CoachGAN

Mike Brodie[1], Brian Rasmussen[1], Chris Tensmeyer[2], Scott Corbitt[1], and Tony Martinez[1]

[1]Brigham Young University [2]Adobe Research
{mikebrodie, rasmussen, corbitt}@byu.edu tensmeye@adobe.com martinez@cs.byu.edu

Abstract

CoachGAN provides an inference time method to improve outputs from GAN generator models. Similar to creating adversarial examples to fool neural network classifiers, CoachGAN exploits gradient information, in this case from a pretrained discriminator model. Unlike the generating adversarial examples, which uses gradient descent to alter outputs directly, CoachGAN alters the inputs of generator models. This allows for output enhancements at test time without additional model training. CoachGAN adapts easily to existing algorithms and is architecture agnostic. In addition to qualitative samples, we quantitatively show that CoachGAN improves IS and FID scores on a variety of GAN architectures and tasks.

Figure 1. Applying CoachGAN to images generated by a GAN trained on the CelebA dataset. Each row shows the transition from original image (left) to final output (right).

1. Introduction

Generative Adversarial Networks (GANs) consistently produce impressive results for a variety of tasks[].The traditional GAN setup involves generator and discriminator models in a mini-max training scenario trying to optimize opposing loss functions. It is common to discard the discriminator after training and use only the generator model to produce novel synthetic outputs. We introduce an efficient post-training algorithm, CoachGAN, that exploits information in the discriminator at inference time to generate more realistic outputs.

At inference time, CoachGAN depends on a well-trained discriminator model that can accurately classify images as real or fake. This provides the generator with otherwise unavailable feedback on output quality at inference time. Rather than update the weights of the generator, which might reduce future generation quality, CoachGAN alters the input to improve realism. Metaphorically, CoachGAN takes on the role of an advisor that provides feedback to the generator at inference time.

Previous work [37, 26] explore input-centric methods to generate adversarial examples to improve the robustness of an auxiliary neural network classifier. CoachGAN instead uses information synthesized by a discriminator model to refine generator inputs to be more realistic. Unlike previous methods, CoachGAN does not require training additional models [37] or access to real training data during inference [37, 26]. Figure 1 demonstrates the gradual output refinement of CoachGAN using a DCGAN [24] generator and discriminator trained on the CelebA dataset.

CoachGAN provides an efficient inference time method that requires no modification of the generator and discriminator architectures. In addition, CoachGAN easily adapts to any GAN architecture with differentiable models and loss functions. We demonstrate this in our experiments with several unique GAN architectures and a variety of datasets. This work provides the following contributions:

- The CoachGAN algorithm, which uses a pretrained discriminator to improve generator outputs at inference time.
- A wide variety of empirical results that demonstrate the effectiveness and adaptability of CoachGAN.

- A quantitative comparison of CoachGAN and non-CoachGAN outputs using Inception Score and Fréchet Inception Distance.

In Section 2 we briefly review relevant work. In Section 3, we introduce and discuss the CoachGAN algorithm. Section 4 outlines experiments and discusses qualitative and quantitative results. Section 5 summarizes this work and examines implications and paths for future work.

2. Related work

To the best of our knowledge, no previous work exploits information in the discriminator at test time for the sake of improving generator output. However, previous work has explored generator input adjustments to favor a more natural look for adversarial examples, which are then used to increase the overall robustness of a classifier model.

Early adversarial methods ([16, 19, 23]) use backpropagation to add gradient-based noise and create 'adversarial' examples that convincingly fool neural network classifiers. Such approaches employ algorithms such as the Fast Gradient Sign Method [10] to exploit the linear behavior of neural networks when dealing with high dimensional inputs [20]. Similarly, [21] synthesizes the preferred inputs of a classification model for each class by performing activation maximization on the output neurons. While these various methods can effectively fool neural network classifiers, generated images often contain unrealistic curves, distortions, and color blending.

Several works map an adversarially altered training instance, $\hat{x}$, to a latent space vector, z^*, such that $\hat{x} \approx G(z^*)$. For instance, [37] uses an inverter network, I_γ, to map $\hat{x}$ to z^*. Alternatively, z^* can be found by optimization to minimize differenes between $G(z^*)$ and $\hat{x}$ [26]. By projecting adversarial images onto the range of G, these methods can remove unrealistic blurs and artifacts to produce more natural-looking images. While these approaches produce more natural-looking adversaries, [37] requires training an additional inversion model, and [26] samples several z and attempts to minimize a non-convex optimization task.

