Supplementary for DeltaGAN: Towards Diverse Few-shot Image Generation with Sample-Specific Delta

Yan Hong, Li Niu*, Jianfu Zhang, and Liqing Zhang

MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University, China
yanhong.sjtu@gmail.com, {ustcnewly,c.sis}@sjtu.edu.cn,
zhang-lq@cs.sjtu.edu.cn

In this document, we provide additional material to support our main submission. In Section 1, we detail the split setting of datasets used in our paper. In Section 2, we describe the structure of our reconstruction subnetwork, generation subnetwork, and discriminators. In Section 3, we visualize more images generated from our DeltaGAN. In Section 4, we report the results of low-data classification augmented by generated images. In Section 5, we show the interpolation results of our DeltaGAN on three datasets. In Section 6, we show some reconstructed images from our reconstruction subnetwork. In Section 7, we show some images by exchanging deltas. In Section 8, we compare our DeltaGAN with baselines in K-shot setting with different K. In Section 9, we report comparison results between our method and other few-shot image generation methods relying on finetuning in the test phase. In Section 10, we compare our DeltaGAN with few-shot image translation method FUNIT. In Section 11, we further analyze the limitation of our DeltaGAN.

1 Datasets and Implementation Details

Datasets We conduct experiments on six few-shot image datasets: EMNIST [5], VGGFace [3], Flowers [17], Animal Faces [6], NABirds [20], and Foods [11]. Following the split setting of MatchingGAN [9] (*resp.*, FUNIT [15]), we split VGGFace and EMNIST (*resp.*, Animal Faces, Flowers, NABirds, and Foods) into seen categories and unseen categories. In detail, for VGGFace (*resp.*, EMNIST) dataset, we randomly select 1802 (*resp.*, 28) categories from all categories as seen training categories and select 96 (*resp.*,10) categories from remaining categories as unseen testing categories. On Flowers (*resp.*, Animal Faces, NABirds, and Foods) dataset, a total of 102 (*resp.*, 149, 555, and 256) categories are split into 85 (*resp.*, 119, 444, and 224) seen categories and 17 (*resp.*, 30, 111, and 32) unseen categories. In Table 1, we summarize the number of seen/unseen categories and the number of seen/unseen images.

Baselines We compare our DeltaGAN with FIGR [4], DAWSON [14], GMN [2], DAGAN [1], MatchingGAN [9], F2GAN [10], and LoFGAN [7]. For fair comparison, we conduct evaluation experiments for these methods using the same conditional images for each dataset.

* Corresponding author.

Table 1. The splits of seen/unseen images ("img") and categories ("cat") on six datasets

Dataset	Seen		Unseen	
	#img	#cat	#img	#cat
EMNIST [5]	78400	28	28000	10
VGGFace [3]	180200	1802	9600	96
Flowers [17]	7121	85	1068	17
Animal Faces [6]	96621	119	20863	30
NABirds [20]	38306	444	10221	111
Foods [11]	27471	224	3924	32

Table 2. Comparison of the number of model parameters (Million) and test time (second) among different few-shot image generation methods

Method	Model parameters		Test time
	Training phase	Testing phase	
FIGR [4]	23.9M	6.7M	0.0083s
GMN [2]	25.1M	18.5M	0.0324s
DAWSON [14]	26.6M	7.1M	0.0091s
DAGAN [1]	28.1M	8.7M	0.0104s
MatchingGAN [9]	28.9M	9.3M	0.0113s
F2GAN [10]	29.6M	9.5M	0.0121s
LoFGAN [7]	39.2M	7.9M	0.0149S
DeltaGAN	31.5M	7.7M	0.0098s

Implementation We implement our model using the TensorFlow 1.13.1 environment on Ubuntu 16.04 LTS equipped by GEFORCE RTX 2080 Ti GPU and Intel(R) Xeon(R) CPU $E5-2660$ v3 @ 2.60GHz CPU.

2 Details of Network Architecture

Generator Our generator consists of a reconstruction subnetwork and a generation subnetwork. Our reconstruction subnetwork (*resp.*, generation subnetwork) is constructed by 3 encoders including E_Δ, E_c, and E_r (*resp.*, E_f) and 1 decoder G. Encoder E_Δ has 5 residual blocks (ResBlks), which consists of 4 encoder blocks and 1 intermediate block. Each encoder block contains 3 convolutional layers with leaky ReLU and batch normalization followed by one downsampling layer, while the intermediate block contains 3 convolutional layers with leaky ReLU and batch normalization. The structure of encoder E_c is the same as encoder E_Δ without parameters sharing. Encoder E_r consists of two Conv-LRelu-BN blocks, in which each block contains 1 convolutional layer with leaky ReLU and batch normalization. Encoder E_f also has two Conv-LRelu-BN blocks. The decoder G consists of 4 residual blocks (ResBlks), in which each block contains 3 convolutional layers with leaky ReLU and batch normalization followed by one upsampling layer.

