ТЕХНОСФЕРА

Лекция 11
Generative adversarial networks

Галков Михаил, Храбров Кузьма

20 ноября 2017 г.

План лекции

Обучение без учителя и GAN

Применения GAN

- DCGAN
- Domain transfer network
- SRGAN
- Text to image (StackedGAN)
- Image to image (cGAN)

Проблема плохих градиентов и подходы к решению

- LSGAN
- WGAN
- BEGAN

Understanding data

When we're learning to see, nobody's telling us what the right answers are — we just look. Every so often, your mother says "that's a dog", but that's very little information. You'd be lucky if you got a few bits of information — even one bit per second — that way. The brain's visual system has 10^{14} neural connections. And you only live for 10^9 seconds. So it's no use learning one bit per second. You need more like 10^5 bits per second. And there's only one place you can get that much information: from the input itself. — Geoffrey Hinton, 1996 (quoted in (Gorder 2006)).

Problems in supervised learning

Проблемы с размеченными датасетами

- Данных много, но меток обычно мало/очень шумные/дорогие
- Transfer learning - нетривиальная проблема
- Распределение на обучающем множестве не совпадает с распределением 'in the wild'
- Распределение данных меняется со временем/адаптируется к вашей системе

Подходы к решению

- Semi-supervised learning (в широком смысле)
- Online learning (например: stock prediction)

Generative modeling

Наша задача - научиться генерировать реалистичные изображения. Под generative modeling понимают решение двух задач.

1. Понимание того, как устроено распределение $p_{data}(x)$.
2. Построение функции G, которая по заданному вектору будет генерировать изображение.

Learning to generate Chairs, Tables and Cars pt.1

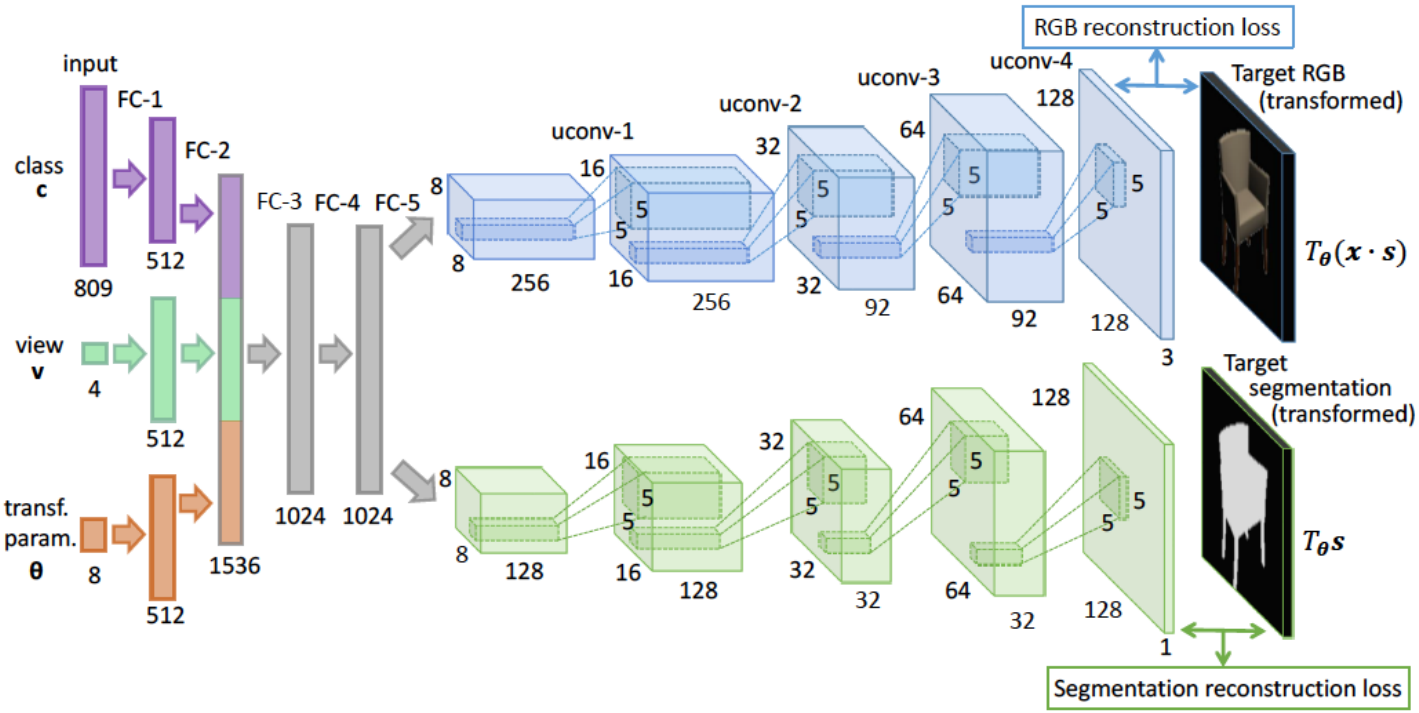


Fig. 1. Architecture of a 2-stream network that generates 128×128 pixel images. Layer names are shown above: FC - fully connected, uconv - unpooling+convolution.

$$\min_{\mathbf{W}} \sum_{i=1}^{N} L_{RGB}\left(T_{\theta^i}(\mathbf{x}^i \cdot \mathbf{s}^i),\, u_{RGB}(h(\mathbf{c}^i, \mathbf{v}^i, \theta^i))\right)$$
$$+\lambda \cdot L_{segm}\left(T_{\theta^i}\mathbf{s}^i,\, u_{segm}(h(\mathbf{c}^i, \mathbf{v}^i, \theta^i))\right),$$

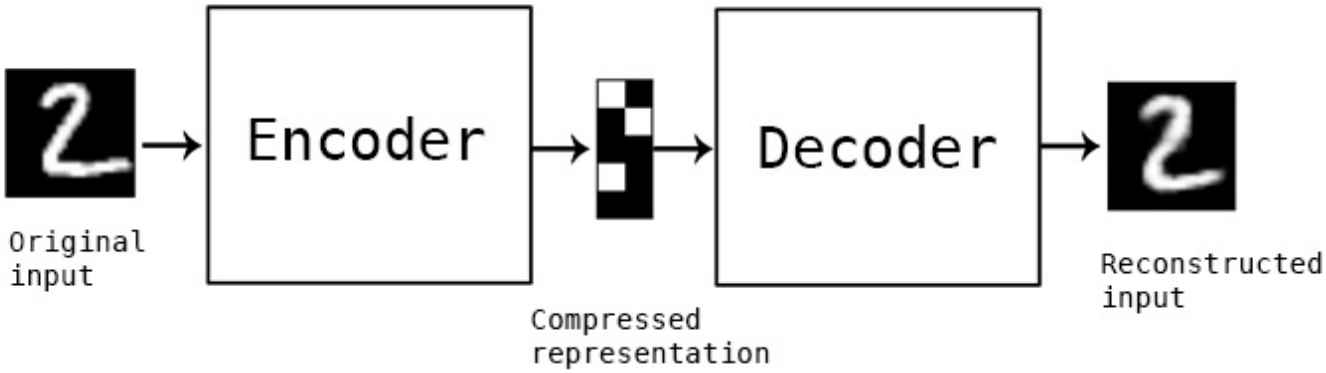
Learning to generate Chairs, Tables and Cars pt.2

Почему получилось так удачно? - **Огромное** количество информации на входе

1. категория семпла
2. углы и повороты
3. заданные параметры трансформаций
4. сегментация

Можно ли автоматизировать извлечение этих фичей?

