

## Appendix

### CIFAR10 Experiments

#### BigGAN + AMAT Experiments

For the baseline we use the author’s official PyTorch implementation<sup>1</sup>. For our experiments on AMAT + BigGAN, we kept the optimizer as Adam [17] and used the hyperparameters  $\beta_1 = 0.0, \beta_2 = 0.9$ . We did not change the model architecture parameters in any way. The best performance was achieved with learning rate = 0.0002 for both the Generator and all the Discriminators. The batch size for the  $G$  and all  $D$  is 50, and the latent dimension is chosen as 128. The initial value of  $T_i = 5$  epochs, and after the first discriminator is added,  $T_i$  is increased by 5 epochs every  $T_i$  epochs. The initial value of  $\alpha_i = 1.5$ , and it is increased by a factor of 3.5 every time a discriminator is added. To check whether to add another discriminator or not, we use 10 exemplar images, 1 from each CIFAR10 class. While assigning datapoints to each discriminator, we use an epsilon greedy approach. We chose  $\varepsilon = 0.25$ , where the datapoint is assigned to a random discriminator with a probability  $\varepsilon$ . The number of discriminator(s) updates per generator update is fixed at 4. We also use exponential moving average for the generator weights with a decay of 0.9999.

#### SN-GAN + AMAT Experiments

We used the SN-GAN implementation from <https://github.com/GongXinyuu/sngan.pytorch>, which is the PyTorch version of the authors’ Chainer implementation [https://github.com/pfnet-research/sngan\\_projection](https://github.com/pfnet-research/sngan_projection). We kept the optimiser as Adam and used the hyperparameters  $\beta_1 = 0.0, \beta_2 = 0.9$ . The batch size for generator is 128 and for the discriminators is 64, and the latent dimension is 128. The initial learning rate is 0.0002 for both generator and the discriminators. The number of discriminator(s) updates per generator update is fixed at 7. The initial value of  $T_i = 2$  epochs, and is increased by 1 epoch after every discriminator is added.  $\alpha_i$  is initialized as 1.5, and is increased by a factor of 1.3 after a discriminator is added, till 20 epochs, after which it is increased by a factor of 3.0. These larger increases in  $\alpha_i$  are required to prevent too many discriminators from being added over all iterations. We chose  $\varepsilon = 0.3$ , where the datapoint is assigned to a random discriminator with a probability  $\varepsilon$ . We use 10 exemplar images, 1 from each CIFAR10 class.

**ResNet GAN:** We use the same ResNet architecture as above, but remove the spectral normalization from the model. The optimizer parameters, learning rate and batch sizes remain the same as well. The number of discriminator(s) updates per generator update is fixed at 5. The initial value of  $T_i = 10$  epochs, and is increased by 5 epochs after every discriminator is added.  $\alpha_i$  is initialized as 1.5, and is increased by a factor of 2.0 after a discriminator is added. We chose  $\varepsilon = 0.2$ , where the datapoint is assigned to a random discriminator with a probability  $\varepsilon$ . We use 10 exemplar images, 1 from each CIFAR10 class.

**ResNet WGAN-GP:** In the above model, the hinge loss is replaced by the Wasserstein loss with gradient penalty. The optimizer parameters, learning rate and batch sizes remain the same as well. The number of discriminator(s) updates per generator update is fixed at 2. The initial value of  $T_i = 5$  epochs, and is increased by 5 epochs after a discriminator is added.  $\alpha_i$  is initialized as 1.5, and is increased by a factor of 3.0 after a discriminator is added. We

<sup>1</sup><https://github.com/ajbrock/BigGAN-PyTorch>

chose  $\varepsilon = 0.2$ , where the datapoint is assigned to a random discriminator with a probability  $\varepsilon$ . We use 10 exemplar images, 1 from each CIFAR10 class. These images are chosen randomly from each class, and may not be the same as the ones for other CIFAR10 experiments.

## DCGAN + AMAT Experiments

We used standard CNN models for our DCGAN as shown in Table 7. We use Adam optimizer with hyperparameters  $\beta_1 = 0.0, \beta_2 = 0.9$ . The learning rate for generator was 0.0002, and the learning rate for the discriminator(s) was 0.0001. The number of discriminators updates per generator was fixed at 1. The initial value of  $T_t = 4$  epochs, and is increased 5 epochs after a discriminator is added.  $\alpha_t$  is initialized as 1.5 and is increased by a factor of 1.5 every time a discriminator is added. We chose  $\varepsilon = 0.3$ , where the datapoint is assigned to a random discriminator with a probability  $\varepsilon$ . We use 10 exemplar images, 1 from each CIFAR10 class.

## Stacked MNIST Experiments

Stacked MNIST provides us a test-bed to measure mode collapse. A three channel image is generated by stacking randomly sampled MNIST classes, thus creating a data distribution if 1000 modes. We use this dataset to show that, when generator oscillates to a different set of modes, catastrophic forgetting is induced in discriminator and this prevents the generator to recover previous modes. To study this phenomenon, we need to measure the correlation between number of modes the generator covered and the catastrophic forgetting in the discriminator. Measuring the number of modes is straight forward, we can by simply classify each channels of the generated images using a MNIST pretrained classifier to find its corresponding mode. However, to measure catastrophic forgetting in the discriminator, we use a proxy setting, where we take the high-level features of the real images from the discriminator and train a simple classifier on top of that. The discriminative quality of the features taken from the discriminator indirectly measure the ability of the network to remember the modes. Finally, as a control experiment we randomize the weights of the discriminator, and train a classifiers on the feature taken from randomized discriminator. This is to show that, extra parameters in the classifier does not interfere our proxy measure for the catastrophic forgetting. Finally, we train a DCGAN with a single discriminator, and a similar DCGAN architecture with our proposed AMAT procedure, and measure the number of modes covered by the generator and the accuracy of the discriminator.