Several works introduce image editing tools that manipulate low-level latent spaces that approximate the natural image manifold. For example, users suggest facial feature changes [6] or color and structure edits [38] in pixel space and a model performs the gradient updates in the latent space. While these methods generally result in more coherent and visually pleasing images, they require training models that predict the latent vector, z^*, that most closely matches a user's edits. A technique similar to CoachGAN is used for image inpainting [35], but some components of the algorithm (e.g. pixel distance-weighting to corrupted image regions) do not generalize to other GAN tasks.

The Deep Image Prior (DIP) method [30] asserts that a randomly initialized neural network is an effective image prior as the result of low-level, structural information that exists implicitly in the network architecture. However, DIP requires an iterative, computationally-expensive optimization of an *entire* randomly-initialized neural network for every single input instance. CoachGAN, on the other hand, can improve large batches of outputs simultaneously. Furthermore, CoachGAN naturally extends to non-image domains, while DIP is exclusively suited to the image domain.

Other methods, such as the GLO framework [3], similarly perform optimization in the latent space. While GLO gives a training-time approach to optimize an embedding space using Laplacian pyramid and ℓ_2 losses, CoachGAN provides an inference time method that works with arbitrary differentiable losses. Also, despite encouraging results on the CelebA dataset, GLO performs poorly compared to GANs on larger datasets such as LSUN [3].

The introspective generative modeling approach generates textures by performing gradient accent in pixel-space over a series of $T = 20$ trained CNN classifiers [17]. In contrast, CoachGAN uses a single discriminator and performs gradient descent in latent space to improve generator outputs at inference time. An advisor analogy is also used in [34] to describe training two distinct generative models using MCMC sampling. Despite a similar metaphor, CoachGAN differs in purpose and approach. Our method is architecture-agnostic and runs at inference time.

3. Method

CoachGAN provides a post-training, modular approach to improve the outputs of a generator. Given input z and pretrained G and D models with frozen weights, CoachGAN improves output realism by gradually altering z using backpropagation and gradient descent. The input z is not limited to latent space vectors, but can take the form of any continuous input space. CoachGAN uses the same differentiable loss, L_G, used during training for G. In this case, however, CoachGAN backpropagates L_G through D and G and performs a gradient descent update *only* on z. Thus at timestep t CoachGAN computes the loss

$$L_t = L_G(G(z_t)) \tag{1}$$

and z receives receives a gradient update according to the chosen optimization method. Under basic stochastic gradient descent optimization, z_{t+1} would update as

$$z_{t+1} = z_t - \eta \frac{\partial L_G}{\partial z} \tag{2}$$

where η is the learning rate. At time $t + 1$, CoachGAN computes the loss as

$$L_{t+1} = L_G(G(z_{t+1})) \tag{3}$$

This CoachGAN optimization process repeats for a user-specified number of iterations, κ.

As a specific example, consider the original minimax GAN objective function:

$$\min_G \max_D \log(D(x)) + \log(1 - D(G(z))) \quad (4)$$

Traditionally, an optimizer for G minimizes $L_G = \log(1 - D(G(z)))$, or it maximizes $\log(D(G(z))$, which helps avoid vanishing gradient problems [9]. CoachGAN uses the the same L_G as in training, but does not compute gradients with respect to θ_G, the weights of G. Instead, CoachGAN computes gradients with respect to the original *input*, z. In this work, we use the improved Wasserstein GAN loss [11], where CoachGAN attempts to minimize $L_G = -D(G(z))$. In the following subsection, we discuss the CoachGAN loss surface and path taken by gradient descent.

3.1. Theory

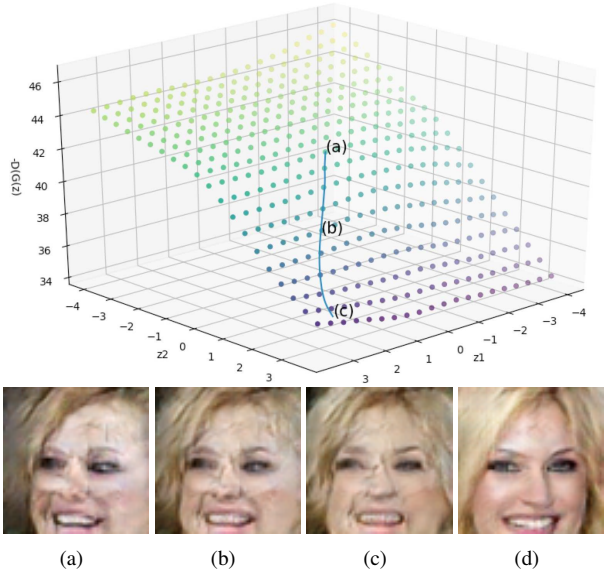
The work of [28] shows that the latent space manifolds of deep neural networks approximate zero curvature. This suggests that movement in latent space z resembles geodesics, which minimize the distance between output points [31]. This idea is commonly used in GAN spherical interpolation methods [29, 33], which produce more realistic output transitions than linear interpolations between outputs. From this viewpoint, we can consider CoachGAN as a partial optimization of latent sample z in an $\mathbb{R}^{|z|}$-dimensional manifold defined by L_G. Similar to geodesics, small movements in the z latent space can quickly produce realistic transitions in the output.

