tuning

We use FreezeD [10] for fine-tuning experiments. Mo *et al.* [10] found that freezing low-level discriminator layers improves fine-tuning performance. We freeze three discriminator layers for fine-tuning results. We note that StyleGAN-ADA [6] also uses FreezeD along with their proposed adaptive discriminator augmentation. We have not used the adaptive discriminator augmentation in this paper, but adding it to FreezeD or our model potentially would improve results, especially when the number of data in target domain is small.

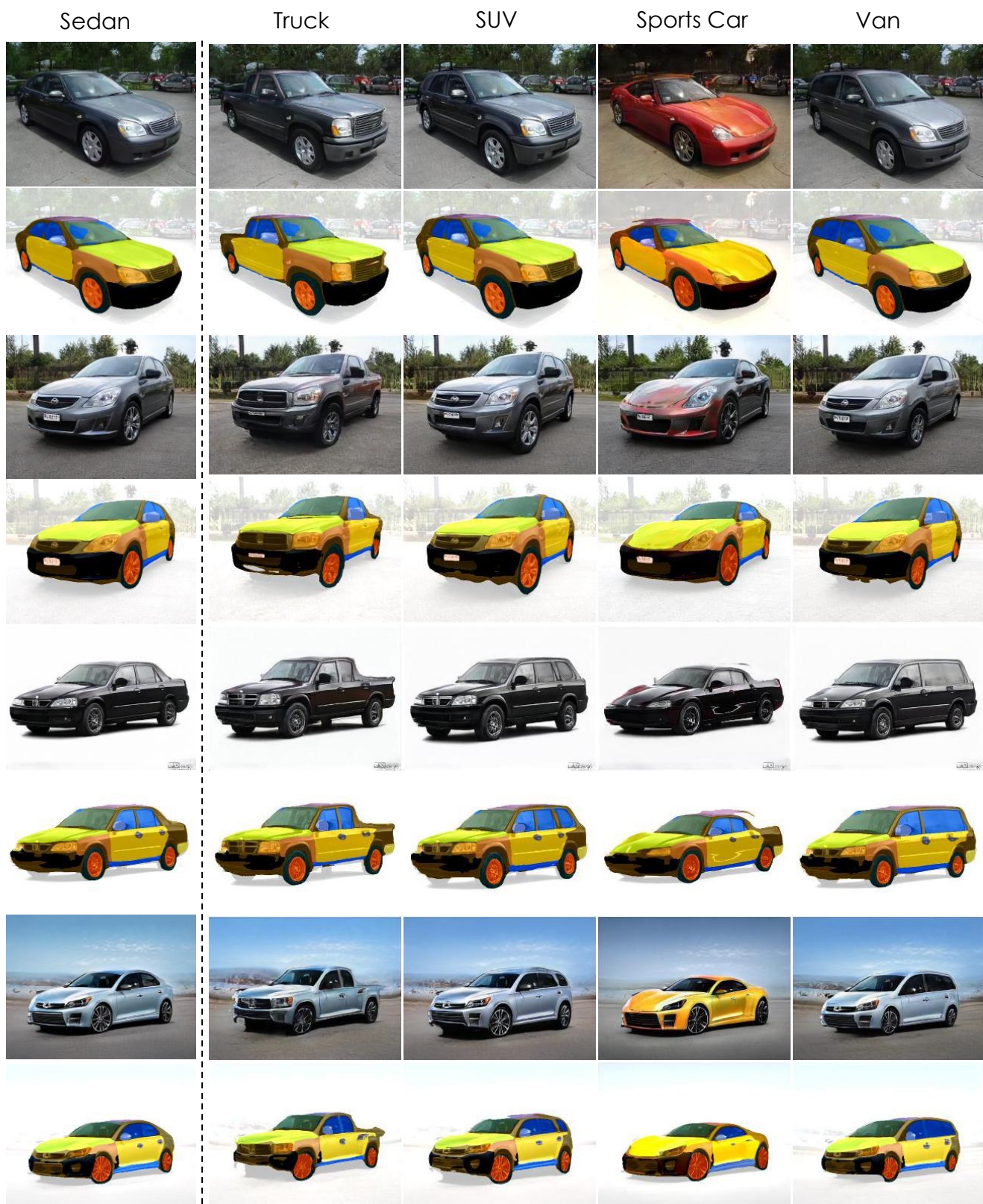


Figure 9. Zero-shot Segmentation Transfer on Cars. The segmentation mask from the leftmost column is transferred to all other domains.

