

Sparsity Aware Normalization for GANs

Idan Kligvasser, Tomer Michaeli

Technion–Israel Institute of Technology, Haifa, Israel
{kligvasser@campus, tomer.m@ee}.technion.ac.il

Abstract

Generative adversarial networks (GANs) are known to benefit from regularization or normalization of their critic (discriminator) network during training. In this paper, we analyze the popular spectral normalization scheme, find a significant drawback and introduce *sparsity aware normalization* (SAN), a new alternative approach for stabilizing GAN training. As opposed to other normalization methods, our approach explicitly accounts for the sparse nature of the feature maps in convolutional networks with ReLU activations. We illustrate the effectiveness of our method through extensive experiments with a variety of network architectures. As we show, sparsity is particularly dominant in critics used for image-to-image translation settings. In these cases our approach improves upon existing methods, in less training epochs and with smaller capacity networks, while requiring practically no computational overhead.

Introduction

Generative adversarial networks (GANs) (Goodfellow et al. 2014) have made a dramatic impact on low-level vision and graphics, particularly in tasks relating to image generation (Radford, Metz, and Chintala 2015; Karras et al. 2017), image-to-image translation (Isola et al. 2017; Zhu et al. 2017; Choi et al. 2018), and single image super resolution (Ledig et al. 2017; Wang et al. 2018; Bahat and Michaeli 2019). GANs can generate photo-realistic samples of fantastic quality (Karras, Laine, and Aila 2019; Brock, Donahue, and Simonyan 2018; Shaham, Dekel, and Michaeli 2019; Ledig et al. 2017), however they are often hard to train and require careful use of regularization and/or normalization methods for making the training stable and effective.

A factor of key importance in GAN training, is the way by which the critic (discriminator) network is optimized. An overly-sharp discrimination function can lead to gradient vanishing when updating the generator, while an overly-smooth function can lead to poor discrimination between real and fake samples and thus to insufficient supervision for the generator. One of the most successful training approaches, is that arising from the Wasserstein GAN (WGAN) (Arjovsky, Chintala, and Bottou 2017) formulation, which asserts that the critic should be chosen among

the set of Lipschitz-1 functions. Precisely enforcing this constraint is impractical (Virmaux and Scaman 2018), yet simple approximations, like weight clipping (Arjovsky, Chintala, and Bottou 2017) and gradient norm penalty (Gulrajani et al. 2017), are already quite effective.

Perhaps the most effective approximation strategy is *spectral normalization* (Miyato et al. 2018). This method normalizes the weights of the critic network after every update step, in an attempt to make each layer Lipschitz-1 individually (which would guarantee that the end-to-end function is Lipschitz-1 as well). Due to its simplicity and its significantly improved results, this approach has become the method of choice in numerous GAN based algorithms (*e.g.* (Miyato and Koyama 2018; Park et al. 2019; Brock, Donahue, and Simonyan 2018; Armanious et al. 2020)).

In this paper, we present a new weight normalization strategy that outperforms spectral normalization, as well as all other methods, by a significant margin on many tasks and with various network architectures (see *e.g.*, Fig. 1). We start by showing, both theoretically and empirically, that normalizing each layer to be Lipschitz-1 is overly restrictive. In fact, as we illustrate, such a normalization leads to very poor GAN training if done correctly. We identify that the real reason for the success of (Miyato et al. 2018) is actually its systematic bias in the estimation of the Lipschitz constant for convolution layers, which is typically off by roughly a factor of 4. Following our analysis, we show that a better way to control the end-to-end smoothness of the critic, is to normalize each layer by its amplification of the *typical signals* that enter it (rather than the worst-case ones). As we demonstrate, in convolutional networks with ReLU activations, these signals are typically channel-sparse (namely many of their channels are identically zero). This motivates us to suggest *sparsity aware normalization* (SAN).

Our normalization has several advantages over spectral normalization. First, it leads to better visual results, as also supported by quantitative evaluations with the Inception score (IS) (Salimans et al. 2016) and the Fréchet Inception distance (FID) (Heusel et al. 2017). This is true in both unconditional image generation and conditional tasks, such as label-to-image translation, super-resolution, and attribute transfer. Second, our approach better stabilizes the training, and it does so at practically no computational overhead. In particular, even if we apply only a single update step of