Like heuristic activation and weight regularization techniques, CoachGAN does not provably guarantee improved output realism. However, empirical results suggest that CoachGAN tends to improve generated outputs.

3.2. Intuition

To demonstrate the behavior of CoachGAN, we conduct a simple experiment using the pretrained DCGAN G and D that generated the results shown in Figure 1. We sample a single 100-dimensional z vector from a spherical Gaussian. Since visualizing the effects of CoachGAN in $\mathbb{R}^{100}$ space is infeasible, we perform the following simplifications: We hold all z_k constant, where $k \in [3, 100]$, and only allow CoachGAN to change z_1 and z_2. For reference, we plot the outputs of $-D(G(z))$ when varying z_1 and z_2 from -4 to 4 by increments of 0.4. This provides an intuitive illustration of CoachGAN's loss landscape and behavior throughout optimization.

We initialized CoachGAN with $z_1 = z_2 = 0.0$ and allowed the algorithm to run for 200 iterations using an Adam optimizer with a learning rate of 0.01. Figure 2 plots the path traveled by CoachGAN as well as sample outputs.



(a) (b) (c) (d)

Figure 2. Starting at $z_1 = z_2 = 0.0$, CoachGAN uses an optimizer (Adam used here) to follow the direction in z-space that minimizes that value of $-D(G(z))$. To visualize the effects of our method, we only allow CoachGAN to update z_1 and z_2. Images (a-c) and the associated marks on the graph correspond with CoachGAN outputs at iteration 1, 200, and 400, respectively. Image (d) shows the output generated when CoachGAN optimizes the full z vector.

Subfigures 2(a-c) and the associated marks on the graph correspond with CoachGAN outputs at iterations 1, 100, and 200, respectively. The realism of the images increase with more iterations, though the results are limited when only optimizes 2 out of 100 dimensions. Figure 2(d) shows the final output at 200 iterations when CoachGAN can optimize all z_k, not just z_1 and z_2.

In later experiments, we use a smaller learning rate and fewer coaching iterations to prevent significant changes in the identity of the original output (e.g. Subfigure 2(d)). We employed the high η and κ values solely to demonstrate the movement through the loss surface.

3.2.1 Basins of Attraction

One consideration in evaluating the usefulness of CoachGAN is whether or not CoachGAN descreases the number of possible outputs from G. In other words, does CoachGAN guide z to a limited number of basins in the input space? Also, does CoachGAN simply push z toward previous z values encountered during training?

To answer these questions, we conducted a simple experiment using DCGAN G and D models and the MNIST dataset. We train the models for 10 epochs on the training set and record all 600,000 z_t used in training. After training, we sample an additional 5, 000 random z_i and perform CoachGAN for 10 iterations with a learning rate of 0.01,

which yields the coached input, z_{ic}. To test whether similar but unique z converge to the same basin, we also perform CoachGAN on $z'_i = z_i + \mathcal{N}(0, \epsilon)$, where $\epsilon = 0.001$. This finds the coached input z'_{ic}. We tested other parameter values (i.e. learning rate, number of CoachGAN iterations, and ϵ) and report those results in our supplementary materials.

To quantify the relationship between z_i and z_{ic}, as well as z_{ic} and z'_{ic}, we compute a number of statistics. First, we calculate distance of z_c to the the nearest z_t in z-space. To calculate this value we average the L2-norm of the vector difference between z_i and the nearest z_t from the training set.

$$Nearest\text{-}Training\text{-}Z\text{-}Distance = \frac{1}{n}\sum_i \min_t ||z_i - z_t|| \tag{5}$$

Next we calculate the distance in pixel space between $G(z_i)$ and $G(z'_{ic})$ as the L2-Norm between the outputs, which we average across all 5,000 samples.

$$Pixel\text{-}Distance = \frac{1}{n}\sum_i ||G(z_i) - G(z'_{ic})|| \tag{6}$$

Finally, we calculate the average z-space distance between z_{ic} and z'_{ic}.

$$Z\text{-}Distance = \frac{1}{n}\sum_i ||z_{ic} - z'_{ic}|| \tag{7}$$

This measures the similarity of the z vector coaching paths given only a small amount of Gaussian noise differentiating the inputs.

