Supplementary: Multi-Agent Diverse Generative Adversarial Networks

Arnab Ghosh*
University of Oxford, UK
arnabg@robots.ox.ac.uk

Viveka Kulharia *
University of Oxford, UK
viveka@robots.ox.ac.uk

Vinay Namboodiri
IIT Kanpur, India
vinaypn@iitk.ac.in

Philip H.S. Torr
University of Oxford, UK
philip.torr@eng.ox.ac.uk

Puneet K. Dokania
University of Oxford, UK
puneet@robots.ox.ac.uk

Here we first give better insights about the Theorem 1 (Eq. (2)) in the main paper and discuss how and when MAD-GAN leads to diverse generations. In addition to that, we provide additional experiments (both synthetic and real-world datasets) for ablation study, and then architectures and data preparation for all the experiments reported in the paper.

1. Insights for Diversity in MAD-GAN

One obvious question that could arise is that *is it possible that all the generators learn to capture the same mode?*. The short answer is, theoretically *yes* and in practice *no*. Let us begin with the discussion to understand this. Theoretically, if $p_{g_i} = p_d$, for all i, then also the minimum objective value can be achieved (Theorem 1 in the main paper). This implies, in worst case, MAD-GAN would perform same as the standard GAN. However, as discussed below, this is possible in following *highly unlikely* situations:

- all the generators *always generate exactly similar* samples so that the discriminator is not able to differentiate them. In this case, the discriminator will learn a uniform distribution over the generator indices, thus, the gradients passed through the discriminator will be exactly the same for all the generators. However, this situation in general is not possible as all the generators are initialized differently. Even a slight variation in the samples from the generators will be enough for the discriminator to identify them and pass different gradient information to each generator. In addition, the objective function of generators is *only* to generate *real* samples, thus, there is nothing that encourages them to generate exactly the same samples.

- the discriminator does not have enough capacity to learn the optimal parameters. This is in contrast to the assumption made in Theorem 1, which is that the discriminator is *optimal*. Thus, it should have enough capacity to learn a feature representation such that it can correctly identify samples from different generators. In practice, this is a very easy task and we did not have to modify anything up to the feature representation stage of the architecture of the discriminator. We used the standard architectures (explained in Section 3) for all the tasks.

Hence, with random initializations and sufficient capacity generator/discriminator, we can easily avoid the trivial solution in which all the generators focus on exactly the same region of the true data distribution. This has been very clearly supported by various experiments showing diverse generations by MAD-GAN.

# Generators	Chi-square ($\times 10^7$)	KL-Div
1	1.27	0.57
2	1.38	0.42
3	3.15	0.71
4	0.39	0.28
5	3.05	0.88
6	0.54	0.29
7	0.97	0.78
8	4.83	0.68

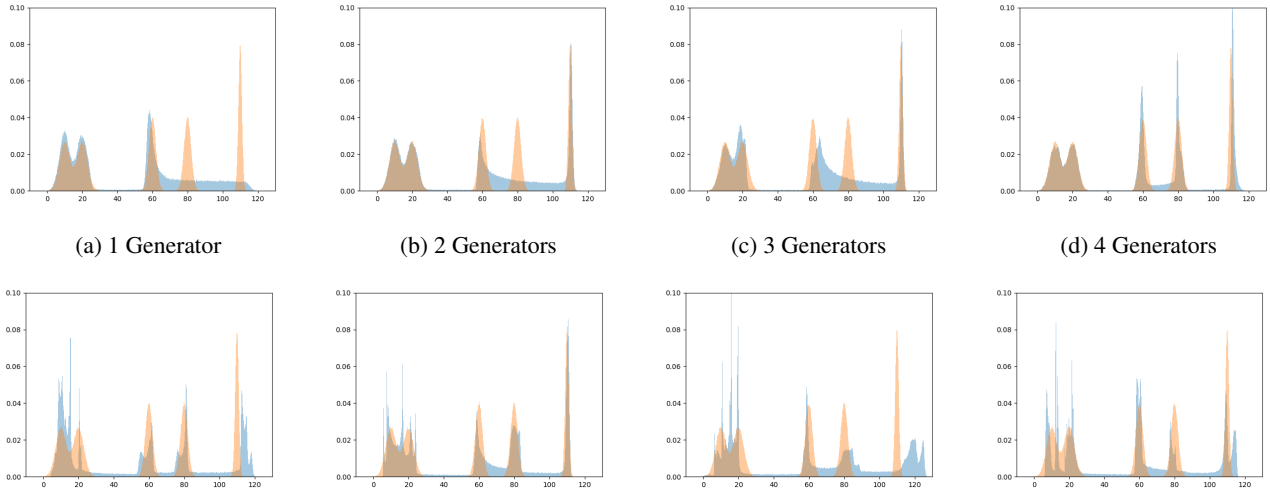
Table 1: Synthetic experiment with different number of MAD-GAN generators as Fig. 1.

