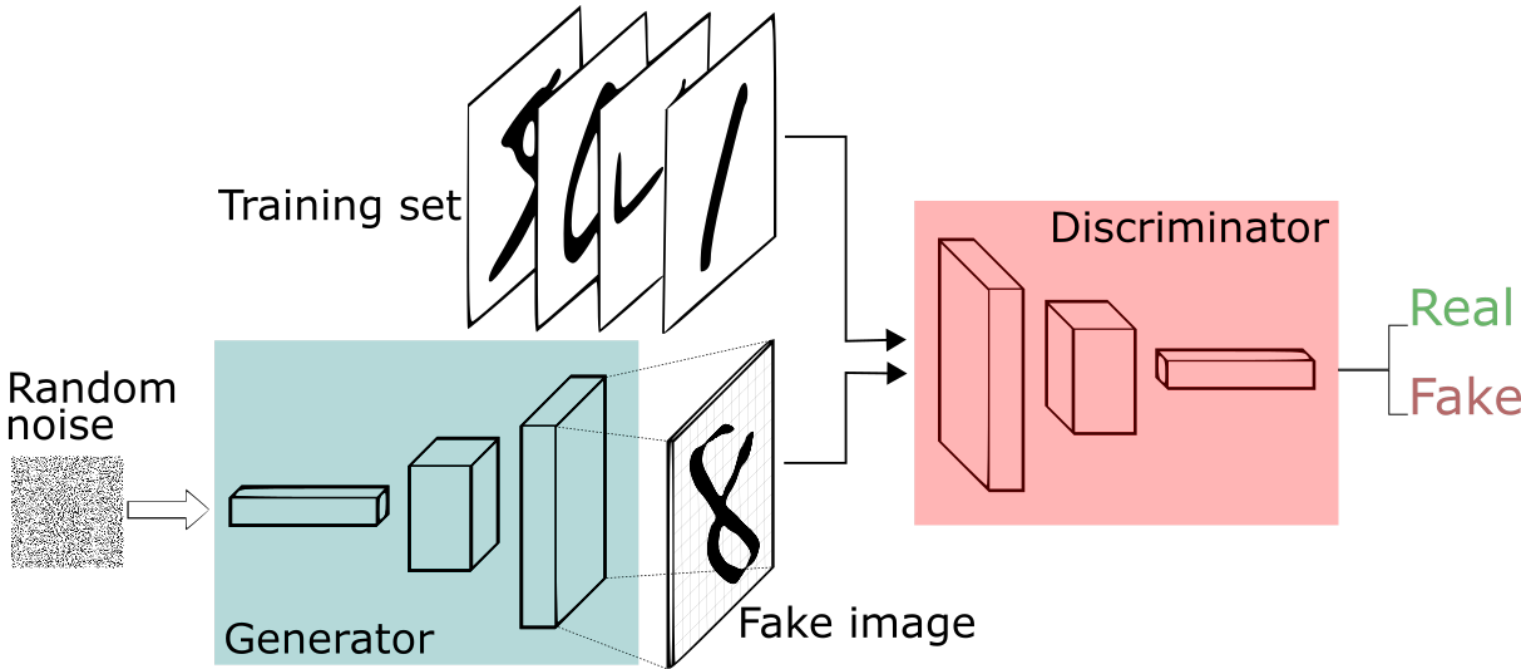
Spectral Normalization for GAN

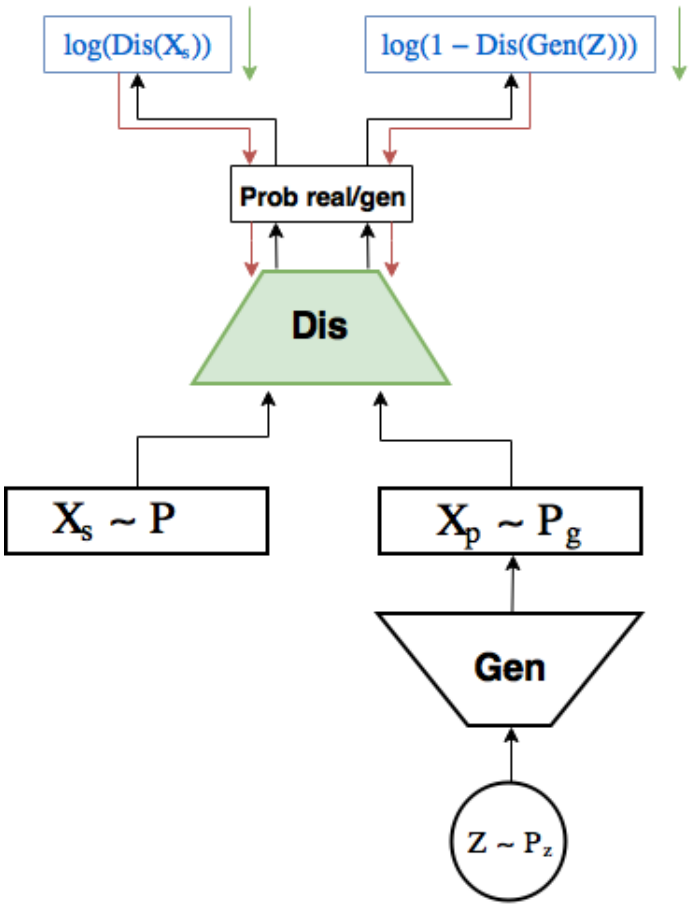
Выступающий: А.В.Соловьев

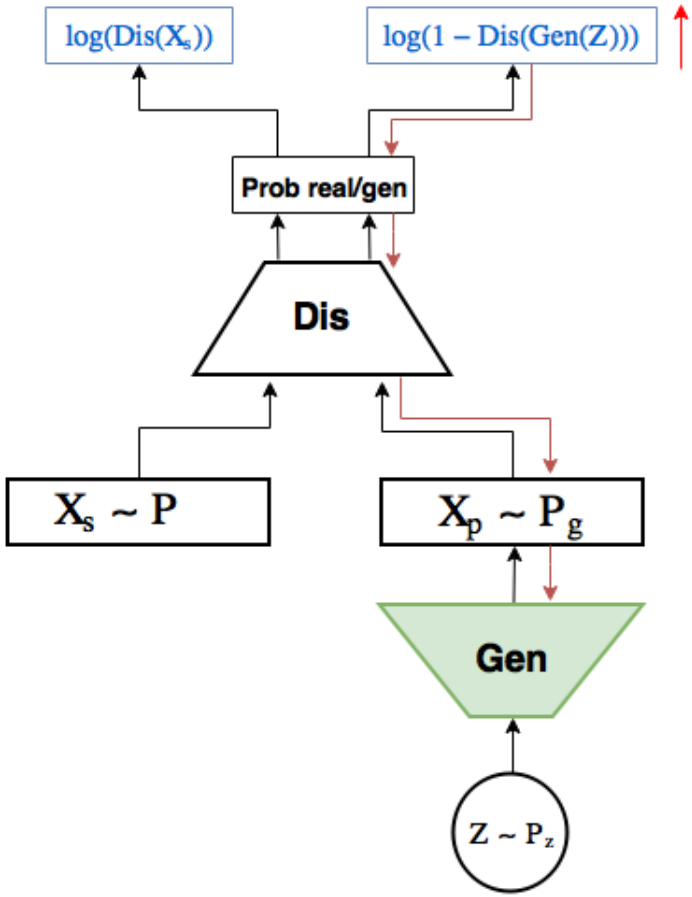
НИУ ВШЭ

GAN


Training set
Discriminator
Real
Fake
Random noise
Generator
Fake image

GAN


log(Dis(X_s))
log(1 – Dis(Gen(Z)))
Prob real/gen
Dis
X_s ~ P
X_p ~ P_g
Gen
Z ~ P_z


log(Dis(X_s))
log(1 – Dis(Gen(Z)))
Prob real/gen
Dis
X_s ~ P
X_p ~ P_g
Gen
Z ~ P_z

$$V(G, D) = E_{x \sim q_{data}}[\log D(x)] + E_{x' \sim p_G}[\log(1 - D(x'))]$$

$$\min_G \max_D V(G, D)$$

q_{data}-data distribution
p_G-generator distribution.

Дискриминатор

$$D(x, \theta) = \mathcal{A}(f(x, \theta)) = \mathcal{A}(W^{L+1} a_L(W^L ... a_1(W^1 x)...))$$

$$\theta := \{W^1, ..., W^{L+1}\}, W^l \in \mathbb{R}^{d_l \times d_{l-1}}, d_{L+1} = 1$$

При фиксированном генераторе:

$$D_G^*(x) := \frac{q_{data}(x)}{q_{data}(x) + p_G(x)}, f^*(x) = \log \frac{q_{data}(x)}{p_G(x)}$$

Какие проблемы?

$$\nabla_x f^*(x) = \frac{1}{q_{data}(x)} \nabla_x q_{data}(x) - \frac{1}{p_G(x)} \nabla_x p_G(x)$$

derivative can be:

- unbounded
- incomputable

Выход: введем условия регуляризации на функцию $f(X, \theta)$

$$arg \max_{||f||_{Lip} \leq K} V(G, D)$$

Определения

Lipschitz norm:

$$||g||_{Lip} := \sup_{h} \sigma(\nabla g(h))$$

Спектральная норма матрицы:

$$\sigma(A) := \max_{h:h\neq 0} \frac{||Ah||_2}{||h||_2} = \max_{||h||_2 \leq 1} ||Ah||_2 = \sigma_{max}$$

Спектральная нормализация:

$$\bar{W}_{SN}(W) := \frac{W}{\sigma(W)}$$

Контроль константы Липшица

$$||g||_{Lip} = \sigma(W), ||a||_{Lip} = 1, \sigma(\bar{W}_{SN}(W)) = 1$$

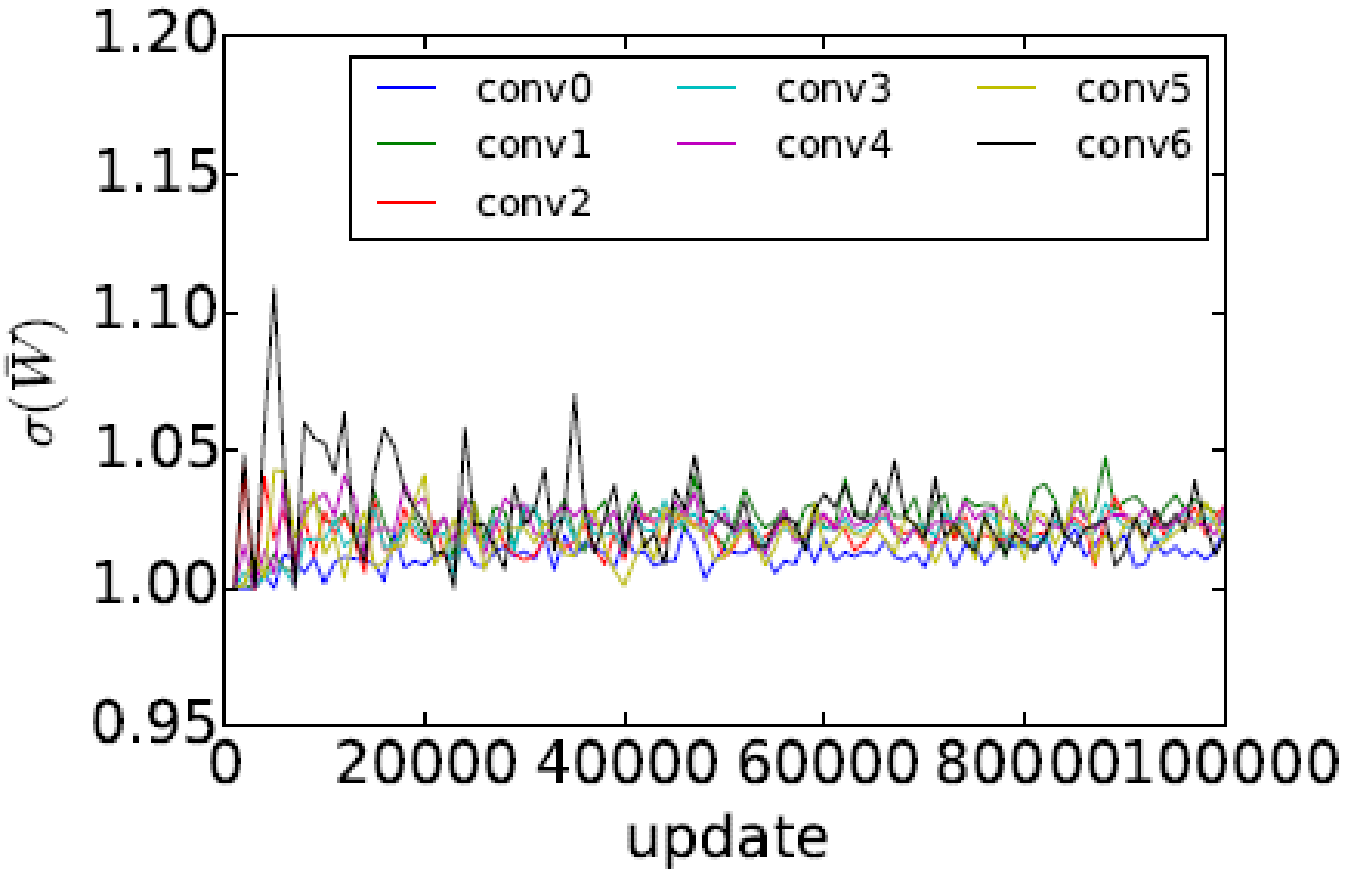
$$||g_1 \circ g_2||_{Lip} \leq ||g_1||_{Lip} \cdot ||g_2||_{Lip}$$

$$||f||_{Lip} \leq \prod_{l=1}^{L+1} ||(h_{l-1} \mapsto W^l h_{l-1})||_{Lip} = \prod_{l=1}^{L+1} \sigma(W^l)$$