The top-left plot in Figure 3 shows both that z_i and z'_i yield measurably different output images, and CoachGAN does not simply push z_i toward z encountered during training. The top-right plot shows that z_i and z'_i end up at distinct z after undergoing CoachGAN refinement. The bottom plot confirms the positive correlation between pixel-distance and z-space distance.

Figure 4 shows sample outputs from this experiment. Our results demonstrate that CoachGAN does not push z vectors into large basins of attraction, or significantly reduce the number of possible outputs. Even when z_i and z_{ic} differ by just $\mathcal{N}(0, 0.001)$, the resultant outputs often show visual distinctions. This suggests that CoachGAN makes dimension specific updates based on the overall state of an embedding vector. A tiny amount of noise added to an input vector can alter which dimensions CoachGAN updates at inference time.

3.2.2 Discussion

Dense, high-dimensional target domain spaces makes the generation of realistic outputs for all z highly improbable. Training algorithms like Wasserstein GAN [1] and Improved Wasserstein GAN [11] encourage smoother interpolations in the output space and generally improve convergence. However, these methods still struggle to achieve full mode-coverage due to factors like insufficient model capacity or a poorly enforced Lipschitz gradient contraint [27].

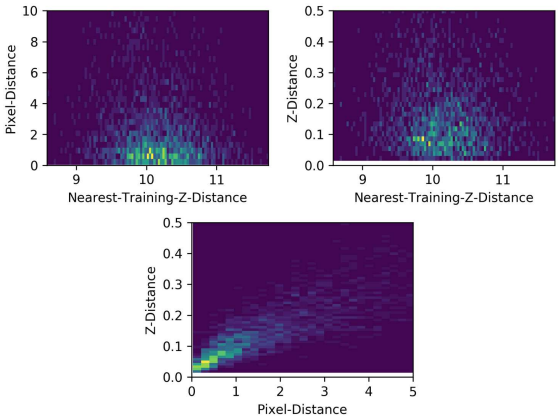


Figure 3. 2D histograms comparing the relationships of pixel distance, z-space distance, and nearest training z distance metrics. Even with a difference of just $\mathcal{N}(0, 0.001)$, z_i and z'_i often produce visually distinct outputs. Furthermore, CoachGAN does not push z vectors toward basins of attraction around z_t.

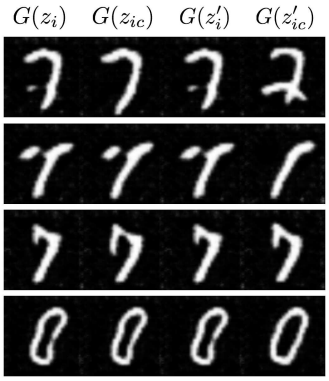


Figure 4. Output comparison of the original and coached samples, $G(z_i)$ and $G(z_{ic})$, to $G(z'_i)$ and $G(z'_{ic})$, where $z'_{ic} = z_i + \mathcal{N}(0, 0.001)$. Even with a small amount of random noise (which does not noticeably alter the original output), visual differences appear in the resultant coached outputs.

CoachGAN provides a method to push portions of the input into domain areas that lead to more confidently realistic outputs. As an example, consider the output shown in Figure 5. The leftmost image shows the faint outline of a pair of glasses around the man's eyes. CoachGAN performs element-wise adjustments on z using gradient descent, which results in a more pronounced pair of glasses in the final right image. Effectively, CoachGAN encourages small coordinated steps in the continuous domain space to generate a more believable final output.

Figure 5. Left to right: transition from original image to final CoachGAN output. The leftmost image shows a faint, partially generated set of glasses. By pushing the source input vector toward the closest element-wise modes, CoachGAN allows G to generate a clear set of glasses.

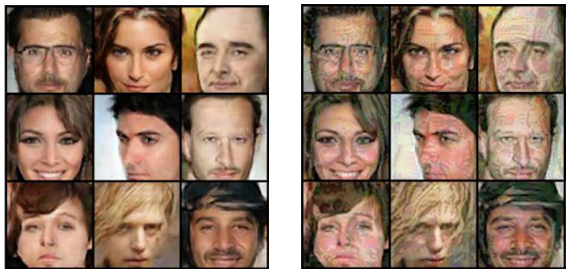


Figure 6. Introducing adversarial noise to the the original outputs (left) creates blurred and artifact-laden images (right) which fool D. By instead altering the input directly, CoachGAN allows for enhancements in output realism.