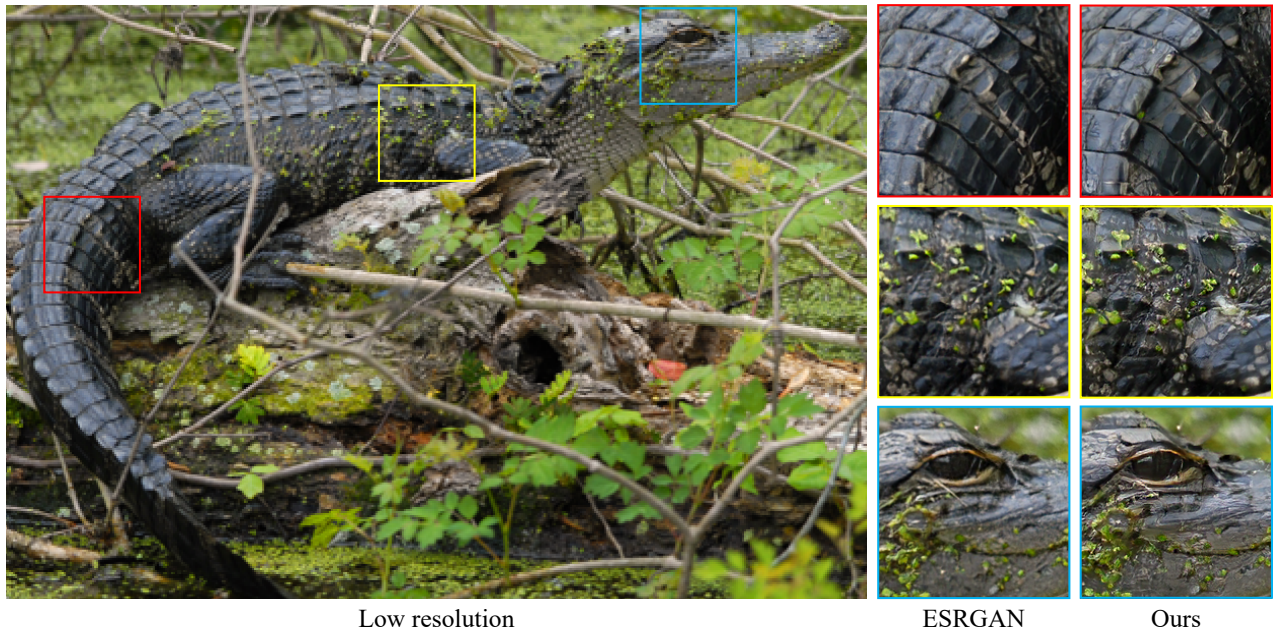


Figure 1: Super resolution with our sparsity aware normalization. Our technique can boost the performance of any GAN-based method, while allowing less training epochs and smaller models. For example, in the task of $4\times$ super-resolution, we achieve more photo-realistic reconstructions than the state-of-the-art ESRGAN network (Wang et al. 2018), while using a model with only 9% the number of parameters of ESRGAN (1.5M for ours and 16.7M for ESRGAN).

the critic for each update of the generator, and normalize its weights only once every 1K steps, we still obtain an improvement over spectral normalization. Finally, while spectral normalization benefits from different tuning of the optimization hyper-parameters for different tasks, our approach works well with the precise same settings for all tasks.

Rethinking Per-Layer Normalization

GANs (Goodfellow et al. 2014) minimize the distance between the distribution of their generated “fake” samples, \mathbb{P}_F , and the distribution of real images, \mathbb{P}_R , by diminishing the ability to discriminate between samples drawn from \mathbb{P}_F and samples drawn from \mathbb{P}_R . In particular, the Wasserstein GAN (WGAN) (Arjovsky, Chintala, and Bottou 2017) targets the minimization of the Wasserstein distance between \mathbb{P}_F and \mathbb{P}_R , which can be expressed as

$$W(\mathbb{P}_R, \mathbb{P}_F) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_R}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_F}[f(x)]. \quad (1)$$

Here, the optimization is over all critic functions $f : \mathbb{R}^n \rightarrow \mathbb{R}$ whose Lipschitz constant is no larger than 1. Thus, the critic’s goal is to output large values for samples from \mathbb{P}_R and small values for samples from \mathbb{P}_F . The GAN’s generator attempts to shape the distribution of fake samples, \mathbb{P}_F , so as to minimize $W(\mathbb{P}_R, \mathbb{P}_F)$ and so to rather decrease this gap.

The Lipschitz constraint has an important role in the training of WGANs, as it prevents overly sharp discrimination functions that hinder the ability to update the generator. However, since f is a neural network, this constraint is impractical to enforce precisely (Virmaux and Scaman 2018),

and existing methods resort to rather inaccurate approximations. Perhaps the simplest approach is to clip the weights of the critic network (Arjovsky, Chintala, and Bottou 2017). However, this leads to stability issues if the clipping value is taken to be too small or too large. An alternative, is to penalize the norm of the gradient of the critic network (Gulrajani et al. 2017). Yet, this often has poor generalization to points outside the support of the current generative distribution.

To mitigate these problems, Miyato *et al.* (2018) suggested to enforce the Lipschitz constraint on each layer individually. Specifically, denoting the function applied by the i th layer by $\phi_i(\cdot)$, we can write

$$f(x) = (\phi_N \circ \phi_{N-1} \circ \dots \circ \phi_1)(x). \quad (2)$$

Now, since $\|\phi_1 \circ \phi_2\|_L \leq \|\phi_1\|_L \cdot \|\phi_2\|_L$, we have that

$$\|f\|_L \leq \|\phi_N\|_L \cdot \|\phi_{N-1}\|_L \cdot \dots \cdot \|\phi_1\|_L. \quad (3)$$

This implies that restricting each ϕ_i to be Lipschitz-1, ensures that f is also Lipschitz-1. Popular activation functions, such as ReLU and leaky ReLU, are Lipschitz-1 by construction. For linear layers (like convolutions), ensuring the Lipschitz condition merely requires normalizing the weights by the Lipschitz constant of the transform, which is the top singular value of the corresponding weight matrix.

This per-layer normalization strategy has gained significant popularity due to its simplicity and the improved results that it provides when compared to the preceding alternatives. However, close inspection reveals that normalizing each layer by its top singular value is actually too conservative. That is, restricting each layer to be Lipschitz-1, typically leads to a *much smaller set of permissible functions*

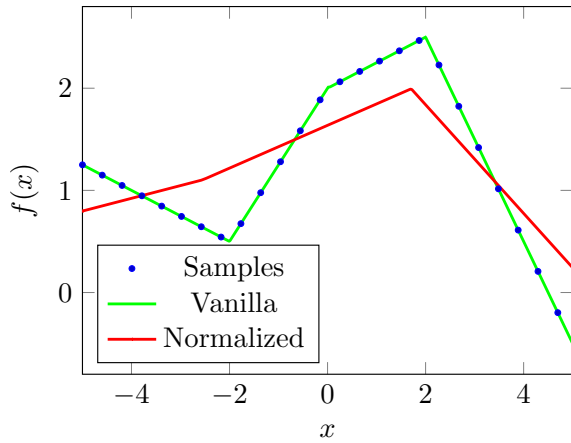


Figure 2: Fitting to a Lipschitz-1 function. Here, we trained a network with one hidden layer to fit to samples of a Lipschitz-1 function (blue dots). When using vanilla training without any normalization, the fit is perfect (green). However, when using layer-wise spectral normalization, the fit is poor (red). This illustrates the fact that the set of functions that can be represented by a network with Lipschitz-1 layers is often significantly smaller than the set of all Lipschitz-1 functions that can be represented by the same architecture.

than the set of functions whose end-to-end Lipschitz constant is 1. As a simple illustration, consider the following example (see proof in the Supplementary).

