Spectral Normalization for Generative Adversarial Networks

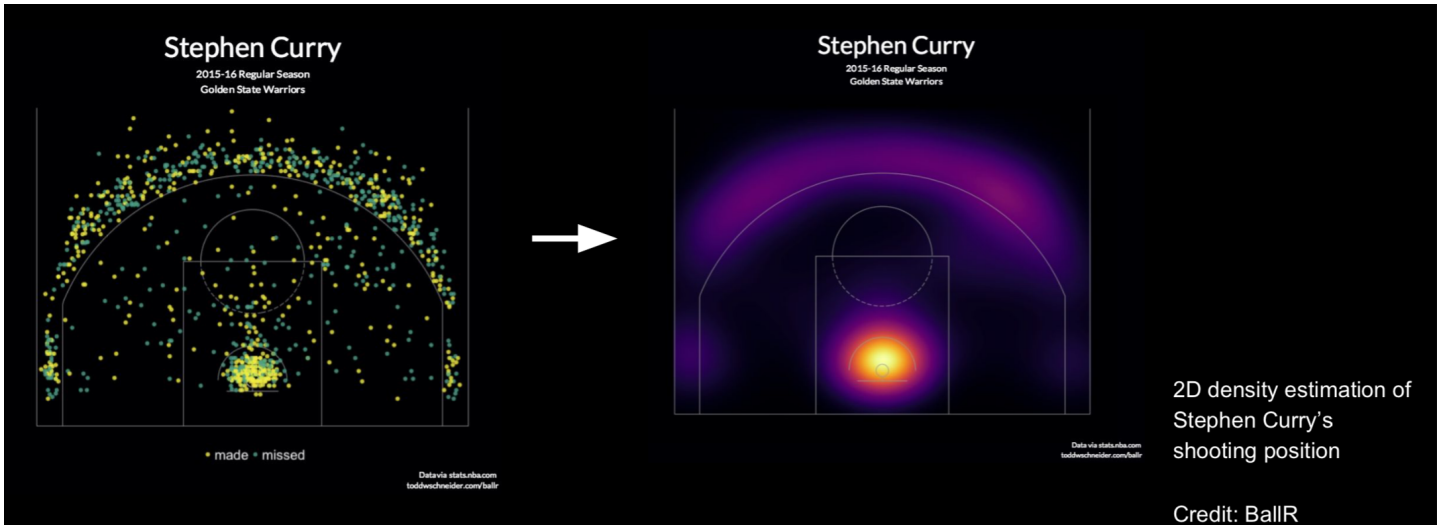
Takeru Miyato, Toshiki Kataoka, Masanori Koyama, Yuichi Yoshida