2. Additional Experiments

2.1. Varying number of generators in MAD-GAN

Here we experiment by varying the number of generators in MAD-GAN. We use toy synthetic experiments to

*Joint first author



(a) 1 Generator (b) 2 Generators (c) 3 Generators (d) 4 Generators

(e) 5 Generators (f) 6 Generators (g) 7 Generators (h) 8 Generators

Figure 1: A toy example to understand the behavior of MAD-GAN with different number of generators (each method was trained for $1,98,000$ iterations). The orange bars show the density estimate of the training data and the blue ones for the generated data points. After careful cross-validation, we chose the bin size of 0.1.

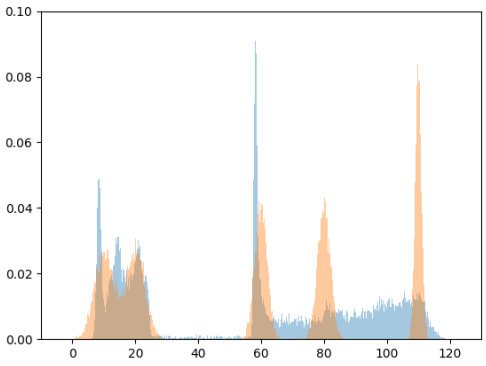


Figure 2: MA-GAN performance corresponding to "non-parametric density estimation" (Fig. 4) in the main paper.

understand the behavior. The real data distribution is the same as that in the main paper (GMM with 5 Gaussians). For better non-parametric estimation, we use 1 million sample points from real distribution (instead of $65,536$ used in the main paper). We generate equal number of points from each of the generators such that they sum up to 1 million. The results are shown in Fig. 1 and corresponding Tab. 1. It is quite clear that as the number of generators are increased up to 4, the sampling keeps getting more realistic. In case when multiple modes are significantly overlapped/clustered, a generator can capture cluster of modes. Therefore, for this real data distribution, 4 generators are enough to capture all the 5 modes. With 5 or more generators, all the modes were still captured, but the two overlapping modes have more than two generation peaks. This is mainly because multiple generators are capturing this region and all the generators (mixture components) were assigned *equal weights* during sampling.

Other works using more than one generators [5, 1] also use the number of generators as a hyper-parameter as knowing a-priori the number of modes in a real-world data (*e.g.* images) in itself is an open problem.

2.2. Non-Parametric Density Estimation

Fig. 2 shows the performance of MA-GAN in 'non-parametric density estimation' experiment described in Section 5.1 of the main paper.

2.3. InfoGAN for Edges-to-Handbags Task

To make our baseline as strong as possible, we did some more experiments with InfoGAN for edges-to-handbags task. The results shown in main paper (Fig. 6) for InfoGAN are obtained by not sharing the discriminator and Q network parameters. Here we did two experiments by sharing all the initial layers of the discriminator and Q network. In the first experiment, the input is the categorical code besides the conditional image. In the second experiment, noise is also added as an input. The architecture details are given in Section 3.3.2. In Fig. 4, we show the results of both these experiments side by side. There are not much perceivable changes as we vary the categorical code values. Generator simply learn to ignore the input noise as was also pointed

Figure 4: InfoGAN for *edges-to-handbags* task by using three categorical code values. In each sub-figure, the first column represents the input, columns $2-4$ represents generations when input is categorical code besides conditioning image, and columns $5-7$ are generations with noise as an additional input. The generations for each of the two architectures are visually the same irrespective of the categorical code value, which clearly indicates that it is not able to capture diverse modes.

Figure 6: Face generations using MAD-GAN. Each sub-figure represents generations by a single generator. The first generator is generating faces with very dark background. The second generator is generating female faces with long hair in very light background, while the third one is generating faces with colored background and casual look (based on direction of viewing and expression) .

by [4].

2.4. Diverse Face Generation

In the main paper (Fig. 9), we showed diverse face generations using MAD-GAN with three generators. To get better understanding about the possible diversities, we show additional generations in Fig. 6.