Table 3. Accuracy(%) of different methods on two datasets in low-data setting. Among few-shot image generation methods, only DAGAN and our DeltaGAN are applicable in 1-sample setting

Method	EMNIST				VGGFace			
	1	5	10	15	1	5	10	15
Standard	50.14	83.64	88.64	91.14	5.08	8.82	20.29	39.12
Traditional	52.82	84.62	89.63	92.07	8.87	9.12	22.83	41.63
FIGR [4]	-	85.91	90.08	92.18	-	6.12	18.84	32.13
GMN [2]	-	84.12	91.21	92.09	-	5.23	15.61	35.48
DAWSON [14]	-	83.63	90.72	91.83	-	5.27	16.92	30.61
DAGAN [1]	57.84	87.45	94.18	95.58	13.27	19.23	35.12	44.36
MatchingGAN [9]	-	91.75	95.91	96.29	-	21.12	40.95	50.12
F2GAN [10]	-	93.18	97.01	97.82	-	24.76	43.21	53.42
LoFGAN [7]	-	93.56	97.35	97.94	-	24.56	43.89	54.02
DeltaGAN	**84.56**	**96.02**	**98.12**	**98.87**	**22.91**	**28.91**	**50.19**	**58.72**

Discriminator Our discriminator D_I is analogous to that in [15], which consists of one convolutional layer followed by four groups of ResBlk. Each group of ResBlk is as follows: ResBlk-k $\rightarrow$ ResBlk-k $\rightarrow$ AvePool2x2, where ResBlk-k is a ReLU first residual block [16] with the number of channels k set as 64, 128, 256, 512 in four groups. We use one fully connected (FC) layer with 1 output following global average pooling (GAP) to obtain the discriminator score. Our discriminator D_M is constructed by four FC layers following GAP. The classifier shares the feature extractor with the discriminator D_I and only replaces the last FC layer with another FC layer with the number of outputs being the number of seen categories.

The number of model parameters We compare the number of model parameters of our DeltaGAN with MatchingGAN [9] and F2GAN [10] in the training stage and testing stage, respectively. In the training stage, the model parameters of generator and discriminator are trainable to complete two-player adversarial learning with seen categories, while only generator is used to generate new images for each unseen category in the testing stage. In particular, our DeltaGAN only uses generation subnetwork to generate new images without reconstruction subnetwork. In Table 2, we can see that our DeltaGAN uses fewer model parameters to generate images of better quality compared with MatchingGAN and F2GAN in the testing stage.

3 More Visualization Results

In this section, we visualize some example images generated by our DeltaGAN on EMNIST, VGGFace, Flowers, Animal Faces, NABirds, and Foods datasets in Fig. 1 and Fig. 2. On all datasets, our DeltaGAN can generally generate diverse and plausible images based on a single conditional image from unseen category.