Presented by: Faruk Ahmed, Alex Zhang

March 29, 2019

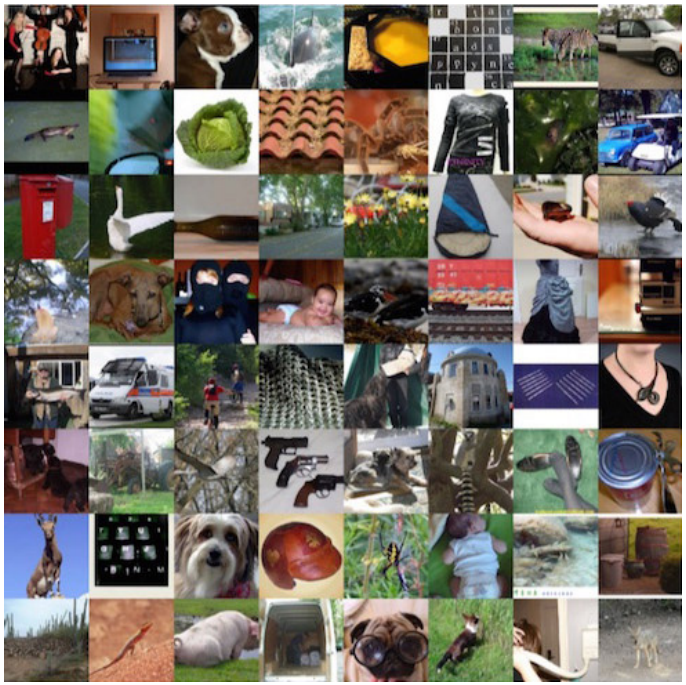
Generative modelling

We want to learn a density model



Generative modelling

Goal: Sample from a parameterized generating model



$x \sim p_D(x)$

$\Rightarrow$

$x \sim p_\theta(x)$

Generative adversarial networks, Goodfellow et al., 2014

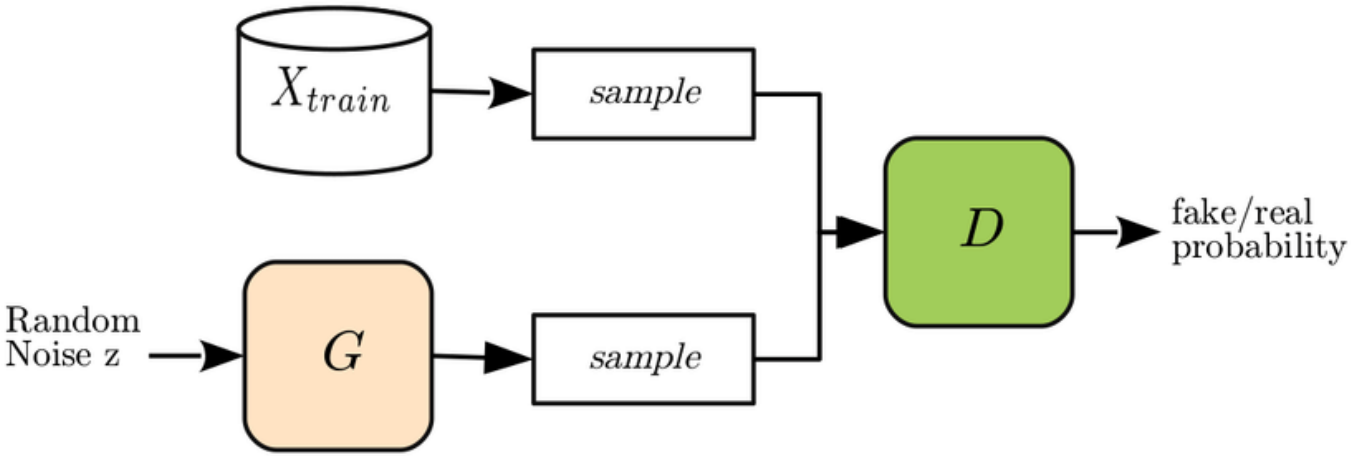


Figure credit: LOGAN, Hayes et al.

GANvolution

Slide credit: Sebastian Nowozin

Generative modelling

Optimization problem

$$\text{argmin}_\theta D(p_D, p_\theta)$$

Choice of D (GAN)

For classical GAN, we can show it is the Jensen-Shannon divergence (for an optimal D)

Generative modelling

Optimization problem

$$\text{argmin}_\theta D(p_D, p_\theta)$$

Choice of D (WGAN)

Wasserstein Distance

Intuition: Minimum effort to move probability mass from p_θ to p_D.

Wasserstein distance

Primal formulation

$$W(p_D, p_\theta) = \inf_{\gamma \in \Pi(p_D, p_\theta)} \mathbb{E}_{(x,y)\sim\gamma} \left[\|x - y\|_2 \right]$$

Kantorovich-Rubinstein Dual

$$W(p_D, p_\theta) = \sup_{\|f\|_L = 1} \left(\mathbb{E}_{x\sim p_D} \left[f(x) \right] - \mathbb{E}_{x\sim p_\theta} \left[f(x) \right] \right)$$