Autoencoder



Идея очень похожая на PCA: найти сжимающее отображение исходных данных (Encoder) в пространство меньшей размерности, такое, что из него возможно восстановить исходное изображение (Decoder).

$$X \to Z \to X' \tag{1}$$

$$X \approx X' \tag{2}$$

Наивный подход

Естественно, хочется взять MSE как лосс-функцию и попробовать оптимизировать.
Но, к сожалению, результаты не впечатляют.

Возможно, мы взяли слишком простой лосс, можно придумать что-то лучше?

Deep reconstructions pt.1

layer	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
name	conv1	relu1	mpool1	norm1	conv2	relu2	mpool2	norm2	conv3	relu3	conv4	relu4	conv5	relu5	mpool5	fc6	relu6	fc7	relu7	fc8
type	cnv	relu	mpool	nrm	cnv	relu	mpool	nrm	cnv	relu	cnv	relu	cnv	relu	mpool	cnv	relu	cnv	relu	cnv
channels	96	96	96	96	256	256	256	256	384	384	384	384	256	256	256	4096	4096	4096	4096	1000
rec. field	11	11	19	19	51	51	67	67	99	99	131	131	163	163	195	355	355	355	355	355

Table 2. **CNN-A structure.** The table specifies the structure of CNN-A along with receptive field size of each neuron. The filters in layers from 16 to 20 operate as "fully connected": given the standard image input size of 227×227 pixels, their support covers the whole image. Note also that their receptive field is larger than 227 pixels, but can be contained in the image domain due to padding.

$\Phi : \mathbb{R}^{HxWxC} \rightarrow \mathbb{R}^d$ - representation function

$\Phi_0 = \Phi(x_0)$ - representation of specific image

L - loss function

R - regularization term

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^{H \times W \times C}}{\operatorname{argmin}} \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x})$$

Deep reconstructions pt.2

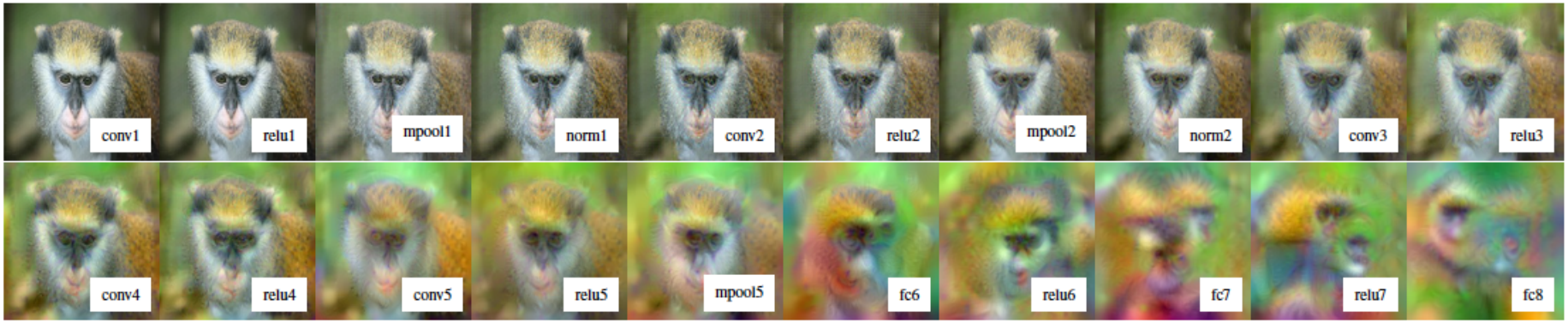


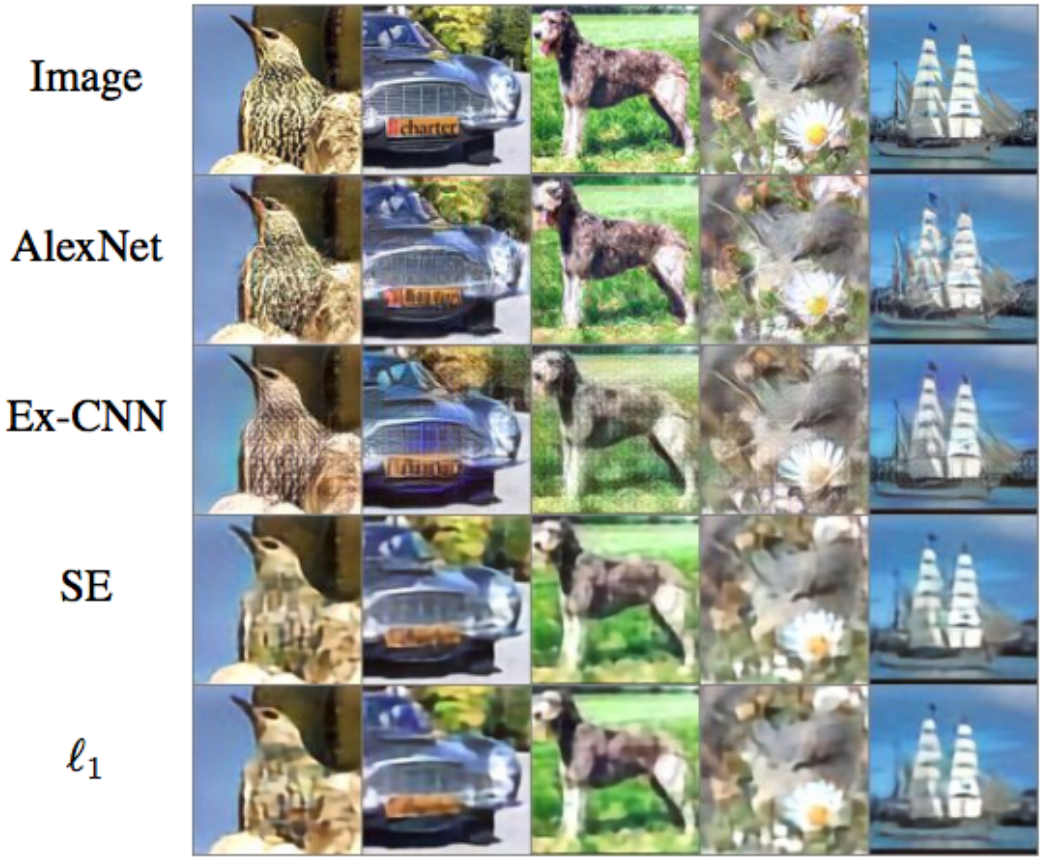
Figure 6. **CNN reconstruction.** Reconstruction of the image of Fig. 5.a from each layer of CNN-A. To generate these results, the regularization coefficient for each layer is chosen to match the highlighted rows in table 3. This figure is best viewed in color/screen.