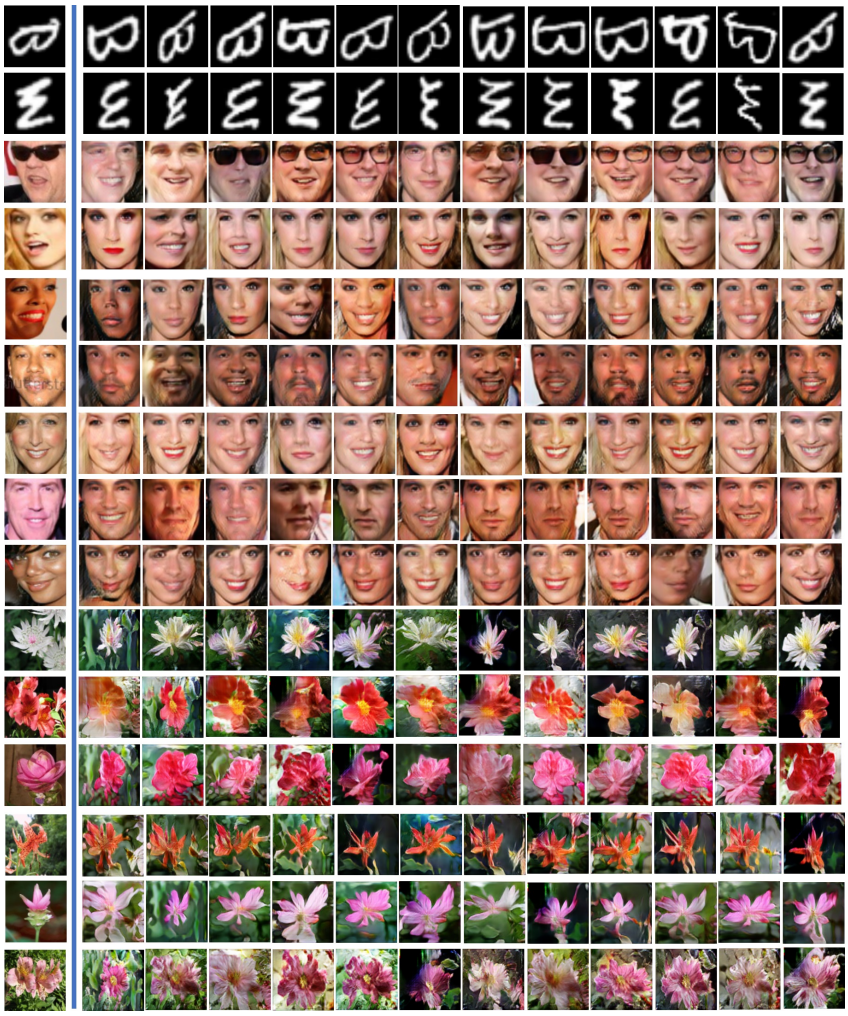

Fig. 1. Images generated by our DeltaGAN in 1-shot setting on three datasets. From top to bottom: EMNIST, VGGFace, and Flowers. The conditional images are in the leftmost column.

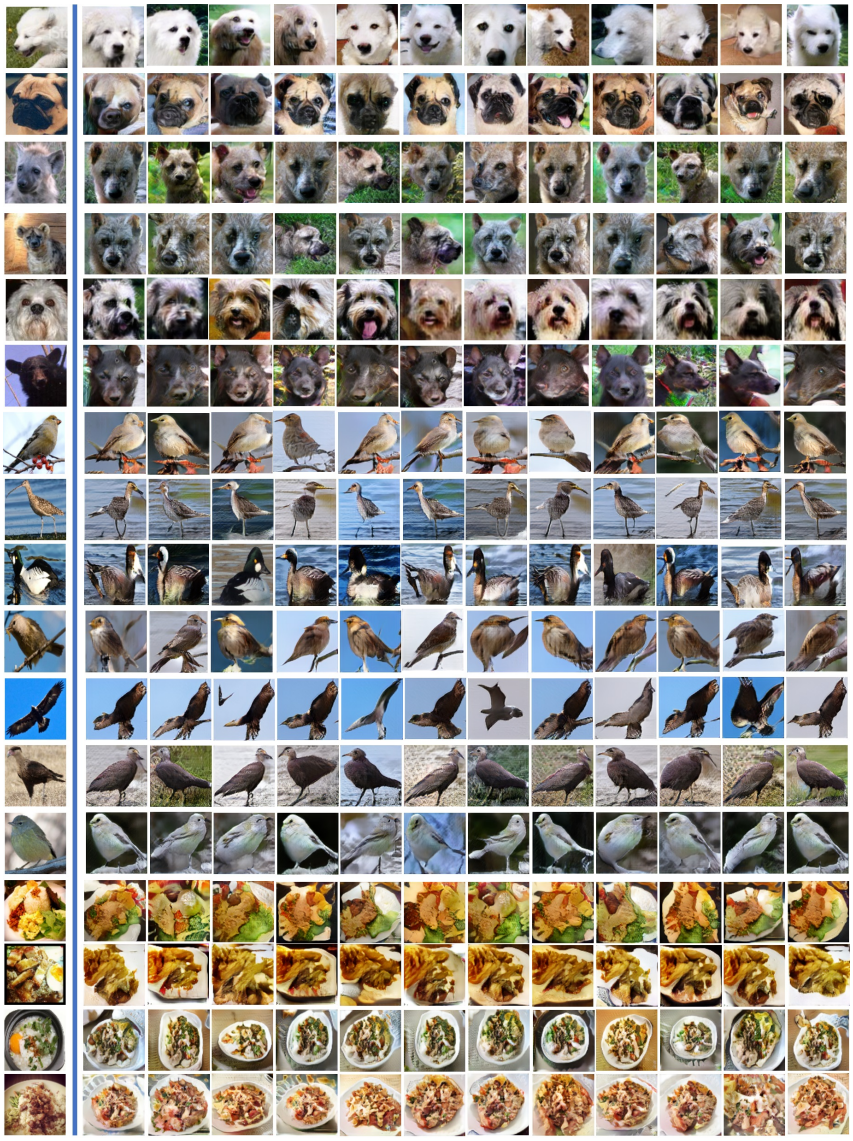

Fig. 2. Images generated by our DeltaGAN in 1-shot setting on three datasets. From top to bottom: Foods, Animal Faces, and NABirds. The conditional images are in the leftmost column.