Neural network approximation

Parameterize a generator

$$z \sim p(z), x = g_\theta(z)$$

Estimate the distance $W(p_D, p_\theta)$ by parameterizing f with a neural network with parameters ϕ

$$\max_{\|f\|_L=1} \underbrace{\left(\mathbb{E}_{x\sim p_D}\left[f_\phi(x)\right] - \mathbb{E}_{z\sim p(z)}\left[f_\phi(g(z))\right] \right)}_{\ell}$$

f is a critic, the equivalent of the discriminator in the classical GAN

Observation 1

The Lipschitz constant of f is the max-spectral norm of $\nabla_x f$

$$\|f\|_L = \sup_x \sigma(\nabla_x f(x))$$

Example: For an affine layer

For an affine f, we have

$$\begin{aligned}
\|f\|_L &= \frac{\|f(x+h)-f(x)\|_2}{\|h\|_2} \\
&= \frac{\|W(\cancel{x}+h)+\cancel{b}-\cancel{Wx}-\cancel{b}\|_2}{\|h\|_2} \\
&= \frac{\|Wh\|_2}{\|h\|_2} = \|W\hat{h}\|_2 \quad \text{with} \quad \|\hat{h}\|_2 = 1 \\
&\leq \sup_{\|\hat{h}\|_2=1} \|W\hat{h}\|_2 = \sigma(W)
\end{aligned}$$

Observation 2

$$\|g_1 \circ g_2\|_L \leq \|g_1\|_L \|g_2\|_L$$

Neural network implementation: f is a sequence of linear operations and fixed-slope (≤ 1) piecewise-linear activations:

$$\|f\|_L \leq \prod_{\text{layers}} \sigma(W^{\text{layer}})$$

Learning f

Algorithm 1 Pseudocode for training f

1: **for** each layer l **do**
2: Estimate spectral norm for the current layer, $\sigma(W^l)$
3: Normalize: $\bar{W}^l = W^l / \sigma(W^l)$
4: Update with SGD: $W^l \leftarrow W^l + \alpha \nabla_{W^l} \ell(\bar{W}^l, \text{minibatch})$
5: **end for**

Learning g

Algorithm 2 Pseudocode for training g

1: **while** not converged **do**
2: Train f
3: Update θ for g_θ with SGD: $\theta \leftarrow \theta - \alpha \nabla_\theta W$(minibatch)
4: **end while**

Estimating spectral norm

Power Iteration

Efficient computation of top singular value

Will converge (with high probability)

Full SVD computed by repeated applications on successive *deflations*

Power Iteration: How it works

Start with a random vector x

Apply W on x (and normalize): $y \leftarrow \frac{Wx}{\|Wx\|}$

Apply $W^\top$ on y (and normalize): $x \leftarrow \frac{W^\top y}{\|W^\top y\|}$

Iterate until convergence of x and y

$$y_{final} \approx u_1$$

$$x_{final} \approx v_1$$

$$\sigma(W) \approx y_{final}^\top W x_{final}$$

Power Iteration: Why it works

The sequence of operations amounts to

$$y_{final} = WW^\top \cdots W x_{init}$$

and

$$x_{final} = W^\top WW^\top \cdots W x_{init}$$

We have that

$$\overbrace{WW^\top \cdots W}^{k \text{ terms}} = U\Sigma V^\top V\Sigma U^\top \cdots V\Sigma U^\top U\Sigma V^\top$$

Power Iteration: Why it works

The sequence of operations amounts to

$$y_{final} = WW^\top \cdots W x_{init}$$

and

$$x_{final} = W^\top WW^\top \cdots W x_{init}$$

We have that

$$\overbrace{WW^\top \cdots W}^{k \text{ terms}} = USV^\top \cancelto{I}{VV^\top} \Sigma U^\top \cdots V\Sigma \cancelto{I}{U^\top U} \Sigma V^\top$$

Power Iteration: Why it works

The sequence of operations amounts to

$$y_{final} = WW^\top \cdots W x_{init}$$

and

$$x_{final} = W^\top WW^\top \cdots W x_{init}$$

We have that

$$\overbrace{WW^\top \cdots W}^{k \text{ terms}} = U\Sigma^k V^\top \quad \text{and} \quad W^\top \overbrace{WW^\top \cdots W}^{k \text{ terms}} = V\Sigma^{k+1} V^\top$$