Figure 9. **CNN receptive field.** Reconstructions of the image of Fig. 5.a from the central 5×5 neuron fields at different depths of CNN-A. The white box marks the field of view of the 5×5 neuron field. The field of view is the entire image for conv5 and relu5.

Основные выводы

- Как глубина влияет на степень реконструкции? (текстура, цвет, положение)
- Каков эффект max-pooling?
- Как можно использовать готовые классификаторы?

Эмпирические результаты



На практике

Действительно работает, настроить параметры несложно.

- Лучше всего использовать комбинацию loss-функций
- Image-to-image loss: L1
- Deep features loss: VGG 16/19
- Регуляризация: total variation

Но можно еще лучше!

Generative adversarial networks pt.1

Введем основные определения:

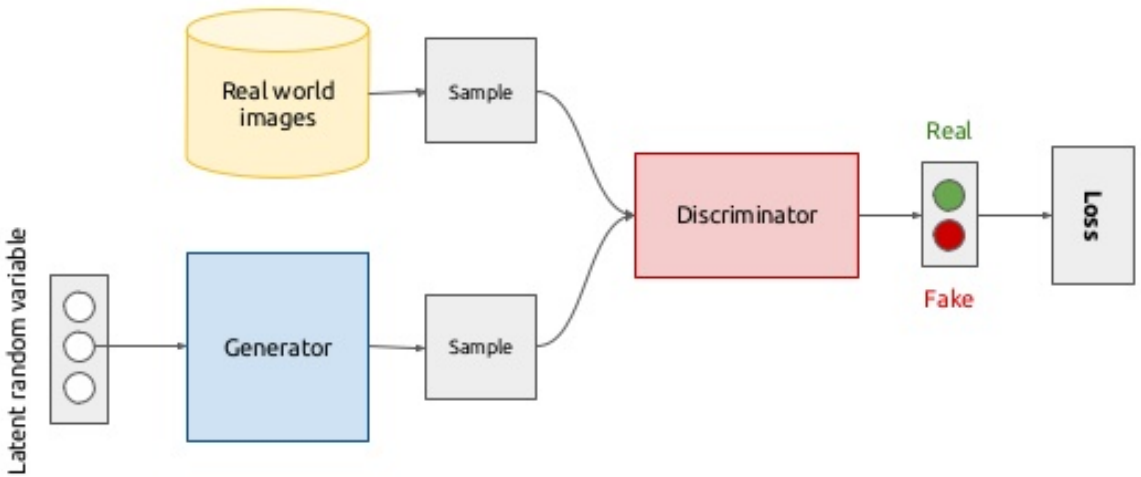
$z \sim p_z(z)$ - noise vector

$p_g(z)$- распределение сгенерированных картинок из noise

$p_{data}(x)$- распределение настоящих картинок

$G(z)$- генератор (генерирует картинку из z)

$D(x)$- дискриминатор (отличает реальные от сгенерированных)



Generative adversarial networks pt.2

Задача генератора - сгенерировать картинку, которую дискриминатор сочтет реалистичной.
Задача дискриминатора - отличить сгенерированную от реальной.
Математически - это игра двух игроков:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Generative adversarial networks pt.3

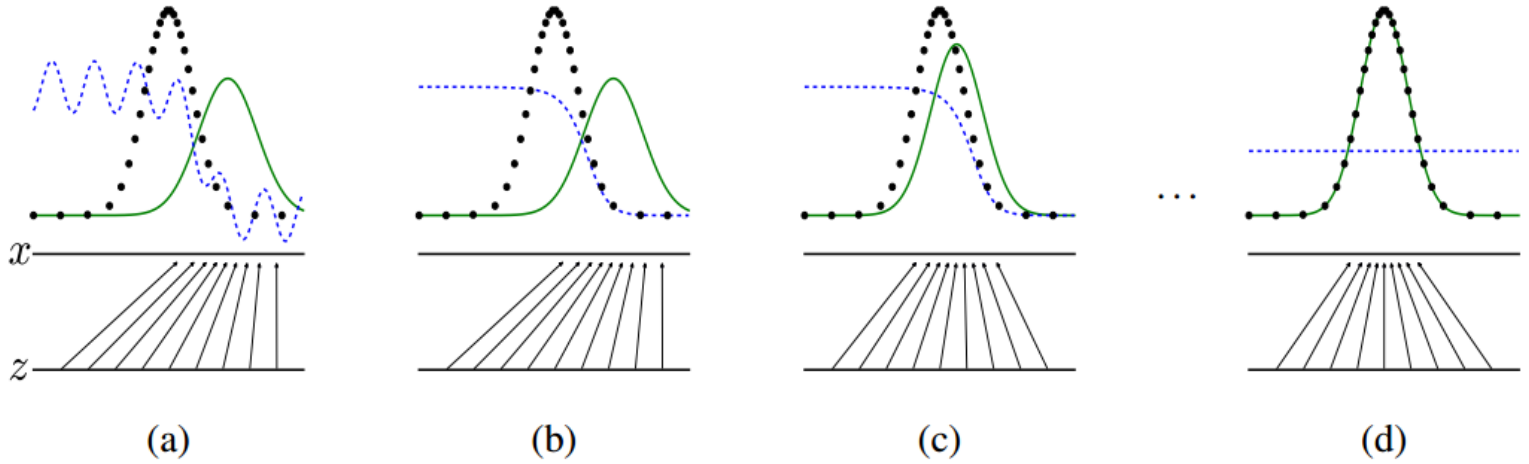


Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the **g**enerative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which $\boldsymbol{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$. (c) After an update to G, gradient of D has guided $G(\boldsymbol{z})$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(\boldsymbol{x}) = \frac{1}{2}$.