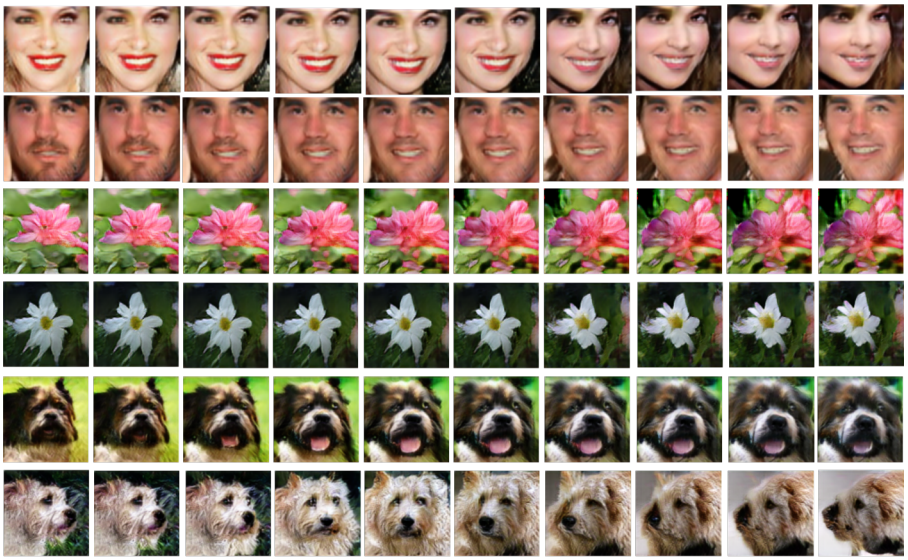

Fig. 3. Images generated by our DeltaGAN by interpolating random vectors between $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ on three datasets (from top to bottom: VGGFace, Flowers, and Animal Faces)

4 Low-data Classification

To further evaluate the quality of our generated images, we conduct downstream classification tasks in low-data setting by using generated images to augment unseen categories. Following F2GAN [10], for each unseen category, we randomly select a few (*e.g.*, $K = 1, 5, 10, 15$) training images and use the remaining images as test images, which is referred to as K-sample in Table 3. We initialize ResNet18 [8] backbone based on seen categories, then finetune the whole network with the training images of unseen categories, and finally apply the trained classifier to the test images of unseen categories. This setting is referred to as "Standard" in Table 3.

Then, we augment unseen training images with new images generated by different few-shot image generation methods. For each unseen category, one method generates 512 images by randomly sampling conditional images from the training set of this unseen category. Then, we augment the original training set of unseen categories with generated images, which are used to finetune the ResNet18 classifier. In addition, we compare with traditional data augmentation (*e.g.*, flip, crop, color jittering), which also generates 512 new images for each unseen category. The setting of traditional data augmentation is referred to as "Traditional" in Table 3. The results of different methods are reported in Table 3. We can see that our DeltaGAN achieves better results than traditional data augmentation methods as well as few-shot image generation baselines, which shows the effec-

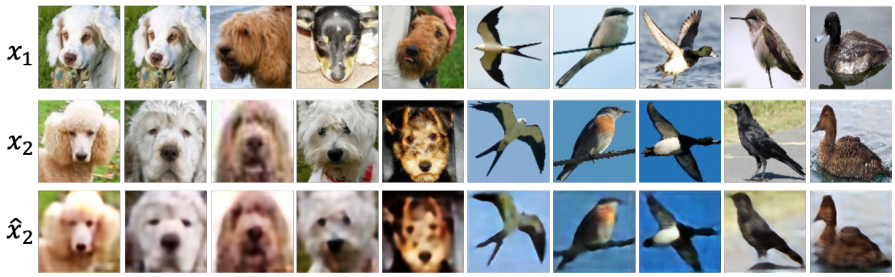

Fig. 4. Reconstruction results of our DeltaGAN on Animal Faces (left) and NABirds (right) datasets. The conditional images $\boldsymbol{x}_1$ are in the first row, the target images $\boldsymbol{x}_2$ are in the second row, and the reconstructed images $\hat{\boldsymbol{x}}_2$ are in the third row

tiveness of using augmented images produced by our DeltaGAN for low-shot classification task.

5 Delta Interpolation

To evaluate whether the delta space of DeltaGAN is densely populated, we perform linear interpolation based on two random vectors $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. In detail, we calculate the interpolated random vector $\boldsymbol{z} = a_1\boldsymbol{z}_1 + a_2\boldsymbol{z}_2$ $(a_1 + a_2 = 1)$ by gradually decreasing (*resp.*, increasing) a_1 (*resp.*, a_2) from 1 (*resp.*, 0) to 0 (*resp.*, 1) with step size 0.1. We use one conditional image and interpolated $\boldsymbol{z}$ to generate sample-specific deltas, which are used to produce interpolation results in Fig. 3. We can see that our DeltaGAN can generate diverse images with smooth transition between two random vectors $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, including the transition between different colors, shapes, and poses.

