WGAN

Towards Principled Methods for Training Generative Adversarial Networks

ICLR 2017 (arXiv 2017.1.17)

Wasserstein GAN (arXiv 2017.5.9)

Improved Training of Wasserstein GANs (arXiv 2017.5.29)

2017.7.24

01 Problem

Despite their success, there is little to no theory explaining the unstable behavior of GAN training.

$$z \sim p(z) \xrightarrow{\quad g_\theta(z) \quad} P_g \longrightarrow P_r$$

Always, g_θ is a neural network parameterized by θ , and the main difference is how g_θ is trained.

MLE ⟷ min(KL divergence)

$$KL(P_r||P_g) = \int P_r log \frac{P_r}{P_g}$$

$$JS(P_r||P_g) = \frac{1}{2}KL(P_r||P_A) + \frac{1}{2}KL(P_g||P_A)$$

$$P_A = \frac{P_r + P_g}{2}$$

02 Problem

The reason of GANs success at producing reallistically looking images is due to the switch from the traditional maximum likelihood approaches.

$$L(D, g_\theta) = E_{x\sim P_r}[logD(x)] + E_{x\sim P_g}[\log(1 - D(x))] \longrightarrow D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)}$$

$$L(D^*, g_\theta) = 2JS(P_r||P_g) - 2log2$$

In practice, as the discriminator gets better, the updates to the generator get consistently worse.

The original GAN paper argued that this issue arose from saturation, and switched to another similar cost function that doesn't have this problem.

03 Problem

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

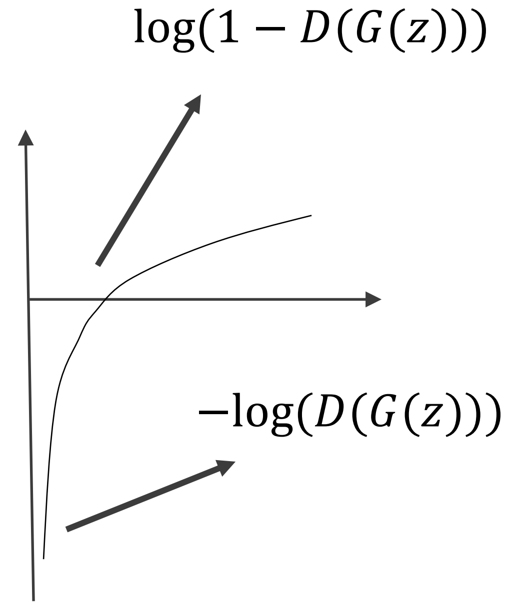
  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

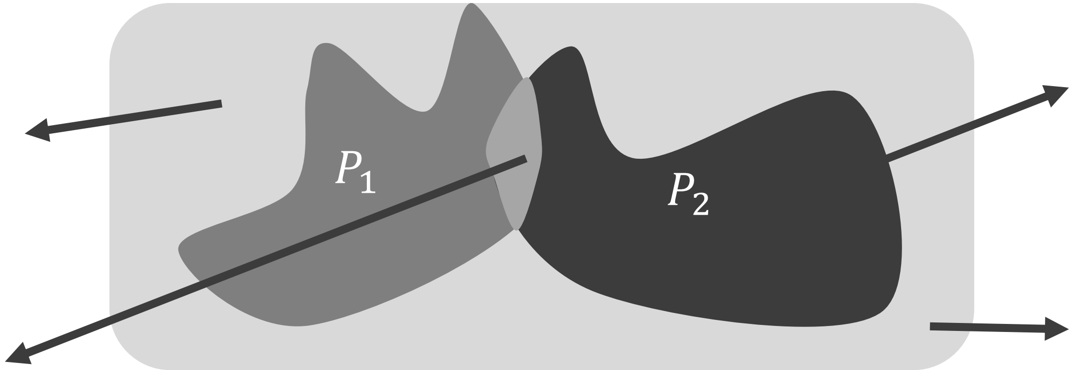


04 Problem

However, even with this new cost function, updates tend to get worse and optimization gets massively unstable.

$P_1(x) = 0 \& P_2(x) = 0$

$P_1(x) \neq 0 \ \& P_2(x) \neq 0$



$P_1(x) = 0 \& P_2(x) \neq 0$

$P_1(x) \neq 0 \& P_2(x) = 0$

$$\mathrm{JS} = \log \frac{P_2}{\frac{1}{2}(P_2 + 0)} = \log 2$$

05 Problem

The only way this can happen is if the distributions are not continuous, or they have disjoint supports.

Their supports lie on low dimensional manifolds.

1. support
2. manifold
3. measure

Lemma

… …, if the dimension of Z is less than the one of X, g(Z) will be a set of measure 0 in X.

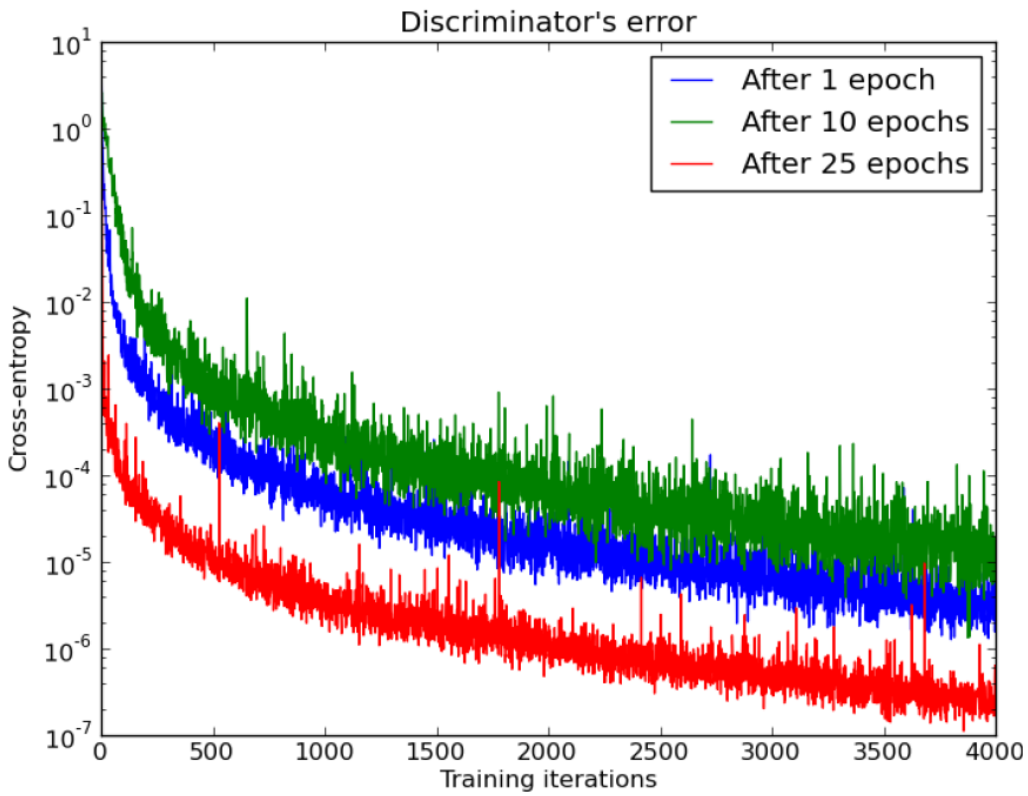
06 Problem

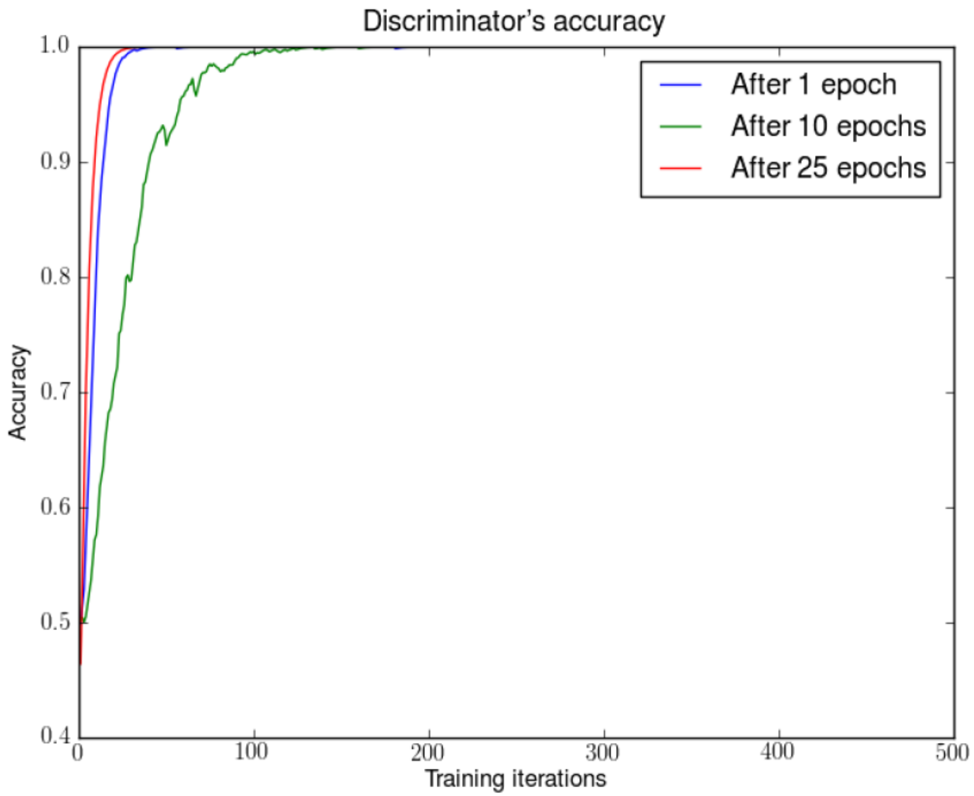
Theorem 1 If two distributions P_r and P_g have support contained on two disjoint compact subsets M and P respectively, then there is a smooth optimal discriminator D^*: that has accuracy 1 and $\nabla_x D^*(x) = 0$ for all $\mathrm{x} \in \mathrm{MUP}$

Theorem 2 Let P_r and P_g be two distributions that have support contained in two closed manifolds and P that don’ t perfectly align and don’ t have full dimension. We further assume that P_r and P_g are continuous in their respective manifolds, meaning that if there is a set A with measure 0 in M, then $P_r(A) = 0$(and analogously for P_g). Then, there exists an optimal discriminator D^* that has accuracy 1 and for almost any x in M or P, D is smooth in a neighborhood of x and $\nabla_x D^*(x) = 0$.

07 Problem

In practice, if we just train D till convergence, its error will go to 0.





08 Problem

Theorem 3 (Vanishing gradients on the generator)

Let $g_\theta: \mathcal{Z} \sim \mathcal{X}$ be a differentiable function that induces a distribution P_g. Let P_r be the real data distribution. Let D be a differentiable discriminator. If the conditions of Theorems 2.1 or 2.2 are satisfied,$\|D - D^*\| < \epsilon$,and $E_{z\sim p(z)}[\|J_\theta g_\theta(z)\|_2^2] \leq M^2, then$

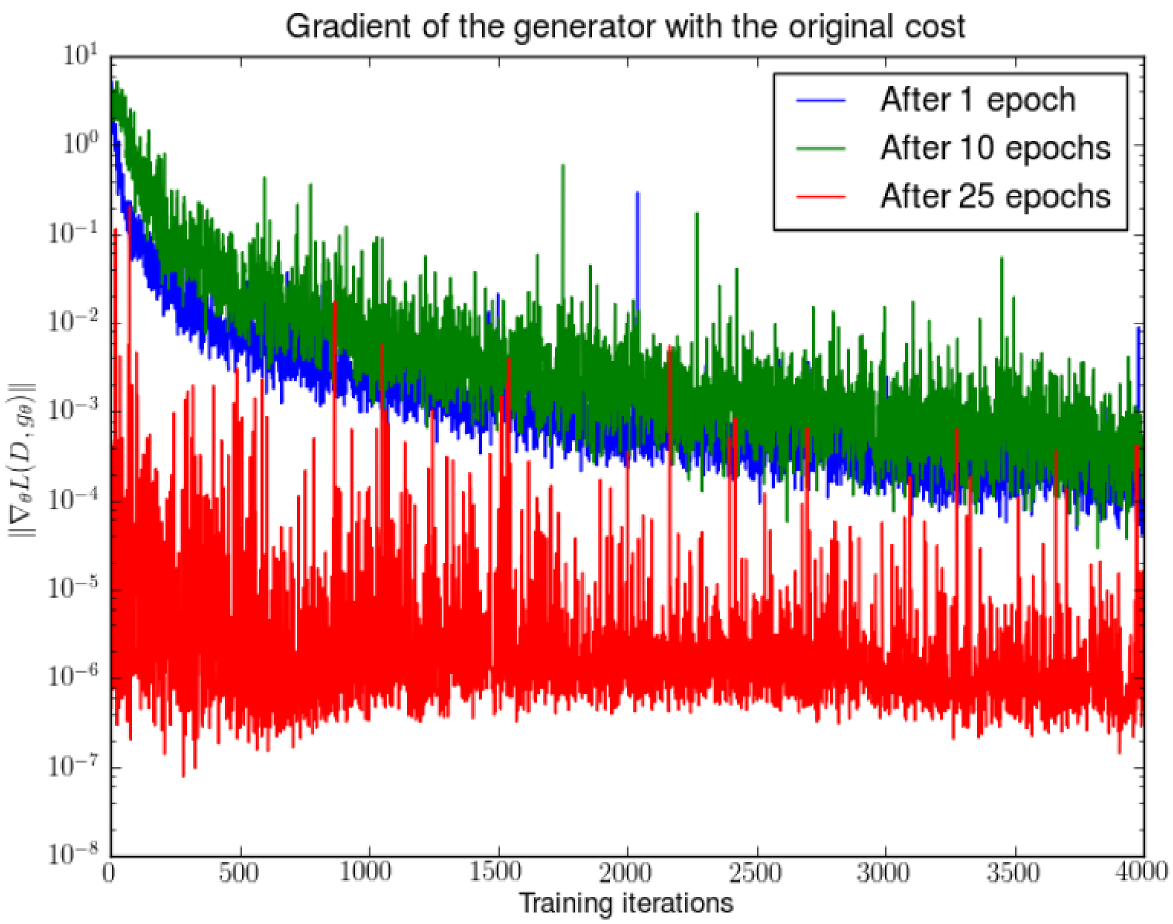
$$\left\|\nabla_\theta E_{z\sim p(z)}[\log(1 - D(g_\theta(z)))]\right\|_2 < M\frac{\epsilon}{1-\epsilon}$$

$$\|D\| = \sup_{x\in\mathcal{X}}|D(x)| + ||\nabla_x D(x)||_2$$

Corollary Under the same assumptions of Theorem 3

$$\lim_{||D-D^*||\to 0} \nabla_\theta E_{z\sim p(z)}[\log(1 - D(g_\theta(z)))] = 0$$