Example 1 Let $f : \mathbb{R} \rightarrow \mathbb{R}$ be a two-layer network with

$$\phi_1(x) = \sigma(w_1x + b_1), \quad \phi_2(z) = w_2^T z + b_2, \quad (4)$$

where σ is the ReLU activation function, $w_1, w_2, b_1 \in \mathbb{R}^n$, and $b_2 \in \mathbb{R}$. Such a critic can implement any continuous piece-wise linear function with $n+1$ segments. Now, the end-to-end constraint $\|f\|_L \leq 1$, restricts the slope of each segment to satisfy $|f'(x)| \leq 1$. But the layer-wise constraints¹ $\|w_1\| \leq 1, \|w_2\| \leq 1$, allow a much smaller set of functions, as they also impose for example that $|f'(-\infty) + f'(\infty)| \leq 1$. In particular, they rule out the identity function $f(x) = x$, and also any function with slope larger than 0.5 or smaller than -0.5 simultaneously for $x \rightarrow \infty$ and $x \rightarrow -\infty$. This is illustrated in Fig. 2.

This example highlights an important point. When we normalize a layer by its top singular value, we restrict how much it can amplify an arbitrary input. However, this is overly pessimistic since not all inputs to that layer are admissible. In the example above, for most choices of w_1 the input to the second layer is necessarily sparse because of the ReLU. Specifically, if w_1 has k_p positive entries and k_n negative ones, then the output of the first layer cannot contain more than $\max\{k_p, k_n\}$ non-zero entries. This suggests that when normalizing the second layer, we should only consider how much it amplifies sparse vectors.

¹Since w_1 and w_2 are $n \times 1$, their top singular value is simply their Euclidean norm.

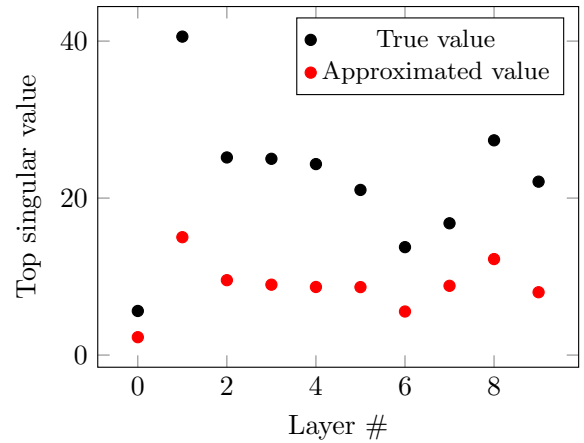


Figure 3: Top singular value. We plot the top singular value (black) of each convolution layer of a trained ResNet critic network, as well as its approximation employed by (Miyato et al. 2018) (red). The approximation is typically much smaller than the actual value, implying that the weights after normalization are in fact much larger than intended.

As a network gets deeper, the attenuation caused by such layer-wise normalization accumulates, and severely impairs the network’s representation power. One may wonder, then, why the layer-wise spectral normalization of (Miyato et al. 2018) works in practice after all. The answer is that for convolutional layers, this method uses a very crude approximation of the top singular value, which is typically $4\times$ smaller than the true top singular value². We empirically illustrate this in Fig. 3 for a ResNet critic architecture, where we use the Fourier domain formulation of (Sedghi, Gupta, and Long 2018) to compute the true top singular value. This observation implies that in (Miyato et al. 2018), the weights after normalization are in fact much larger than intended.

What would happen had we normalized each layer by its true top singular value? As shown in Fig. 4, in this case, the training completely fails. This is because the weights become extremely small and the gradients vanish.

Sparsity Aware Normalization

We saw that the spectral normalization of (Miyato et al. 2018) is effective because of the particular approximation used for $\|\phi_i\|_L$. A natural question, then, is whether we can somehow improve upon this normalization scheme. A naive approach would be to set a multiplier parameter σ , to adjust their normalization constant. However, as the authors of (Miyato et al. 2018) themselves indicate, such a parameter does not improve their results. This implies that the set of discriminator functions satisfying their per-layer constraints does not overlap well with the set of Lipschitz-1 functions as neither dilation nor erosion of this set improves their results.

²In (Miyato et al. 2018), the top singular value of the convolution operation is approximated by the top singular value of a 2D matrix obtained by reshaping the 4D kernel tensor.

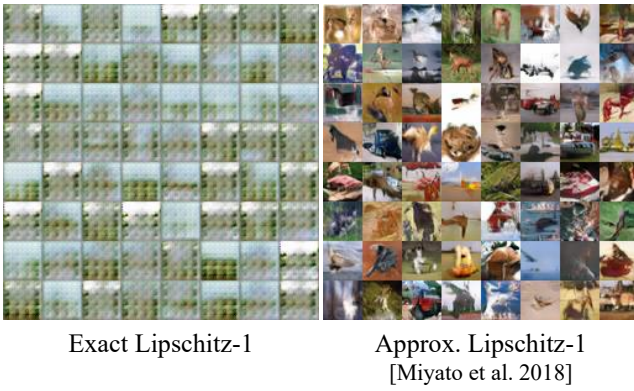


Figure 4: The effect of normalization. Here, we trained a WGAN on the CIFAR-10 dataset using exact per-layer spectral normalization and the approximate method of (Miyato et al. 2018), both initialized with the same draw of random weights. The exact normalization does not converge, while the approximate one rather leads to good conditioning.

A more appropriate strategy is therefore to seek for a normalization method that explicitly accounts for the statistics of signals that enter each layer. An important observation in this respect, is that in convolutional networks with ReLU activations, the features are typically channel-sparse. That is, for most input signals, many of the channels are identically zero. This is illustrated in Fig. 5, which shows a histogram of the norms of the channels of the last layer of a trained critic³, computed over 2048 randomly sampled images from the training set.

In light of this observation, rather than normalizing a layer $\phi(x) = Wx + b$ by its Lipschitz constant,

$$\|\phi\|_L = \sup_{\|x\| \leq 1} \|Wx\|, \quad (5)$$

here we propose to modify the constraint set to take into account only channel-sparse signals. Moreover, since we know that many output channels are going to be zeroed out by the ReLU that follows, we also modify the objective of (5) to consider the norm of each output channel individually.