3.3. Optimizing output directly

For sake of comparison, we experimented with optimizing the output of G directly, rather than altering the original input. This removes the need to backpropagate through G and slightly speeds up the CoachGAN process. Unfortunately, this approach does not produce the same high quality results as the input-altering CoachGAN algorithm. Figure 6 shows an example output of CoachGAN when optimizing the outputs for the CelebA dataset. Instead of increasing the realism of outputs, this method simply adds adversarial-like noise [10].

4. Experiments

CoachGAN does not assume a particular optimization algorithm, but we use Adam for our experiments. We found a basic tradeoff between the chosen number of iterations, κ, and the optimizer learning rate, η. For example, in Figure 7, we show the output refinement for a single image with $\eta \in \{0.002, 0.004, 0.01\}$ (rows) and $\kappa \in \{5, 10, 20, 40\}$ (columns). The diagonal from top right to bottom left reveals that the various settings produced similar output images. For faster convergence, CoachGAN can use a larger η and smaller κ. In general, however, we found the best results with smaller η such as 0.001 and κ between 50 and 100. This appears to hold true for a variety of algorithms and datasets, which we discuss in subsequent sections.

CoachGAN adapts easily to various training algorithms and provides noticeable improvements to generated outputs. We evaluate CoachGAN on the original unconditional GAN architecture [9] using the CelebA and LSUN bedroom datasets. Adding CoachGAN required minimal code alteration and often led to substantial qualitative improvements, as we demonstrate in the following subsections.

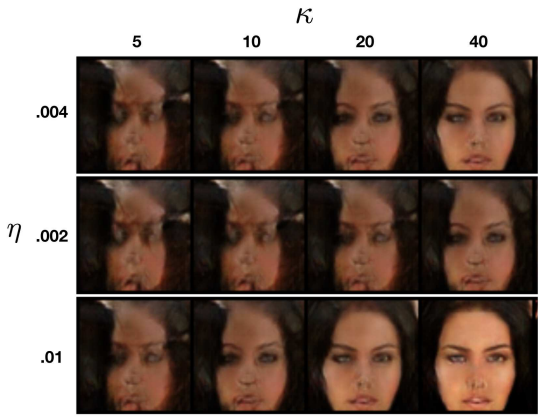


Figure 7. Effects of different learning rates, η, and iterations, κ, for CoachGAN using G and D trained for 40 epochs on the CelebA dataset. Left to right: transition from original image to final CoachGAN output.

We emphasize that CoachGAN improves the majority of GAN outputs. Under certain circumstances, such as poorly trained models or blurry training images, we observed that CoachGAN magnifies existing noise in outputs. Additionally, if the initial output, $G(z)$, already appears realistic, CoachGAN does not directly improve realism, but adjusts image characteristics to match those most favored by D. We provide specific examples of this D favoritism in our results. We also provide randomly sampled CoachGAN outputs and quantitative evaluation to demonstrate the general behavior of CoachGAN.

4.1. Unconditional GAN

We first present results for the basic unconditional GAN algorithm. We employ the DCGAN [24] G and D models and the WGAN-GP [11] training algorithm for these experiments.

4.1.1 CelebA

For the CelebA experiments, we trained G and D for 40 epochs using the same parameters as [24]: A learning rate of 0.0002, batch size of 64, and the Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We center cropped the training data and resized images to 64x64 for faster processing.

As demonstrated in Figures 1, 2, and 5, CoachGAN can provide remarkable improvements to low quality outputs from G. We include additional CelebA output transformations from randomly sampled z vectors in the supplementary materials. These results illustrate the effectiveness

Figure 8. Left to right: transition from original image to final CoachGAN output. CoachGAN reveals learned rotation biases of D. Both the man and woman are centered with eyes facing forward.

Figure 9. 5-Nearest Neighbors in the training set for the original generated output (top-left) and the CoachGAN refined output (bottom-left).

of CoachGAN in performing a wide variety of realism enhancements at inference time.

When the output images of G already appear realistic, CoachGAN pushes outputs to resemble the modes of the dataset, as captured by D during training. For instance, Figure 8 reveals that D favors centered, face-forward images. This is not unexpected, as most images in the CelebA dataset possess these characteristics. Other modes of D that we empirically observed include brightening images, removing bangs, and reducing baldness.

4.1.2 Nearest neighbor

Similar to Section 3.2.1, we verify that CoachGAN does not push generated outputs toward existing training samples. In order to test this, we perform a 5-Nearest Neighbor search on the training set. Rather than use a distance metric in the output space, we adopt the method of [7], which computes the distance in feature space using the activations of several layers of a pretrained VGG-19 network. Figure 9 displays the nearest neighbor results for both an original output sample and the CoachGAN refined output. The results show that CoachGAN does not push outputs toward existing training set samples. Rather, CoachGAN greedily adjusts inputs in Z space to generate outputs that better fool D.