Theoretical results pt. 1

Theorem

Пусть G - фиксирован, тогда $D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$

Доказательство.

$$
\begin{aligned}
V(G,D) &= \mathop{\mathbb{E}}_{x\sim p_{data}(x)} \log D(x) + \mathop{\mathbb{E}}_{z\sim p_z(z)} \log(1-D(G(z))) \\
&= \int_x p_{data}(x)\log D(x) + p_g(x)\log(1-D(x))
\end{aligned} \tag{3}
$$

Теперь воспользуемся тем, что функционал $f(y) = a\log(y) + b\log(1-y)$ на $[0,1]$ достигает максимума в точке $\frac{a}{a+b}$. □

Theoretical results pt. 2

Подставим оптимальный дискриминатор в value-function

$$
\begin{aligned}
C(G) &= \int_x p_{data}(x) \log D^*(x) + p_g(x) \log(1 - D^*(x)) \\
&= \int_x p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \\
&+ p_g(x) \log \frac{p_g(x)}{p_{data}(x) + p_g(x)}
\end{aligned}
$$

Если подставить $p_{data}(x)$ вместо $p_g(x)$, получим:
$C(G) = -log4$, и если вынести $-log4$, то

$$
C(G) = -log4 + KL(p_{data} \parallel \frac{p_{data} + p_g}{2}) + KL(p_g \parallel \frac{p_{data} + p_g}{2}) \tag{4}
$$

Training algorithm

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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ 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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ 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.

Takeaway

Преимущества GAN

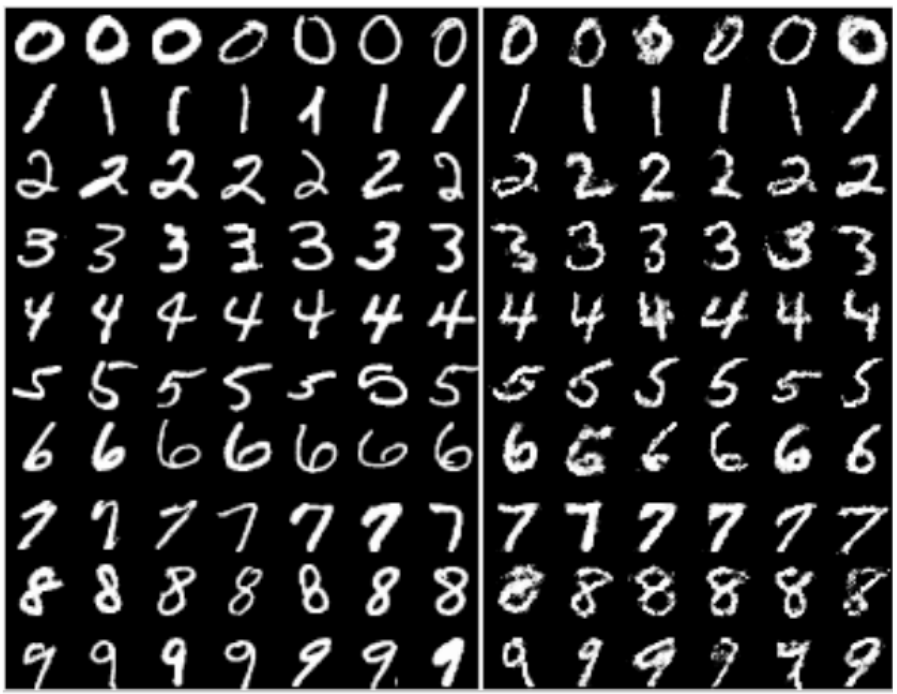
- Теоретические гарантии сходимости
- Можно обучать обычным SGD
- Решает в явном виде задачу generative modeling
- Но неявным образом (нейросети)

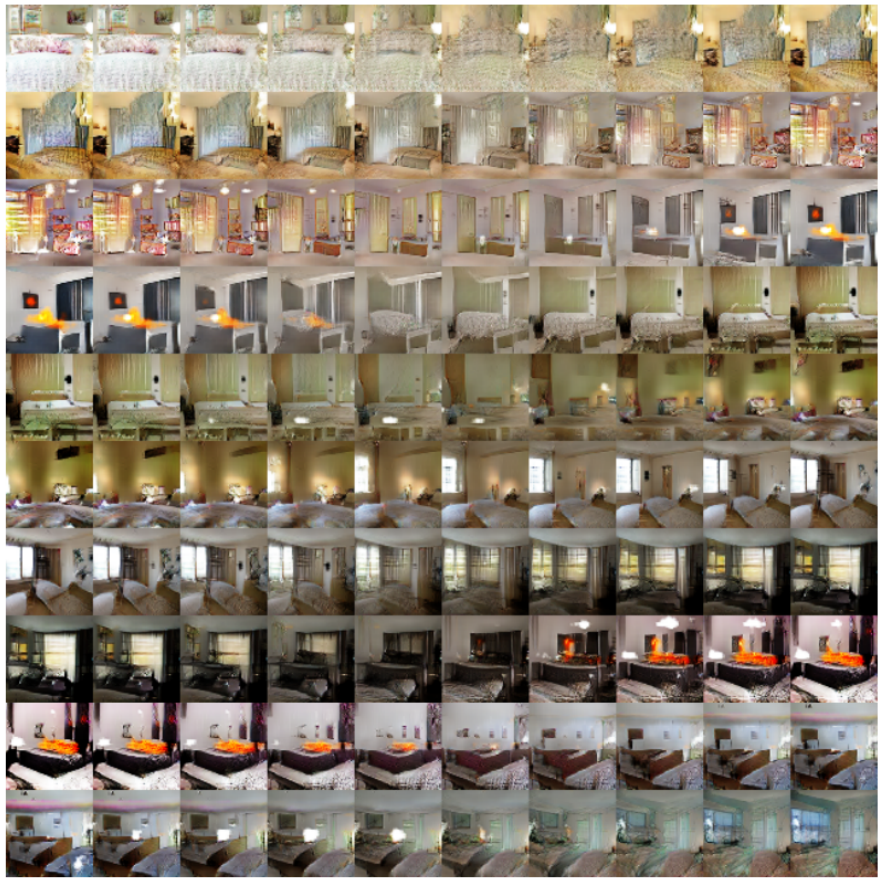
Недостатки GAN

- Нестабильное обучение
- Очень долгая сходимость
- Mode-collapsing
- Generator/Discriminator starvation
- Поиск оптимальных параметров - pure luck

Тем не менее, на практике GAN почти всегда дает заметные улучшения. Следующая лекция о том, как улучшить GAN.

