## A. Comparison of Terminologies between Game Theory and GAN

Table 1. Comparison of Terminologies between Game Theory and GAN

| Game Theory terminology | GAN terminology |
|---|---|
| Player | Generator/ discriminator |
| Strategy | The parameter setting of generator/ discriminator, e.g., $\pi_g$ and $\pi_d$ |
| Policy | The sequence of parameters (strategies) till epoch $t$, e.g., $(\pi_g^1, \pi_g^2, ..., \pi_g^t)$<br>Note: Not used in DO-GAN. |
| Game | The minmax game between generator and discriminator |
| Meta-game/ meta-matrix | The minmax game between generator & discriminator with<br>their respective set of strategies at epoch $t$ of DO framework |
| Meta-strategy | The mixed NE strategy of generator/discriminator at epoch $t$ |

## B. Full Algorithm of DO-GAN

**Algorithm 1:** `GeneratorOracle(`$\sigma_d^{t*}, \mathcal{D}$`)`

Initialize a generator $G$ with random parameter setting $\pi_g'$;
**for** *iteration* $k_0 \dots k_n$ **do**
  Sample noise $\mathbf{z}$;
  $\pi_d =$ Sample a discriminator from $\mathcal{D}$ with $\sigma_d^{t*}$;
  Initialize a discriminator $D$ with parameter setting $\pi_d$;
  Update the generator $G$'s parameters $\pi_g'$ via Adam optimizer:

$$\bigtriangledown_{\pi_g'} \log\left(1 - D(G(\mathbf{z}))\right)$$

**Algorithm 2:** `DiscriminatorOracle(`$\sigma_g^{t*}, \mathcal{G}$`)`

Initialize a discriminator $D$ with random parameter setting $\pi_d'$;
**for** *iteration* $k_0 \dots k_n$ **do**
  Sample a minibatch of data $\mathbf{x}$;
  **for** *a minibatch* **do**
    Sample noise $\mathbf{z}$;
    $\pi_g =$ Sample a generator from $\mathcal{G}$ with $\sigma_g^{t*}$;
    Initialize a generator $G$ with a parameter setting $\pi_g$;
    Generate and add to mixture $G(\mathbf{z})$;
  Update the discriminator $D$'s parameters $\pi_d'$ via Adam optimizer:

$$\bigtriangledown_{\pi_d'} \log D(\mathbf{x}) + \log\left(1 - D(G(\mathbf{z}))\right)$$

We train the oracles for some iterations which we denote as $k_{0,1,2,...}$. For experiments, we train each oracle for an epoch for the real-world datasets and 50 iterations for the 2D Synthetic Gaussian Dataset. At each iteration $t$, we sample the generators from the support set $\mathcal{G}$ with the meta-strategy $\sigma_g^{t*}$ to generate the images for evaluation. Similarly, we conduct the performance

evaluation with the generators sampled from $\mathcal{G}$ with the final $\sigma_g^*$ at termination. SGAN consists of a top-down stack of GANs, e.g, for a stack of 2, Generator 1 is the first layer stacked on Generator 0 with each of them connected to Discriminator 1 and 0 respectively. Hence, in DO-SGAN, we store the meta-strategies for the Generator 0 and 1 in $\sigma_g^{t*}$ and the Discriminator 1 and 0 for $\sigma_d^{t*}$. In `GeneratorOracle`(), we first sample Discriminator 1 and 0 from discriminator distribution $\sigma_d^{t*}$ and train Generator 1 first then followed by calculating loss with Discriminator 1 and train Generator 0 subsequently, and finally calculate final loss with Discriminator 0 and train the whole model end to end. We perform the same process for `DisciminatorOracle`().

## C. Implementation Details

Table 2. Training Hyperparameters

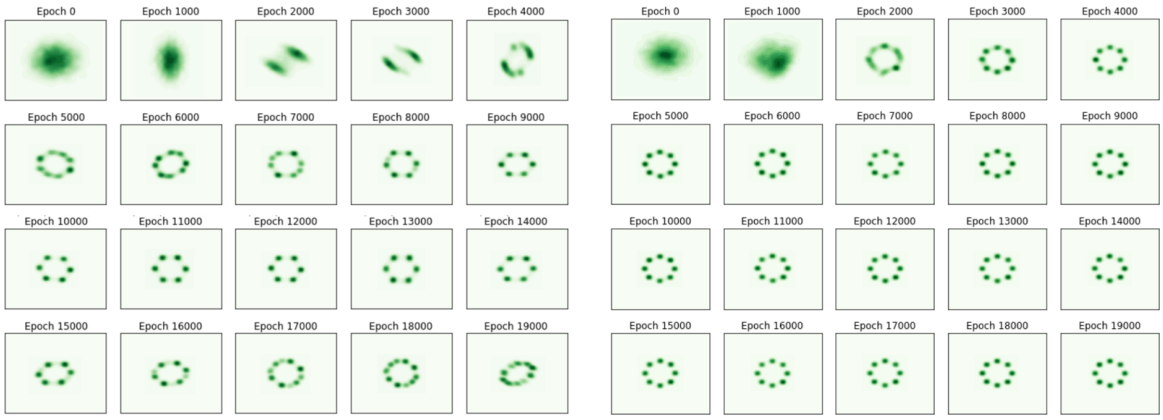
| | GAN | DCGAN | SNGAN | SGAN |
|---|---|---|---|---|
| Generator Learning Rate | 0.0002 | 0.0002 | 0.0002 | 0.0001 |
| Discriminator Learning Rate | 0.0002 | 0.0002 | 0.0002 | 0.0001 |
| batch size | 64 | 64 | 64 | 100 |
| Adam: beta 1 | 0.5 | 0.5 | 0.5 | 0.5 |
| Adam: beta 2 | 0.999 | 0.999 | 0.999 | 0.999 |

We implement our proposed method with Python 3.7, Pytorch=1.4.0 and Torchvision=0.5.0. We set the hyperparameters as the original implementations. We present the hyperparameters set in Table 2. We use Nashpy to compute the equilibria of the meta-matrix game.