4.1.3 LSUN

Using the same DCGAN and WGAN-GP setup as the CelebA experiments, we further explored the effects of

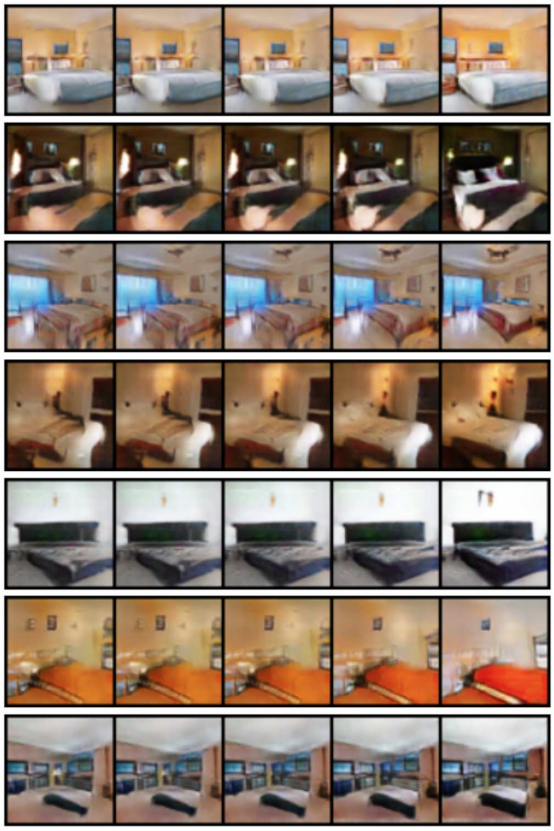


Figure 10. Randomly sampled CoachGAN outputs using a G and D trained on the LSUN bedroom dataset. Left to right: transition from original output to final CoachGAN output.

CoachGAN on downsampled 64x64 LSUN bedroom images. Figure 10 shows the CoachGAN transformations for 7 randomly sampled z vectors. These outputs demonstrate that CoachGAN is not limited to highly regular datasets, such as the centered and aligned CelebA face images. For the generated LSUN outputs, CoachGAN clarifies the edges of bedding and walls, removes blurring, and even adds details to windows and reflective surfaces.

4.1.4 Progressive Growing of GANs

An increasing number of research groups provide access to their pretrained GAN weights online. This means that CoachGAN can readily be used with a variety of publically available state-of-the-art methods. To demonstrate the ease with which CoachGAN adapts to other models, we applied CoachGAN to the Progressive Growing of GANs (PGGAN) [14] inference method. Expanding upon earlier pyramid-based [8] or multi-step GAN training algorithms [36], PGGAN generates outputs of increasingly higher resolution - ultimately producing a 1024×1024 output images.

With only a few lines of additional code, we successfully augmented PGGAN inference with CoachGAN. Fig-

ure 11 shows samples comparing original (left) and CoachGAN outputs (right). In the top image, CoachGAN largely removes the color blob covering the man's head. As for the middle image, CoachGAN improves the hair texture and removes artifacts from the forehead. In the bottom image, CoachGAN seals a gap in the woman's neck and completes a partially created earing. We include additional PGGAN samples with corresponding CoachGAN outputs in our supplementary materials.

4.1.5 StyleGAN

The recent StyleGAN model [15] builds upon style transfer methods [13] to yield an unprecedented level of high-definition sample quality as well as a better disentangled feature space. Besides introducing and using the larger 70,000-image FFHQ dataset (compared to 30,000 images in CelebA-HQ), StyleGAN relies on the truncation trick (TT) [5], which reduces the latent sampling space in order to improve image quality. While the resampling approach of TT performs a similar function as CoachGAN, TT reduces image variation – especially when using low truncation values. In contrast, CoachGAN can yield noticeable improvements with just tiny changes to the original latent vector, as demonstrated in Section 3.2.1.

TT also does not scale well to arbitrary generator architectures. For instance, [5] notes that orthogonal regularization of the weights of each layer of the generator is needed for TT to work effectively in their BigGAN method. CoachGAN, on the other hand, works with any differentiable GAN architecture and does not require layer modification or access to the generator weights. We note that TT relies on random resampling of the input latent vector, whereas CoachGAN uses gradient descent to more intelligently search for a 'improved' latent vector. We expect that CoachGAN can yield improvements even after applying TT. To test this claim, we compare FID scores of StyleGAN using TT with a threshold of 0.7 (as used in [15]) and TT + CoachGAN in Section 4.4.