Concretely, for a multi-channel signal x with channels x_1, \dots, x_k , let us denote by $\|x\|_\infty$ its largest channel norm, $\max\{\|x_i\|\}$, and by $\|x\|_0$ its number of nonzero channels, $\#\{\|x_i\| > 0\}$. With these definitions, we take our normalization constant to be⁴

$$\|W\|_{0,\infty} \triangleq \sup_{\substack{\|x\|_0 \leq 1 \\ \|x\|_\infty \leq 1}} \|Wx\|_\infty. \quad (6)$$

Normalizing by $\|W\|_{0,\infty}$ ensures that there exists no 1-sparse input signal (*i.e.* with a single nonzero channel) that can cause the norm of some output channel to exceed 1.

For convolutional layers, computing $\|W\|_{0,\infty}$ is simple. Specifically, if W has n input channels and m output chan-

³We used no normalization, but chose a run that converged.

⁴Note that $\|\cdot\|_{0,\infty}$ is not a norm since ℓ_0 is not a norm.

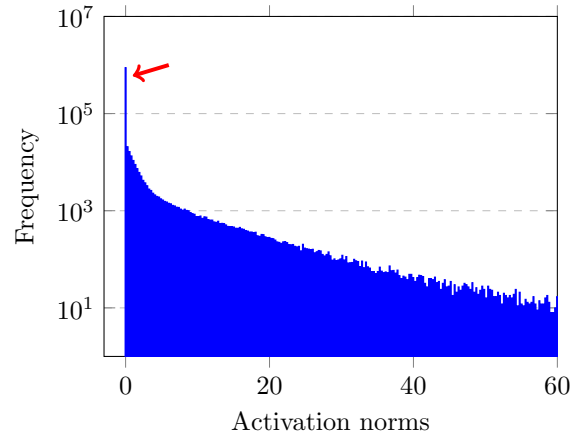


Figure 5: Channel sparsity. The histogram of the channel norms at the last layer of a critic trained without normalization. For most input images, many of the channels are identically zero.

nels, then the i th channel of $y = Wx$ can be expressed as

$$y_i = \sum_{j=1}^n w_{i,j} * x_j. \quad (7)$$

Here, ‘*’ denotes single-input-single-output convolution and $w_{i,j}$ is the kernel that links input channel j with output channel i . Now, using the kernels $\{w_{i,j}\}$, we can compute $\|W\|_{0,\infty}$ as follows (see proof in the Supplementary).

Lemma 1 For a multiple-input-multiple-output filter W with cyclic padding,

$$\|W\|_{0,\infty} = \max_{i,j} \|\mathcal{F}\{w_{i,j}\}\|_\infty, \quad (8)$$

where $\mathcal{F}\{w_{i,j}\}$ is the discrete Fourier transform of $w_{i,j}$, zero-padded to the spatial dimensions of the channels.

Thus, to compute our normalization constant, all we need to do is take the Fourier transform of each kernel, find the maximal absolute value in the transform domain, and then take the largest among these $m \times n$ top Fourier values.

Efficiency

To take advantage of Lemma 1, we use cyclic padding for all convolutional layers of the critic. This allows us to employ the fast Fourier transform (FFT) for computing the normalization constants of the layers. For fully-connected layers, we use the top singular value of the weight matrix, as in (Miyato et al. 2018). The overhead in running time is negligible. For example, on CIFAR-10, each critic update takes the same time as spectral normalization and 20% less than gradient-penalty regularization (see Supplementary).

In models for large images, storing the FFTs of all the filters of a layer can be prohibitive. In such settings, we compute the maximum in (8) only over a random subset of the filters. We compensate for our under-estimation of the maximum by multiplying the resulting value by a scalar g . As we show in the Supplementary, the optimal value of g varies

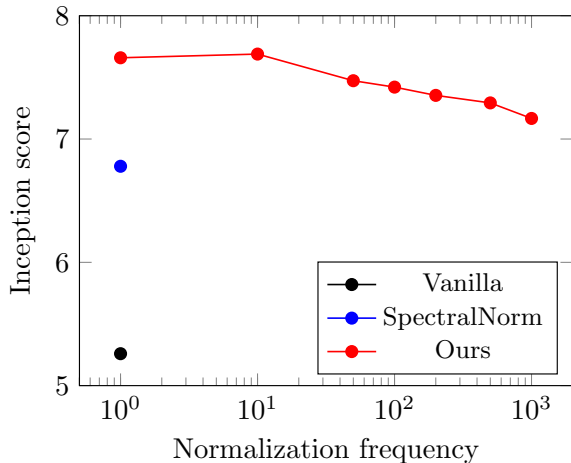


Figure 6: Efficiency. Here we compare three WGANs, trained for 100 epochs on the CIFAR-10 dataset (Krizhevsky and Hinton 2009): (i) without normalization, (ii) with spectral-normalization (Miyato et al. 2018), (iii) with our normalization. The training configurations and the initial seed are the same for all networks. In contrast to (Miyato et al. 2018), which performs weight normalization after each critic update, in our method we can normalize the layers much less frequently. Pay attention that even if we normalize only once every 1000 steps, less than 80 updates in total, we still outperform spectral normalization by a large margin.

very slowly as a function of the percentage of chosen filters (*e.g.* it typically does not exceed 1.3 even for ratios as low as 25%). This can be understood by regarding the kernels’ top Fourier coefficients as independent draws from some density. When this density decays fast, the expected value of the maximum over k draws increases very slowly for large k . For example, for the exponential distribution (which we find to be a good approximation), we show in the Supplementary that the optimal g for ratio r is given by

$$g = \frac{\sum_{j=1}^{m \cdot n} \frac{1}{m \cdot n - j + 1}}{\sum_{j=1}^{\lceil m \cdot n \cdot r \rceil} \frac{1}{\lceil m \cdot n \cdot r \rceil - j + 1}}, \quad (9)$$

leading to *e.g.* $g \approx 1.2$ for $r = 25\%$ with $m \cdot n = 64^2$ filters.

The effect of our normalization turns out to be very strong and therefore, besides a boost in performance, it also allows more efficient training than spectral normalization (Miyato et al. 2018) and other WGAN methods. Particularly:

Critic updates: For every update step of the generator, we perform only one update step of the critic. This is in contrast to other WGAN schemes, which typically use at least three (Arjovsky, Chintala, and Bottou 2017; Gulrajani et al. 2017; Miyato et al. 2018). Despite the fewer updates, our method converges faster (see Supplementary).