После спектральной нормализации:

$$||f||_{Lip} \leq 1$$

Спектральная норма каждого слоя



Ускоряем подсчет сингулярного числа

Algorithm 1 SGD with spectral normalization

- Initialize $\tilde{u}_l \in \mathcal{R}^{d_l}$ for $l = 1, \ldots, L$ with a random vector (sampled from isotropic distribution).
- For each update and each layer l:
  1. Apply power iteration method to a unnormalized weight W^l:

$$\tilde{v}_l \leftarrow (W^l)^{\mathrm{T}} \tilde{u}_l / \|(W^l)^{\mathrm{T}} \tilde{u}_l\|_2 \tag{20}$$

$$\tilde{u}_l \leftarrow W^l \tilde{v}_l / \|W^l \tilde{v}_l\|_2 \tag{21}$$

  2. Calculate $\bar{W}_{\mathrm{SN}}$ with the spectral norm:

$$\bar{W}^l_{\mathrm{SN}}(W^l) = W^l / \sigma(W^l), \text{ where } \sigma(W^l) = \tilde{u}_l^{\mathrm{T}} W^l \tilde{v}_l \tag{22}$$

  3. Update W^l with SGD on mini-batch dataset $\mathcal{D}_M$ with a learning rate α:

$$W^l \leftarrow W^l - \alpha \nabla_{W^l} \ell(\bar{W}^l_{\mathrm{SN}}(W^l), \mathcal{D}_M) \tag{23}$$

Обучение

$$\frac{\partial \bar{W}_{SN}(W)}{\partial W_{ij}} = \frac{E_{ij}}{\sigma(W)} - \frac{W \partial \sigma(W)}{\sigma(W)^2 \partial W_{ij}} = \frac{E_{ij} - [u_1 v_1^T]_{ij} \bar{W}_{SN}}{\sigma(W)}$$

E_{ij} - матрица с 1 на (i, j) месте

u_1 - левый сингулярный вектор W

v_1 - правый сингулярный вектор W

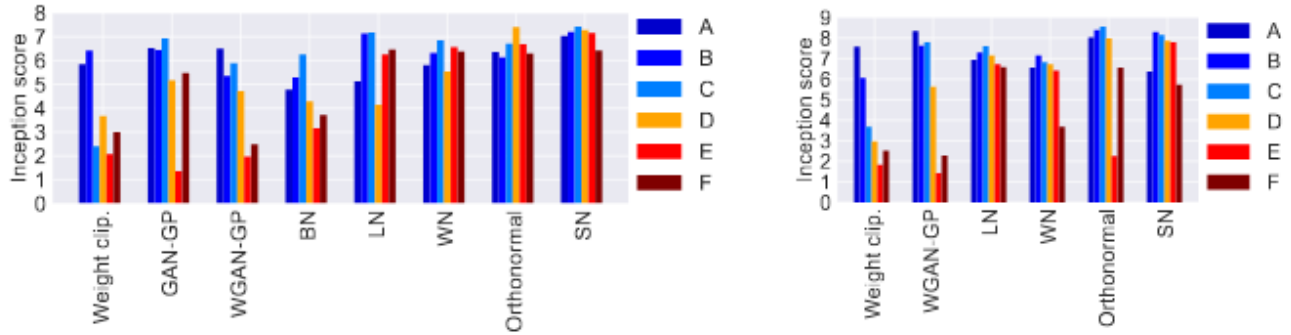
Особенности

В данном методе нормализации:

- Константа Липшица является единственным гиперпараметром.
- Простая реализация.
- Маленькие дополнительные вычислительные затраты.