### C.1. Value of $\lambda$

The experiment in [17] has done ablation studies for FFHQ dataset which are emoji faces and hence we used the $\lambda$ value for CelebA. Meanwhile, we adopted the results from [34] and set 1000 for MNIST and the maximum value of ablation study for SVHN dataset to train CIFAR-10 as we want to use $\lambda$ values from the most similar datasets. The experiments in [34] reported that they observed little difference in visual quality regarding with ablation study but high values of $\lambda$ cause no loss in visual fidelity when beginning training on a new task rather than lower value of $\lambda$. Hence, we use the maximum value.

## D. Full Training Process of 2D Gaussian Dataset




(a) GAN (b) DO-GAN/P

Figure 1. Full comparison of GAN and DO-GAN/P on 2D Synthetic Gaussian Dataset

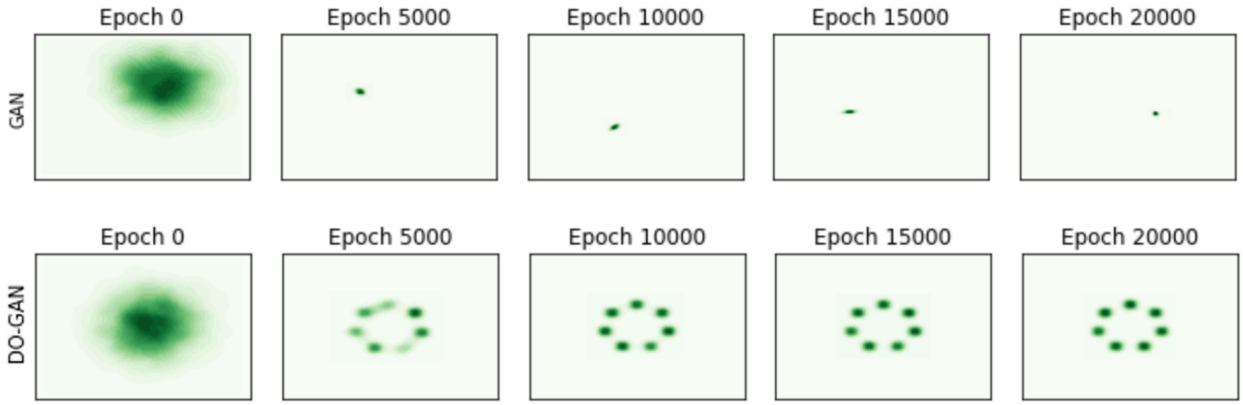



Figure 2. GAN and DO-GAN/P comparison with Gaussian Mixture 7 modes

Figure 1 shows the full training process of DO-GAN/P and GAN on 2D Synthetic Gaussian Dataset. From the results, we find that GAN struggles to generate the samples into 8 modes while DO-GAN/P can generate all the 8 modes of the distribution. Furthermore, DO-GAN/P takes shorter time (less than 5000 iterations) to identify all 8 modes of the data distribution. Moreover, we present the experiment results on 7 mode and 9-mode Gaussian Mixtures in Figure 2 and 3.