3. Network Architectures and Parameters

Here we provide all the details about the architectures and the parameters used in various experiments shown in the main paper. For the experiment concerning non-parametric density estimation, the MAD-GAN parameters are randomly initialized using xavier initialization with normal distributed random sampling [3]. For all the other experiments, the initialization done is same as the base architecture used to adapt MAD-GAN.

3.1. Non-Parametric Density Estimation

Architecture Details: The generator has two fully connected hidden layers with 128 neurons (each of which are followed by exponential linear unit) and fully connected outer layer. In case of MAD-GAN and MA-GAN, we used 4 generators with parameters of first two layers shared. Generator generates 1D samples. Input to each generator is a uniform noise $U(-1, 1)$ of 64 dimension. In case of InfoGAN, 5 dimensional categorical code is further con-

DCGAN, Unrolled GAN, InfoGAN, MA-GAN Disc	Mode-Reg DCGAN Disc	Mode-Reg DCGAN Enc	WGAN, GoGAN Disc	BEGAN Enc	BEGAN Dec	MAD-GAN Disc	InfoGAN QNet
Input: 1	1	1	32	1	1	1	1
fc: 128, leaky relu							
fc: 128, leaky relu							
fc: 1	1	64	1	32	1	5 (nGen+1)	5
sigmoid		identity				softmax	

Table 2: Non-Parametric density estimation architecture for discriminators (Disc), encoders (Enc), decoders (Dec), and Q Network (QNet). nGen is number of generators, fc is fully connected layer.

catenated with the uniform noise to form the input. The categorical code is randomly sampled from the multinomial distribution. The discriminator architecture for respective networks is shown in Tab. 2. Mode-Regularized GAN architecture has encoder, BEGAN has encoder and decoder, and InfoGAN has Q Network whose details are also present in Tab. 2.

MAD-GAN has multi-label cross entropy loss. MA-GAN has binary cross entropy loss. For training, we use Adam optimizer with batch size of 128 and learning rate of $1e-4$. In each mini batch, for MAD-GAN we have 128 samples from each of the generators as well as real distribution, while for MA-GAN 128 samples are chosen from real distribution as well as all the generators combined.

Dataset Generation We generated synthetic 1D data using GMM with 5 Gaussians and select their means at $10, 20, 60, 80$ and 110. The standard deviation used is $3, 3, 2, 2$ and 1. The first two modes overlap significantly while the fifth one is peaky and stands isolated.

3.2. Stacked and compositional MNIST Experiments

Architecture details: The architecture for stacked-MNIST is similar to the one used in [6]. Please refer to the Tab. 3 for generator architecture and Tab. 4 for discriminator architecture and Q network architecture of InfoGAN. The architecture for compositional-MNIST experiment is same as DCGAN [8]. Please refer to the Tab. 5 for discriminator architecture and Q network architecture of InfoGAN. In both the experiments, Q network of InfoGAN shares all except the last layer with the discriminator.

Dataset preparation: MNIST database of hand written digits are used for both the tasks.

	number outputs	stride
Input: $z \sim \mathcal{N}(0, I_{256})$		
Fully connected	4 * 4 * 64	
Reshape to image 4,4,64		
Transposed Convolution	32	2
Transposed Convolution	16	2
Transposed Convolution	8	2
Convolution	3	1

Table 3: Generator architecture for 1000 class stacked-MNIST experiment. For MAD-GAN, all the layers except those mentioned in last two rows are shared.

	number outputs	stride
Input: 32x32 Color Image		
Convolution	4	2
Convolution	8	2
Convolution	16	2
Flatten		
Fully Connected	1	

Table 4: Discriminator architecture for 1000 class stacked-MNIST experiment. For MAD-GAN, with k generators, it is adapted to have $k+1$ dimensional last layer output. For InfoGAN, with 156 dimensional salient variables and 100 dimensional incompressible noise, it is adapted to have 156 dimensional output for Q network.

3.3. Image-to-Image Translation

3.3.1 MAD-GAN

Architecture details: The network architecture is adapted from [4] and the experiments were conducted with the U-Net architecture and patch based discriminator.

In more detail, let Ck denote a Convolution-BatchNorm-ReLU layer with k filters and CDk represent a Convolution-

	number outputs	stride
Input: Color Image (64x64)		
Convolution	64	2
Convolution	128	2
Convolution	256	2
Convolution	512	2
Flatten		
Fully Connected	1	

Table 5: Discriminator architecture for 1000 class compositional-MNIST experiment. For MAD-GAN, with k generators, it is adapted to have $k+1$ dimensional last layer output. For InfoGAN, with 156 dimensional salient variables and 100 dimensional incompressible noise, it is adapted to have 156 dimensional output for Q network.

