Sequential training of GANs against GAN-classifiers reveals correlated "knowledge gaps" present among independently trained GAN instances

Supplementary Material

1. Experiment details

1.1. Data

We use MNIST or FFHQ (depending on the DCGAN or StyleGAN2 setting) as the dataset of natural images in our experiments. They both consist of 70,000 images. We use 80% of the data for training, and 20% for evaluation, i.e. we use a fixed sample of 56,000 images for training in all the experiments, and use the rest (14,000) for the evaluation of classifiers. Note that this means that we use the same data of natural images for training both GAN and GAN classifiers, across all iterations. MNIST images are used as 28×28 grayscale images, and FFHQ images are used as 256×256 RGB images in all our experiments, for both training and evaluation.

1.2. GAN Training

In the DCGAN setting, we trained a simplistic DCGAN architecture well suited for the MNIST generation task (unconditional generation of all digits). Specifically, the generator network is modeled as follows: the random noise variable of 100 dimensions is passed through a fully connected layer of 12544 units, followed by 3 transposed convolution layers of 128, 64 and 32 units, each with a kernel size of 5, before the final transposed convolution unit for image output. All hidden layers use batchnorm and the LeakyReLU activation.

For the second setting, we have used the unmodified StyleGAN2 as the GAN architecture of choice. StyleGAN2 is generally considered a SOTA GAN model, capable of generating high-resolution, diverse, photo-realistic images, especially of human faces. There are several components and techniques used in its training framework that cause the generated images to be of high quality and of greater diversity. We specifically emphasize the additional inputs to the generator network: a latent code being output by a non-linear mapping network, and the random noise inputs, both are fed to the individual layers of the generator network. These techniques help the outputs capture the stochasticity and variance present in the real world.

We use proprietary implementation of StyleGAN2, which replicates the TensorFlow [1] implementation available online (https://github.com/NVlabs/stylegan2). We did not tweak any training parameters. For training a single GAN instance, we use 8 NVIDIA Tesla P100 GPUs and each GAN instance roughly required 1 week to train.

For the modified loss functions, we have used $\phi = 0.001$ in all experiments in the StyleGAN2 setting.

We have not used any pre-training for the GANs in the main paper, we discuss this in Sec. 3.

1.3. Classifier Training

In the DCGAN setting, we use a basic CNN classifier that performed well for this task. The classifier includes two convolutional layers of 32 and 64 units, each with a kernel size of 3. Both layers use ReLU activation and followed by a max pooling of 2 in both dimensions. We train directly on the grayscale images without any compression.

In the StyleGAN2 setting, we have used ResNet-50 (version 1.5), Inception-v3 and MobileNetV2 architectures when training the classifiers. ResNet-50 is a high-performing CNN architecture, particularly for image classification, and has been shown to be effective for our task in previous research [2, 7]. One of the distinguishing features of the ResNet-50 architecture is the use of residual connections, that generally enables efficient learning by sharing information between the hidden layers of a deep network. The two other CNN architectures in our study, Inception-v3 ([6]) and MobileNetV2 ([5]), are chosen for their differences to ResNet-50. Like ResNet-50, Inception-v3 is also a large CNN architecture but it does not include residual connections and uses a different "module" that is repeated across the layers. We include MobileNetV2 as a relatively lighter capacity architecture, when compared to the other two architectures.

For the StyleGAN2 classifiers, we use the publicly available implementations as part of the TensorFlow library. When training the classifiers, we pass both the natural and GAN-generated images through JPEG encoding. For each classifier, we train using a single Tesla P100 GPU, and the models roughly require 1 day to train. We do not use any pre-training for the classifiers (unlike [7]), to avoid any external influence in our experiments. We let the classifiers train till they reach convergence, and did not need to finetune the parameters for better performance.

We always train the classifiers on a balanced sample of

natural and generated images. Therefore, when we train a classifier using a sample of 15 GAN instances, we train with 15×56000 generated and an equal number (15×56000) of natural images, and use 0.5 as the classification decision threshold. For this, the natural images are simply repeated 15 times to obtain a balanced training dataset. When evaluating the fooling ability of GAN generators, the held-out test classifiers are trained using 10 held-out GAN instances.