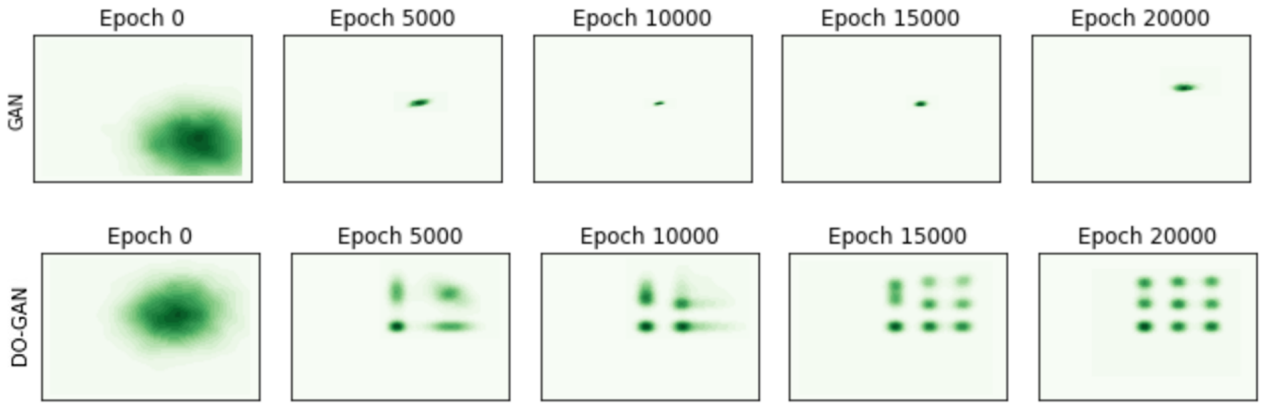



Figure 3. GAN and DO-GAN/P comparison with Gaussian Mixture 9 modes

## E. Investigation of Support Set Size for DO-GAN/P

We vary the support set size $s$ to $5, 10, 15$ and record the training evolution and the running time as presented in Table 3 and Figure 4. We find that if the support size is too small, e.g., $s = 5$, the best responses which are not optimal yet have better utilities than the models in the support set are added and pruned from the meta-matrix repeatedly making the training not able to converge. However, $s = 15$ takes a significantly longer time as the time for the augmenting of meta-matrix becomes exponentially long with the support set size. Hence, we chose $s = 10$ as our experiment support set size since we observed that there is no significant trade-off and shorter runtime.

Table 3. Runtime of DO-GAN/P on 2D Gaussian Dataset with $s = 5, 10, 15$

| Support Set Size | Runtime (GPU hours) |
|---|---|
| $s = 5$ | $> 1$ |
| $s = 10$ | 0.5627 |
| $s = 15$ | 0.9989 |

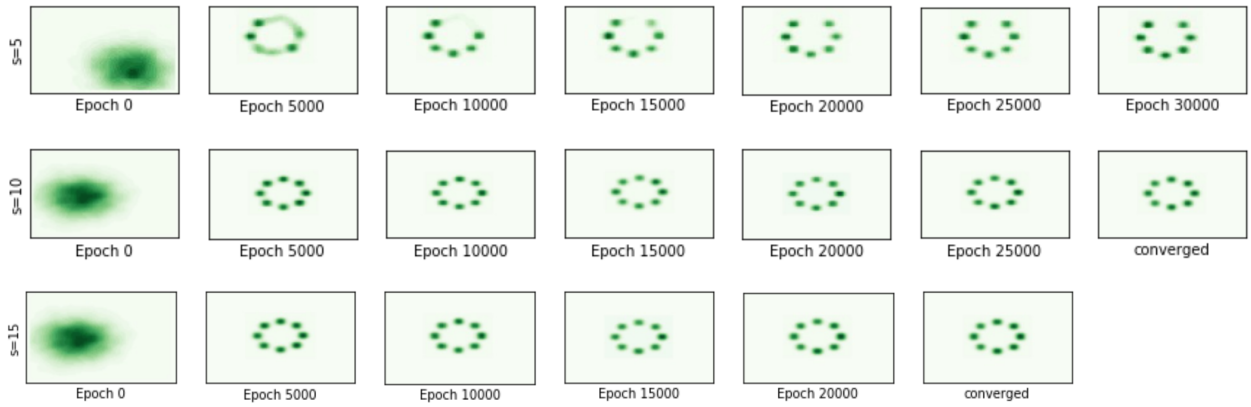



Figure 4. Training evolution on 2D Gaussian Dataset with $s = 5, 10, 15$

## F. Generated images of CelebA and CIFAR-10

In this section, we present the training images of CelebA and CIFAR-10 datasets. We do not evaluate the performance of vanilla GAN and its DO variant on CelebA dataset since DCGAN and SGAN outperform vanilla GAN in image generation tasks [28].