09 Problem



Gradient of the generator with the original cost
After 1 epoch
After 10 epochs
After 25 epochs
$\|\nabla_\theta L(D, g_\theta)\|$
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
0
500
1000
1500
2000
2500
3000
3500
4000
Training iterations

10 -logD

$$\nabla_\theta E_{z\sim p(z)}[-logD(g_\theta(z))]|_{\theta_0}$$

$$KL(P_{g_\theta}||P_r) = E_{x\sim P_{g_\theta}}\left[log\frac{P_{g_\theta}(x)}{P_r(x)}\right] \qquad = E_{x\sim P_{g_\theta}}\left[log\frac{P_{g_{\theta_0}}(x)}{P_r(x)}\right] - E_{x\sim P_{g_\theta}}\left[log\frac{P_{g_\theta}(x)}{P_{g_{\theta_0}}(x)}\right]$$

$$= -E_{x\sim P_{g_\theta}}\left[log\frac{D^*(x)}{1-D^*\ (x)}\right] - KL(P_{g_\theta}||P_{g_{\theta_0}}) \qquad = -E_{z\sim p(z)}\left[log\frac{D^*(g_\theta(z))}{1-D^*\ (g_\theta(z))}\right] - KL(P_{g_\theta}||P_{g_{\theta_0}})$$

$$\nabla_\theta KL(P_{g_\theta}||P_r)|_{\theta=\theta_0} = -\nabla_\theta E_{z\sim p(z)}\left[log\frac{D^*(g_\theta(z))}{1-D^*(g_\theta(z))}\right]|_{\theta=\theta_0} - \nabla_\theta KL(P_{g_\theta}||P_{g_{\theta_0}})$$

$$D^* = \frac{P_r}{P_{g_{\theta_0}+P_r}}$$

11 -logD

$$= \nabla_\theta E_{z\sim p(z)}\left[-log\frac{D^*(g_\theta(z))}{1-D^*(g_\theta(z))}\right]|_{\theta=\theta_0} \qquad E_{z\sim p(z)}\left[\nabla_\theta \log(1-D^*(g_\theta(z)))|_{\theta=\theta_0}\right] = \nabla_\theta 2JS(P_{g_\theta}||P_r)|_{\theta=\theta_0}$$

$$= \nabla_\theta E_{z\sim p(z)}[-logD^*(g_\theta(z))] + \nabla_\theta E_{z\sim p(z)}[\log(1-D^*(g_\theta(z)))]$$

$$E_{z\sim p(z)}\left[-\nabla_\theta logD^*(g_\theta(z))|_{\theta=\theta_0}\right] = \nabla_\theta[KL(P_{g_\theta}||P_r) - 2JS(P_{g_\theta}||P_r)]|_{\theta=\theta_0}$$

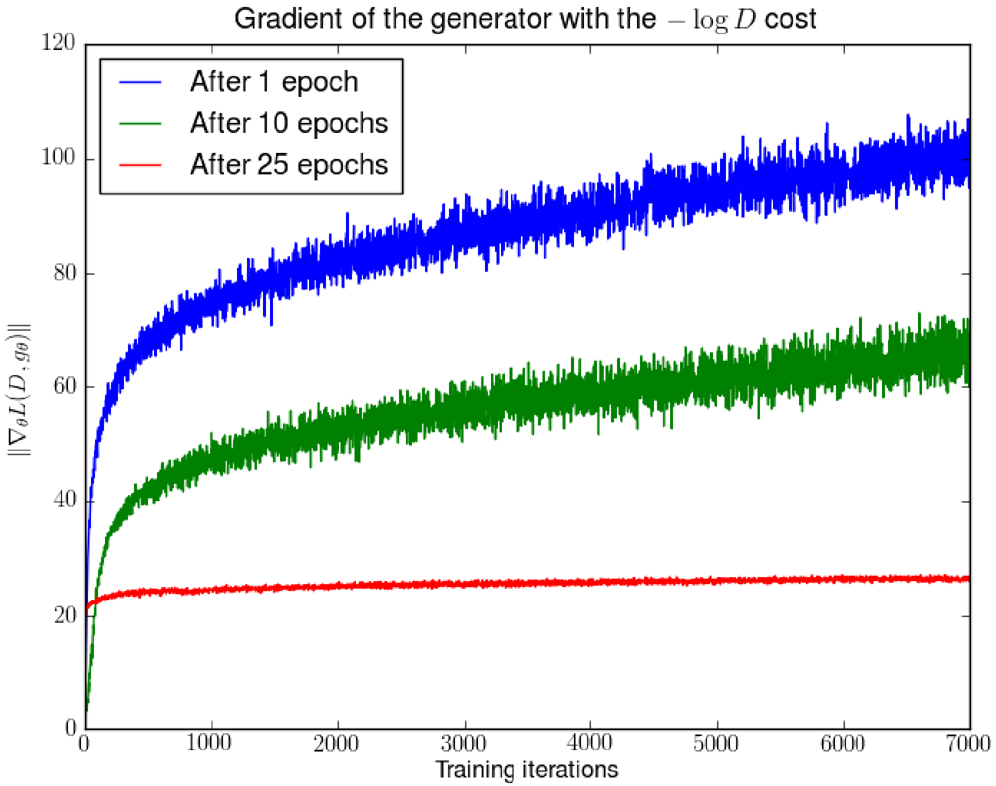
The JSs are in the opposite sign, which means they are pushing for the distributions to be different, which seems like a fault in the update.

12 -logD

The gradient norms grow quickly.

Furthermore, the noise in the curves shows that the variance of the gradients is also increasing.

All these gradients lead to updates that lower sample quality notoriously.



13 -logD

$$E_{z\sim p(z)}\left[-\nabla_\theta \log D^*(g_\theta(z))|_{\theta=\theta_0}\right] = \nabla_\theta[KL(P_{g_\theta}||P_r) - 2JS(P_{g_\theta}||P_r)]|_{\theta=\theta_0}$$

The KL appearing in the equation is $KL(P_g||P_r)$,not the one equivalent to maximum likelihood.

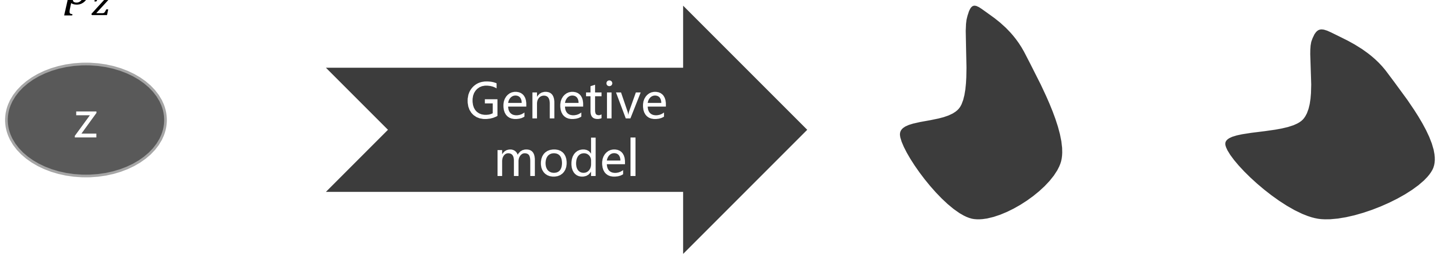
This KL assigns an extremely high cost to generating fake looking samples, and an extremely low cost on mode dropping, JS is symetrical so it shouldn' t alter this behaviour.

$P_g(x) \to 0 \quad P_r \to 1$ $\qquad P_g(x)\log\frac{P_g(x)}{P_r(x)} \to 0$

$P_g(x) \to 1 \quad P_r \to 0$ $\qquad P_g(x)\log\frac{P_g(x)}{P_r(x)} \to +\infty$

14 MLE

p_z $\quad G(z) = x \quad P_G(x;\theta) \quad P_{data}(x)$



$$D_{KL}(P||Q) = \int_{-\infty}^{+\infty} p(x) log \frac{p(x)}{q(x)}$$

$x^1, x^2, \dots, x^m$

$$L = \prod_{i=1}^{m} P_G(x^i;\theta)$$

$$D_{KL}(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)}$$

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{m} p_G(x^i, \theta) \iff \underset{\theta}{\operatorname{argmax}}\, log \prod_{i=1}^{m} p_G(x^i, \theta)$$

15 MLE

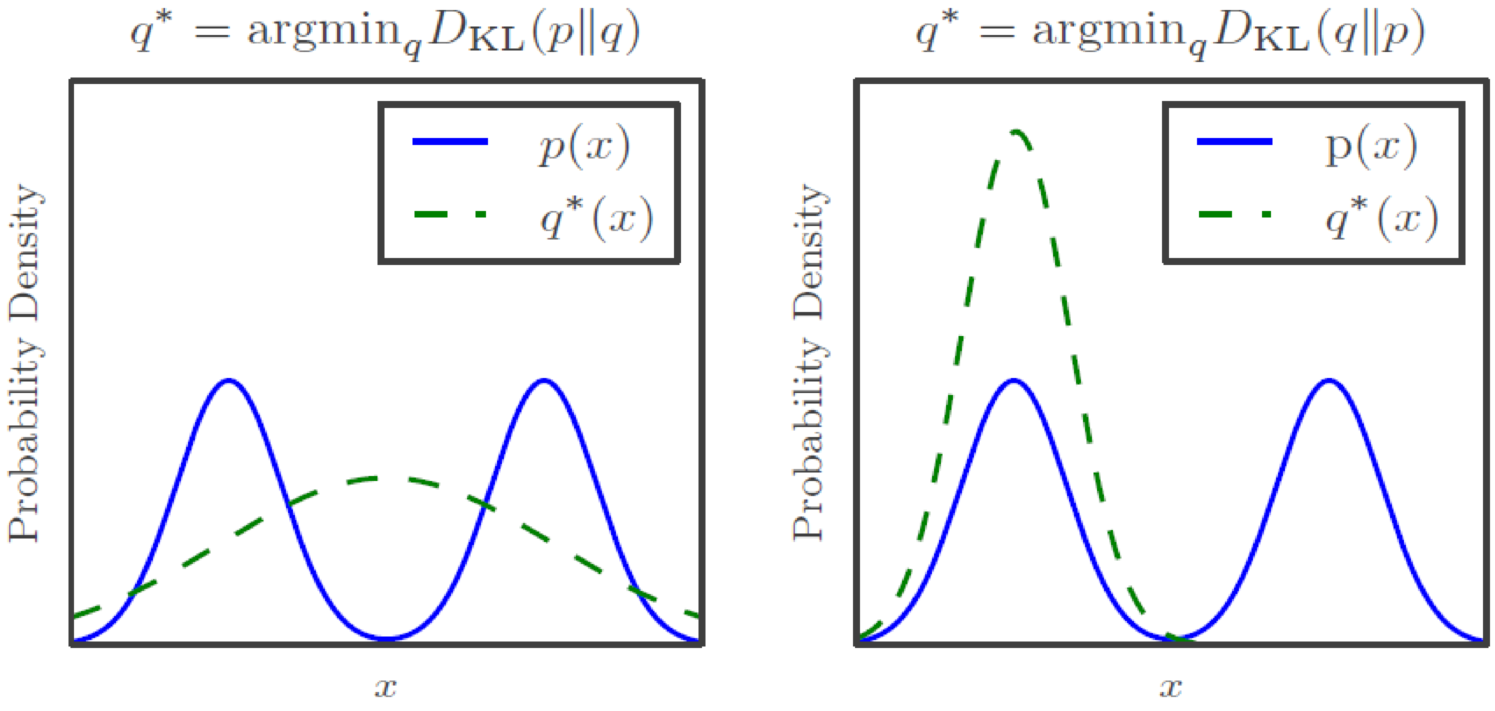
$$= \underset{\theta}{\operatorname{argmax}} \sum_{i}^{m} logP_G(x^i, \theta) \approx \underset{\theta}{\operatorname{argmax}} E_{x \sim P_{data}}[logP_G(x; \theta)]$$

$$\Leftrightarrow \underset{\theta}{\operatorname{argmax}} \int P_{data}(x) logP_G(x; \theta) dx - \int P_{data}(x) logP_{data}(x) dx$$

$$= \underset{\theta}{\operatorname{argmax}} \int P_{data}(x) log \frac{P_G(x;\theta)}{P_{data}(x)} dx$$

$$\underset{\theta}{\operatorname{argmin}} KL(P_{data} || P_G(x; \theta))$$

16 Comparison



Maximum likelihood

Reverse KL

WGAN

Wasserstein GAN (arXiv 2017.5.9)