Картинки для привлечения внимания





GAN gif

DCGAN

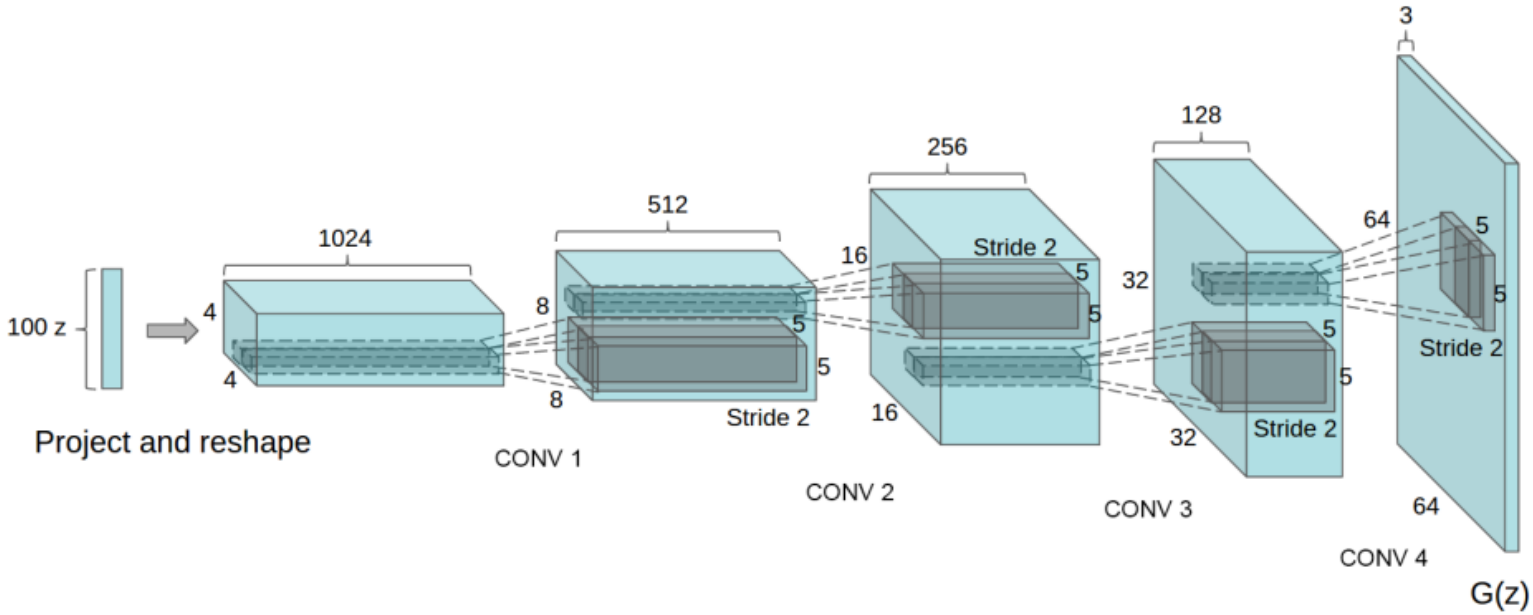
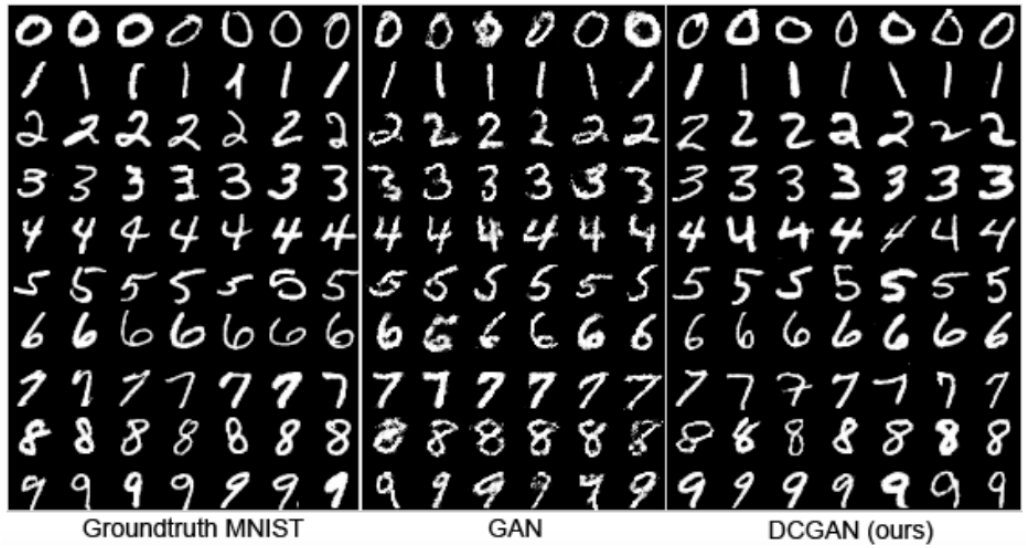


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

DCGAN results



DCGAN interpolation

Почему интерполяция возможна?

Domain transfer network

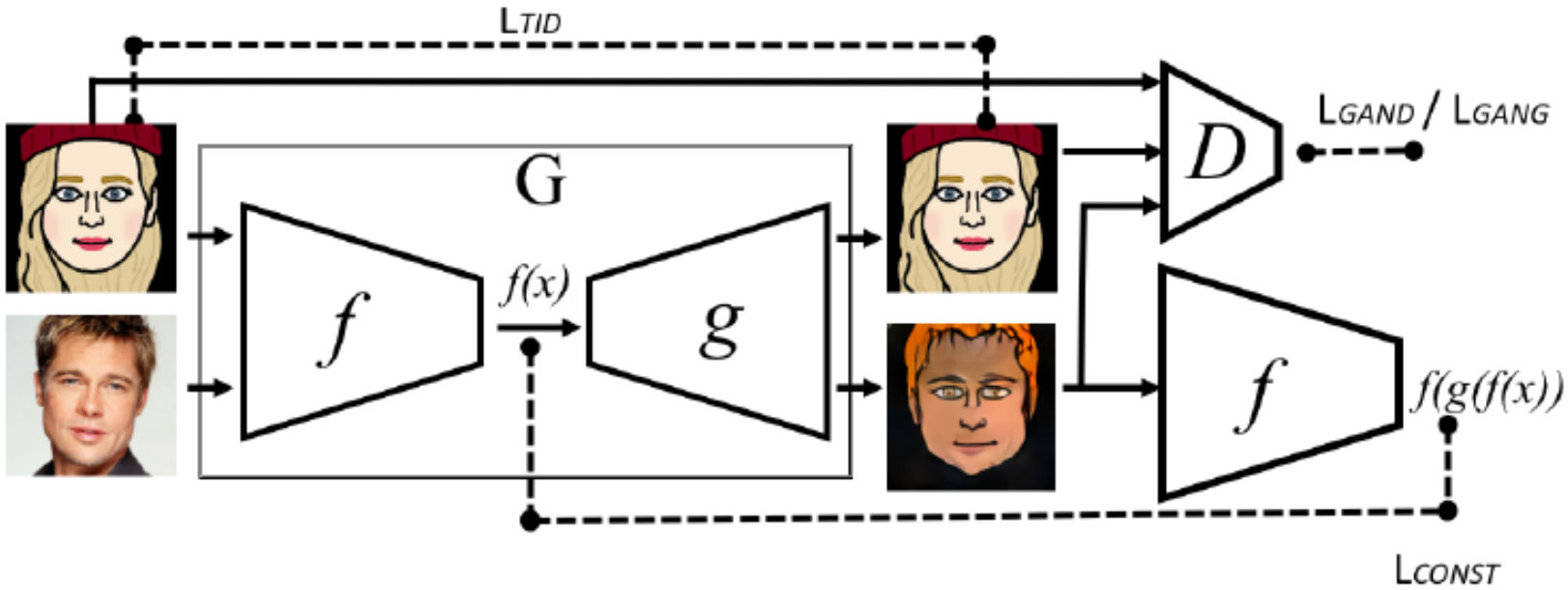


Figure 1: The Domain Transfer Network. Losses are drawn with dashed lines, input/output with solid lines. After training, the forward model G is used for the sample transfer.