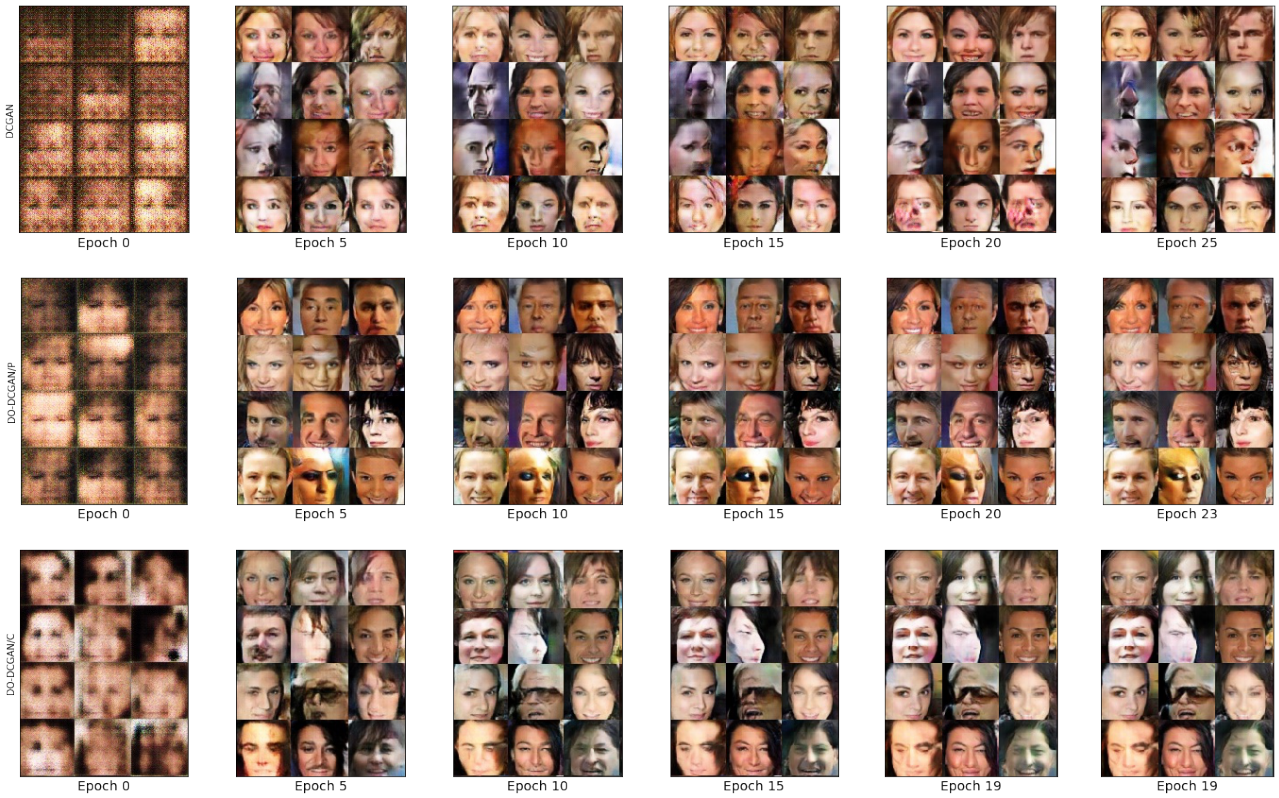



Figure 5. Training images with fixed noise for DCGAN and DO-DCGAN until termination.

Figure 5 shows the training samples of DCGAN, DO-DCGAN/P and DO-DCGAN/C through the training process. Figure 6 also shows those of SNGAN which is trained for 40 epochs with termination $\epsilon$ of $5 \times 10^{-5}$ for DO-SNGAN/P and DO-SNGAN/C. Figure 7 shows those of SGAN which is trained until 20 epochs as well as DO-SGAN/P and DO-SGAN/C with the same termination settings. The results show that DCGAN suffers from mode-collapse, generating similar face while DO-DCGAN/P can generate more plausible and varying faces. We also present the generated images of DCGAN, DO-DCGAN/P, DO-DCGAN/C, SNGAN, DO-SNGAN/P, DO-SNGAN/C, SGAN, DO-SGAN/P and DO-SGAN/C of CIFAR-10 dataset showing that the variants of DO-DCGAN, DO-SNGAN and DO-SGAN can generate better and more identifiable images than DCGAN, SNGAN and SGAN respectively. We present more of the generated samples from SNGAN, DO-SNGAN/P, DO-SNGAN/C, SGAN, DO-SGAN/P, DO-SGAN/C on CelebA dataset in Figure 9.

## G. FID score against iterations

To compute FID score, we use Inception_v3 model with max pool of 192 dimensions and the last layer as coding layer as mentioned in [9]. We resized MNIST, CIFAR-10 generated and test images to $32 \times 32$ and CelebA images to $64 \times 64$. The FID score against training epochs for CIFAR-10 dataset is as follows:

Figure 10 presents the FID score against each epoch of training for SGAN and DO-SGAN/P on CIFAR-10. While both perform relatively well in generating plausible images, we can see that DO-SGAN/P terminates early at epoch 288 and has a better FID score of 16.56 compared to 24.83 at 300 epoch until 21.284 at 500 epoch for the training of SGAN.