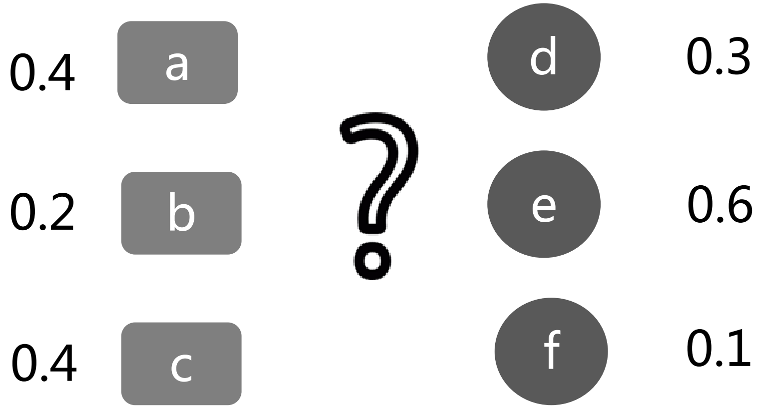
$$KL(P||Q) = \int_{-\infty}^{+\infty} p(x)log\frac{p(x)}{q(x)} \qquad p \sim N(0,\epsilon^3) \qquad q \sim N(\epsilon,\epsilon^3) \qquad KL(P||Q) = \frac{1}{2\epsilon}$$

$$W(P_r,P_g) = \inf_{\gamma \in \prod(P_r,P_g)} \left(\int d(x,y)^p d\gamma(x,y) \right)^{1/p} = (\inf_{\gamma} E[d(x,y)^p])^{1/p}$$

Earth-Mover (EM) distance $W(P_r,P_g) = \inf_{\gamma \in \prod(P_r,P_g)} E_{(x,y)\sim\gamma}[||x-y||]$

$\prod(P_r,P_g)$denotes the set of all joint distributions $\gamma(x,y)$ whose marginal are respectively P_r and P_g. Intuitively, $\gamma(x,y)$indicates how much "mass" must be transported from x to y in order to transform the distributions P_r into the distribution P_r. The EM distance is the "cost" of the optimal transport plan.

EM



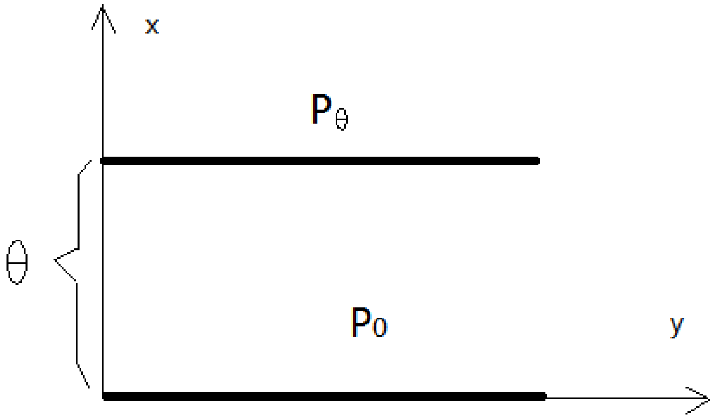
$d(s,t)$ $m(s,t)$

$$min \sum_{\pi(s,t)} d(s,t) * m(s,t)$$

$$\sum_{s} m(s,t) = b(t)\ \forall t$$

$$\sum_{t} m(s,t) = a(s)\ \ \forall s$$

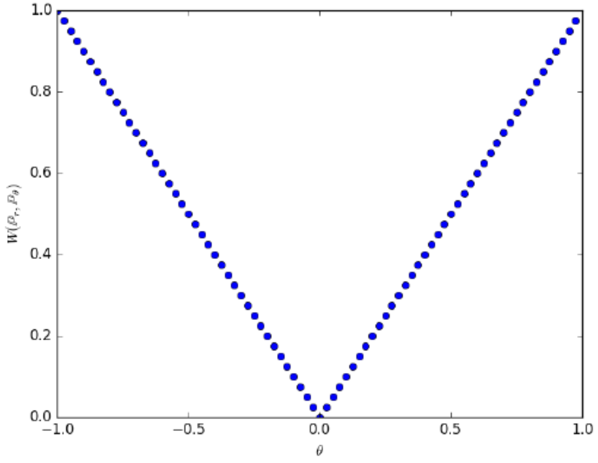
Comparison

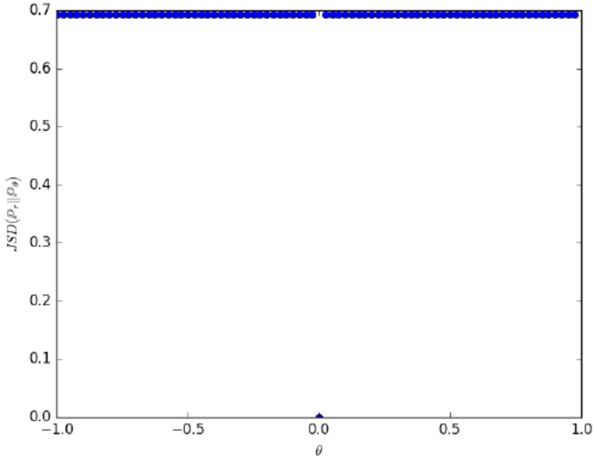


$$W(P_0, P_\theta) = |\theta|$$

$$JS(P_0, P_\theta) = \begin{cases} log2 & if\ \theta \neq 0 \\ 0 & if\ \theta = 0 \end{cases}$$

$$KL(P_0, P_\theta) = KL(P_\theta, P_0) = \begin{cases} +\infty & if\theta \neq 0 \\ 0 & if\theta = 0 \end{cases}$$





$$W(P_r, P_g) = \inf_{\gamma \in \prod(P_r, P_g)} E_{(x,y)\sim\gamma}[\|x - y\|]$$

However, the infimum is highly intractable.

Transformation

Kantorovich-Rubinstein[22] duality tells us that

$$W(P_r, P_\theta) = \sup_{||f||_{\leq 1}} E_{x\sim P_r}[f(x)] - E_{x\sim p_\theta}[f(x)]$$

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$$

We replace $||f||_L \leq 1$ for $||f||_L \leq K$

$$W(P_r, P_\theta) = \frac{1}{K} \sup_{||f||_{\leq K}} E_{x\sim P_r}[f(x)] - E_{x\sim p_\theta}[f(x)]$$

If we have a parameterized family of functions $\{f_w\}_{w\in W}$ that are all K-Lipschitz for some K, we could consider solving the problem

$$KW(P_r, P_\theta) \approx \max_{w:|f_w|_{L\leq K}} E_{x\sim P_r}[f_w(x)] - E_{z\sim p(z)}[f_w(g_\theta(z))]$$

Clamp the weights to a fixed box (say $\mathcal{W} = [-0.01, 0.01]^l$) after each gradient update.

Theory

Corollary Let g_θ be any feedforward neural network parameterized by θ , and p(z) a prior over z such that $E_{z\sim p(z)}[\|z\|] < \infty$ (e.g. Gaussian, uniform, etc.). then assumption is satisfied and therefore $W(P_r, P_\theta)$ is continuous everywhere and differentiable almost everywhere.

The corollary tells us that learning by minimizing the EM distance makes sense (at least in theory) with neural networks.

Furthermore, we could consider differentiating $W(P_r, P_\theta)$ by back-proping via estimating $E_{z\sim p(z)}[\nabla_\theta f_w(g_\theta(z))]$.

$$\max_{w:|f_w|_L\leq K} E_{x\sim P_r}[f_w(x)] - E_{z\sim p(z)}[f_w(g_\theta(z))]$$

$$\nabla_\theta W(P_r, P_\theta) = -E_{z\sim p(z)}[\nabla_\theta f(g_\theta(z))]$$

Framework

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
    for $t = 0, ..., n_{\text{critic}}$ **do**
        Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
        Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
        $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
        $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
        $w \leftarrow \text{clip}(w, -c, c)$
    end for
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
    $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

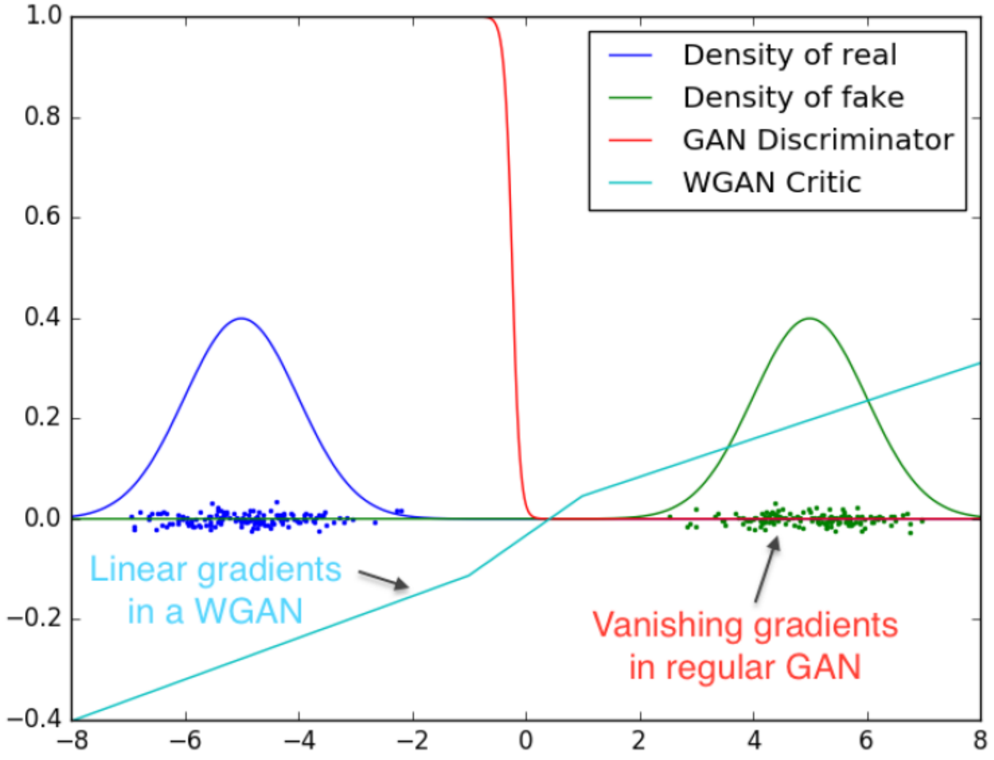
23 Comparison

critic=$E_{x\sim P_r}[f_w(x)] - E_{z\sim p(z)}[f_w(g_\theta(z))]$

The discriminator learns very quickly to distinguish between fake and real, and as expected provides no reliable gradient information.

The critic, however, can't saturate, and converges to a linear function that gives remarkably clean gradients everywhere.

The fact that the EM distance is continuous and dierentiable a.e. means that we can (and should) train the critic till optimality. The argument is simple, the more we train the critic, the more reliable gradient of the Wasserstein we get, which is actually useful by the fact that Wasserstein is dierentiable almost everywhere.



Summary

Weight clipping is a clearly terrible way to enforce a Lipschitz constraint. If the clipping parameter is large, then it can take a long time for any weights to reach their limit, thereby making it harder to train the critic till optimality. If the clipping is small, this can easily lead to vanishing gradients when the number of layers is big, or batch normalization is not used (such as in RNNs).