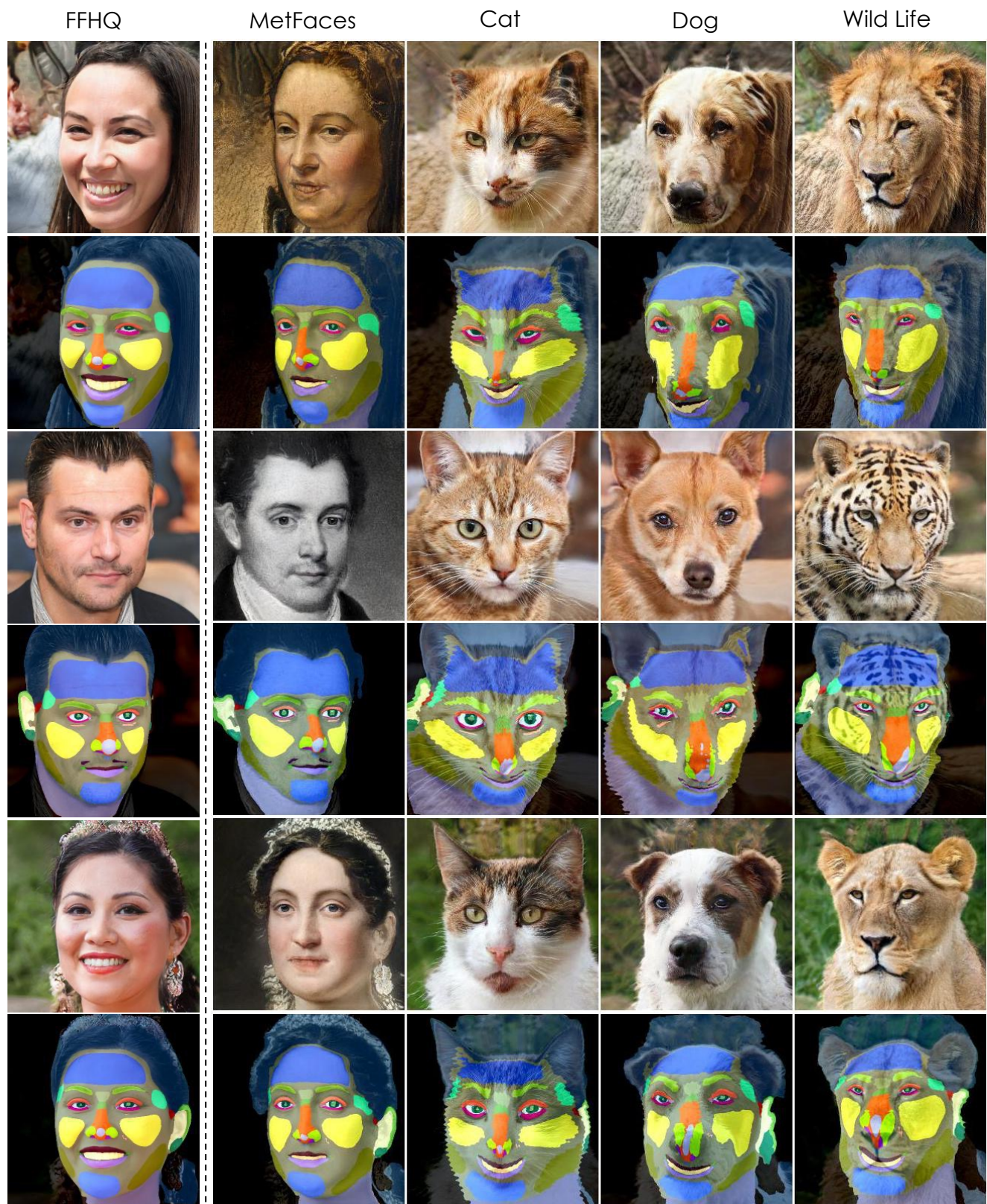


Figure 10. Zero-shot Segmentation Transfer on Faces. The segmentation mask from the leftmost column is transferred to all other domains.