BatchNorm-Dropout-ReLU layer with a dropout rate of 50%. All Convolutions are 4×4 spatial filters with a stride of 2. Convolutions in the encoder, and in the discriminator, downsample by a factor of 2, whereas in the decoder they upsample by a factor of 2.

Generator Architectures We used the U-Net generator based architecture from [4] as follows:

- U-Net Encoder: C64-C128-C256-C512-C512-C512-C512-C512
- U-Net Decoder: CD512-CD1024-CD1024-C1024-C1024-C512-C256-C128. Note that, in case of MAD-GAN, the last layer does not share parameters with other generators.

After the last layer in the decoder, a convolution is applied to map to the number of output channels to 3, followed by a tanh function. BatchNorm is not applied to the first C64 layer in the encoder. All ReLUs in the encoder are leaky, with a slope of 0.2, while ReLUs in the decoder are not leaky. The U-Net architecture has skip-connections between each layer i in the encoder and layer $n-i$ in the decoder, where n is the total number of layers. The skip connections concatenate activations from layer i to layer $n-i$. This changes the number of channels in the decoder.

Discriminator Architectures The patch based 70×70 discriminator architecture was used in this case : C64-C128-C256-C512.

Diversity term After the last layer, a convolution is applied to map the output layer to the dimension of $k+1$ (where k is the number of generators in MAD-GAN) followed by the softmax layer for the normalization.

For the training, we used Adam optimizer with learning rate of $2e-4$ (for both generators and discriminator), $\lambda_{L1} = 10$ (hyperparameter corresponding to the L_1 regularizer) and batch size of 1.

3.3.2 InfoGAN

The network architecture is adapted from [4] and the experiments were conducted with the U-Net architecture and patch based discriminator.

Generator Architectures The U-Net generator is exactly same as in [4] except that the number of input channels are increased from 3 to 4. For the experiment done in Section 2.3, to take noise as input, input channels are increased to 5 (one extra input channel for noise).

Discriminator Architectures The discriminator is exactly same as in [4]: C64-C128-C256-C512

Q network Architectures The Q network architecture is C64-C128-C256-C512-Convolution3-Convolution3. Here first Convolution3 gives a output of 30×30 patches with 3 channels while second Convolution3 just gives 3 dimensional output. All the layers except last two are shared with the discriminator to perform the experiments as mentioned in Section 2.3.

Diversity term To capture three kinds of distinct modes, the categorical code can take three values. Hence, in this case, the categorical code is a 2D matrix in which one third of entries are set to 1 and remaining to 0 for each category. The generator is fed input image along with categorical code appended channel wise to the image. For the experiment done in Section 2.3, to take noise as input, the generator input is further channel wise appended with a 2D matrix of normal noise.

For the training, we used Adam optimizer with learning rate of $2e-4$ (for both generator and discriminator), $\lambda_{L1} = 10$ (hyperparameter corresponding to the L_1 regularizer) and batch size of 1.

Dataset Preparation:

- Edges-to-Handbags: We used $137,000$ Amazon handbag images from [10]. The random split into train and test was kept the same as done by [10].
- Night-to-Day: We used $17,823$ training images extracted from 91 webcams. We thank Jun-Yan Zhu for providing the dataset.

Discriminator D
Input 64x64 Color Image
4x4 conv. 64 leakyRELU. stride 2. batchnorm
4x4 conv. 128 leakyRELU. stride 2. batchnorm
4x4 conv. 256 leakyRELU. stride 2. batchnorm
4x4 conv. 512 leakyRELU. stride 2. batchnorm
4x4 conv. output leakyRELU. stride 1

Table 6: DCGAN Discriminator: It is adapted to have $k+1$ dimensional last layer output for MAD-GAN with k generators. (normalizer is softmax).

3.4. Diverse-Class Data Generation

Architecture details: The network architecture is adapted from DCGAN [8]. Concretely, the discriminator architecture is described in Table 8 and the generator architecture in Table 7. We use three generators without sharing any parameter. The residual layers helped in improving the image quality since the data manifold was much more complicated and the discriminator needed more capacity to accommodate it.

Diversity terms For the training, we used Adam optimizer with the learning rate of $2e-4$ (both generator and discriminator) and batch size of 64.