6 Image Reconstruction Results

In the training stage, our reconstruction network can reconstruct $\boldsymbol{x}_2$ based on $\boldsymbol{x}_1$ and $\boldsymbol{\Delta}^r_{x_1}$. To demonstrate that the reconstruction ability of reconstruction subnetwork can be transferred from seen categories to unseen categories, we visualize the reconstructed unseen images on Animal Faces and NABirds datasets in Fig. 4. To be exact, we randomly sample same-category unseen image pairs $\{\boldsymbol{x}_1, \boldsymbol{x}_2\}$, which pass through our reconstruction network to yield $\hat{\boldsymbol{x}}_2$. From Fig. 4, we can see that the reconstructed images $\hat{\boldsymbol{x}}_2$ are quite close to the target images $\boldsymbol{x}_2$.

7 Visualization of Exchanging Delta

Note that our learnt delta is sample-specific delta. To check whether the delta is transferable across different images, we first show some generated images after exchanging delta in the reconstruction subnetwork. As shown in Fig. 5, we

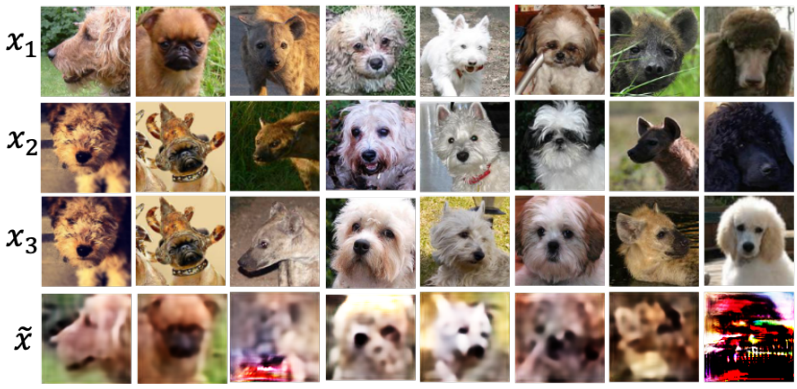

Fig. 5. Visualization results of exchanging delta in the reconstruction subnetwork on Animal Faces. From top to bottom: conditional image $\boldsymbol{x}_1$, a pair of $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ providing real delta, $\tilde{\boldsymbol{x}}$ generated based on $\boldsymbol{x}_1$ and the real delta

extract real delta $\boldsymbol{\Delta}_{23}^r$ from one pair of images $\{\boldsymbol{x}_2, \boldsymbol{x}_3\}$ from the same category as $\boldsymbol{x}_1$, after which $\boldsymbol{\Delta}_{23}^r$ is applied to the conditional image $\boldsymbol{x}_1$ to generate a new image $\tilde{\boldsymbol{x}}$. In column 1-2, we investigate a special case $\boldsymbol{x}_2 = \boldsymbol{x}_3$. In this case, the delta is vacuous and thus the generated image is close to the conditional image $\boldsymbol{x}_1$. Recall that the real delta $\boldsymbol{\Delta}_{23}^r$ between $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ contains the necessary information required to transform $\boldsymbol{x}_2$ to $\boldsymbol{x}_3$. In column 3-6, we show some cases, in which the delta appearance information or delta pose information encoded in $\boldsymbol{\Delta}_{23}^r$ influences $\boldsymbol{x}_1$ to some degree. For example, in column 3, the fur color of $\boldsymbol{x}_3$ is lighter than $\boldsymbol{x}_2$, so the fur color of generated image $\tilde{\boldsymbol{x}}$ is also lighter than $\boldsymbol{x}_1$. In column 5, $\boldsymbol{x}_3$ turns face to the left compared with $\boldsymbol{x}_2$, so $\tilde{\boldsymbol{x}}$ also turns face to the left compared with $\boldsymbol{x}_1$. However, the generated images are generally of low quality. In column 7-8, the generated images $\tilde{\boldsymbol{x}}$ are corrupted when there is huge difference between $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ in appearance and pose.