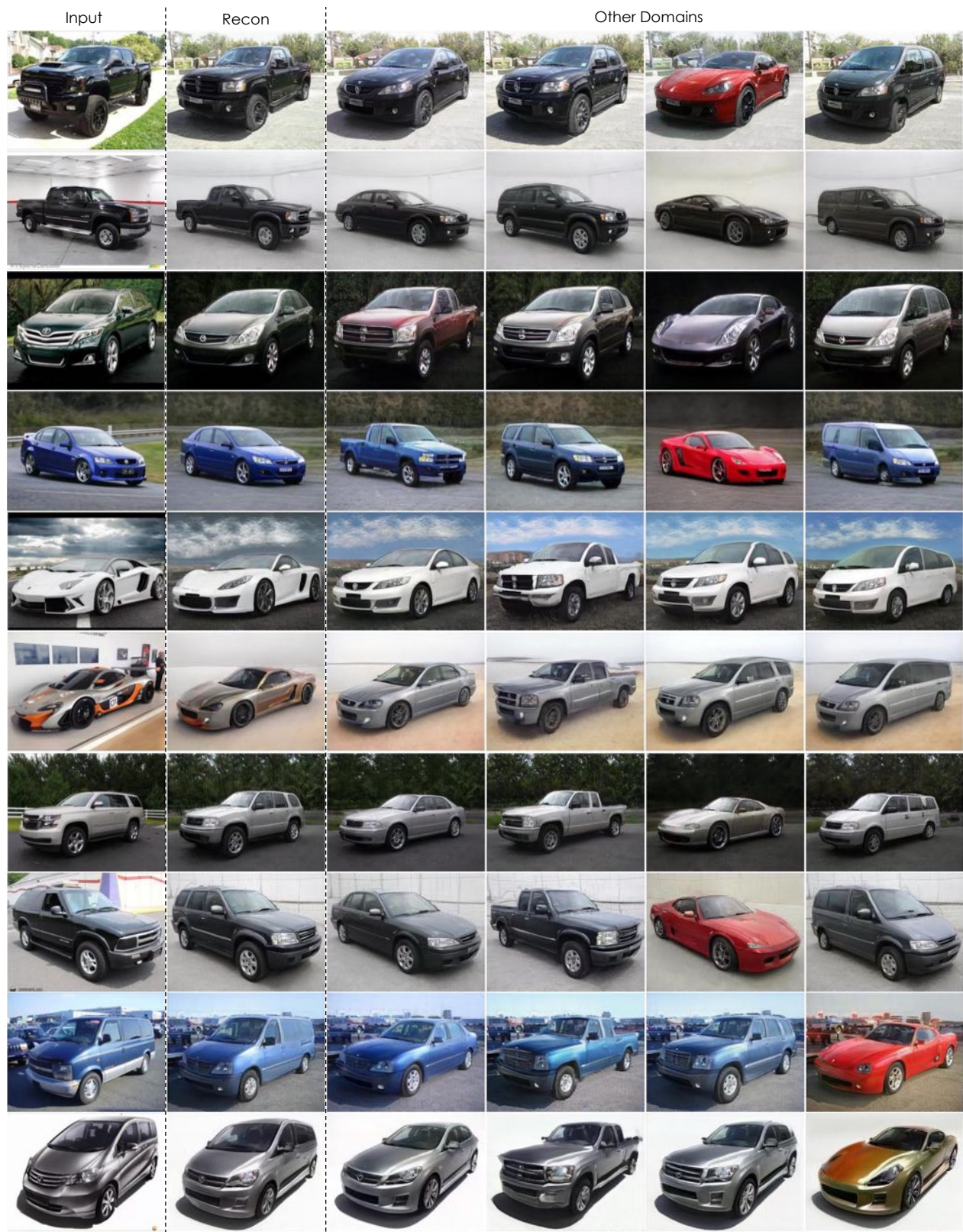


Figure 11. Image-to-Image translation results on Cars dataset.