2. Training a StyleGAN2 classifier in the presence of the "truncation trick"

The "truncation trick" is often used with StyleGAN2 (followed from the StyleGAN model) to avoid generating unrealistic images. The approach shrinks the distribution, in order to remove the regions of low density that might be poorly represented by the GAN model. The expression used to shrink the latent distribution is:

$$\mathbf{w}' = \bar{\mathbf{w}} + \psi(\mathbf{w} - \bar{\mathbf{w}}) \tag{1}$$

, where $\bar{\mathbf{w}}$ is the the expected value of the mapped latent space. Here, ψ is the coefficient of truncation: $\psi = 1$ implies an absence of truncation and $\psi = 0$ would correspond to using the (fixed) expected value of the mapped latent space as the latent input for sample generation. Typically [3, 4], $\psi = 0.5$ is effective in practice.

The truncation trick is used with StyleGAN2 if sampling realistic images, when trained with the FFHQ dataset. We note that we also used this trick when we visually compared the image quality. However, the use of this trick effectively shrinks the diversity and brings the samples closer to the "average face" that is learned by the model. And therefore, we don't use the truncation trick in our experiments, since our study is directly measuring the extent of diversity present in the GAN models.

In our experiments, we have also identified that the FID is negatively affected when employing the truncation trick. The FID (lower is better) for a sample of generated images without truncation is ~ 37, whereas with truncation ($\psi = 0.5$) is ~ 81.

Since the diversity is significantly reduced if sampling images with truncation, we have also identified that we do not require multiple generators when training a classifier to be able to achieve generalization: i.e., training a classifier using truncated samples from just one SG2 generator instance suffices to detect truncated samples from an independent SG2 generator instance.

Moreover, we note that by training a classifier which can detect the full range of GAN-generated samples, we also achieve a perfect accuracy when detecting samples generated with the truncation trick.

3. Finetuning StyleGAN2 to transform to the next iteration

The experiments mentioned in the paper train GANs from scratch using the modified loss, with random weight initialization. However, we acknowledge that training GANs is an expensive process, where modern GAN models like StyleGAN2 require weeks to train using multiple accelerators. To this point, we have observed that we can "transform" a GAN to the succeeding iteration by finetuning a pre-trained GAN, and including a pre-trained detector in the finetuning steps. For instance, using the same modification to the generator loss, finetuning an $i = 0$ iteration GAN results in a model that exhibits the same artifacts as what's present in an $i = 1$ iteration GAN generator trained from scratch. We arrive at this finding because they are both detected by a held-out $i = 1$ iteration detector and they both fool a held-out $i = 0$ iteration detector.

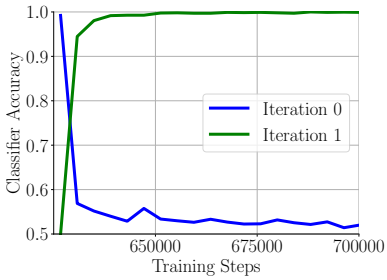

Figure 1. **Finetuning GAN of iteration $i = 0$.** We finetune a fully trained GAN of iteration 0, using the modified loss in the finetuning steps. As depicted, a held-out classifier of iteration $i = 0$ quickly gets fooled, and a held-out classifier of iteration $i = 1$ starts to detect the artifacts that were not present before.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org. 1

[2] Diego Gragnaniello, Davide Cozzolino, Francesco Marra, Giovanni Poggi, and Luisa Verdoliva. Are GAN generated images easy to detect? A critical analysis of the state-of-the-art. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021. 1

[3] 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

[4] 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

[5] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520. IEEE, 2018. 1

[6] C Szegedy, V Vanhoucke, S Ioffe, J Shlens, and ZB Wojna. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, volume 2016, pages 2818–2826. IEEE, 2016. 1

[7] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A. Efros. CNN-Generated Images Are Surprisingly Easy to Spot... for Now. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1