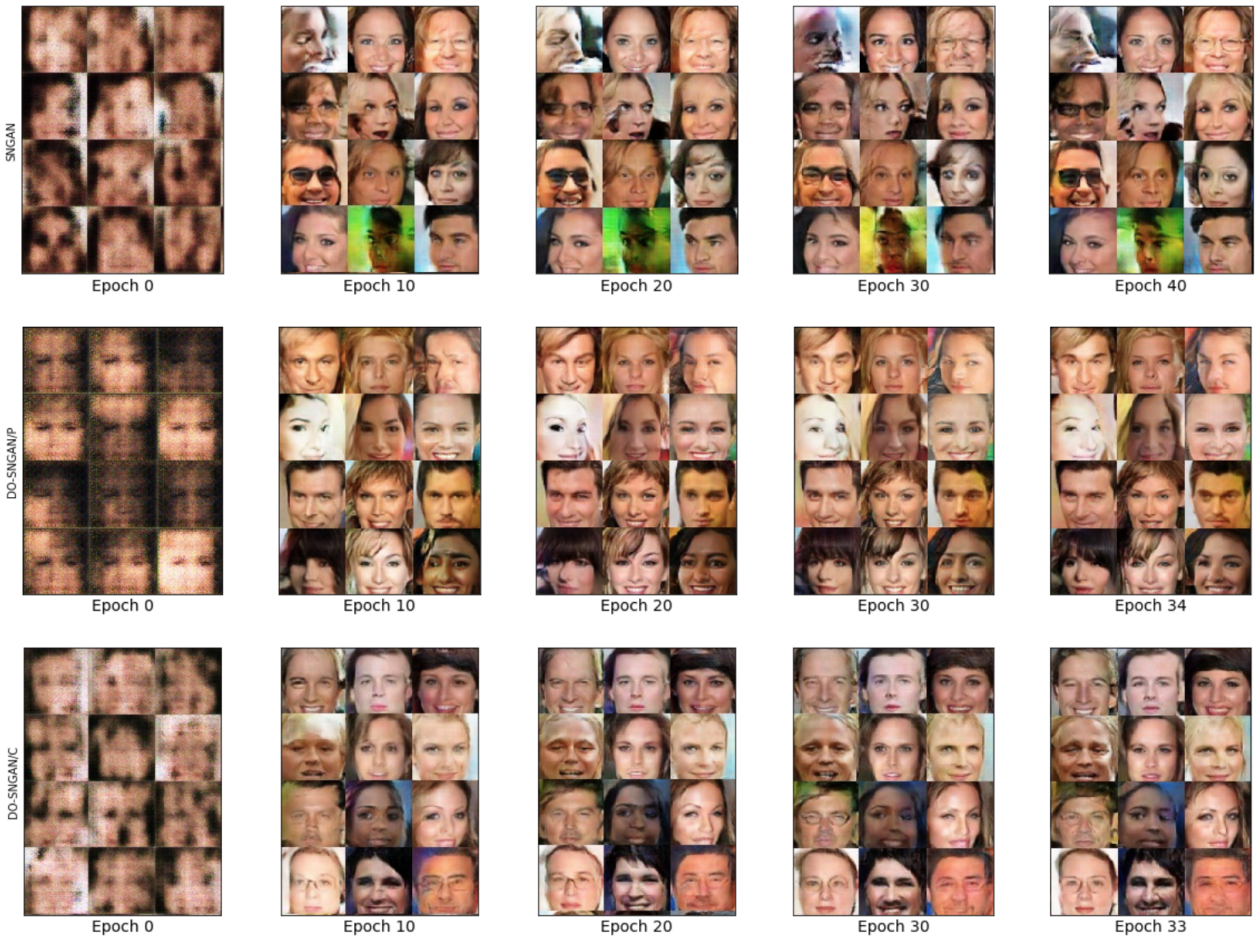



Figure 6. Training images with fixed noise for SNGAN and DO-SNGAN until termination.

## H. Choice of GAN Architectures for Experiments

We carried out experiments with the variants of GANs to evaluate the performance of our DO-GAN framework. We refer to the taxonomy of GANs [37] and choose each architecture from the groups of GANs focused on Network Architecture, Latent Space and Loss: DCGAN, SNGAN and SGAN as shown in Figure 11. We have also included comparisons with mixture architectures such as MIXGAN and MGAN.