Power Iteration: Why it works

We have that

$$US^kV^\top x = \sum_i \sigma_i^k u_i v_i^\top x = \sigma_1^k \sum_i \left(\frac{\sigma_i}{\sigma_1}\right)^k u_i v_i^\top x$$

and similarly

$$VS^{k+1}V^\top x = \sum_i \sigma_i^{k+1} v_i v_i^\top x = \sigma_1^{k+1} \sum_i \left(\frac{\sigma_i}{\sigma_1}\right)^{k+1} v_i v_i^\top x$$

Power Iteration: Why it works

We have that

$$US^kV^\top x = \sum_i \sigma_i^k u_i v_i^\top x = \sigma_1^k \sum_i \left(\frac{\sigma_i}{\sigma_1}\right)^k u_i v_i^\top x$$

and similarly

$$VS^{k+1}V^\top x = \sum_i \sigma_i^{k+1} v_i v_i^\top x = \sigma_1^{k+1} \sum_i \left(\frac{\sigma_i}{\sigma_1}\right)^{k+1} v_i v_i^\top x$$

Power Iteration: Why it works

As k gets larger,

$$US^kV^\top x \to \sigma_1^k u_1 v_1^\top x$$

$$VS^{k+1}V^\top x \to \sigma_1^{k+1} v_1 v_1^\top x$$

Power Iteration: Why it works

As k gets larger,

$$US^kV^\top x \to \sigma_1^k u_1 v_1^\top x$$

$$VS^{k+1}V^\top x \to \sigma_1^{k+1} v_1 v_1^\top x$$

Power Iteration: Why it works

As k gets larger,

$$y_{final} \approx u_1 \text{ and } x_{final} \approx v_1$$

and so we have the spectral norm estimate:

$$\sigma_1 \approx y_{final}^{\top} W x_{final}$$

In practice

In practice one iteration of power iteration per critic update works

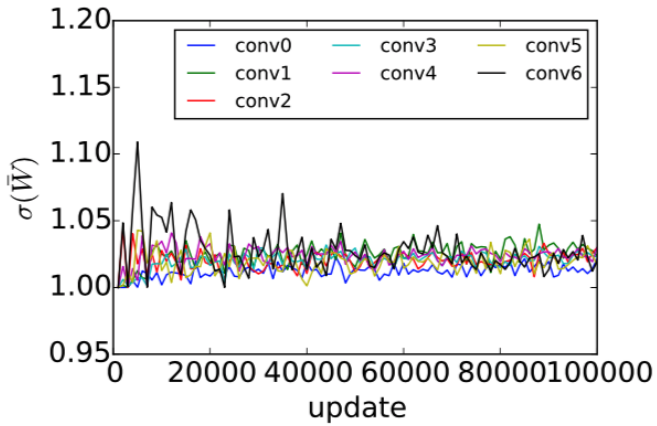
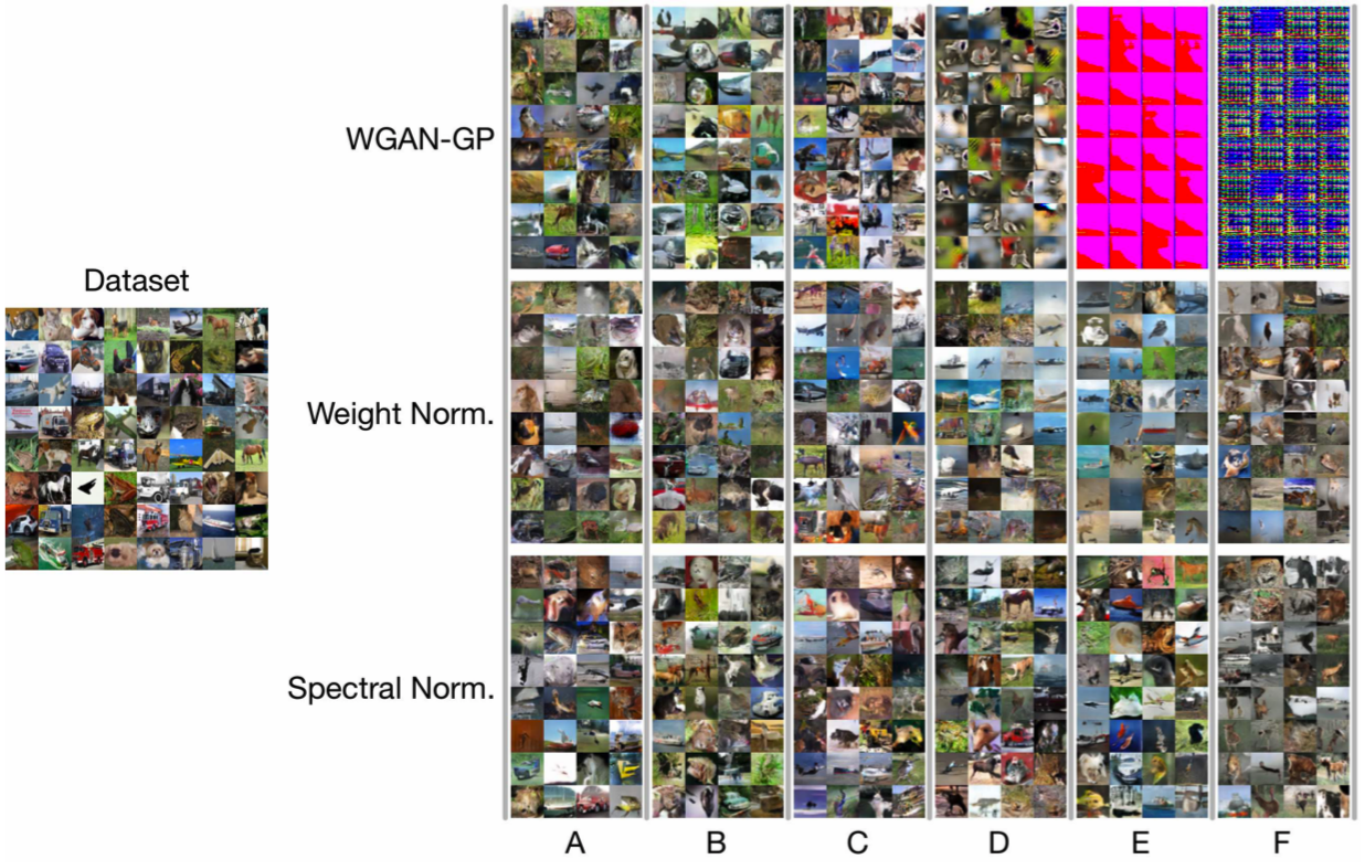


Figure 9: Spectral norms of all seven convolutional layers in the standard CNN during course of the training on CIFAR 10.