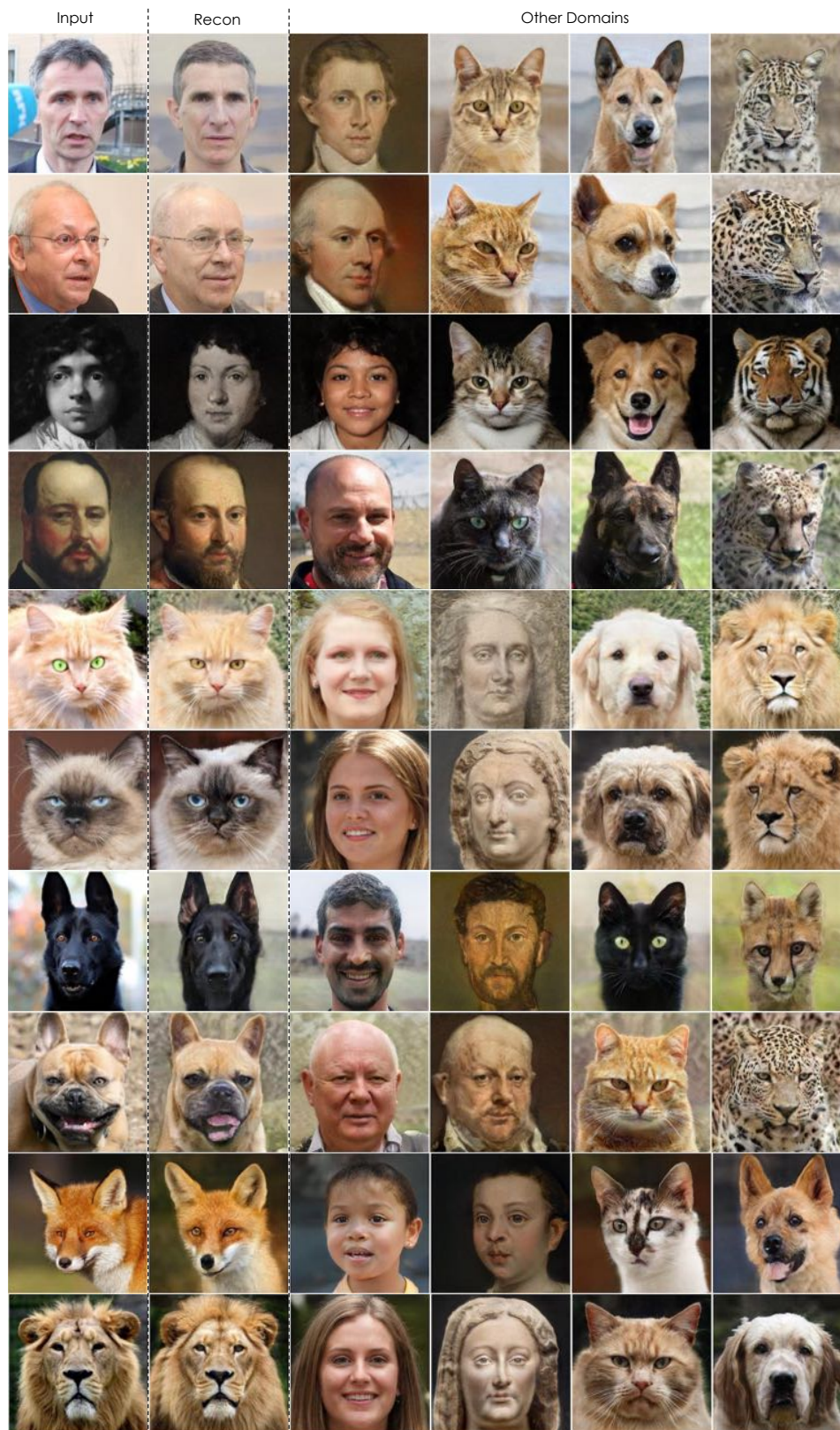


Figure 12. Image-to-Image translation results on Faces dataset.