Примеры работы

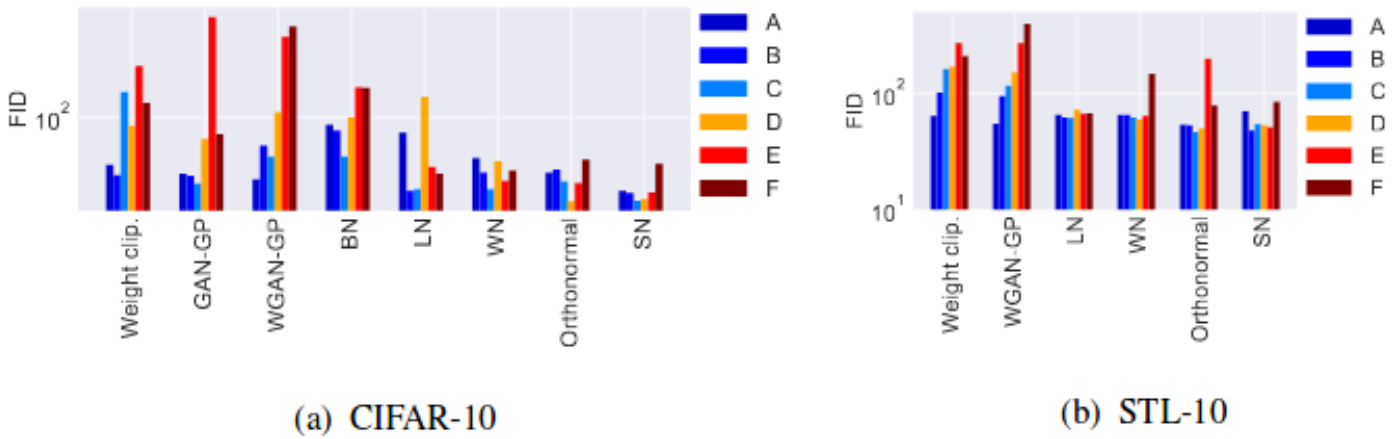
Сравнение с другими методами



(a) CIFAR-10 (b) STL-10

Figure 1: Inception scores on CIFAR-10 and STL-10 with different methods and hyperparameters (higher is better).

Сравнение с другими методами



(a) CIFAR-10

(b) STL-10

Figure 2: FIDs on CIFAR-10 and STL-10 with different methods and hyperparameters (lower is better).

Архитектуры

Table 3: Standard CNN models for CIFAR-10 and STL-10 used in our experiments on image Generation. The slopes of all lReLU functions in the networks are set to 0.1.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
dense $\rightarrow M_g \times M_g \times 512$
4×4, stride=2 deconv. BN 256 ReLU
4×4, stride=2 deconv. BN 128 ReLU
4×4, stride=2 deconv. BN 64 ReLU
3×3, stride=1 conv. 3 Tanh

(a) Generator, $M_g = 4$ for SVHN and CIFAR10, and $M_g = 6$ for STL-10

RGB image $x \in \mathbb{R}^{M \times M \times 3}$
3×3, stride=1 conv 64 lReLU 4×4, stride=2 conv 64 lReLU
3×3, stride=1 conv 128 lReLU 4×4, stride=2 conv 128 lReLU
3×3, stride=1 conv 256 lReLU 4×4, stride=2 conv 256 lReLU
3×3, stride=1 conv. 512 lReLU
dense $\rightarrow$ 1

(b) Discriminator, $M = 32$ for SVHN and CIFAR10, and $M = 48$ for STL-10

Архитектуры

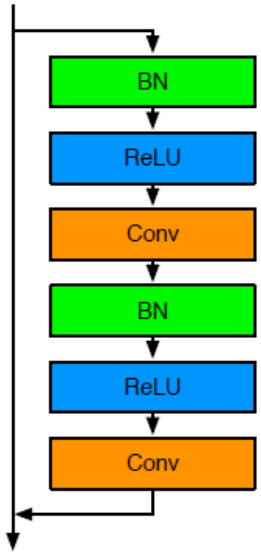


Figure 8: ResBlock architecture. For the discriminator we removed BN layers in ResBlock.

Table 4: ResNet architectures for CIFAR10 dataset. We use similar architectures to the ones used in Gulrajani et al. (2017).

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
dense, $4 \times 4 \times 256$
ResBlock up 256
ResBlock up 256
ResBlock up 256
BN, ReLU, 3×3 conv, 3 Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{32 \times 32 \times 3}$
ResBlock down 128
ResBlock down 128
ResBlock 128
ResBlock 128
ReLU
Global sum pooling
dense $\rightarrow$ 1

(b) Discriminator

Архитектуры

Table 5: ResNet architectures for STL-10 dataset.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
dense, $6 \times 6 \times 512$
ResBlock up 256
ResBlock up 128
ResBlock up 64
BN, ReLU, 3×3 conv, 3 Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{48\times48\times3}$
ResBlock down 64
ResBlock down 128
ResBlock down 256
ResBlock down 512
ResBlock 1024
ReLU
Global sum pooling
dense $\rightarrow$ 1

(b) Discriminator

Литература

- Takeru Miyato, Toshiki Kataoka, Masanori Koyama, Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks
- Generated images and pretrained models
- Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio.Generative Adversarial Networks