Additionally, we show the visualization results of exchanging delta in the generation subnetwork in Fig. 6, in which we apply the delta provided by $\boldsymbol{x}_2$ to the conditional image $\boldsymbol{x}_1$ to generate a new image $\tilde{\boldsymbol{x}}$. The left (*resp.*, right) four columns correspond to "SC delta" (*resp.*, "DC delta") in Table 3 in main paper, in which $\boldsymbol{x}_2$ is from the same (*resp.*, different) category of $\boldsymbol{x}_1$. We can see that exchanging delta usually leads to severely degraded quality of generated images compared with DeltaGAN. Another observation is that "DC delta" is more inclined to generated unreasonable images compared with "SC delta". These observations coincide with the quantitative results of "DC delta" and "SC delta" in Table 3 in main paper.

As shown in Fig. 5 and Fig. 6, a plausible delta for one conditional image may be unsuitable for another conditional image, so we target at learning sample-specific delta. The learnt sample-specific delta has weak transferability across images and weaker transferability across categories. Hence, it is not suggested

Fig. 6. Visualization results of exchanging delta in the generation subnetwork on Animal Faces. From top to bottom: conditional image $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ providing the delta, $\tilde{\boldsymbol{x}}$ generated based on $\boldsymbol{x}_1$ and the delta of $\boldsymbol{x}_2$

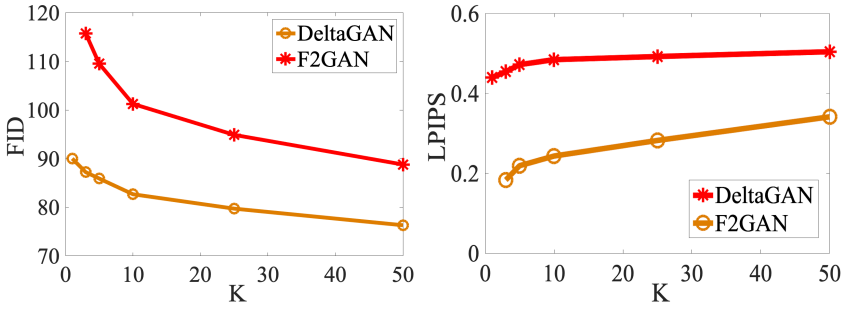

Fig. 7. FID and LPIPS comparison between F2GAN and our DeltaGAN with different numbers (K) of conditional images on Animal Faces

to generate new images by exchangeably applying delta. However, the ability of generating effective sample-specific delta is transferable from seen categories to unseen categories, so our DeltaGAN is effective in producing new realistic images based on a conditional unseen image.

8 Few-shot Generation Ability

Here, we repeat the experiments in Section 4.1 in the main paper except tuning K in a wide range. Recall that K in K-shot setting means that K real images are provided for each unseen category. We use our DeltaGAN and competitive baseline F2GAN to generate 128 new images for each unseen category in K-shot setting. We also adopt the same quantitative evaluation metrics (FID and LPIPS) to measure image quality and diversity as in Section 4.1 in the main paper. We plot the FID curve and LPIPS curve of two methods with increasing K in Fig. 7. It can be seen that our DeltaGAN outperforms F2GAN by a large

Setting	FID ↓	LPIPS ↑
[18]	109.89	0.2879
Ours	**109.78**	**0.3912**

Table 4. FID (↓) and LPIPS (↑) of images generated by [18] and our method on Flowers dataset

margin with all values of K, especially when K is very small. These results demonstrate that our DeltaGAN can generate abundant diverse and realistic images even if only a few (*e.g.*, 10) real images are provided.

9 Comparison with Other Few-shot Image Generation

As discussed in Section 2 in the main paper, our setting is quite different from recent works [18,13,19,21]. Our method can achieve instant adaptation to multiple unseen categories without finetuning, while the abovementioned methods need to finetune the trained model for each unseen category, which is very resource-consuming and time-consuming. We compare the performance of [18] with our method on Flowers dataset. For [18], we train a source model on all seen images during training. At test stage, we finetune the source model on each selected unseen category with one image (1-shot setting in Section 4.1 in main paper) and produce 128 images by sampling random vectors for evaluation. We report the results in Table 4. We can see that our method slightly outperforms [18] and produces more diverse and realistic images. However, [18] requires finetuning for each unseen category, while our model can be instantly adapted to unseen categories without finetuning.