Normalization frequency: In spectral normalization, the weights are normalized after each critic update (using a single iteration of the power method). In contrast, we can normalize the layers much less frequently and still obtain a boost in performance. For example, as shown in Fig. 6, even

Method	CIFAR-10	STL-10
Real-data	11.24 ± .12	26.08 ± .26
Unconditional GAN		
(Standard CNN)		
Weight clipping	6.41 ± .11	7.57 ± .10
WGAN-GP	6.68 ± .06	8.42 ± .13
Batch norm	6.27 ± .10	
Layer norm	7.19 ± .12	7.61 ± .12
Weight norm	6.84 ± .07	7.16 ± .10
Orthonormal	7.40 ± .12	8.67 ± .08
SN-GANs	7.58 ± .12	8.79 ± .14
(ours) SAN-GANs	7.89 ± .09	9.18 ± .06
Unconditional GAN		
(ResNet)		
Orthonormal	7.92 ± .04	8.72 ± .06
SN-GANs	8.22 ± .05	9.10 ± .04
(ours) SAN-GANs	8.43 ± .13	9.21 ± .10
Conditional GAN		
(ResNet)		
BigGAN	9.24 ± .16	
(ours) SAN-BigGAN	9.53 ± .13	

Table 1: Inception scores for image generation on the CIFAR-10 and STL-10 datasets. The SAN method outperforms all other regularization methods.

if we normalize only once every 1000 steps, we still outperform spectral normalization by a large margin.

Hyper-parameters: As opposed to other normalization methods, like (Arjovsky and Bottou 2017; Gulrajani et al. 2017; Ioffe and Szegedy 2015; Salimans and Kingma 2016; Brock et al. 2016; Miyato et al. 2018), our algorithm does not require special hyper-parameter tuning for different tasks. All our experiments use the same hyper-parameters.

Experiments

We now demonstrate the effectiveness of our approach in several tasks. In all our experiments, we apply normalization after each critic update step to obtain the best results.

Image Generation

We start by performing image generation experiments on the CIFAR-10 (Krizhevsky and Hinton 2009) and STL-10 (Coates, Ng, and Lee 2011) datasets. We use these simple test-beds only for the purpose of comparing different regularization methods on the same architectures. Here, we use $r = 100\%$ of the filters (and thus a compensation of $g = 1$).

Our first set of architectures is that used in (Miyato et al. 2018). But to showcase the effectiveness of our method, in our STL-10 ResNet critic we remove the last residual block, which cuts its number of parameters by 75%, from 19.5M to 4.8M (the competing methods use the 19.5M variant). The architectures are described in full in the Supplementary. As in (Miyato et al. 2018), we use the hinge loss (Wang et al. 2017) for the critic’s updates. We train all networks for 200 epochs with batches of 64 using the Adam optimizer

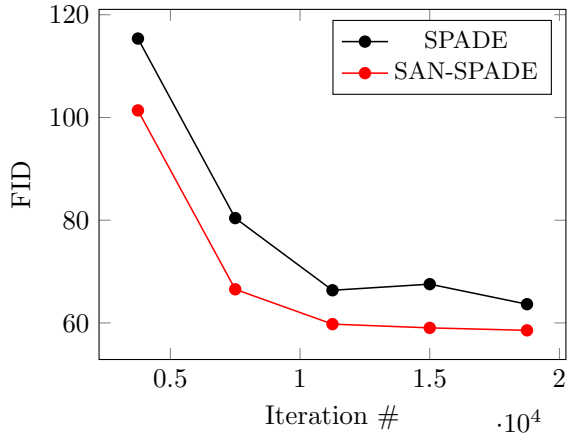


Figure 7: Model convergence in image translation. The FID score of our SAN-SPADE converges faster and to a better result than the original SPADE.

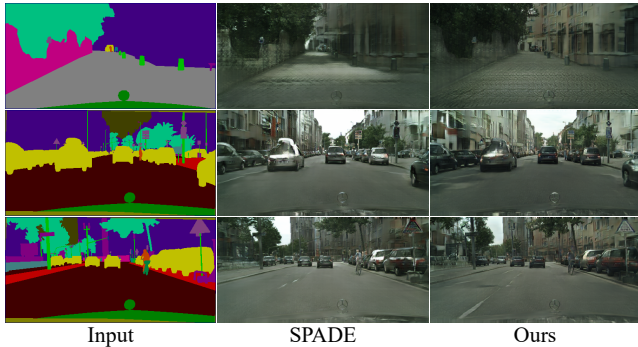


Figure 8: Visual comparison for image translation. The images synthesized by our SAN-SPADE have less artifacts and contain more fine details than the original SPADE.

(Kingma and Ba 2015). We use a learning rate of $2 \cdot 10^{-4}$ and momentum parameters $\beta_1 = 0.5$ and $\beta_2 = 0.9$.

Additionally, we experiment with the more modern BigGAN architecture (Brock, Donahue, and Simonyan 2018) for conditional generation on CIFAR-10. We replace the spectral normalization by our SAN in all critic’s res-blocks, and modify Adam’s first momentum parameter to $\beta_1 = 0.5$.

Table 1 shows comparisons between our approach (SAN-GAN) and other regularization methods in terms of Inception score (Salimans et al. 2016). The competing methods include weight clipping (Arjovsky and Bottou 2017), gradient penalty (WGAN-GP) (Gulrajani et al. 2017), batch norm (Ioffe and Szegedy 2015), layer norm (Ba, Kiros, and Hinton 2016), weight norm (Salimans and Kingma 2016), orthonormal regularization (Brock et al. 2016), and spectral normalization (SN-GAN) (Miyato et al. 2018). As can be seen, our models outperform the others by a large gap. Most notably, SAN-BigGAN performs substantially better than the original BigGAN, and sets a new state-of-the-art in conditional image generation on CIFAR-10.