## A Fair comparison on discriminator capacity

Our AMAT approach incrementally adds new discriminators to the GAN frameworks, and its overall capacity increases over time. Therefore, it is not fair to compare a model with AMAT training procedure with its corresponding the single discriminator model. As a fair comparison to our AMAT algorithm, we ran single discriminator model with approximately matching its discriminator capacity to the final AMAT model. For example, SN-GAN with AMAT learning scheme uses 4 discriminators at the end of its training. Therefore we use a discriminator with four times more parameters for the single discriminator SN-GAN model. This is done by increasing the convolutional fillters in the discriminator. Table 6 shows that,

	Scores	DCGAN	ResNetGAN	WGAN-GP	SN-GAN	BigGAN
w/o AMAT	#of Param of D	1.10 M	3.22 M	2.06 M	4.20 M	8.42 M
	IS	$5.97 \pm 0.08$	$6.59 \pm 0.09$	$7.72 \pm 0.06$	$8.24 \pm 0.05$	$9.14 \pm 0.05$
	FID	34.7	36.4	19.1	14.5	10.5
+ AMAT	#of Param of D	1.02 M	$3 \times 1.05$ M	$2 \times 1.05$ M	$4 \times 1.05$ M	8.50 M
	IS	$6.32 \pm 0.06$	$8.1 \pm 0.04$	$7.80 \pm 0.07$	$8.34 \pm 0.04$	$9.51 \pm 0.06$
	FID	30.14	16.35	17.2	13.8	6.11

Table 6: Increasing network capacity alone does not capture more modes. Even after the discriminator capacity is matched, single discriminator GANs do not perform as well as multi-adversarial GANs with AMAT learning

even after matching the network capacity, the single discriminator models do not perform well as compared to our AMAT learning.

## Synthetic Data Experiments

We add flow-based non-linearity (Algorithm 1) to a synthetic 8-Gaussian ring dataset. We chose  $K = 5$  as our non-linearity depth and chose a randomly initiated 5 layer MLP as our non-linear functions. We use an MLP as our GAN generator and discriminator (Table 8). We use the Adam optimizer with hyperparameters  $\beta_1 = 0.0, \beta_2 = 0.9$ . The learning rate for the generator and discriminator was 0.0002. The number of discriminator updates per generator update is fixed to 1, and the batch size is kept 64. The initial value of  $T_i = 5$  epochs, and is increased by 10 every time a discriminator is added.  $\alpha_i$  is initialized as 1.5, and is increased by a factor of 1.5 after a discriminator is added for the first 50 epochs. After that  $\alpha_i$  is increased by a factor of 3. We chose  $\varepsilon = 0.25$ , where the datapoint is assigned to a random discriminator with a probability  $\varepsilon$ . 1 random datapoint from each of the 8 modes is selected as the exemplar image.

Figure 4 shows the difference in performance of a standard MLP GAN (8 and the same MLP GAN with AMAT. The GIF on the lefts shows a cyclic mode collapse due the discriminator suffering from catastrophic forgetting. The same GAN with is able to completely mitigate catastrophic forgetting with just 2 discriminators added during training, on a 728-dimensional synthetic data.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$	$x \in \mathbb{R}^{32 \times 32 \times 3}$
dense $\rightarrow 4 \times 4 \times 512$	$3 \times 3$ , stride=1 conv. 64 lReLU
$4 \times 4$ , stride=2 deconv. BN 256 ReLU	$4 \times 4$ , stride=2 conv. 64 lReLU
$4 \times 4$ , stride=2 deconv. BN 128 ReLU	$3 \times 3$ , stride=1 conv. 128 lReLU
$4 \times 4$ , stride=2 deconv. BN 64 ReLU	$4 \times 4$ , stride=2 conv. 128 lReLU
$3 \times 3$ , stride=1 conv. 3 Tanh	$3 \times 3$ , stride=1 conv. 256 lReLU
Generator	$4 \times 4$ , stride=2 conv. 256 lReLU
	$3 \times 3$ , stride=1 conv. 512 lReLU
	dense $\rightarrow 1$
	Discriminator

Table 7: DCGAN Architecture for CIFAR10

$z \in \mathbb{R}^{25} \sim \mathcal{N}(0, I)$	$x \in \mathbb{R}^2$
dense $\rightarrow 128$ , BN 128 ReLU	dense $\rightarrow 128$ ReLU
dense $\rightarrow 128$ , BN 128 ReLU	dense $\rightarrow 512$ ReLU
dense $\rightarrow 512$ , BN 512 ReLU	dense $\rightarrow 1$ Sigmoid
dense $\rightarrow 1024$ , BN 1024 ReLU	Discriminator
dense $\rightarrow 2$ , Tanh	
Generator	

Table 8: MLP architecture for Synthetic Dataset

Figure 4: **GAN training visualization:** (Figure contains animated graphics, better viewed in Adobe Acrobat Reader) Training trajectories of an MLP in table 8 (leftmost panel) and an MLP trained with our AMAT procedure (Algorithm 3) (rest three panels) on a 784-dimensional synthetic dataset. Green dots represent real samples and the blue dots represent the generated samples. The vanilla GAN samples are overlayed against discriminator’s output heatmap where the warm yellow color indicates a high probability of being real and cold violet indicates fake. In the AMAT + GAN panels, the discriminator landscapes are shown separately for both discriminators with the second discriminator being spawned at iteration 4000 (Algorithm 2). The 2D visualizations of the 784D data space is facilitated by our synthetic data generation procedure (Algorithm 1).