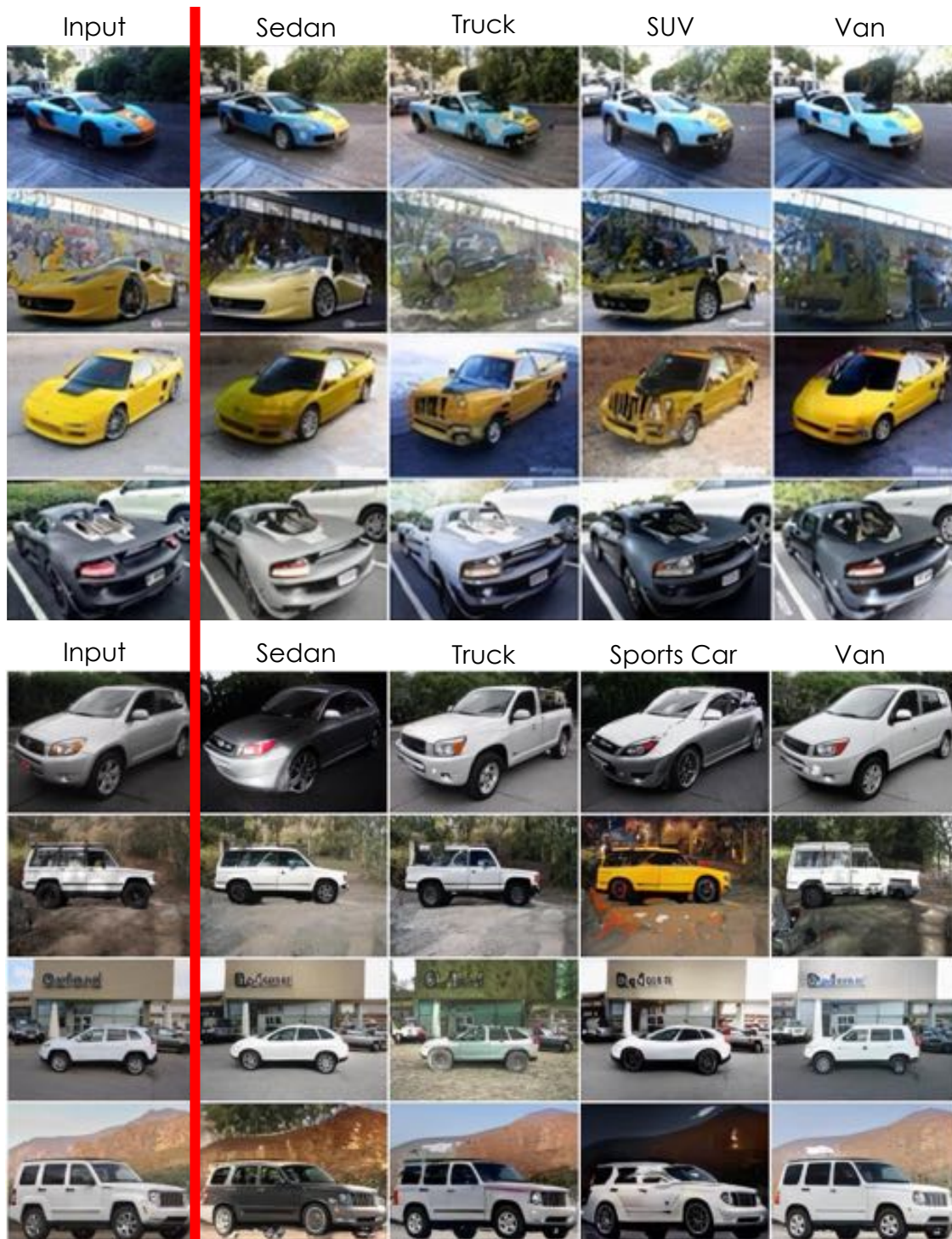


Figure 13. Additional Image-to-Image translation results on Cars dataset from baseline StarGANv2 model. StarGANv2 generally has difficulty changing the geometry of the input.

References

- [1] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2019. 5
- [2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 6
- [3] 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. 2
- [4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5
- [5] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *NeurIPS*, 28:2017–2025, 2015. 2
- [6] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Proc. NeurIPS*, 2020. 2, 9
- [7] 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. 2, 9
- [8] 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. 2
- [9] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013. 1
- [10] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Freeze the discriminator: a simple baseline for fine-tuning gans. In *CVPR AI for Content Creation Workshop*, 2020. 9
- [11] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses, 2021. 2
- [12] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1532–1540, 2021. 6
- [13] Yichun Shi, Debayan Deb, and Anil K Jain. Warpgan: Automatic caricature generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10762–10771, 2019. 2
- [14] 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. 9
- [15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1
- [16] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, 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. 2
- [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. 9
- [18] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. Datasetgan: Efficient labeled data factory with minimal human effort. In *CVPR*, 2021. 6