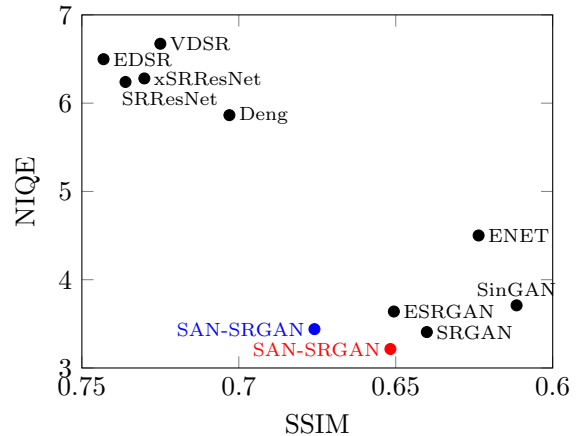


Figure 9: Perception-distortion evaluation for SR. We compare our models ($\lambda_{\text{adversarial}} = 10^{-1}$ and 10^{-2}) to other state-of-the-art super-resolution models in terms of perceptual quality (NIQE, lower is better) and distortion (SSIM). Our method improves upon all existing perceptual methods (those at the bottom right) in both perceptual quality and distortion.

Method	BSD100	URBAN100	DIV2K
SRGAN	25.18 / 3.40		
ENET	24.93 / 4.52	23.54 / 3.79	
ESRGAN	25.31 / 3.64	24.36 / 4.23	28.18 / 3.14
ESRGAN*	25.69 / 3.56	24.36 / 3.96	28.22 / 3.06
Ours (0.1)	25.32 / 3.21	23.86 / 3.70	27.74 / 2.87
Ours (0.01)	26.15 / 3.44	24.85 / 3.83	28.76 / 3.16

Table 2: PSNR/NIQE comparison among different perceptual SR methods on varied datasets. Our models attain a significantly higher PSNR and lower NIQE than other perceptual SR methods.

Image-to-Image Translation

Next, we illustrate our method in the challenging task of translating images between different domains. Here we focus on converting semantic segmentation masks to photo-realistic images. In the Supplementary, we also demonstrate the power of SAN for attribute transfer.

We adopt the state-of-the-art SPADE scheme (Park et al. 2019) as a baseline framework, and enhance its results by applying our normalization. We use the same multi-scale discriminator as (Park et al. 2019), except that we replace the zero padding by circular padding and preform SAN. To reduce the memory footprint, we use $r = 25\%$ of the filters with a compensation factor of $g = 1.3$. All hyper-parameters are kept as in (Park et al. 2019), except for Adam’s first momentum parameter, which we set to $\beta_1 = 0.5$.

We use 512×256 images from the Cityscapes dataset (Cordts et al. 2016). For quantitative evaluation, we use the Fréchet Inception distance (FID). As can be seen in Fig. 7, our method converges faster and leads to a final model that outperforms the original SPADE by a non-negligible mar-

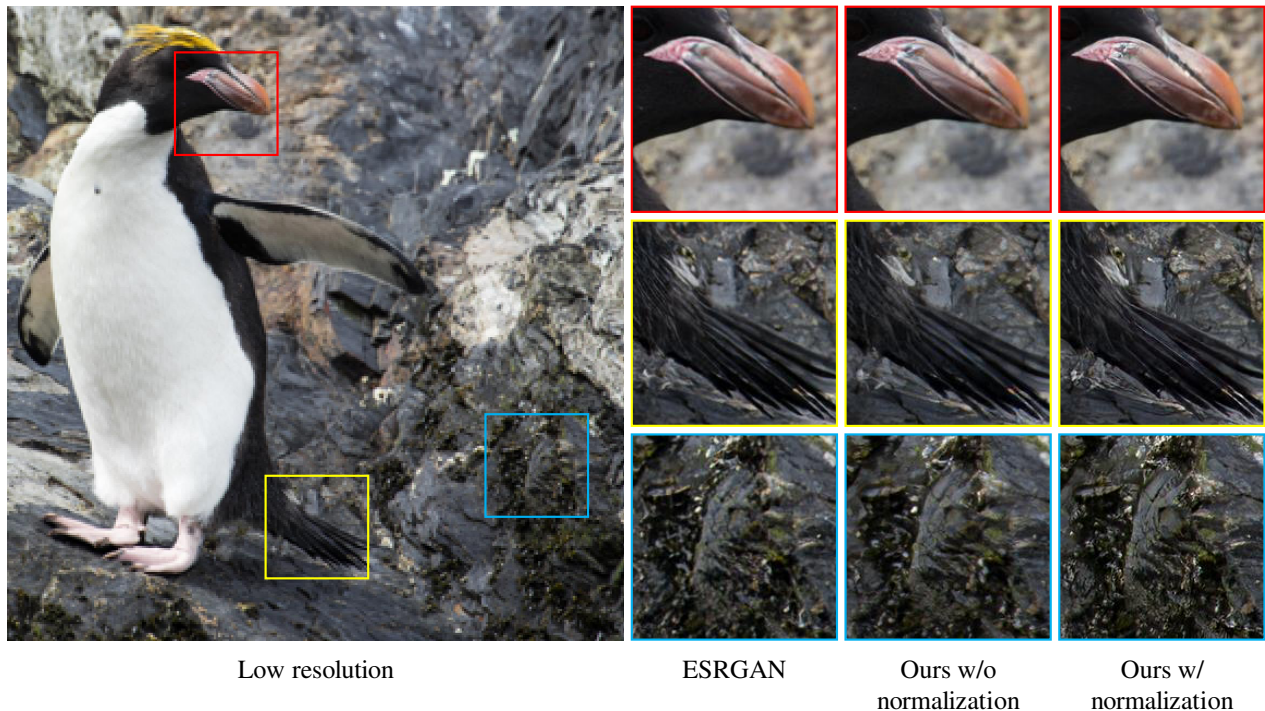


Figure 10: The influence of normalization in super-resolution. We compare the state-of-the-art ESRGAN method to our approach, with and without normalization, at a magnification factor of $4\times$. As can be seen, our normalization leads to sharper and more photo-realistic images.

gin. Specifically, SAN-SPADE achieves an FID of 58.56 while the original SPADE achieves 63.65. Figure 8 shows a qualitative comparison between SPADE and our SAN version after 1.1×10^4 iterations. As can be seen, our synthesized images have less artifacts and contain more details.

Single Image Super Resolution

Finally, we illustrate SAN in single image super resolution (SR), where the goal is to restore a high resolution image from its down-sampled low resolution version. We focus on $4\times$ SR for images down-sampled with a bicubic kernel.