Comparison

1. 判别器最后一层去掉sigmoid
2. 生成器和判别器的loss不取log
3. 每次更新判别器的参数之后把它们的绝对值截断到不超过一个固定常数c
4. 不要用基于动量的优化算法（包括momentum和Adam），推荐RMSProp，SGD也行

http://blog.csdn.net/heyongluoyao8/article/details/52478715

BGD SGD MBGD Momentum Nesterov Momentum

$$g' = \frac{1}{m}\nabla_\theta \sum_i L(f(x_i, \theta_t), y_i)$$

$$v_{t+1} = av_t - \eta g' \qquad \theta_{t+1}=\theta_t + v_t$$

Adagrad

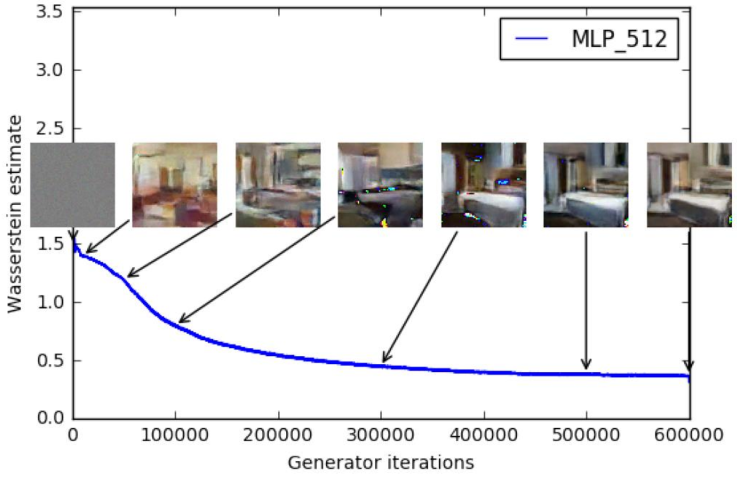
$$g'_{t,i} = \nabla_\theta L(f(x_i, \theta_{t,i}), y_i) \qquad \theta_{t+1,i} = \theta_{t,i} - \eta g_{t,i} \qquad \theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} g_{t,i}$$

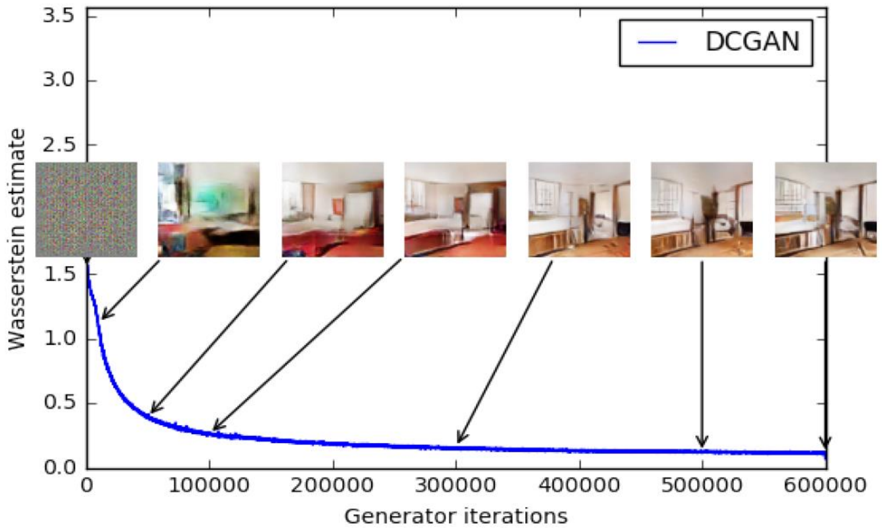
G_t是一个对角矩阵，第i行对角元素e_{ii}为过去到当前第i个参数θ_i的梯度平方和

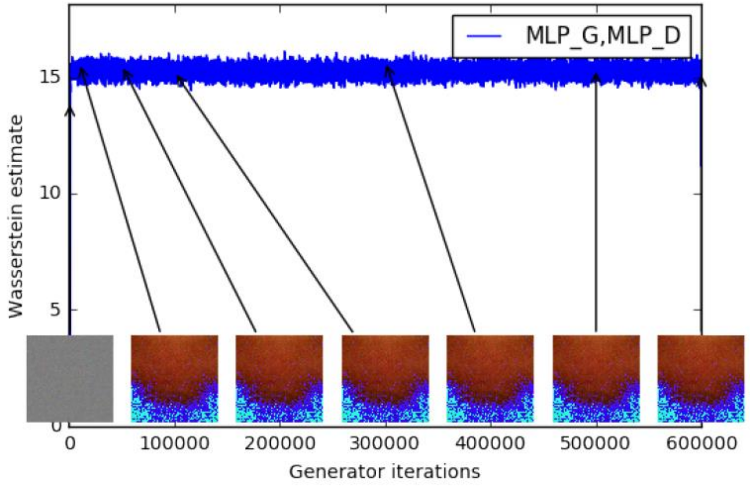
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

Adadelta RMSprop Adam

Results



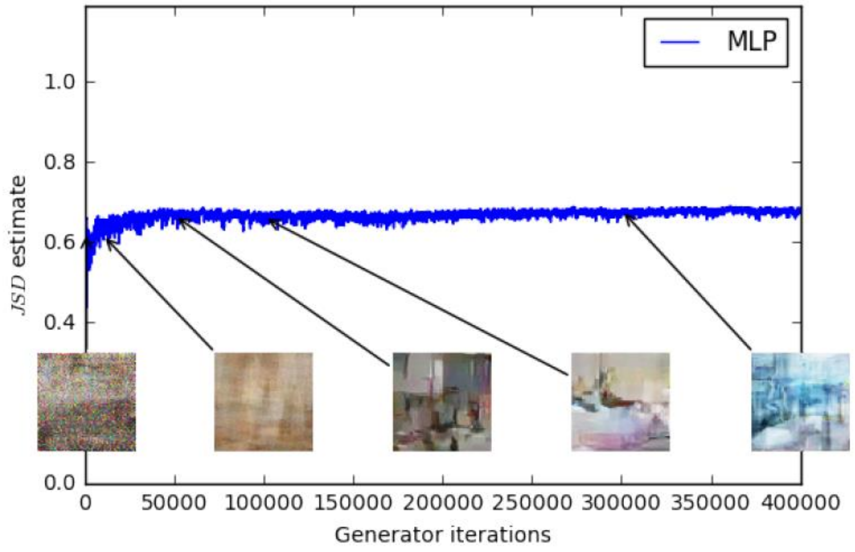


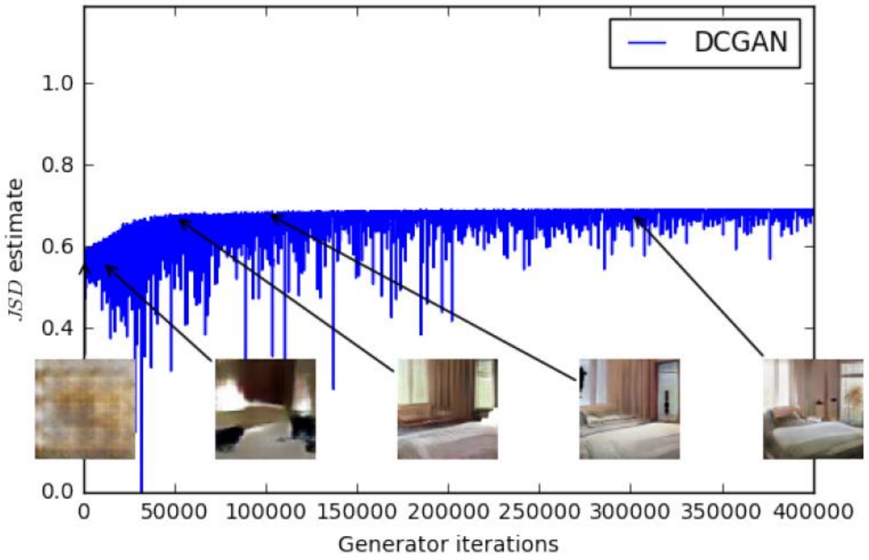


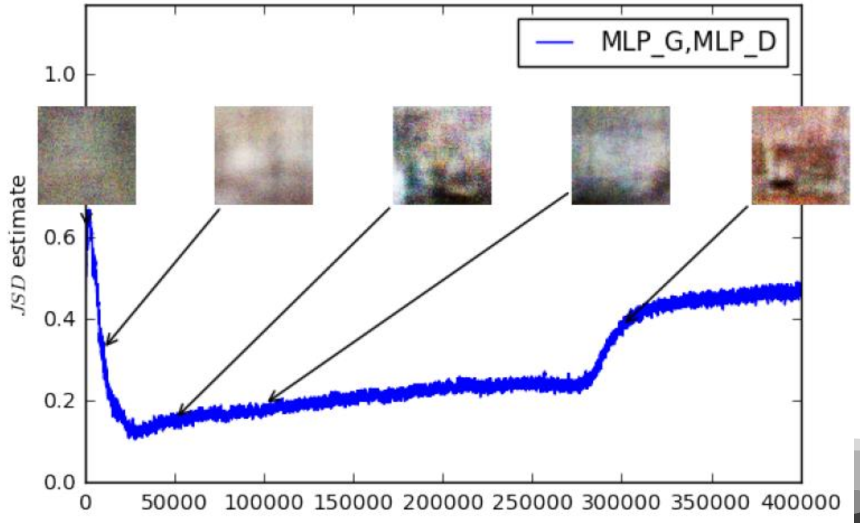
The evolution of the WGAN estimate (3) of the EM distance during WGAN training for all three architectures.

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z))] \tag{3}$$

Results





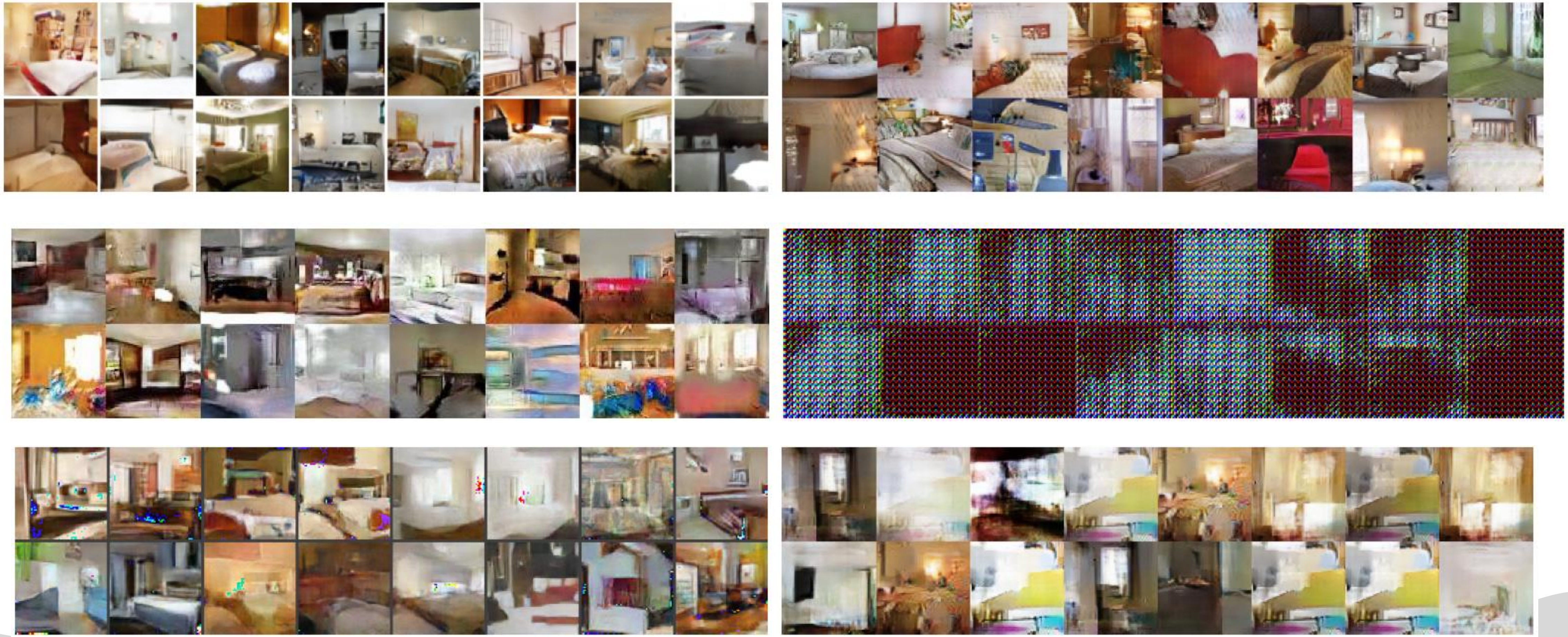


$$L(D, g_\theta) = \mathbb{E}_{x\sim\mathbb{P}_r}[\log D(x)] + \mathbb{E}_{x\sim\mathbb{P}_\theta}[\log(1 - D(x))]$$