$$L_{\mathrm{D}} = -\mathbb{E}_{x\in\mathbf{s}} \log D_1(g(f(x))) - \mathbb{E}_{x\in\mathbf{t}} \log D_2(g(f(x))) - \mathbb{E}_{x\in\mathbf{t}} \log D_3(x)$$

$$L_{\mathrm{GANG}} = -\mathbb{E}_{x\in\mathbf{s}} \log D_3(g(f(x))) - \mathbb{E}_{x\in\mathbf{t}} \log D_3(g(f(x)))$$

$$L_{\mathrm{CONST}} = \sum_{x\in\mathbf{s}} d(f(x), f(g(f(x))))$$

$$L_{\mathrm{TID}} = \sum_{x\in\mathbf{t}} d_2(x, G(x))$$

Results

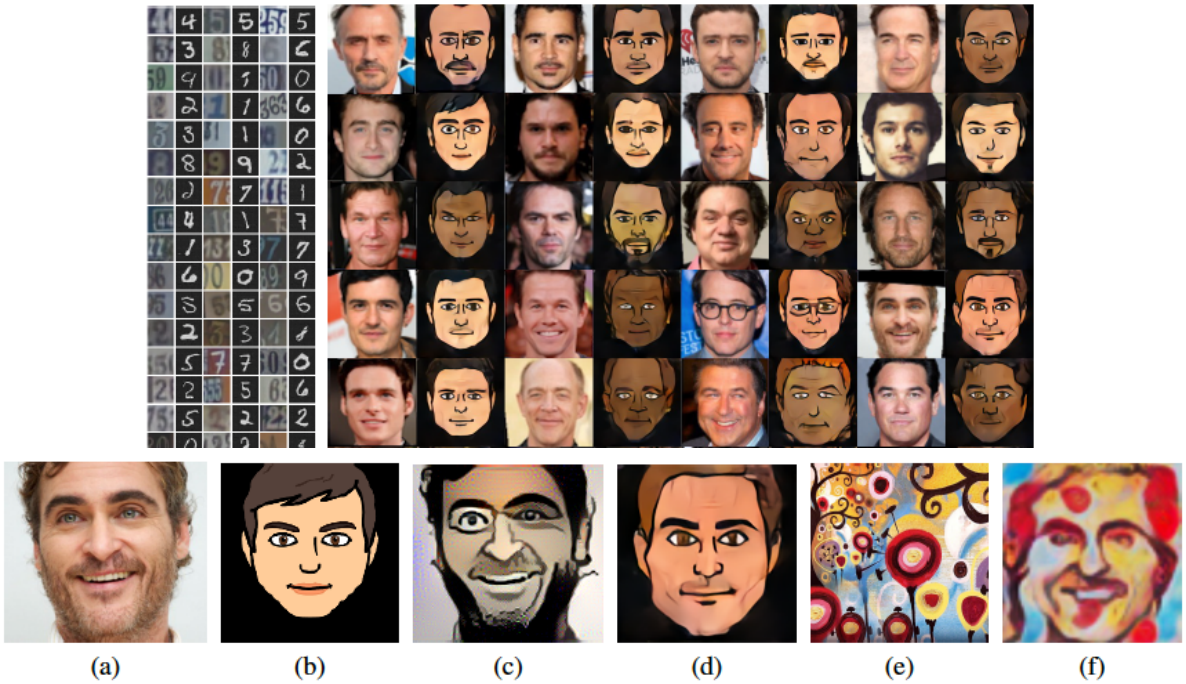
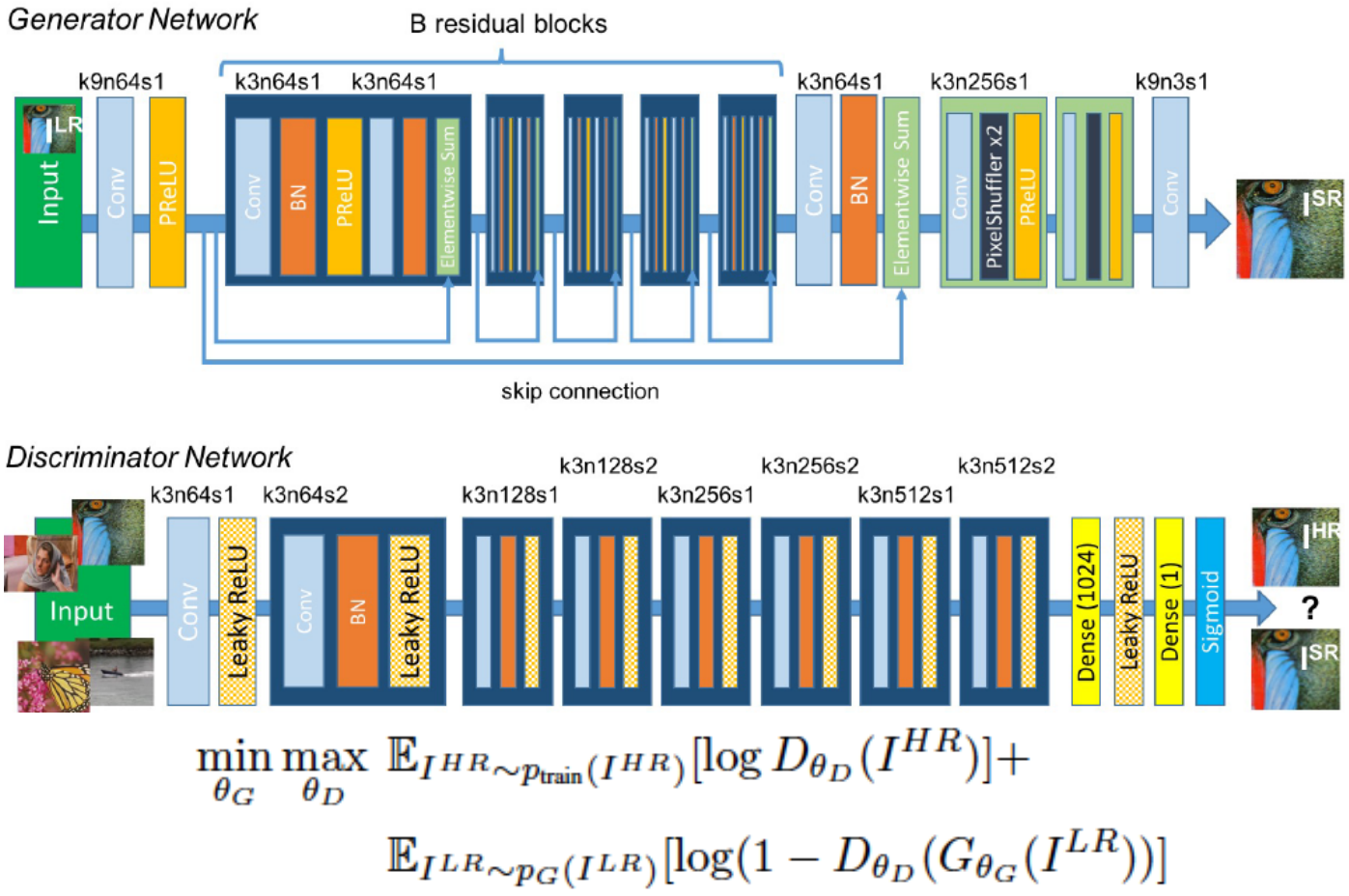