Following the state-of-the-art ESRGAN (Wang et al. 2018) method, our loss function comprises three terms,

$$\mathcal{L} = \lambda_{\text{content}} \cdot \mathcal{L}_{\text{content}} + \mathcal{L}_{\text{features}} + \lambda_{\text{adversarial}} \cdot \mathcal{L}_{\text{adversarial}}. \quad (10)$$

Here, $\mathcal{L}_{\text{content}}$ is the L_1 distance between the reconstructed high-res image \hat{x} and the ground truth image x . The term $\mathcal{L}_{\text{features}}$ measures the distance between the deep features of \hat{x} and x , taken from 4th convolution layer (before the 5th max-pooling) of a pre-trained 19-layer VGG network (Simonyan and Zisserman 2014). Lastly, $\mathcal{L}_{\text{adversarial}}$ is an adversarial loss that encourages the restored images to follow the statistics of natural images. Here, we use again the hinge loss.

For the generator, we use the much slimmer SRGAN network (Ledig et al. 2017), so that our model has only 9% the number of parameters of ESRGAN (1.5M for ours and 16.7M for ESRGAN). As suggested in (Lim et al. 2017), we remove the batch normalization layers from the generator. For the critic network, we choose a simple feed forward

CNN architecture with 10 convolutional layers and 2 fully connected ones (see architectures in the Supplementary).

We train our network using the 800 DIV2K training images (Agustsson and Timofte 2017), enriched by random cropping and horizontal flipping. The generator’s weights are initialized to those of a pre-trained model optimized to minimize mean squared error. We minimize the loss (10) with $\lambda_{\text{content}} = 10^{-2}$, and for the adversarial term, $\lambda_{\text{adversarial}}$ we examine two options of 10^{-1} and 10^{-2} . We use the Adam optimizer (Kingma and Ba 2015) with momentum parameters set to 0.5 and 0.9, as in Section . We use a batch size of 32 for 400K equal discriminator and generator updates. The learning rate is initialized to $2 \cdot 10^{-4}$ and is decreased by a factor of 2 at 12.5%, 25%, 50% and 75% of the total number of iterations.

Following (Blau and Michaeli 2018), we compare our method to other super-resolution schemes in terms of both perceptual quality and distortion. Figure 9 shows a comparison against EDSR (Lim et al. 2017), VDSR (Kim, Kwon Lee, and Mu Lee 2016), SRResNet (Ledig et al. 2017), xSRResNet (Kligvasser, Rott Shaham, and Michaeli 2018), Deng (Deng 2018), ESRGAN (Wang et al. 2018), SRGAN (Ledig et al. 2017), ENET (Sajjadi, Scholkopf, and Hirsch 2017) and SinGAN (Shaham, Dekel, and Michaeli 2019). Here, perceptual quality is quantified using the no-reference NIQE metric (Mittal, Soundararajan, and Bovik 2012) (lower is better), which has been found in (Blau et al. 2018) to correlate well with human opinion scores in this

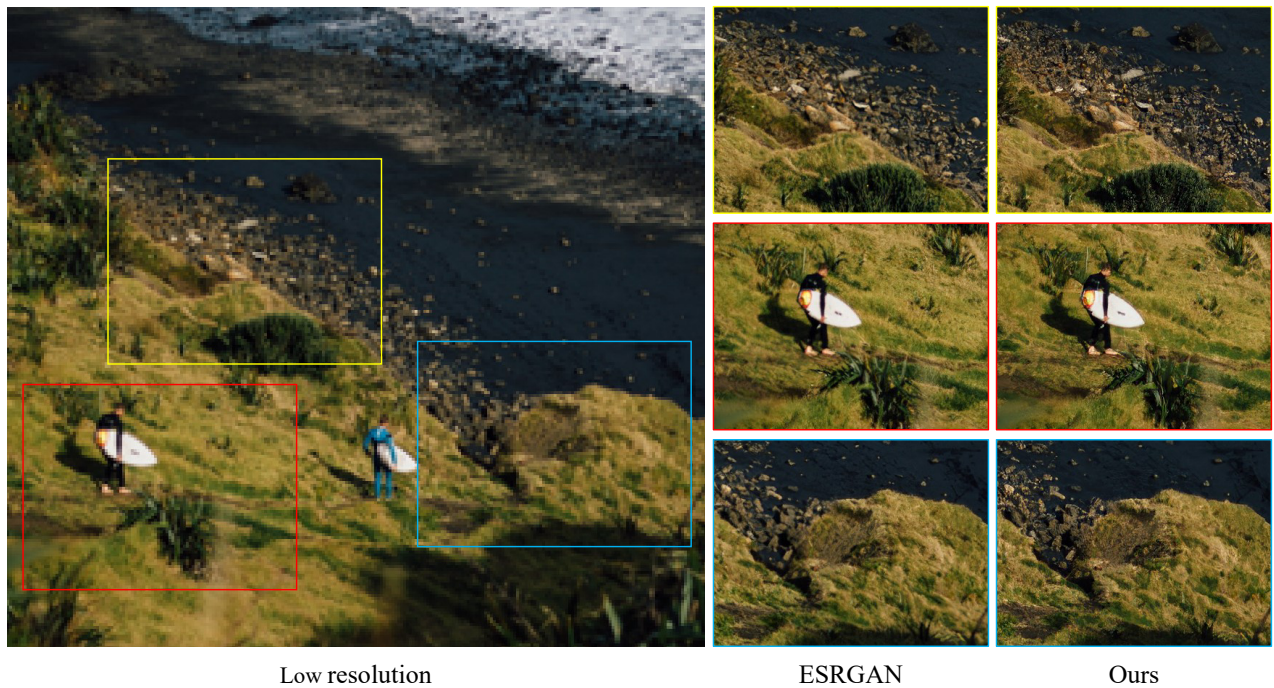


Figure 11: Further super-resolution comparisons. Compared to ESRGAN, our method better recovers textures, like grass and stones.

task. Distortion is measured by SSIM (Wang et al. 2004) (higher is better). We report average scores over the BSD100 test set (Martin et al. 2001). As can be seen, our method achieves the best perceptual quality, and lower distortion levels than the perceptual methods (at the bottom right).