$$2JS(\mathbb{P}_r, \mathbb{P}_\theta) - 2\log 2$$

$$\frac{1}{2}L(D, g_\theta) + \log 2$$

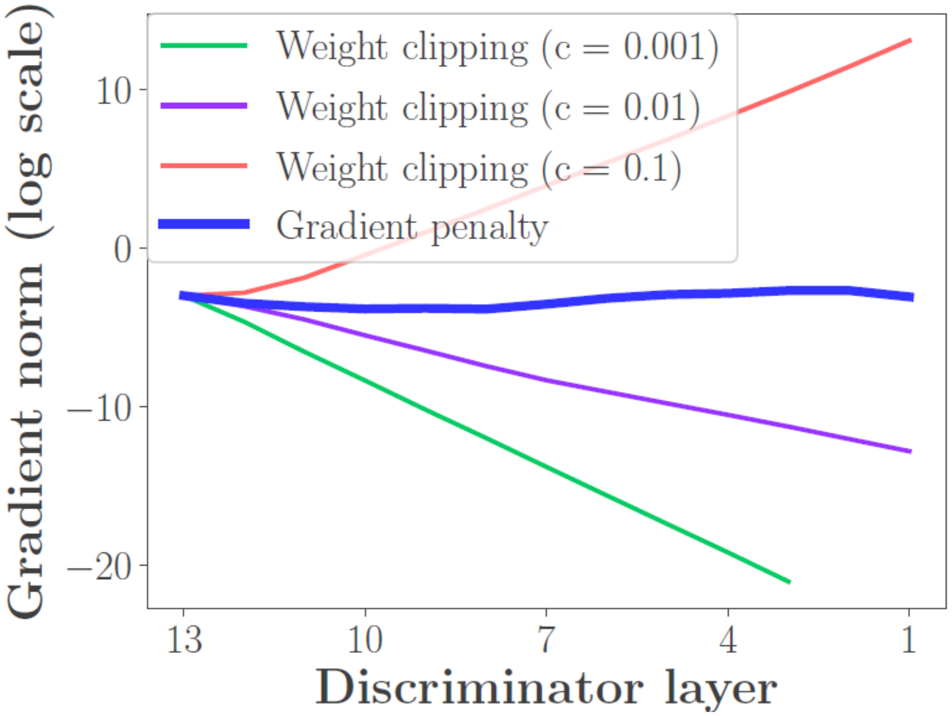
Results

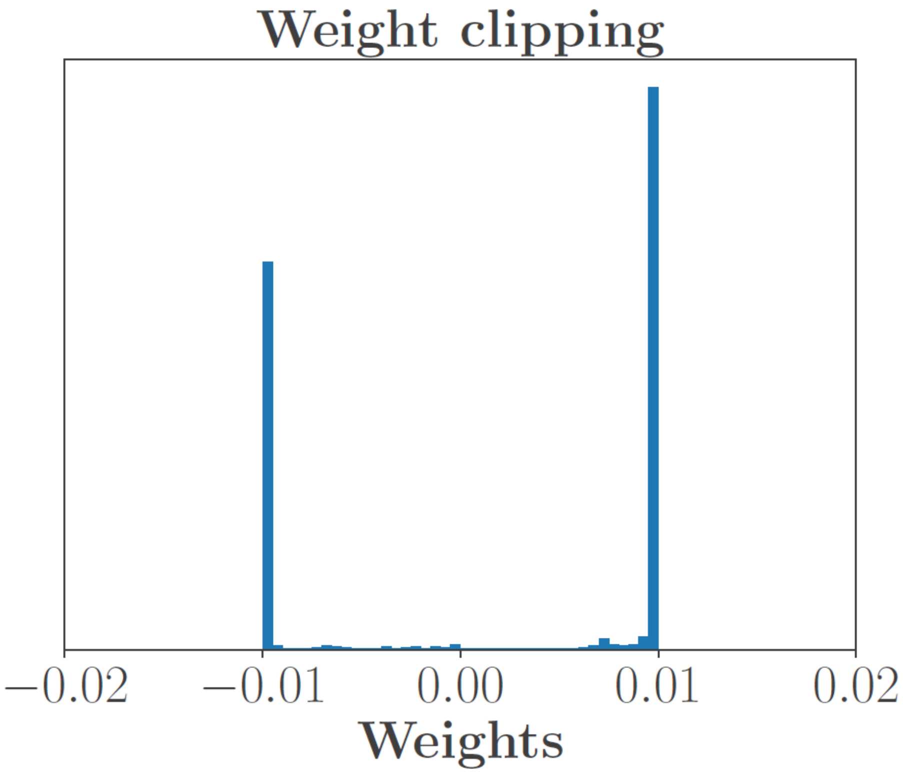


WGAN-GP

Improved Training of Wasserstein GANs

$$L(D) = -E_{x\sim P_r}[D(x)] + E_{x\sim P_g}[D(x)] \qquad L(G) = -E_{x\sim P_g}[D(x)] \qquad ||\nabla_x D(x)||_p < K, \forall x \in \mathcal{X}$$





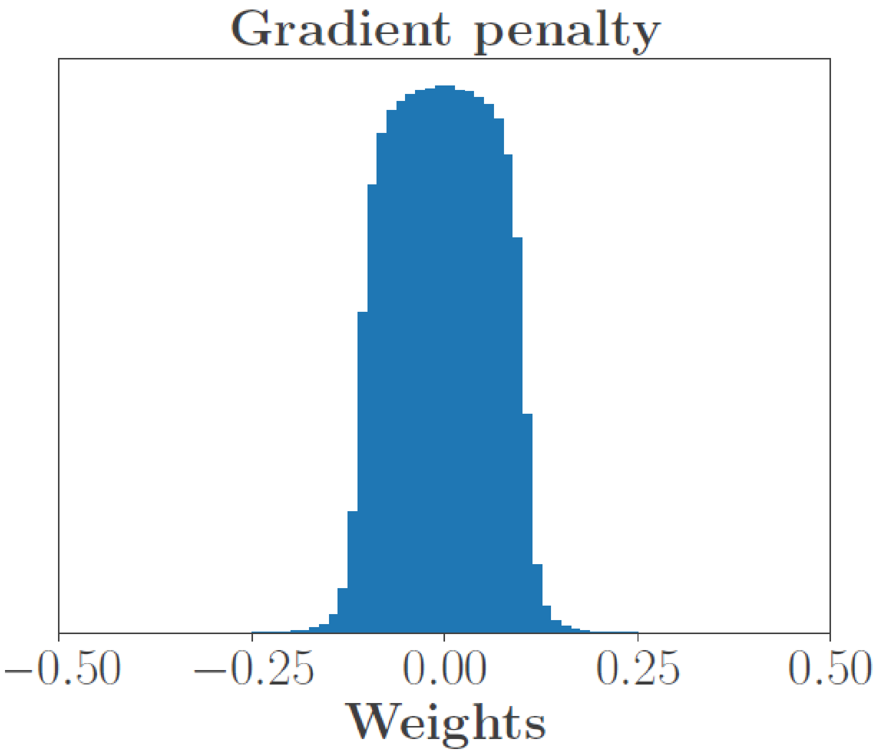
WGAN-GP

$$[||\nabla_x D(x)||_p - K]^2$$

$$L = -E_{x\sim P_r}[D(x)] + E_{x\sim P_g}[D(x)] + \lambda E_{\hat{x}\sim\mathcal{X}}[||\nabla_{\hat{x}} D(\hat{x})||_p - 1]^2$$

$$x_r \sim P_r, x_g \sim P_g, \epsilon \sim Uniform[0,1]$$

$$\hat{x} = \epsilon x_r + (1-\epsilon)x_g$$



WGAN-GP

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

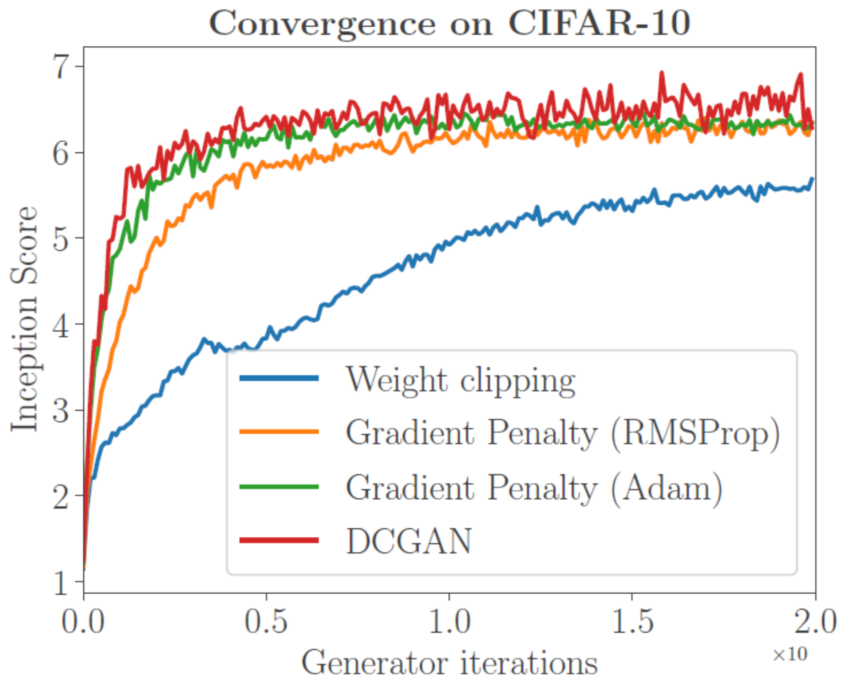
while θ has not converged **do**
    for $t = 1, \ldots, n_{\text{critic}}$ **do**
        for $i = 1, \ldots, m$ **do**
            Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
            $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
            $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
            $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
        end for
        $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
    end for
    Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
    $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

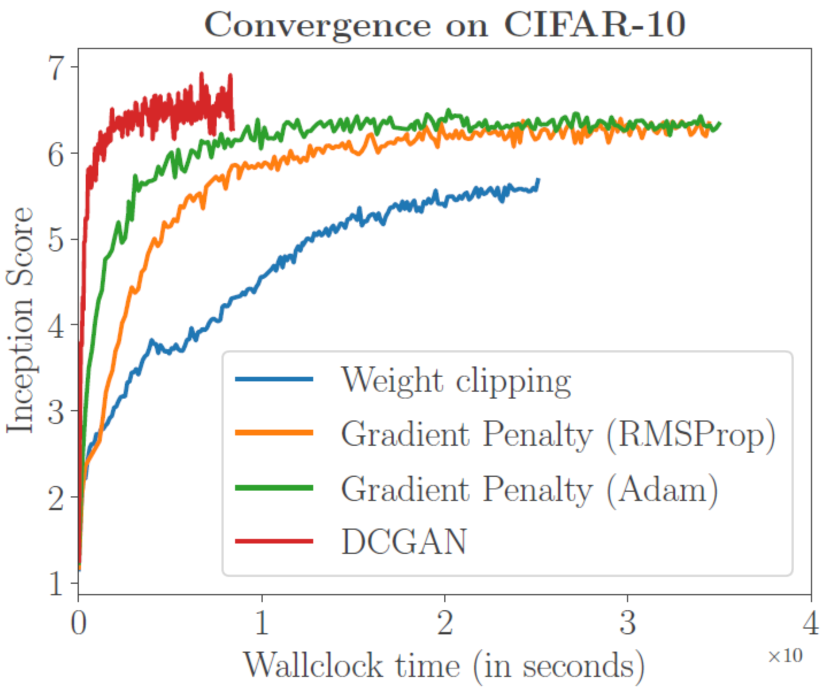
WGAN-GP

No critic batch normalization

Layer normalization

Penalize the norm of the critic’ s gradient with respect to each input independently, and not the entire batch.





WGAN-GP

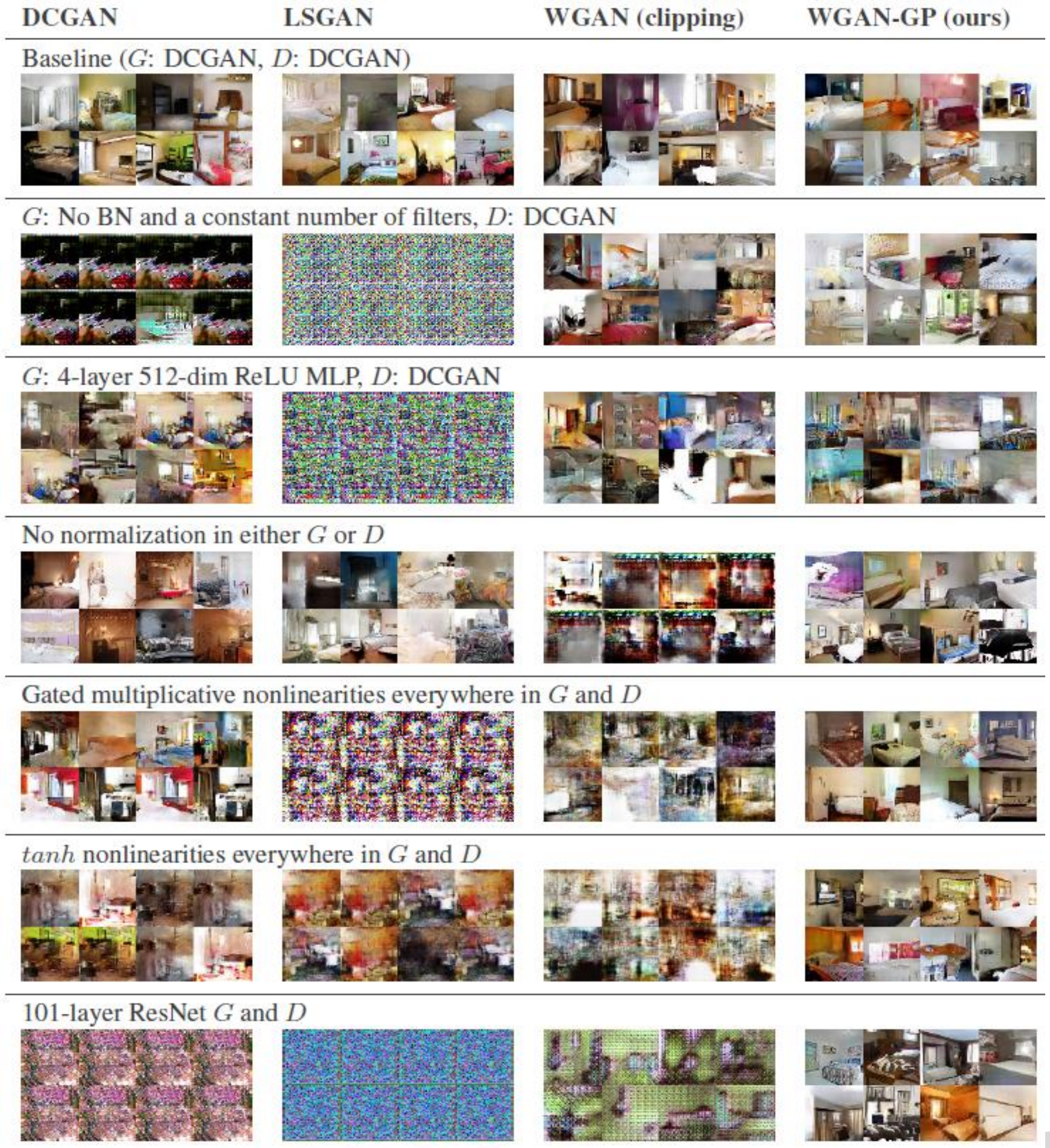

DCGAN
LSGAN
WGAN (clipping)
WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)
G: No BN and a constant number of filters, D: DCGAN
G: 4-layer 512-dim ReLU MLP, D: DCGAN
No normalization in either G or D
Gated multiplicative nonlinearities everywhere in G and D
tanh nonlinearities everywhere in G and D
101-layer ResNet G and D

Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network

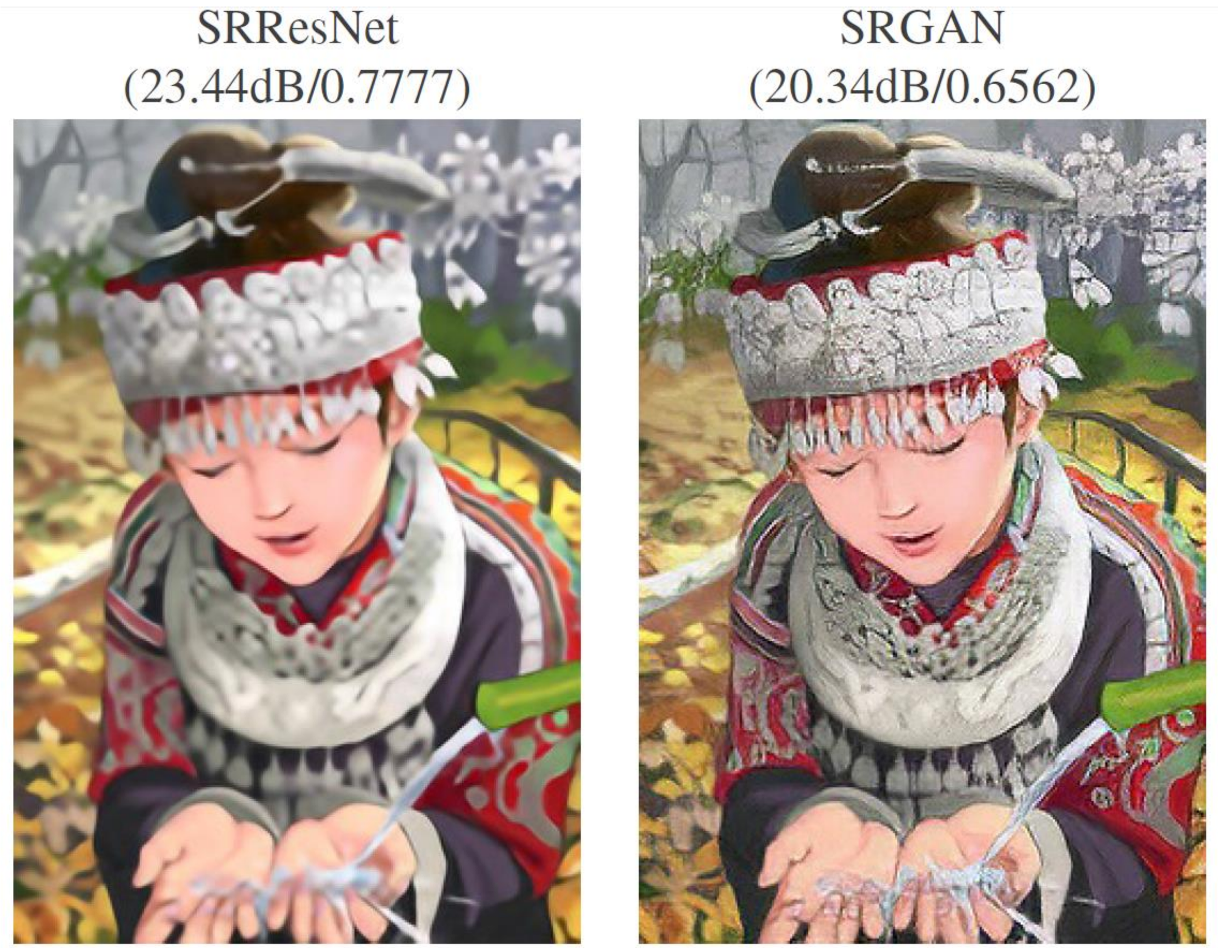
arXiv 2016.9.15
2017.5.25

Image-to-Image Translation with Conditional Adversarial Networks

arXiv 2016.11.21

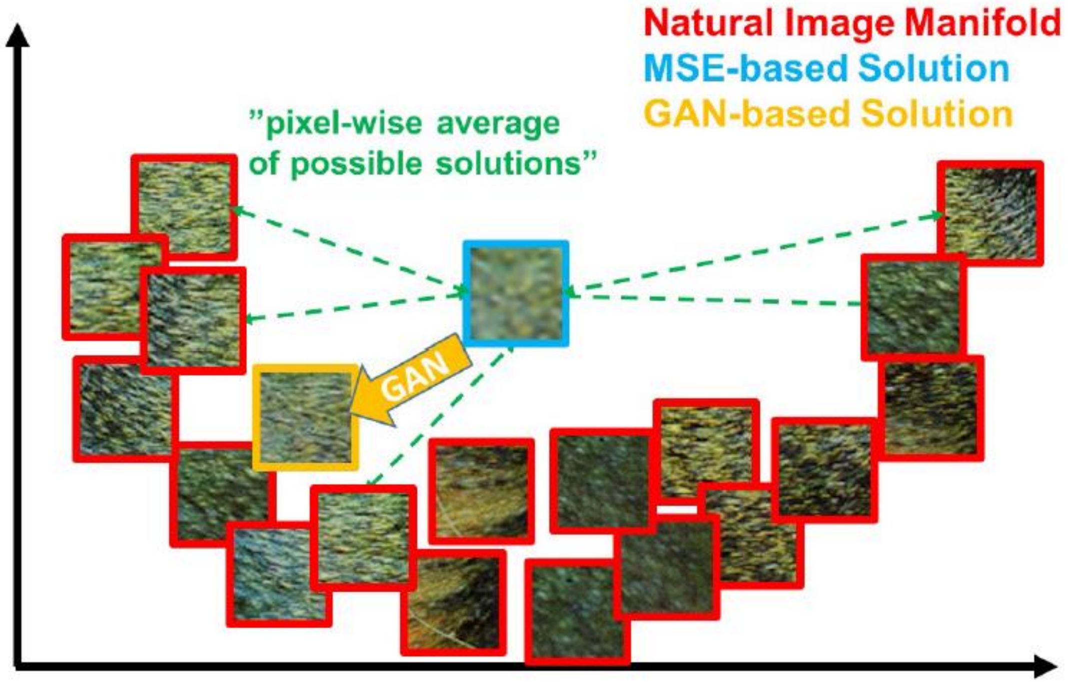
2017.7.24

01 Loss Functions



Pixel-wise loss functions such as MSE, minimizing MSE encourages finding pixel-wise averages of plausible solutions which are typically overly-smooth and thus have poor perceptual quality.

02 Loss Functions



The MSE-based solution appears overly smooth due to the pixel-wise average of possible solutions in the pixel space,

While GAN drives the reconstruction towards the natural image manifold producing perceptually more convincing solutions.

03 SRGAN

In training, I^{LR} is obtained by applying a Gaussian-filter to I^{HR} followed by a downsampling operation with downsampling factor r. For an image with C color channels, we describe I^{SR}, I^{LR} by a real-valued tensor of size $W \times H \times C$ and I^{SR} by $\mathrm{r}W \times rH \times C$ respectively.

Our ultimate goal is to train a generating function G that estimates for a given LR input image its corresponding HR counterpart.

A feed-forward CNN G_{θ_G} θ_G=$\{W_{1:L}, b_{1:L}\}$

Given training images I_n^{HR},$n = 1, \ldots, N$ with corresponding $I_n^{LR}, n = 1, \ldots, N$

$$\hat{\theta}_G = \arg\min_{\theta_G} \frac{1}{N} \sum_{n=1}^{N} l^{SR}(G_{\theta_G}(I_n^{LR}), I_n^{HR})$$

$$\min_{\theta_G} \max_{\theta_D} E_{I^{HR} \sim p_{train}(I^{HR})} \left[log D_{\theta_D}(I^{HR}) \right] + E_{I^{LR} \sim p_G(I^{HR})} \left[\log(1 - D_{\theta_D}(G_{\theta_G}(I^{LR}))) \right]$$

04 Perceptual loss function

$$l^{SR} = \underbrace{l_X^{SR}}_{content\ loss} + \underbrace{10^{-3} l_{Gen}^{SR}}_{adversarial\ loss} \qquad l^{SR} = \underbrace{l_X^{SR}}_{content\ loss} + \underbrace{10^{-3} l_{Gen}^{SR}}_{adversarial\ loss} + \underbrace{2 \cdot 10^{-8} l_{TV}^{SR}}_{\text{regularization}\ loss}$$

Content loss

The pixel-wise MSE loss is calculated as

$$l_{MSE}^{SR} = \frac{1}{r^2 WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} (I_{x,y}^{H,R} - G_{\theta_G}(I^{LR})_{x,y})^2$$

While achieving particularly high PSNR, solutions of MSE optimization problems often lack high-frequency content which result in perceptually unsatisfying solutions with overly smooth textures.

We define the **VGG loss** based on the ReLU activation layers of the pre-trained 19 layer VGG network$\phi_{i,j}$ indicates the feature map obtained by the j-th convolution(after activation) feature map before the i-th maxpooling layer within the VGG19 network, which we consider given.

05

$$l_{VGG/i,j}^{SR} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2$$

Adversarial loss

$$l_{Gen}^{SR} = \sum_{n=1}^{N} -logD_{\theta_D}(G_{\theta_G}(I^{LR}))$$

Regularization Loss

$$l_{TV}^{SR} = \frac{1}{r^2WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} \left\| \nabla G_{\theta_G}(I^R)_{x,y} \right\|$$

For better gradient behavior we minimize $-logD_{\theta_D}(G_{\theta_G}(I^{LR})$ instead of $\log[1 - logD_{\theta_D}(G_{\theta_G}(I^{LR}))]$

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

ICML 2015

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

$$\frac{\partial \ell}{\partial \widehat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i$$

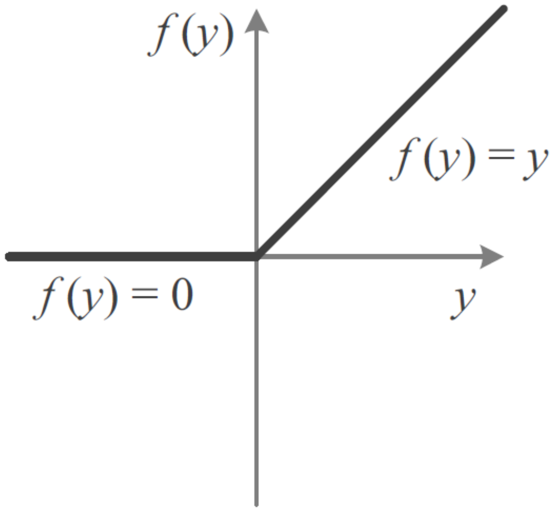
$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

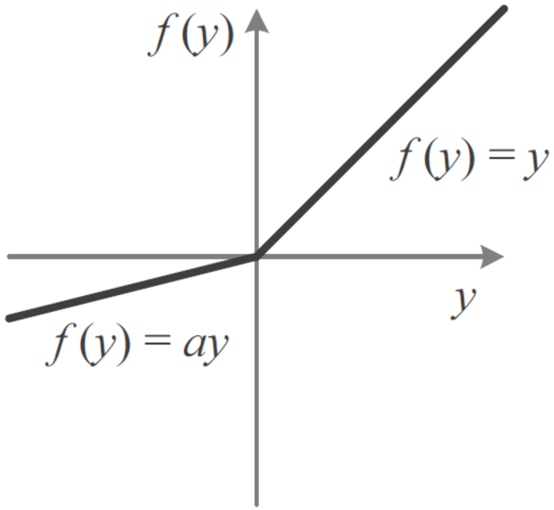
07 ReLUs

Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

ICCV 2015

$$f(y_i) = \begin{cases} y_i & if\, y_i > 0 \\ a_i y_i & if\, y_i \leq 0 \end{cases}$$





If a_i is a small and fixed value, PReLU becomes the Leaky ReLU.

PReLU introduces a very small number of extra parameters. The number of extra parameters is equal to the total number of channels, which is negligible when considering the total number of weights.

08 ReLUs

PReLU can be trained using backpropagation and optimized simultaneously with other layers.

$$\frac{\partial \varepsilon}{\partial a_i} = \sum_{y_i} \frac{\partial \varepsilon}{\partial f(y_i)} \frac{\partial f(y_i)}{\partial a_i}$$

$\frac{\partial \varepsilon}{\partial f(y_i)}$ is the gradient propagated from the deeper layer.