## I. Example of Meta-matrix of DO-SGAN/P on CIFAR-10

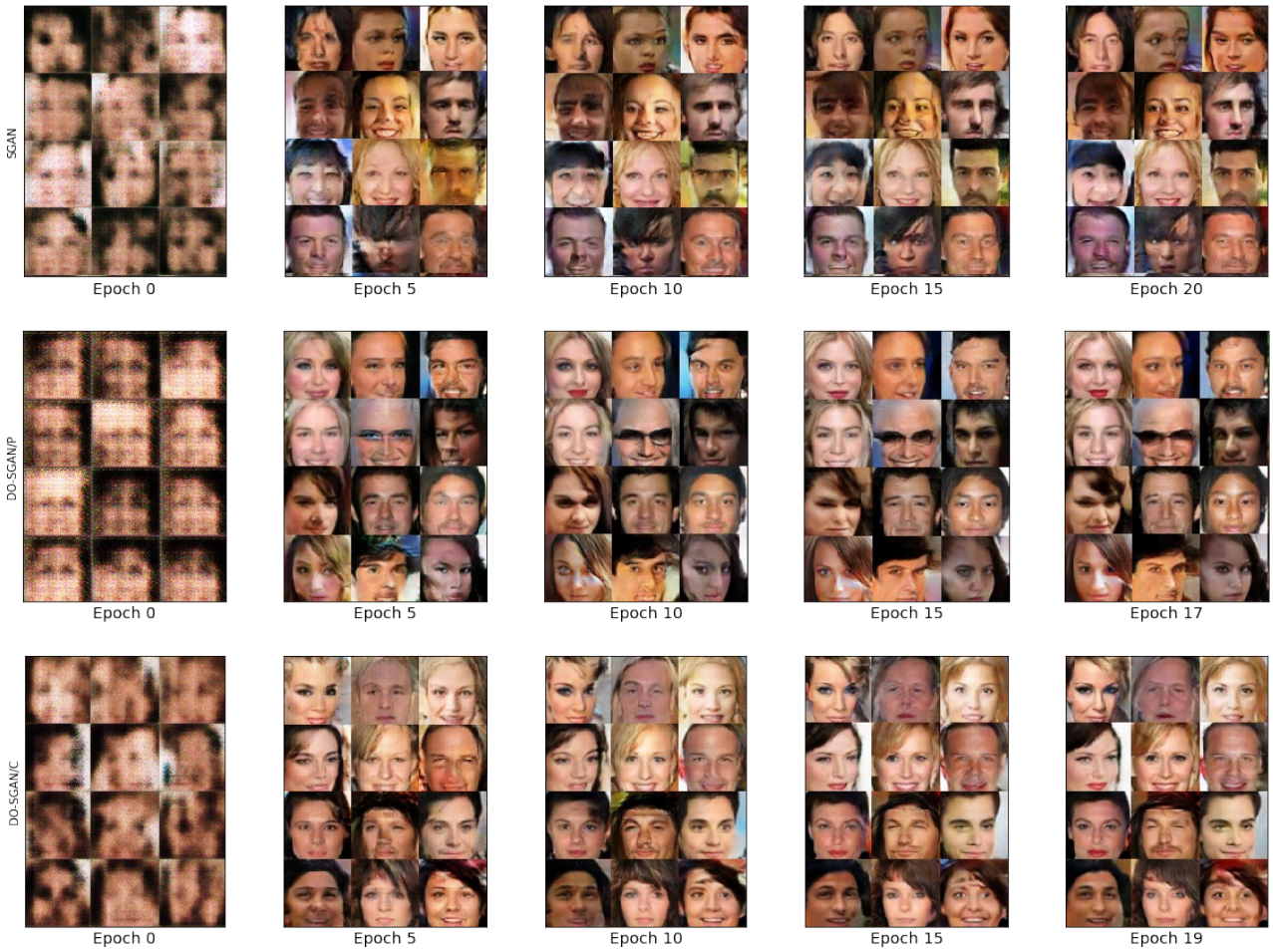
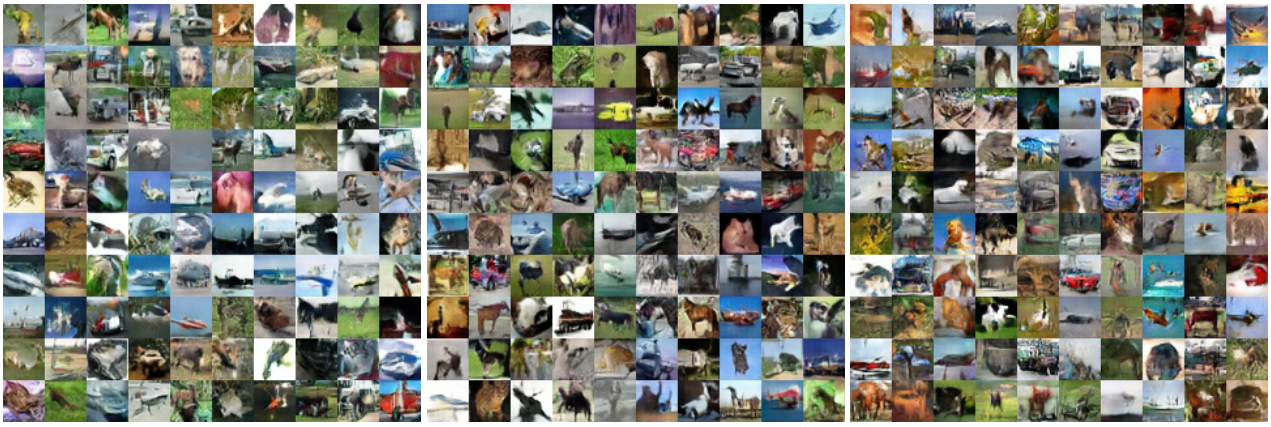
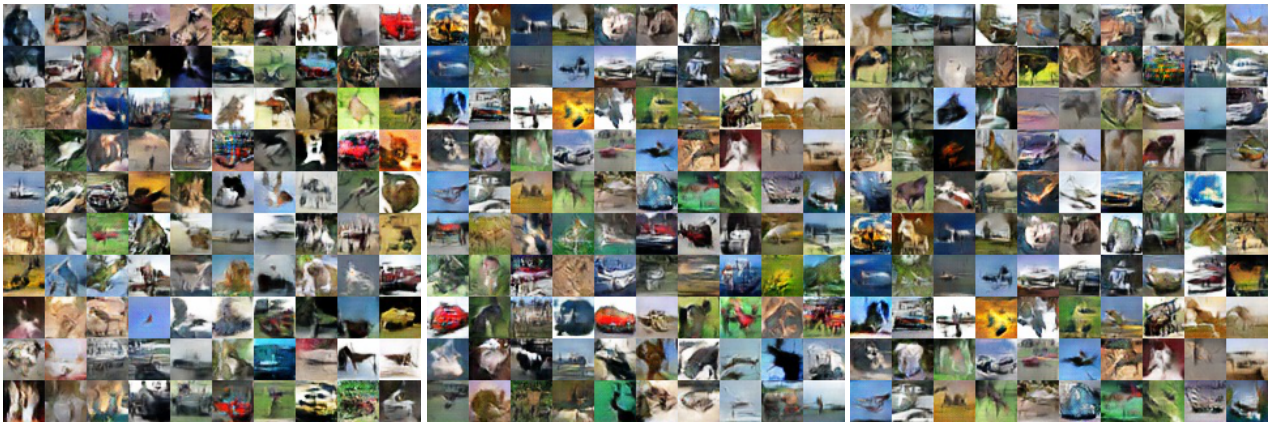



Figure 7. Training images with fixed noise for SGAN and DO-SGAN until termination.

