Stabilizing GANs: A Case for Spectral Normalization

Liam Donovan[1]

[1]Department of Mathematics and Statistics, University of Massachusetts Amherst

WGANs

▷ **Goal:** Generative Adversarial Networks (GANs) aim to "learn" a target, unknown probability distribution, $Q(x)$ from samples: it consists of

- A function, G called the generator to transform random noise,$z \sim p(z)$, into new "samples", a distribution P_z
- A function D, called the discriminator which aims to distinguish the difference between real samples from Q and generated samples from P_g.
- This process is iterated so each function learns how to better perform its task.

▷ **Problem:** optimizing the discriminator

- If P_g and Q have no overlap (i.e., they have completely disjoint support), then D does not learn
  - ∗ I.e., its gradient goes to 0; the Jensen-Shannon distance is constant and so the gradient descent does not move

  Even if there is a small difference in where they sit, there are similar training issues.

▷ **Key idea:** Instead of relying on the Jensen-Shannon divergence, we switch to a different measure of distance

$$W(P_g, Q) = \inf_{\gamma \in \Pi(P_g, Q)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

This is the Wasserstein distance, which provides informative gradients even when P_g and Q don't overlap.

- It is not easy to minimize this, since we must compute overall all possible couplings.
- **Trick**: Kantorovich–Rubinstein duality tells us:

$$W(P_g, Q) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim Q}[f(x)] - \mathbb{E}_{x\sim P_g}[f(x)]$$

where the supremum is over all 1-Lipschitz functions f (i.e., functions whose output changes no faster than their input; $|\nabla f(x)| \leq 1$).

- ∗ Use the optimal f instead of D. Since f scores rather than classifies, it is called a critic, not a discriminator.

- Even finding this maximum is not easy; we can approximate it using a neural network. This is called Wasserstein GANs (WGANs)

▷ **Takeaway:** All we need now is a way to ensure that our critic is 1-Lipschitz.

Motivation

▷ A naïve approach to keeping f small: keep layer weights small; say $[-.01, .01]$. This is called weight clipping.

- **Pro:** Very easy to implement
- **Downside:** Slow learning rate.

▷ A more involved approach: add a new term in backpropagation (optimization) to force gradient close to 1. This is called gradient penalty.

- **Pros:** More stable than weight clipping
- **Cons:** Have to tune an extra parameter, expensive.

Theory

Recall:

▷ **Linear Layer:** A fully connected layer applies

$$h = Wx + b$$

where $W \in \mathbb{R}^{m\times n}$ is the weight matrix and b is the bias vector.

▷ **Activation Function:** A nonlinearity $\phi(\cdot)$ is applied elementwise after the linear layer. Common examples include:

$$\text{ReLU}(x) = \max(0, x), \quad \text{LeakyReLU}(x) = \max(\alpha x, x)$$

▷ A **Neural Network** is a composition of linear layers and activation functions:

$$f(x) = \phi_L(W_L \cdots \phi_2(W_2\,\phi_1(W_1 x + b_1) + b_2) \cdots + b_L)$$

▷ **Spectral Norm:** The spectral norm of W is its largest singular value:

$$\|W\|_2 \coloneqq \sup_{\|x\|_2=1} \|Wx\|_2 = \sigma_{max}(W)$$

Main Result

▷ **Spectral Normalization ensures f is 1-Lipschitz**

Let $\tilde{W}$ be a **spectrally normalized** weight matrix:

$$\tilde{W} \coloneqq \frac{W}{\|W\|_2} = \frac{W}{\sigma_{\max}(W)}$$

Then the neural network:

$$f(x) = \phi_L(\tilde{W}_L \cdots \phi_2(\tilde{W}_2\,\phi_1(\tilde{W}_1 x + b_1) + b_2) \cdots + b_L)$$

satisfies:

$$\|f\|_L = \sup_x \|\nabla f\| \leq 1$$

if all ϕ_i are Lipschitz-1.

- Common activations ReLU (and its adapted form Leaky-ReLu) are both Lipschitz-1. Some like GeLu, Swish are not Lipschitz-1.

Spectral Normalization as an Implicit Variance-Based Model

▷ Another source of gradient instability is the randomness in the weight initialization. If the weight matrix has large variance, then $\|Wx\|$ can grow rapidly across layers.

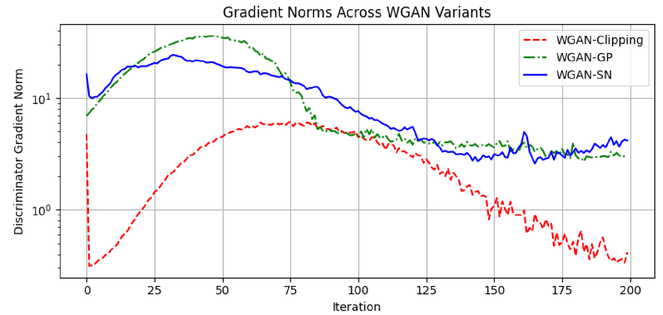
- For example, suppose $x_i \sim \mathcal{N}(0, 1)$ and $W_{ij} \sim \mathcal{N}(0, \sigma^2)$ for an n-dimensional input. Then:

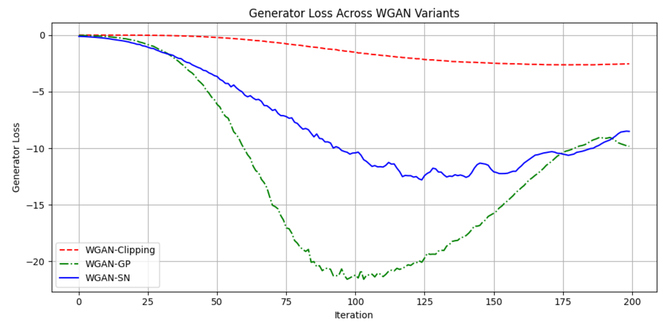
$$\text{Var}((Wx)_i) = n\sigma^2 \quad \Rightarrow \quad \|Wx\| \sim \sqrt{n\sigma^2}$$

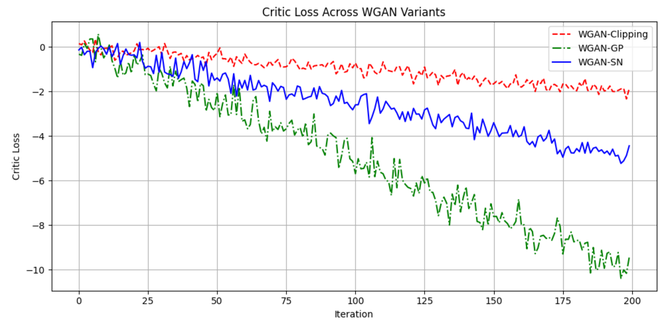
- To keep activations stable, we choose $\sigma^2 \propto \frac{1}{n}$ — this is the principle behind LeCun/Xavier initialization

▷ **Key Insight:** Spectral Normalization indirectly enforces this by bounding $\|W\|_2$, without assuming any distributional form for x

Training Behavior: SN vs. GP vs. Clipping







▷ Weight clipping results in small gradients by 200^{th} iteration.

▷ GP has quickly decreasing critic loss; chance of overfitting

▷ SN is a "middle ground"

References

▷ Lin, Z., Sekar, V., Fanti, G. (2021). *Why Spectral Normalization Stabilizes GANs: Analysis and Improvements.* arXiv:2105.10984.

▷ Plots produced with aid from ChatGPT