Image quality scores

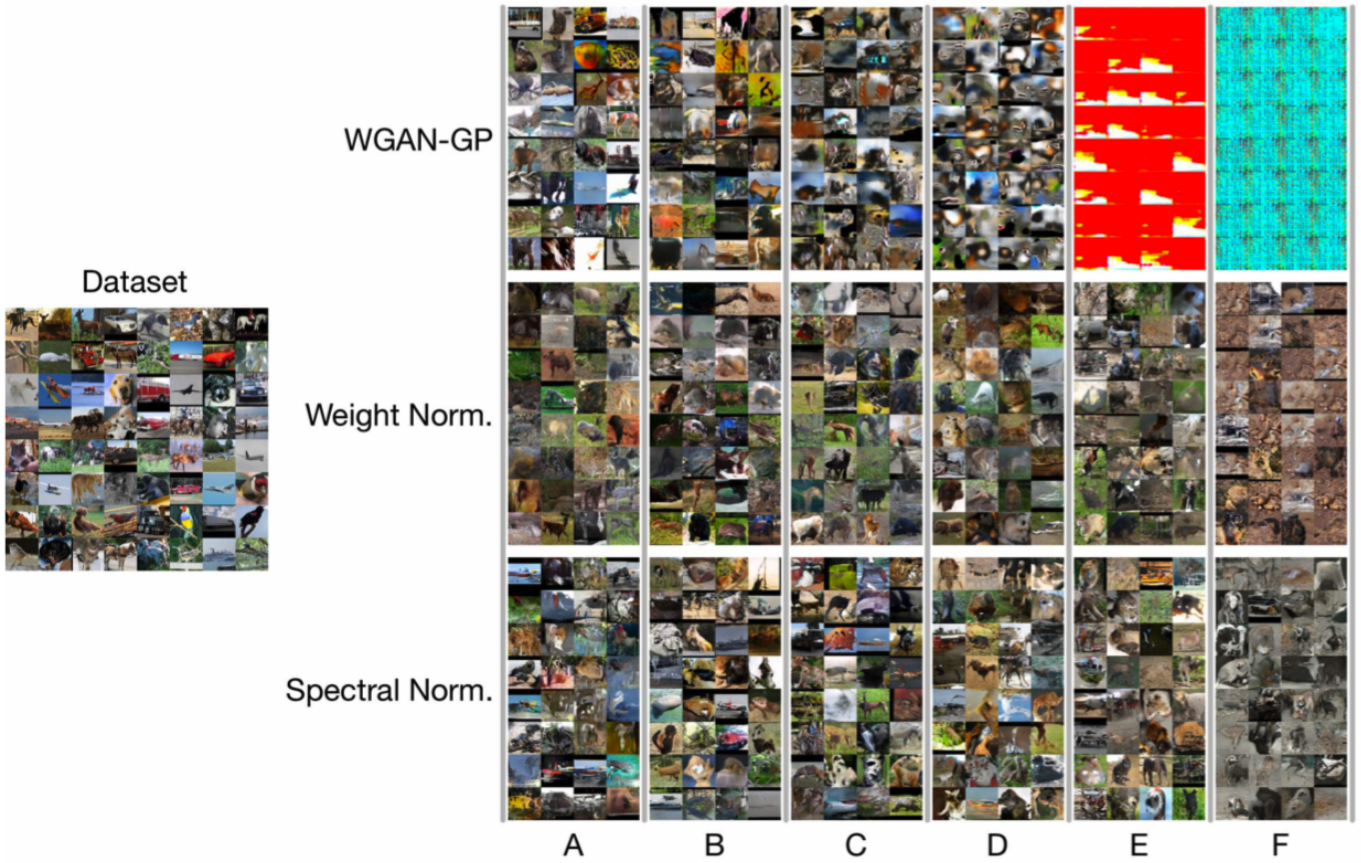
Method	Inception score		FID	
	CIFAR-10	STL-10	CIFAR-10	STL-10
Real data	11.24±.12	26.08±.26	7.8	7.9
-Standard CNN-				
Weight clipping	6.41±.11	7.57±.10	42.6	64.2
GAN-GP	6.93±.08		37.7	
WGAN-GP	6.68±.06	8.42±.13	40.2	55.1
Batch Norm.	6.27±.10		56.3	
Layer Norm.	7.19±.12	7.61±.12	33.9	75.6
Weight Norm.	6.84±.07	7.16±.10	34.7	73.4
Orthonormal	7.40±.12	8.56±.07	29.0	46.7
(ours) SN-GANs	7.42±.08	8.28±.09	29.3	53.1
Orthonormal (2x updates)		8.67±.08		44.2
(ours) SN-GANs (2x updates)		8.69±.09		47.5
(ours) SN-GANs, Eq.(17)	7.58±.12		25.5	
(ours) SN-GANs, Eq.(17) (2x updates)		8.79±.14		43.2
-ResNet-[5]				
Orthonormal, Eq.(17)	7.92±.04	8.72±.06	23.8±.58	42.4±.99
(ours) SN-GANs, Eq.(17)	**8.22**±.05	**9.10**±.04	**21.7**±.21	**40.1**±.50
DCGAN†	6.64±.14	7.84±.07		
LR-GANs‡	7.17±.07			
Warde-Farley et al.*	7.72±.13	8.51±.13		
WGAN-GP (ResNet)††	7.86±.08			

Pretty pictures: CIFAR-10



(a) CIFAR-10

Pretty pictures: STL-10



(b) STL-10

Conclusions

- Provides a "more correct" way to implement WGANs
- Convincing performance
- More diverse samples
- Slow to converge if leading singular value not strongly dominant