The gradient of the activation is given by:

$$\frac{\partial f(y_i)}{\partial a_i} = \begin{cases} 0 & if\, y_i > 0 \\ y_i & if\, y_i < 0 \end{cases}$$

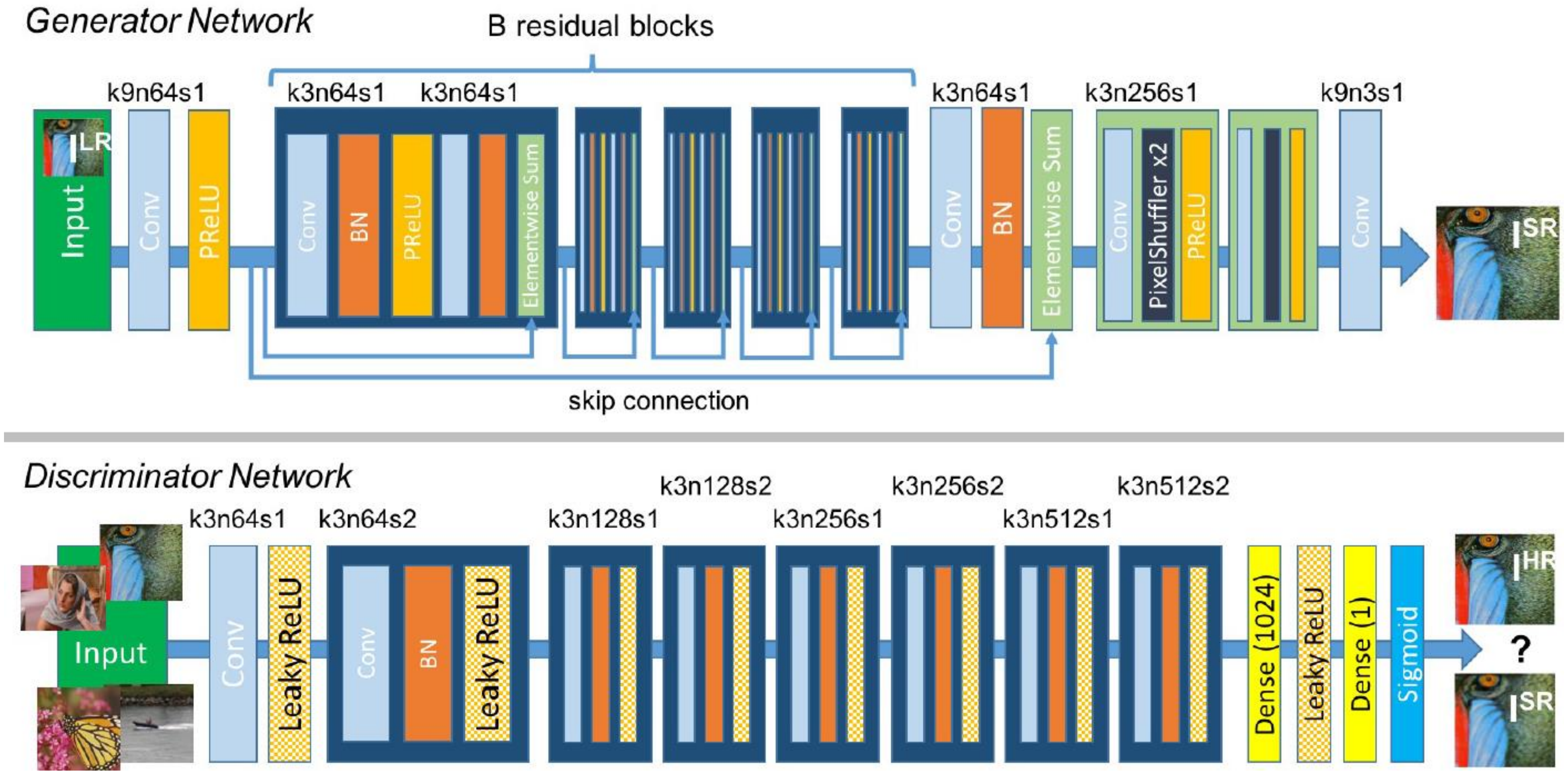
We adopt the momentum method when updating a_i:

$$\Delta a_i := \mu \Delta a_i + \epsilon \frac{\partial \varepsilon}{\partial a_i}$$

μ is the momentum and ϵ is the learning rate

It is worth noticing that we do not use weight decay (L_2 regularization) when updating a_i. A weight decay tends to push a_i to zero, and thus biases PReLU toward ReLU.

09


Generator Network
B residual blocks
k9n64s1
k3n64s1
k3n64s1
k3n64s1
k3n256s1
k9n3s1
I LR
Input
Conv
PReLU
BN
Elementwise Sum
PixelShuffler x2
I SR
skip connection
Discriminator Network
k3n64s1
k3n64s2
k3n128s1
k3n128s2
k3n256s1
k3n256s2
k3n512s1
k3n512s2
Input
Conv
Leaky ReLU
BN
Dense (1024)
Dense (1)
Sigmoid
I HR
?
I SR

10 Results

	SRResNet-		SRGAN-		
Set5	MSE	VGG22	MSE	VGG22	VGG54
PSNR	32.05	30.51	30.64	29.84	29.40
SSIM	0.9019	0.8803	0.8701	0.8468	0.8472
MOS	3.37	3.46	3.77	3.78	3.58
Set14					
PSNR	28.49	27.19	26.92	26.44	26.02
SSIM	0.8184	0.7807	0.7611	0.7518	0.7397
MOS	2.98	3.15*	3.43	3.57	3.72*

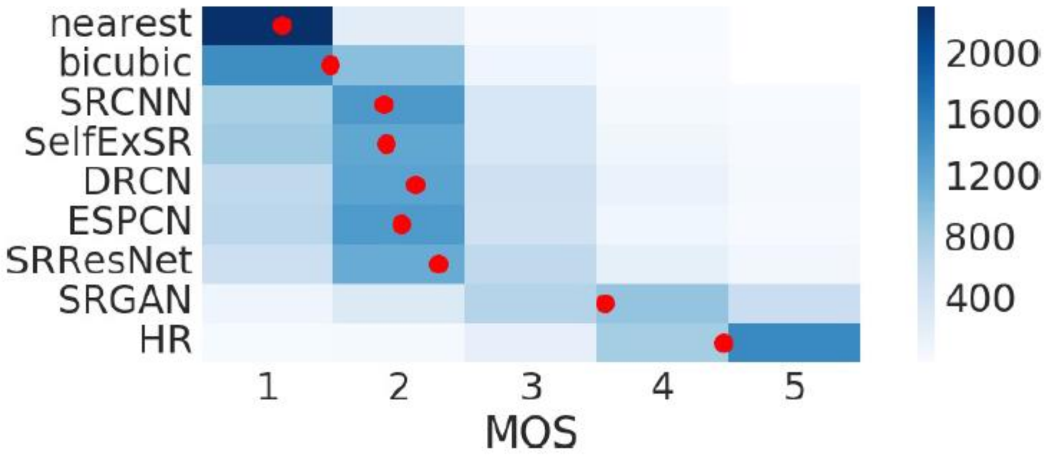
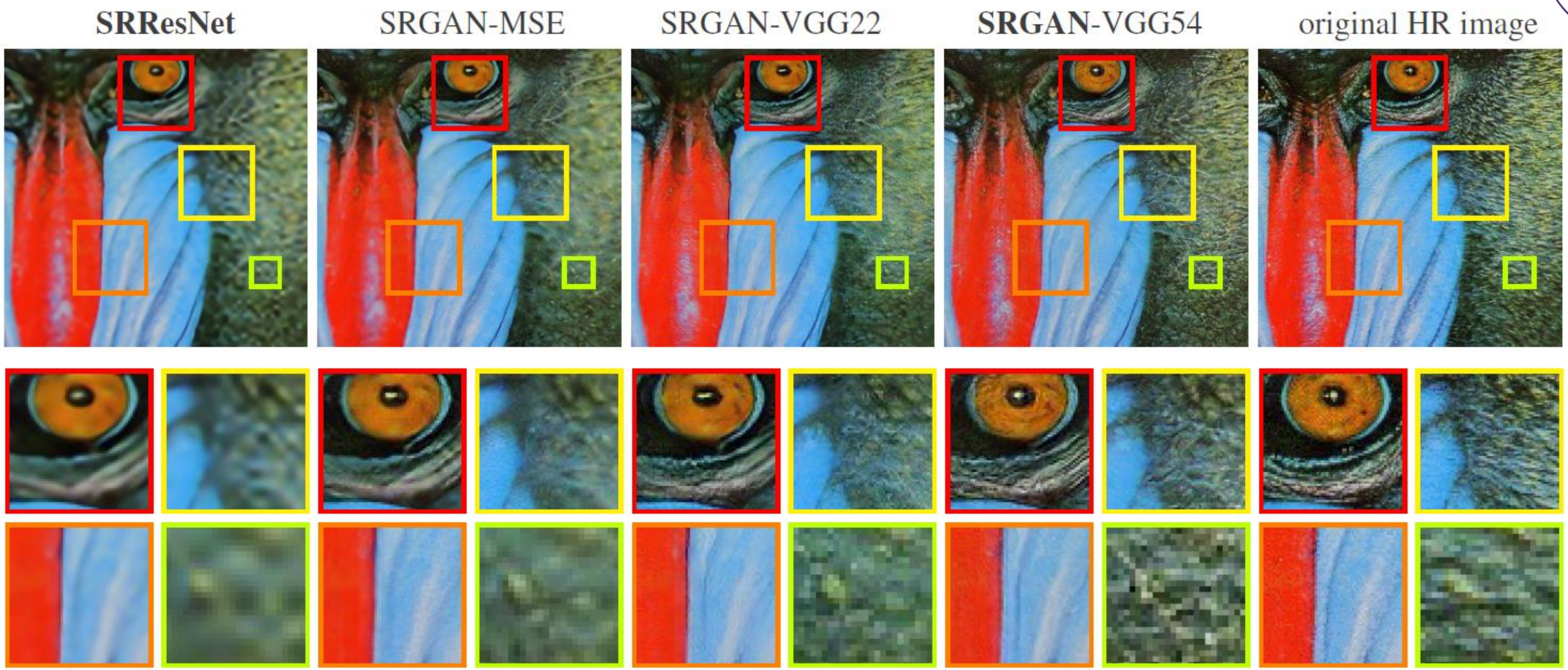


Figure 5: Color-coded distribution of MOS scores on **BSD100**. For each method 2600 samples (100 images × 26 raters) were assessed. Mean shown as red marker, where the bins are centered around value i. [4× upscaling]

11 Results

Set5	nearest	bicubic	SRCNN	SelfExSR	DRCN	ESPCN	**SRResNet**	**SRGAN**	HR
PSNR	26.26	28.43	30.07	30.33	31.52	30.76	**32.05**	29.40	∞
SSIM	0.7552	0.8211	0.8627	0.872	0.8938	0.8784	**0.9019**	0.8472	1
MOS	1.28	1.97	2.57	2.65	3.26	2.89	3.37	**3.58**	4.32
Set14									
PSNR	24.64	25.99	27.18	27.45	28.02	27.66	**28.49**	26.02	∞
SSIM	0.7100	0.7486	0.7861	0.7972	0.8074	0.8004	**0.8184**	0.7397	1
MOS	1.20	1.80	2.26	2.34	2.84	2.52	2.98	**3.72**	4.32
BSD100									
PSNR	25.02	25.94	26.68	26.83	27.21	27.02	**27.58**	25.16	∞
SSIM	0.6606	0.6935	0.7291	0.7387	0.7493	0.7442	**0.7620**	0.6688	1
MOS	1.11	1.47	1.87	1.89	2.12	2.01	2.29	**3.56**	4.46

12 Results

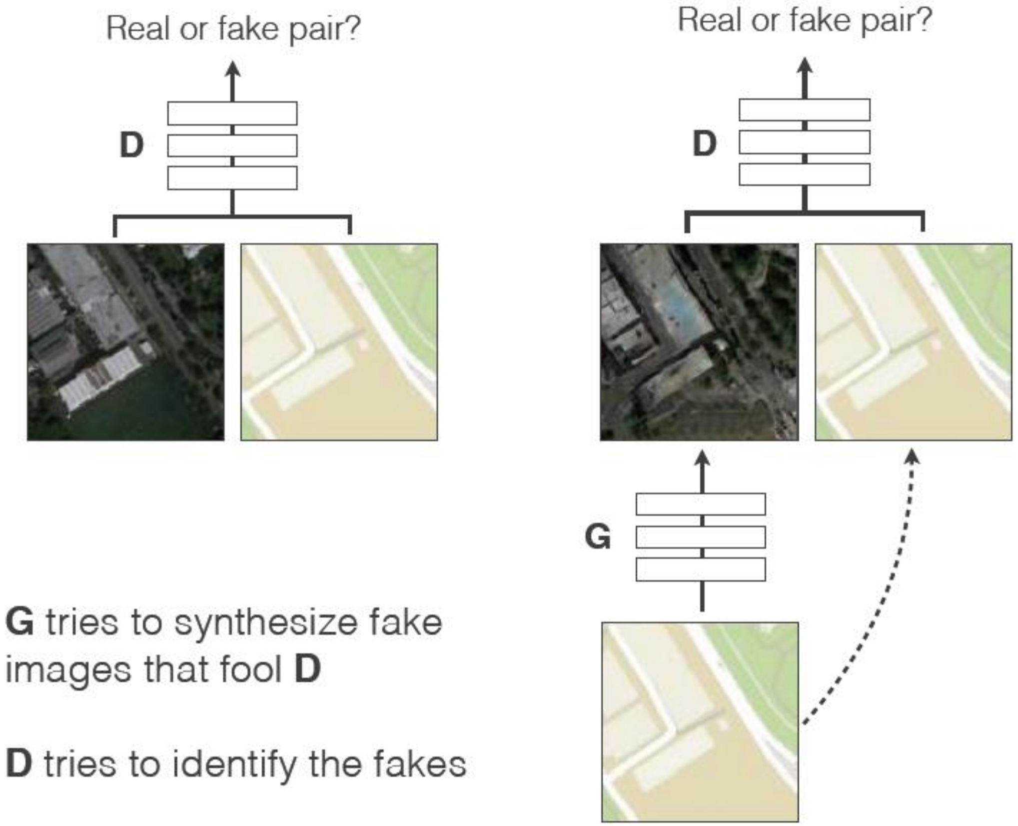


SRResNet
SRGAN-MSE
SRGAN-VGG22
SRGAN-VGG54
original HR image

CGAN-Image

Image-to-Image Translation with Conditional Adversarial Networks

arXiv 2016.11.21



GANs are generative models that learn a mapping from random noise vector z to output image $y: G: z \rightarrow y$, In contrast, conditional GANs learn a mapping from observed image x and random noise vector z, to $y: G: \{x, z\} \rightarrow y$.