(a) DCGAN (b) SNGAN (c) SGAN

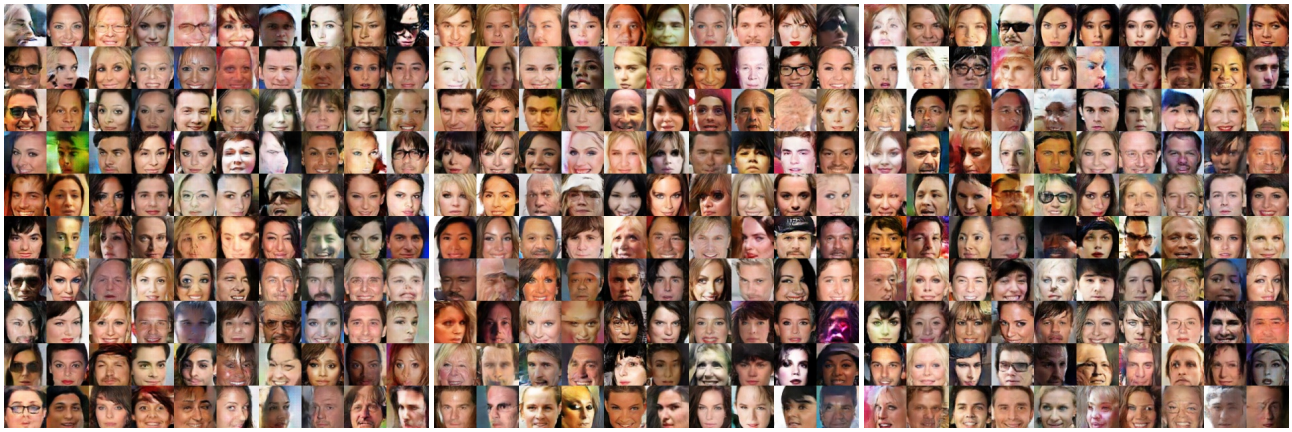


(d) DO-DCGAN/P (e) DO-SNGAN/P (f) DO-SGAN/P



(g) DO-DCGAN/C (h) DO-SNGAN/C (i) DO-SGAN/C

Figure 8. Generated images of CIFAR-10 dataset



(a) SNGAN (b) DO-SNGAN/P (c) DO-SNGAN/C

Figure 9. Generated images of CelebA dataset for DO-SNGAN/P, DO-SNGAN/C and SNGAN