10 Comparison with Few-shot Image Translation

Recently, few-shot image translation methods like FUNIT [15] have been proposed to translate seen images to unseen categories, which can also generate new images for unseen categories given a few images. However, the motivations of few-shot image generation and few-shot image translation are considerably different. Specifically, the former can generate new unseen images without touching seen images, while the latter relies on seen images to generate new unseen images. In particular, FUNIT disentangles the latent representation of an image into category-relevant representation (*i.e.*, class code) and category-irrelevant representation (*i.e.*, content code), in which appearance belongs to class code and pose belongs to content code [15]. In the testing stage, given a few images from one unseen category, FUNIT generates new images for this unseen category by combining the content codes of seen images with the class codes of these unseen images. However, in reality, the disentanglement in FUNIT is not perfect and the content code may also contain appearance information. So when translating seen images to unseen categories, the appearance information of seen images

Table 5. Accuracy(%) of different methods on Animal Faces in few-shot classification setting. Note that MatchingGAN, F2GAN, and LoFGAN are not applicable in 1-shot setting

Method	10-way 1-shot	10-way 5-shot
DPGN [22]	57.18	72.02
DeepEMD [23]	58.01	72.71
MatchingGAN [9]	-	70.89
F2GAN [10]	-	73.19
LoFGAN [7]	-	73.43
FUNIT-1	56.61	69.12
FUNIT-2	53.38	67.87
DeltaGAN	**60.31**	**74.59**

Fig. 8. Images generated by FUNIT [15] in 1-shot setting on Animal Faces. Unseen (*resp.*, seen) images are shown in the first (*resp.*, second) row. In each column, the new image is generated by combining the class code of unseen image and the content code of seen image

may be leaked to the generated new images. To corroborate this point, we visualize some example images generated by FUNIT. As shown in Fig. 8, in column 1-4, the generated new images contain the appearance of seen images, which is against our expectation that the generated new images should be from unseen categories. In column 5-8, the generated images are even corrupted, probably due to incompatible content codes and class codes.

We also compare our DeltaGAN with FUNIT quantitatively for few-shot classification. By using the released model of FUNIT [15] trained on Animal Faces [6], we combine the content codes of seen images and the class codes of unseen images to produce 512 new images for each unseen category. Then, the generated images are used to facilitate few-shot classification, which is recorded as "FUNIT-1" in Table 5. However, "FUNIT-1" leverages seen images, which are not used in our DeltaGAN when generating new unseen images. For fair comparison, we also exchange content codes within the images from the same unseen category to produce new images, which is recorded as "FUNIT-2" in

Table 6. Significance test between DeltaGAN and F2GAN on Animal Faces dataset in 3-shot setting

Setting	Accuracy(%) ↑	FID ↓	LPIPS ↑
F2GAN	75.65±0.32	117.12±0.29	0.1903±0.17
DeltaGAN	**77.13±0.21**	**87.12±0.06**	**0.4661±0.11**

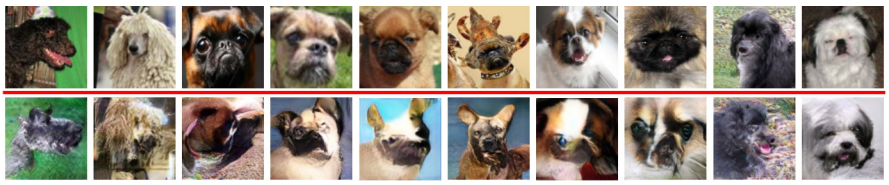

Fig. 9. Failure cases of our DeltaGAN on Animal Faces dataset. The conditional images are shown in the top row and the generated images are shown in the bottom row

Table 5. In this case, we can only generate $(C-1)\times C$ new images for each unseen category in N-way C-shot setting. Based on Table 5, we observe that "FUNIT-2" is much worse than "FUNIT-1", because "FUNIT-1" resorts to a large number of extra seen images to generate more unseen images than "FUNIT-2". We also observe that "FUNIT-1" underperforms some few-shot classification methods and some few-shot image generation methods (*e.g.*, F2GAN, DeltaGAN), which may be attributed to the appearance information leakage or image corruption as shown in Fig. 8.

11 Limitation

Failure cases: Although our model achieves promising results both qualitatively and quantitatively on six datasets, some generated deltas are applied to the conditional images to produce distorted images due to the complexity of transformations between intra-category pairs. We show some failure cases of our DeltaGAN on Animal Faces dataset in Fig. 9.

Generation ability on coarse-grained datasets: Our method can transfer knowledge learned from seen categories to unseen categories to produce compelling results for unseen categories on fine-grained datasets. However, like other competitive few-shot image generation method F2GAN [10] and few-shot image translation method FUNIT [15], our method cannot achieve satisfactory results on coarse-grained datasets, such as CIFAR-100 [12] with large inter-category variance. We randomly divide a total of 100 categories into 80 seen training categories and 20 unseen testing categories to conduct experiments for F2GAN, FUNIT, and our DeltaGAN. Similar to Section 4.1 in main paper, we visualize some example images generated by different methods in 3-shot setting in Fig. 10. For FUNIT in 3-shot setting, we randomly select two seen images as content images to combine with each conditional image to produce new images.