14 CGAN-Image

$$L_{CGAN}(G,D) = E_{x,y\sim p_{data}(x,y)}[logD(x,y)] + E_{x\sim p_{data}(x),z\sim p_z(z)}[\log(1-D(x,G(x,z)))]$$

$$G^* = \arg\min_G \max_D L_{CGAN}(G,D)$$

To test the importance of conditioning the discrimintor, we also compare to an unconditional variant in which the discriminator does not observe x:

$$L_{GAN}(G,D) = E_{y\sim p_{data}(x,y)}[logD(y)] + E_{x\sim p_{data}(x),z\sim p_z(z)}[\log(1-D(x,G(x,z)))]$$

Previous approaches to conditional GANs have found it beneficial to mix the GAN objective with a more traditional loss, such as L2 distance. The discriminator’ s job remains unchanged, but the generator is tasked to not only fool the discriminator but also to be near the ground truth output in an L2 sense.

CGAN-Image

$$L_{L1}(G,D) = E_{x\sim p_{data}(x),z\sim p_z(z)}[\|y - G(x,z)\|_1]$$

$$G^* = \arg\min_G \max_D L_{CGAN}(G,D) + \lambda L_{L1}(G)$$

Without z, the net could still learn a mapping from x to y, but would produce deterministic outputs, and therefore fail to match any distribution other than a delta function.

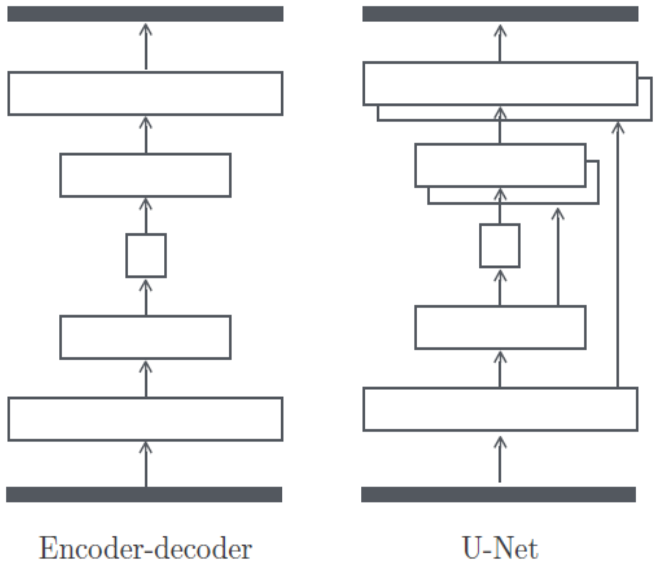
Past conditional GANs have acknowledged this and provided Gaussian noise z as an input to the generator, in addition to x.

In initial experiments, we did not find this strategy effective – the generator simply learned to ignore the noise.

16 CGAN-Image

We provide noise only in the form of dropout, applied on several layers of our generator at both training and test time.

Despite the dropout noise, we observe very minor stochasticity in the output of our nets.



Dropout: A Simple Way to Prevent Neural Networks from Overtting JMLR 2014

U-Net: Convolutional Networks for Biomedical Image Segmentation

International Conference on Medical Image Computing & Computer-assisted Intervention 2015

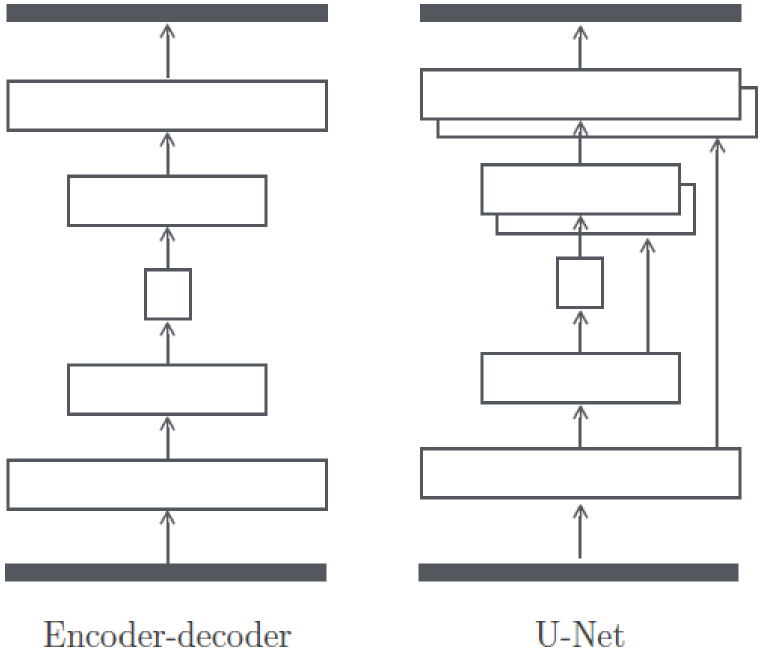
Generator

In such a network, the input is passed through a series of layers that progressively downsample, until a bottleneck layer, at which point the process is reversed.

Specifically, we add skip connections between each layer I and layer $n - i$, where n is the total number of layers.

C_k denote a Convolution-BatchNorm-ReLU layer with k filters

CD_k denotes a Convolution-BatchNorm-Dropout-ReLU layer with a dropout rate of 50%.



Architecture

Encoder C64_C128_C256_C512_C512_C512_C512_C512

Decoder C512_C512_C512_C512_C512_256_C128_C64

After the last layer in the decoder, a convolution is applied to map to the number of output channels (3 in general, except in colorization, where it is 2), followed by a tanh function. As an exception to the above notation, Batch-Norm is not applied to the first C64 layer in the encoder. All ReLUs in the encoder are leaky, with slope 0.2, while ReLUs in the decoder are not leaky.

U-Net decoder CD512-CD1024-CD1024-C1024-C1024-C512-C256-C128

70×70discriminator C64_C128_C256_C512

16×16discriminator C64-C128

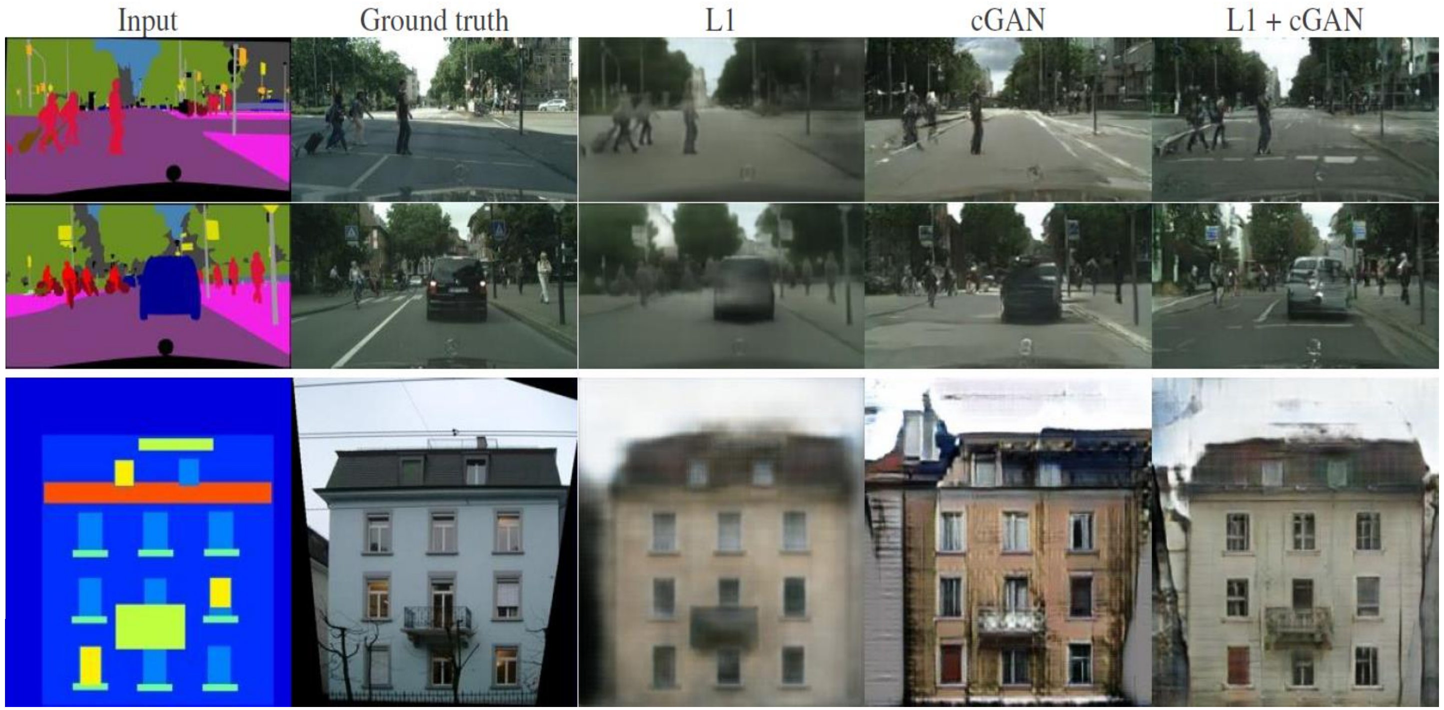
256×256discriminator C64-C128-C256-C512-C512-C512

1×1discriminator C64-C128

Results

Loss	Per-pixel acc.	Per-class acc.	Class IOU
L1	0.44	0.14	0.10
GAN	0.22	0.05	0.01
cGAN	0.61	**0.21**	**0.16**
L1+GAN	**0.64**	0.19	0.15
L1+cGAN	0.63	**0.21**	**0.16**
Ground truth	0.80	0.26	0.21

Table 1: FCN-scores for different losses, evaluated on Cityscapes labels↔photos.



Results

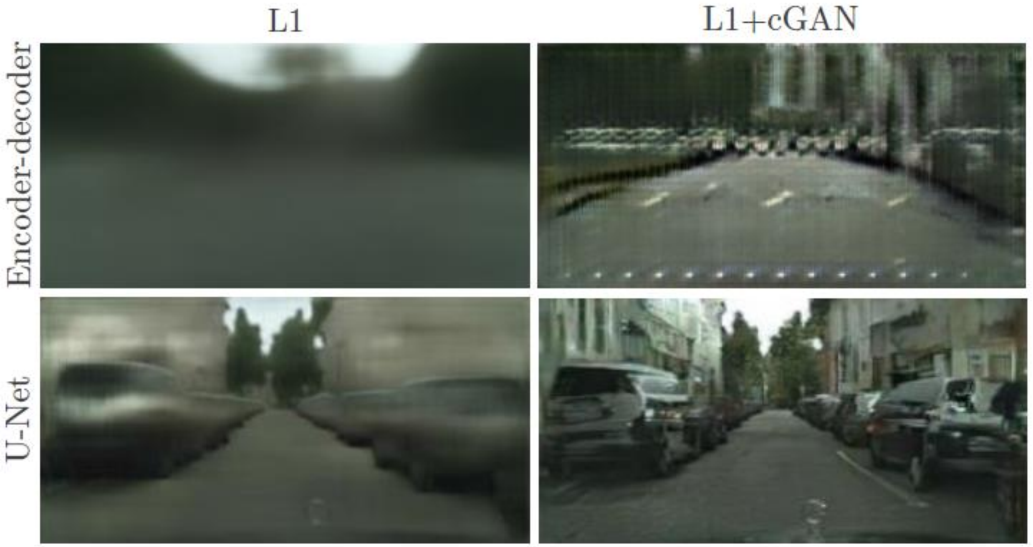


Figure 5: Adding skip connections to an encoder-decoder to create a “U-Net” results in much higher quality results.

Results

Discriminator receptive field	Per-pixel acc.	Per-class acc.	Class IOU
1×1	0.44	0.14	0.10
16×16	0.62	0.20	**0.16**
70×70	**0.63**	**0.21**	**0.16**
256×256	0.47	0.18	0.13

Table 2: FCN-scores for different receptive field sizes of the discriminator, evaluated on Cityscapes labels→photos.

Figure 6: Patch size variations. Uncertainty in the output manifests itself differently for different loss functions. Uncertain regions become blurry and desaturated under L1. The 1x1 PixelGAN encourages greater color diversity but has no effect on spatial statistics. The 16x16 PatchGAN creates locally sharp results, but also leads to tiling artifacts beyond the scale it can observe. The 70x70 PatchGAN forces outputs that are sharp, even if incorrect, in both the spatial and spectral (coforfulness) dimensions. The full 256x256 ImageGAN produces results that are visually similar to the 70x70 PatchGAN, but somewhat lower quality according to our FCN-score metric (Table 2). Please