In Table 2, we report comparisons to the best perceptual methods on two more datasets, the URBAN100 and DIV2K test sets. As the original ESRGAN (Wang et al. 2018) uses gradient penalty as a normalization scheme, for a fair comparison, we train an equivalent version, ESRGAN*, with spectral normalization (Miyato et al. 2018). Note that our model outperforms ESRGAN (winner of the PRIM challenge on perceptual image super-resolution (Blau et al. 2018)) as well as the improved variant ESRGAN*. This is despite the fact that our generator network has only 9% the number of parameters of ESRGAN’s generator. Furthermore, while our model has the same generator architecture as SRGAN, it outperforms it by 1dB in PSNR without any sacrifice in perceptual score.

Figures 1 and 11 shows a visual comparison with ESRGAN. As can be seen, our method manages to restore more of the fine image details, and produces more realistic textures. Figure 10 shows yet another visual result, where we specifically illustrate the effect of our normalization. While without normalization our method is slightly inferior to ESRGAN, when we incorporate our normalization, the visual quality is significantly improved.

Limitations

SAN does not provide a boost in performance when the critic’s feature maps do not exhibit strong channel-sparsity. This happens, for example, in BigGAN for 128×128 images (see Supplementary). There, there is one set of features in each res-block that are less sparse (those after the residual connection). A possible solution could be to use a different compensation factor g for different layers, according to their level or sparsity. However, we leave this for future work.

Conclusion

We presented a new per-layer normalization method for GANs, which explicitly accounts for the statistics of signals that enter each layer. We showed that this approach stabilizes the training and leads to improved results over other GAN schemes. Our normalization adds a marginal computational burden compared to the forward and backward passes, and can even be applied once every several hundred steps while still providing a significant benefit.

Acknowledgements

This research was supported by the Israel Science Foundation (grant 852/17) and by the Technion Ollendorff Minerva Center.

References

Agustsson, E.; and Timofte, R. 2017. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study.

In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Arjovsky, M.; and Bottou, L. 2017. Towards principled methods for training generative adversarial networks. *ArXiv preprint arXiv: 1701.04862* .

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein Generative Adversarial Networks. In *International Conference on Machine Learning*, 214–223.

Armanious, K.; Jiang, C.; Fischer, M.; Küstner, T.; Hepp, T.; Nikolaou, K.; Gatidis, S.; and Yang, B. 2020. MedGAN: Medical image translation using GANs. *Computerized Medical Imaging and Graphics* 79: 101684.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* .

Bahat, Y.; and Michaeli, T. 2019. Explorable Super Resolution. *arXiv preprint arXiv:1912.01839* .

Blau, Y.; Mechrez, R.; Timofte, R.; Michaeli, T.; and Zelnik-Manor, L. 2018. The 2018 PIRM challenge on perceptual image super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 0–0.

Blau, Y.; and Michaeli, T. 2018. The Perception-Distortion Tradeoff. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Brock, A.; Donahue, J.; and Simonyan, K. 2018. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations*.

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

Choi, Y.; Choi, M.; Kim, M.; Ha, J.-W.; Kim, S.; and Choo, J. 2018. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8789–8797.

Coates, A.; Ng, A.; and Lee, H. 2011. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 215–223.

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3213–3223.

Deng, X. 2018. Enhancing image quality via style transfer for single image super-resolution. *IEEE Signal Processing Letters* 25(4): 571–575.

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

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved training of wasserstein gans. In *Advances in neural information processing systems*, 5767–5777.

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

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* .

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1125–1134.

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

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

Kim, J.; Kwon Lee, J.; and Mu Lee, K. 2016. Accurate Image Super-Resolution Using Very Deep Convolutional Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kingma, D.; and Ba, J. 2015. Adam: A method for stochastic optimization. In *The International Conference on Learning Representations (ICLR)*.

Kligvasser, I.; Rott Shaham, T.; and Michaeli, T. 2018. xUnit: Learning a spatial activation function for efficient image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2433–2442.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4681–4690.

Lim, B.; Son, S.; Kim, H.; Nah, S.; and Mu Lee, K. 2017. Enhanced Deep Residual Networks for Single Image Super-Resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Martin, D.; Fowlkes, C.; Tal, D.; and Malik, J. 2001. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, 416–423.

Mittal, A.; Soundararajan, R.; and Bovik, A. C. 2012. Making a “completely blind” image quality analyzer. *IEEE Signal Processing Letters* 20(3): 209–212.

Miyato, T.; Kataoka, T.; Koyama, M.; and Yoshida, Y. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957* .

Miyato, T.; and Koyama, M. 2018. cGANs with projection discriminator. *arXiv preprint arXiv:1802.05637* .

- Park, T.; Liu, M.-Y.; Wang, T.-C.; and Zhu, J.-Y. 2019. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2337–2346.
- Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.
- Sajjadi, M. S. M.; Scholkopf, B.; and Hirsch, M. 2017. EnhanceNet: Single Image Super-Resolution Through Automated Texture Synthesis. In *The IEEE International Conference on Computer Vision (ICCV)*.
- Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X. 2016. Improved techniques for training gans. In *Advances in neural information processing systems*, 2234–2242.
- Salimans, T.; and Kingma, D. P. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, 901–909.
- Sedghi, H.; Gupta, V.; and Long, P. M. 2018. The singular values of convolutional layers. *arXiv preprint arXiv:1805.10408*.
- Shaham, T. R.; Dekel, T.; and Michaeli, T. 2019. SinGAN: Learning a generative model from a single natural image. In *Proceedings of the IEEE International Conference on Computer Vision*, 4570–4580.
- Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Virmaux, A.; and Scaman, K. 2018. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*, 3835–3844. Curran Associates, Inc. URL <http://papers.nips.cc/paper/7640-lipschitz-regularity-of-deep-neural-networks-analysis-and-efficient-estimation.pdf>.
- Wang, R.; Cully, A.; Chang, H. J.; and Demiris, Y. 2017. Magan: Margin adaptation for generative adversarial networks. *arXiv preprint arXiv:1704.03817*.
- Wang, X.; Yu, K.; Wu, S.; Gu, J.; Liu, Y.; Dong, C.; Qiao, Y.; and Change Loy, C. 2018. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 0–0.
- Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13(4): 600–612.
- Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, 2223–2232.