4.2. Conditional GAN

4.2.1 BigGAN

Google's class conditional BigGAN [5] significantly improved state-of-the-art FID and IS scores for condition GAN generation. Although the official online repository provides only pretrained generator weights for several image resolutions, the primary BigGAN author released a PyTorch version of BigGAN, which includes pretrained G and D models. Because this unofficial model trained on just ImageNet (rather than expanded dataset used in the paper), the model does not attain the state-of-the-art results reported in [5]. However, we still observe improvements in IS and FID score, as we demonstrate in the next section. We include a video of before and after CoachGAN image comparisons for BigGAN in our supplementary work. z

4.3. Quantitative evaluation

Inception Score (IS) [11] remains one of the most popular and widely adopted GAN evaluation metrics. Using a representative image sample from generator, G, IS produces class label distributions using a pretrained Inception v3 classification neural network. IS then calculates the Kullback-Leibler divergence between the class distribution of each generated output, $p(y|G(z_i))$, and the average label distribution of all samples, $p(y)$. The exponentiated expectation of these KL-divergences yields the final IS score. We write this as

$$IS(G) = exp(\ \mathbb{E}_{G(z_i)}\ KL(\ p(y|G(z_i))\ ||\ p(y))\ ) \quad (8)$$

The work introducing IS, [25], states that IS tends to correlate well with human opinion of image realism.

Since IS calculates an entropy distribution over 1,000 ImageNet classes, we do not measure IS on the single-class datasets such as CelebA and LSUN-bedroom. We do, however, measure IS on a DCGAN trained on CIFAR-10 for 200 epochs and a BigGAN model trained for 100 epochs. Due to limited model capacity (DCGAN) and reduced training time and data (BigGAN), we do not observe state-of-the-art IS values in these experiments. Rather, we hypothesize that CoachGAN will lead to a relative increase in IS for each of the GAN experiments.

Following the recommendation of [2], we generate 50,000 samples for calculating IS. We report the initial IS and IS after applying CoachGAN with $\kappa = 10$ and $\eta = 0.01$ for CIFAR and $\kappa = 1$ and $\eta = 0.01$ for BigGAN. The results in Table 1 show that CoachGAN improves the baseline IS score.

Dataset	IS-PRE	IS-POST
CIFAR-10	4.79	**5.00**
BigGAN	60.30	**60.48**

Table 1. Inception Score calculated before (PRE) and after CoachGAN (POST), where higher scores are better.

4.4. Fréchet inception distance

Although IS remains a popular metric for GAN evaluation, its use is limited for datasets that do not share classes with ImageNet. In fact, an increasing number of theoretical and empirical analyses [2, 4, 22] demonstrate that IS does not measure intra-class diversity and fails to detect training set memorization. Additionally, IS relies on an Inception model pretrained on the 1000-label ImageNet dataset, which may not be appropriate for non-ImageNet GAN evaluation tasks (e.g. GANs trained on data with image statistics and label distributions that differ noticeably from ImageNet).

Because of this, we evaluate CoachGAN using the Fréchet Inception Distance (FID) [12], which is calculated based on the activations of a 2048-dimension pooling layer in the Inception v3 network. Using real data samples, X, and generated samples, $G(Z)$, FID is calculated as

$$FID = ||\mu_X - \mu_{G(Z)}||^2 + \text{Tr}(\Sigma_X + \Sigma_{G(Z)} - 2(\Sigma_X \Sigma_{G(Z)})^{\frac{1}{2}}) \quad (9)$$

We also calculate FID using coached Z' values.

Using pretrained CIFAR-10, LSUN-bedroom, and CelebA DCGAN models, we use $\kappa = 10$ and $\eta = 0.01$ to compare FID before and after CoachGAN. We calculate these scores using FID statistics computed over the entire training datasets and 50,000 generated or coached images. CoachGAN produces better (lower) FID scores for two out of three of the DCGAN models (see Table 2).

Dataset	PRE	POST
CelebA	17.95	**16.43**
CIFAR-10	35.29	**34.38**
LSUN	**24.22**	24.34
PGGAN	54.36	**54.13**
StyleGAN	16.74	**16.55**
BigGAN	40.35	**40.20**

Table 2. Frechet Inception Distance calculated with z samples before (PRE) and after applying CoachGAN (POST). CoachGAN improves FID (lower is better) for both CelebA and CIFAR-10 pretrained DCGAN models.

We also calculate pre- and post-CoachGAN FID scores for PGGAN, StyleGAN, and BigGAN using $\kappa = 1$ and $\eta = 0.01$ (we observed similar results for various tradeoffs of η and κ). Because Celeb-HQ only contains 30,000 training images, it is not ideal for calculating FID scores with 50,000 samples PGGAN images. The FFHQ dataset, which is an expanded version of the Celeb-HQ dataset from [15], contains 70,000 images. We use FFHQ as the ground truth dataset for scoring both PGGAN and StyleGAN. Table 2 reports the FID results for these experiments. All three models yield improved FID scores when using CoachGAN.