(d) SGAN (e) DO-SGAN/P (f) DO-SGAN/P

Figure 9. Generated images of CelebA dataset for DO-SGAN and SGAN

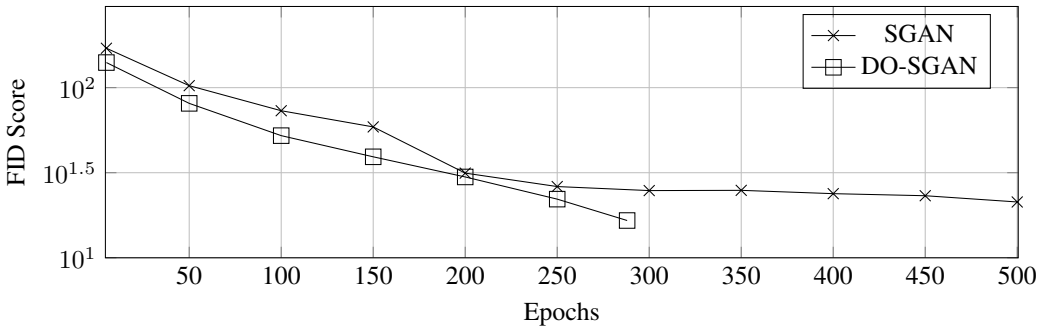



Figure 10. FID score vs. Epochs for SGAN and DO-SGAN trained on CIFAR-10

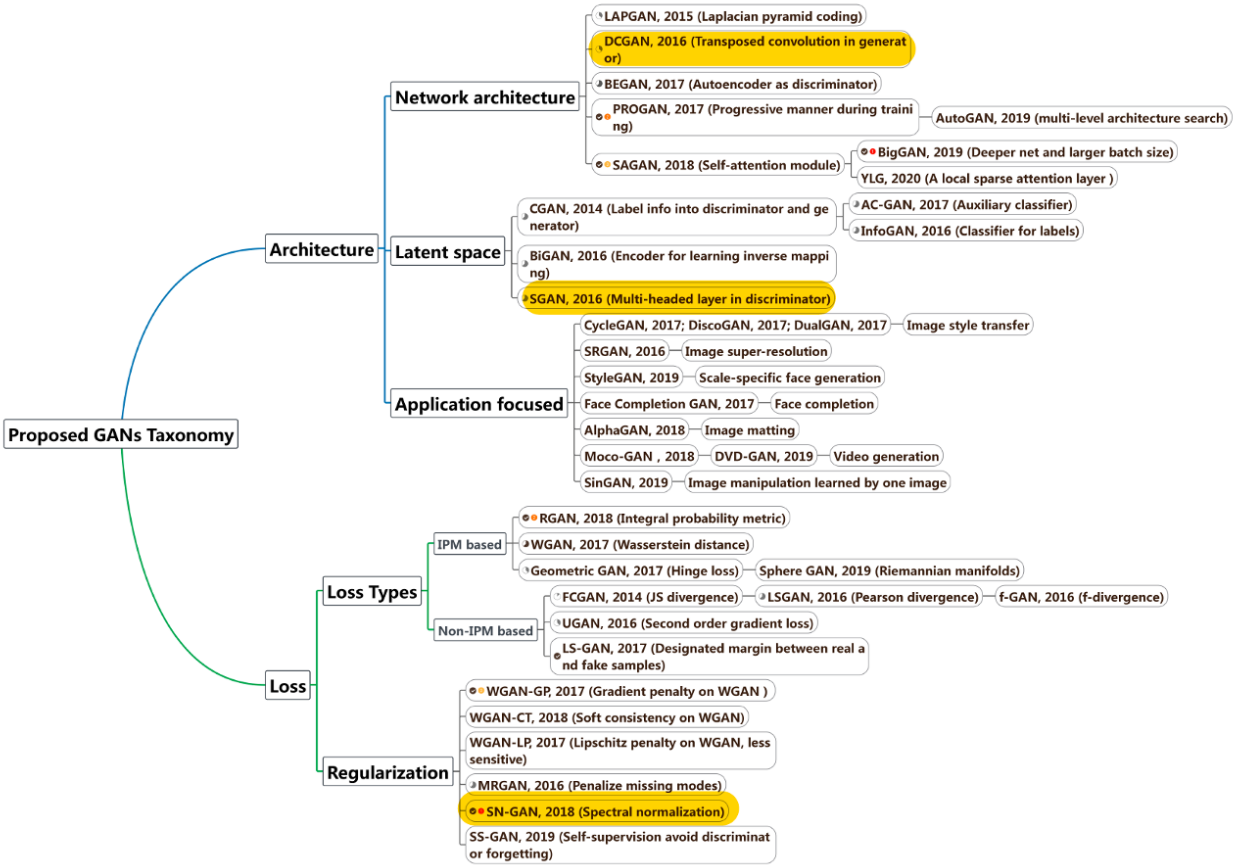
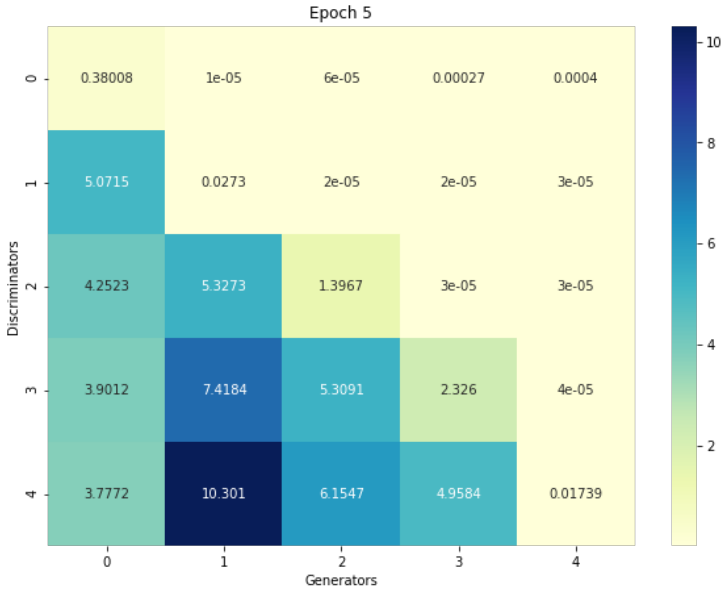
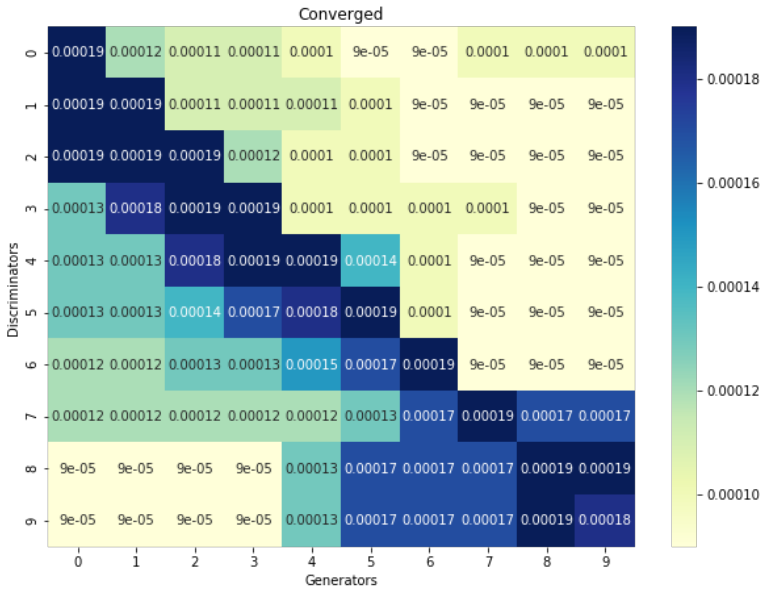



Figure 11. Taxonomy of GAN Architectures from [37]




(a) Meta-matrix at epoch $t = 5$
**Meta-Strategies:** $\sigma_g^{5*} = [0, 0, 0, 0, 1]$, $\sigma_d^{5*} = [0, 0, 0, 0, 1]$
**Expected Payoff:** $U^5(\sigma_g^{5*}, \sigma_d^{5*}) = 0.017$




(b) Meta-matrix at convergence
**Meta-Strategies:**
$\sigma_g^{288*} = [0.015196, 0.025942, 0.141215, 0.000000, 0.0989, 0.163399, 0.000000, 0.535331, 0.000000, 0.020017]$
$\sigma_d^{288*} = [0.320082, 0.004844, 0.000000, 0.111335, 0.141493, 0.071708, 0.046, 0.000000, 0.000000, 0.304538]$
**Expected Payoff:** $U^{288}(\sigma_g^{288*}, \sigma_d^{288*}) = -0.000133$