Figure 5: Style transfer as a specific case of Domain Transfer. (a) The input content photo. (b) An emoji taken as the input style image. (c) The result of applying the style transfer method of Gatys et al. (2016). (d) The result of the emoji DTN. (e) Source image for style transfer. (f) The result, on the same input image, of a DTN trained to perform style transfer.

Super resolution GAN



$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{I^{HR} \sim p_{\text{train}}(I^{HR})}[\log D_{\theta_D}(I^{HR})] +$$

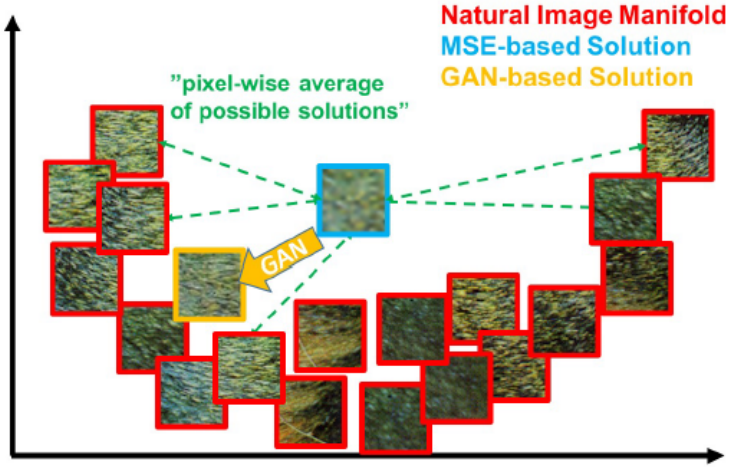
$$\mathbb{E}_{I^{LR} \sim p_G(I^{LR})}[\log(1 - D_{\theta_D}(G_{\theta_G}(I^{LR}))]$$

Можно ли решить как supervised задачу?

SRGAN

SRGAN
(21.15dB/0.6868)

original



Text to image

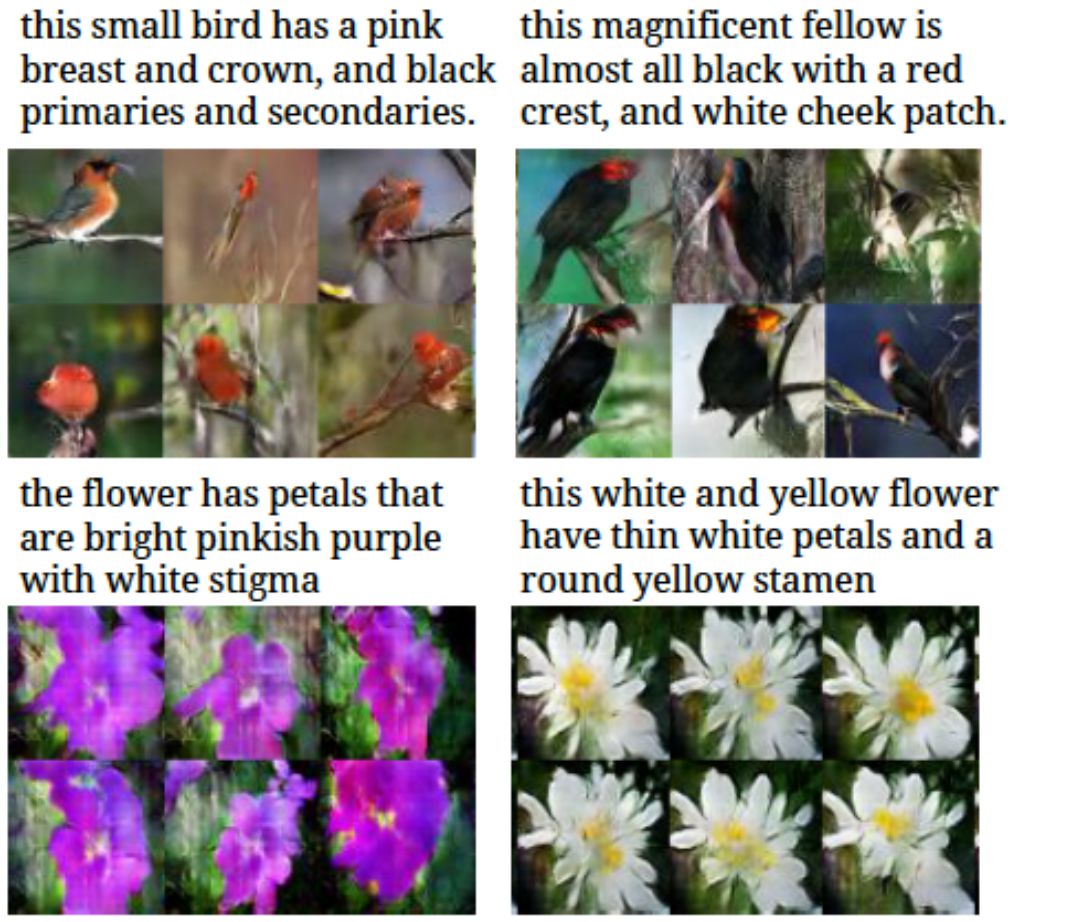
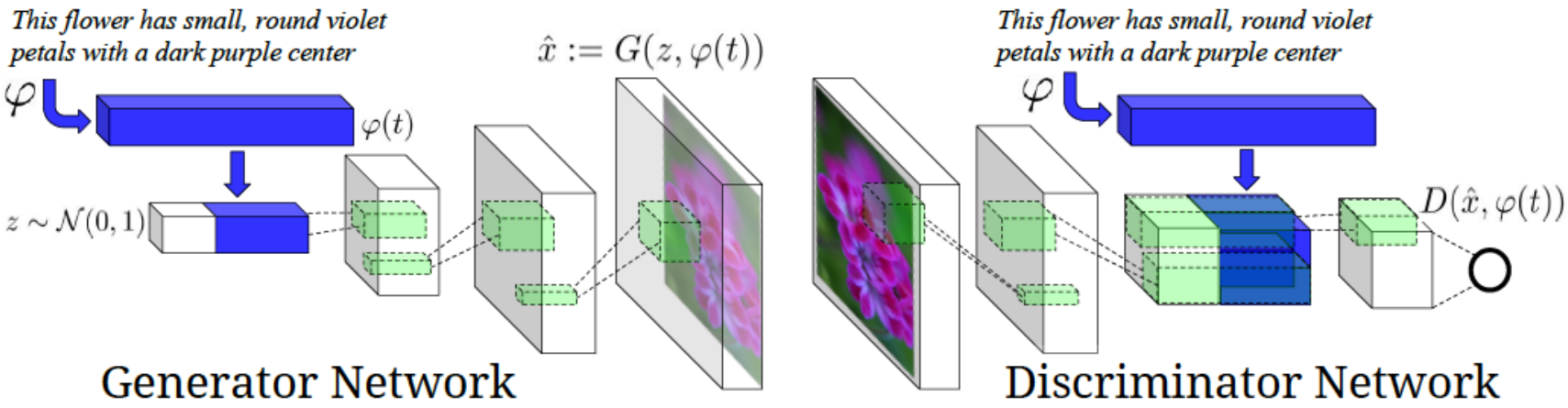