4.4.1 Runtime

While CoachGAN works best with small learning rates and more iterations, we also observe improvements for single-step coaching with larger learning rates (as demonstrated in the PGGAN, StyleGAN, and BigGAN experiments). Single-step coaching can improve results with a moderate increase in running time. To quantify this statement, we compare generation times with and without CoachGAN for DCGAN, PGGAN and BigGAN in Table 3. We use an NVIDIA Tesla P100 GPU for these experiments.

Although using CoachGAN with BigGAN (the model with the greatest complexity) roughly triples inference time, the total time is still under 0.2 seconds. In our experiments, coaching 50,000 samples with a batch-size of 16 took less than an hour on a single GPU. Larger batch sizes and a more powerful GPU could further reduce the time needed for CoachGAN.

Figure 11. Left: Original images generated with the Progressive Growing of GANs model [14]. Right: The resultant images after applying CoachGAN.

	Normal	CoachGAN
DCGAN	0.0009073	0.00535
PGGAN	0.02202	0.0229
BigGAN	0.06157	0.18806

Table 3. Generation runtime (in seconds) with and without CoachGAN.

5. Conclusion

CoachGAN provides a modular, effective approach to improve generator outputs at inference time. We have empirically demonstrated qualitative and quantitative improvements using CoachGAN with a variety of datasets. We also demonstrated the ease of applying CoachGAN to different GAN architectures with distinct loss functions (e.g. DCGAN, PGGAN, StyleGAN, and BigGAN). Regarding current applications, there is an increasing interest in using GANs for image editing and synthesis [6, 38, 32, 18]. Such applications require high quality results and the potential for user control. CoachGAN can both refine poor quality results and allow users to select a precise output from sequences of images generated by CoachGAN's SGD steps.

Future work will consider task-specific constraints to preserve desired features. For instance, CoachGAN could restrict changes that alter characteristics such as gender, hair color, or pose when improving outputs of the CelebA dataset. This could prove useful when applying CoachGAN to video sequence generation, where certain visual features must stay constant from frame to frame.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[2] S. Barratt and R. Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[3] P. Bojanowski, A. Joulin, D. Lopez-Pas, and A. Szlam. Optimizing the latent space of generative networks. In *International Conference on Machine Learning*, pages 599–608, 2018.

[4] A. Borji. Pros and cons of gan evaluation measures. *arXiv preprint arXiv:1802.03446*, 2018.

[5] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[6] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

[7] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, page 3, 2017.

[8] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[10] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[12] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[13] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[14] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[15] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[16] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[17] J. Lazarow, L. Jin, and Z. Tu. Introspective neural networks for generative modeling. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2793–2802, Oct 2017.

[18] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz. A closed-form solution to photorealistic image stylization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 453–468, 2018.

[19] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. *arXiv preprint*, 2017.

[20] A. Nayebi and S. Ganguli. Biologically inspired protection of deep networks from adversarial attacks. *arXiv preprint arXiv:1703.09202*, 2017.

[21] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems*, pages 3387–3395, 2016.

[22] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

[23] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[24] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[25] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[26] P. Samangouei, M. Kabkab, and R. Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

[27] M. Sanjabi, J. Ba, M. Razaviyayn, and J. D. Lee. On the convergence and robustness of training gans with regularized optimal transport. In *Advances in Neural Information Processing Systems*, pages 7091–7101, 2018.

[28] H. Shao, A. Kumar, and P. Thomas Fletcher. The riemannian geometry of deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 315–323, 2018.

[29] K. Shoemake. Animating rotation with quaternion curves. In *ACM SIGGRAPH computer graphics*, volume 19, pages 245–254. ACM, 1985.

[30] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454, 2018.

[31] N. K. Vishnoi. Geodesic convex optimization: Differentiation on manifolds, geodesics, and convexity. *arXiv preprint arXiv:1806.06373*, 2018.

[32] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.

[33] T. White. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

[34] J. Xie, Y. Lu, R. Gao, S.-C. Zhu, and Y. N. Wu. Cooperative training of descriptor and generator networks. *arXiv preprint arXiv:1609.09408*, 2016.

[35] R. A. Yeh, C. Chen, T.-Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with deep generative models. In *CVPR*, volume 2, page 4, 2017.

[36] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5907–5915, 2017.

[37] Z. Zhao, D. Dua, and S. Singh. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*, 2017.

[38] 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*, pages 597–613. Springer, 2016.