Dataset preparation: Training data is obtained by combining dataset consisting of various highly diverse images such as *islets*, *icebergs*, *broadleaf-forest*, *bamboo-forest* and *bedroom*, obtained from the Places dataset [9]. To create the training data, images were randomly selected from each of them, creating a dataset consisting of $24,000$ images.

3.5. Diverse Face Generations with DCGAN

Architecture details: The network architecture is adapted from DCGAN [8]. Concretely, the discriminator architecture is described in Table 8 and the generator architecture in Table 7. In this case all the parameters of the generators except the last layer were shared. The residual layers helped in improving the image quality since the data manifold and the manifolds of each of the generators was much more complicated and the discriminator needed more capacity to accommodate it.

Generator G
Input $\in \mathbb{R}^{100}$
4x4 upconv. 512 RELU.batchnorm.shared
4x4 upconv. 256 RELU. stride 2.batchnorm.shared
4x4 upconv. 128 RELU. stride 2.batchnorm.shared
4x4 upconv. 64 RELU. stride 2.batchnorm.shared
4x4 upconv. 3 tanh. stride 2

Table 7: DCGAN Generator: All the layers except the last one are shared among all the three generators.

Residual Discriminator D
Input 64x64 Color Image
7x7 conv. 64 leakyRELU. stride 2. pad 1. batchnorm
3x3 conv. 64 leakyRELU. stride 2. pad 1. batchnorm
3x3 conv. 128 leakyRELU. stride 2.pad 1. batchnorm
3x3 conv. 256 leakyRELU. stride 2. pad 1. batchnorm
3x3 conv. 512 leakyRELU. stride 2. pad 1. batchnorm
3x3 conv. 512 leakyRELU. stride 2. pad 1. batchnorm
3x3 conv. 512 leakyRELU. stride 2. pad 1. batchnorm
RESIDUAL-(N512, K3, S1, P1)
RESIDUAL-(N512, K3, S1, P1)
RESIDUAL-(N512, K3, S1, P1)

Table 8: Diverse-Class Data Generation and Diverse Face Generation: The last layer output is $k+1$ dimensional for MAD-GAN with k generators (normalizer is softmax). RESIDUAL layer is elaborated in Table 9.

RESIDUAL-Residual Layer
Input: previous-layer-output
c1: CONV-(N512, K3, S1, P1), BN, ReLU
c2: CONV-(N512, K3, S2, P1), BN
SUM(c2,previous-layer-output)

Table 9: Residual layer description for Tab. 8.

Diversity terms For the training, we used Adam optimizer with the learning rate of $2e-4$ (both generator and discriminator) and batch size of 64.

Dataset preparation: We used CelebA dataset as mentioned for face generation based experiments. For Image generation all the images $(14,197,122)$ from the Imagenet-1k dataset [2] were used to train the DCGAN with 3 Generators alongside the MAD-GAN objective. The images from both CelebA and Imagenet-1k were resized into 64×64.

3.6. Unsupervised Representation Learning

Architecture details: Our architecture uses the one proposed in DCGAN [8]. Similar to the DCGAN experiment on SVHN dataset $(32 \times 32 \times 3)$ [7], we removed the penulti-

2 Generators	3 Generators	4 Generators
20.5%	18.2%	17.5%

Table 10: The misclassification error of MAD-GAN on SVHN with different number of generators are shown.

mate layer of generator (second last row in Tab. 7) and first layer of discriminator (first convolution layer in Tab. 6).

Classification task: We trained our model on the available SVHN dataset [7]. For feature extraction using discriminator, we followed the same method as mentioned in the DCGAN paper [8]. The features were then used for training a regularized linear L2-SVM. The ablation study is presented in Tab. 10.

Dataset preparation: We used SVHN dataset [7] consisting of $73,257$ digits for the training, $26,032$ digits for the testing, and $53,1131$ extra training samples. As done in DCGAN [8], we used 1000 uniformly class distributed random samples for training, $10,000$ samples from the non-extra set for validation and 1000 samples for testing.

For the training, we used Adam optimizer with learning rate of $2e-4$ (both generator and discriminator), $\lambda = 1e-4$ (competing objective), and batch size of 64.

References

[1] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *International Conference on Machine Learning*, 2017.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Computer Vision and Pattern Recognition*, 2009.

[3] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.

[4] P. Isola, J.-Y. Zhu, T. Zhou, and A. Efros. Image-to-image translation with conditional adversarial networks. In *Computer Vision and Pattern Recognition*, 2017.

[5] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, 2016.

[6] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. In *International Conference on Learning Representations*, 2017.

[7] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[8] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[9] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[10] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, 2016.