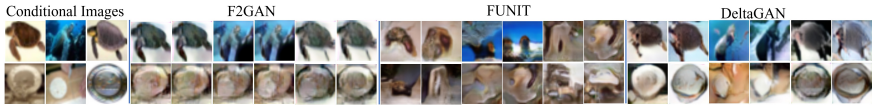

Fig. 10. Images generated by F2GAN, FUNIT and our DeltaGAN in 3-shot setting on CIFAR-100 dataset. The conditional images are in the left three column. The first row is category "turtle", the second row is category "plate".

Fig. 11. Failure cases of our DeltaGAN trained on Animal Faces dataset and tested on Flowers dataset.The conditional images are in the leftmost column.

We can observe that the structures of images generated by F2GAN are similar to conditional images. The generated images are vague and lacking local details. For FUNIT, the generated images do not have clear shape or overall structure. In contrast, the images produced by our DeltaGAN are relatively more diverse with clearer shape. We have to acknowledge that the quality of images generated by all three methods is poor, because it is difficult to achieve instant adaptation on coarse-grained datasets with large inter-category variance.

Adaptation ability between different datasets: We also explore the adaptation ability of our DeltaGAN on Animal Faces dataset and Flowers dataset. In detail, we train DeltaGAN on Animals dataset, and test on Flowers dataset. Similar to Section 4.1 in main paper, we show some example images in Fig. 11. We can see that the generated images belong to the different animal face categories, although the given conditional images are from Flowers dataset. The distributions of sample-specific deltas on different datasets are considerably different, so DeltaGAN trained on Animal Faces dataset fails in capturing sample-specific deltas for Flowers dataset.

References

1. Antoniou, A., Storkey, A., Edwards, H.: Data augmentation generative adversarial networks. arXiv preprint arXiv:1711.04340 (2017)
2. Bartunov, S., Vetrov, D.: Few-shot generative modelling with generative matching networks. In: AISTATS (2018)

3. Cao, Q., Shen, L., Xie, W., Parkhi, O.M., Zisserman, A.: Vggface2: A dataset for recognising faces across pose and age. In: FG (2018)
4. Clouâtre, L., Demers, M.: Figr: Few-shot image generation with reptile. arXiv preprint arXiv:1901.02199 (2019)
5. Cohen, G., Afshar, S., Tapson, J., van Schaik, A.: EMNIST: an extension of MNIST to handwritten letters. In: IJCNN (2017)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
7. Gu, Z., Li, W., Huo, J., Wang, L., Gao, Y.: Lofgan: Fusing local representations for few-shot image generation. In: ICCV (2021)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
9. Hong, Y., Niu, L., Zhang, J., Zhang, L.: Matchinggan: Matching-based few-shot image generation. In: ICME (2020)
10. Hong, Y., Niu, L., Zhang, J., Zhao, W., Fu, C., Zhang, L.: F2gan: Fusing-and-filling gan for few-shot image generation. In: ACM MM (2020)
11. Kawano, Y., Yanai, K.: Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In: ECCV (2014)
12. Krizhevsky, A., et al.: Learning multiple layers of features from tiny images (2009)
13. Li, Y., Zhang, R., Lu, J., Shechtman, E.: Few-shot image generation with elastic weight consolidation. In: NeurIPS (2020)
14. Liang, W., Liu, Z., Liu, C.: Dawson: A domain adaptive few shot generation framework. arXiv preprint arXiv:2001.00576 (2020)
15. Liu, M., Huang, X., Mallya, A., Karras, T., Aila, T., Lehtinen, J., Kautz, J.: Few-shot unsupervised image-to-image translation. In: ICCV (2019)
16. Mescheder, L.M., Geiger, A., Nowozin, S.: Which training methods for gans do actually converge? In: ICML (2018)
17. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: CVGIP (2008)
18. Ojha, U., Li, Y., Lu, J., Efros, A.A., Lee, Y.J., Shechtman, E., Zhang, R.: Few-shot image generation via cross-domain correspondence. In: CVPR (2021)
19. Robb, E., Chu, W.S., Kumar, A., Huang, J.B.: Few-shot adaptation of generative adversarial networks. arXiv preprint arXiv:2010.11943 (2020)
20. Van Horn, G., Branson, S., Farrell, R., Haber, S., Barry, J., Ipeirotis, P., Perona, P., Belongie, S.: Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In: CVPR (2015)
21. Wang, Y., Gonzalez-Garcia, A., Berga, D., Herranz, L., Khan, F.S., van de Weijer, J.: Minegan: Effective knowledge transfer from gans to target domains with few images. In: CVPR (2020)
22. Yang, L., Li, L., Zhang, Z., Zhou, X., Zhou, E., Liu, Y.: Dpgn: Distribution propagation graph network for few-shot learning. In: CVPR (2020)
23. Zhang, C., Cai, Y., Lin, G., Shen, C.: Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In: CVPR (2020)