Figure 1. Examples of generated images from text descriptions. Left: captions are from zero-shot (held out) categories, unseen text. Right: captions are from the training set.

Text to image model



Text to image algo

Algorithm 1 GAN-CLS training algorithm with step size α, using minibatch SGD for simplicity.

Input: minibatch images x, matching text t, mismatching $\hat{t}$, number of training batch steps S
for $n = 1$ **to** S **do**
  $h \leftarrow \varphi(t)$ {Encode matching text description}
  $\hat{h} \leftarrow \varphi(\hat{t})$ {Encode mis-matching text description}
  $z \sim \mathcal{N}(0, 1)^Z$ {Draw sample of random noise}
  $\hat{x} \leftarrow G(z, h)$ {Forward through generator}
  $s_r \leftarrow D(x, h)$ {real image, right text}
  $s_w \leftarrow D(x, \hat{h})$ {real image, wrong text}
  $s_f \leftarrow D(\hat{x}, h)$ {fake image, right text}
  $\mathcal{L}_D \leftarrow \log(s_r) + (\log(1 - s_w) + \log(1 - s_f))/2$
  $D \leftarrow D - \alpha \partial \mathcal{L}_D / \partial D$ {Update discriminator}
  $\mathcal{L}_G \leftarrow \log(s_f)$
  $G \leftarrow G - \alpha \partial \mathcal{L}_G / \partial G$ {Update generator}
end for

Image to image

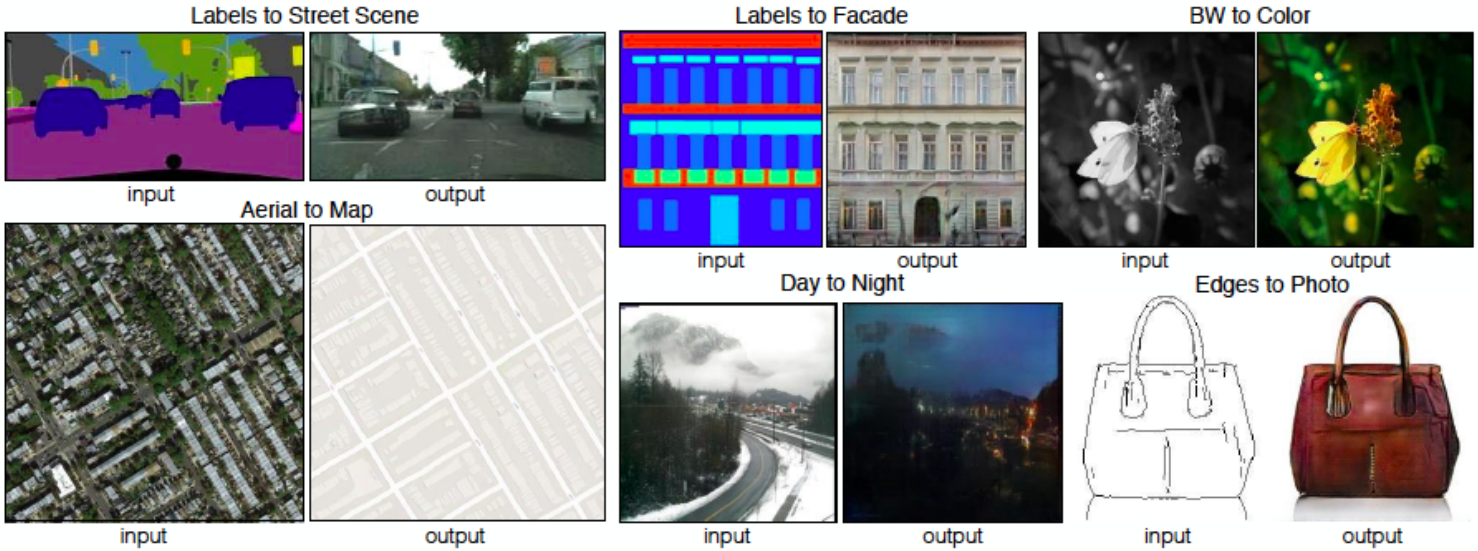


Image to image model

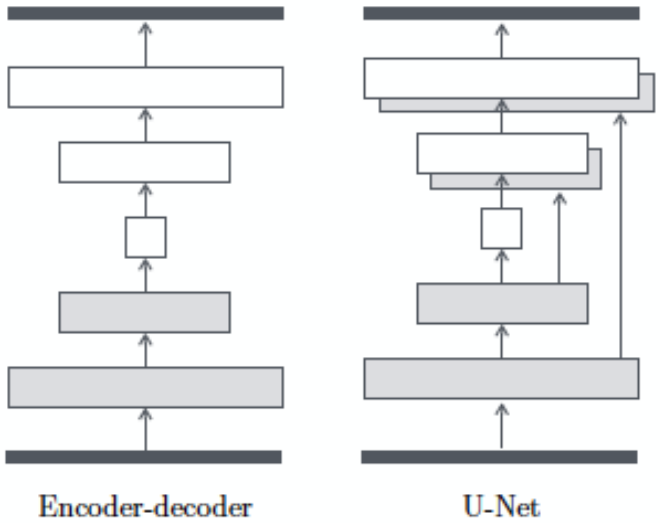
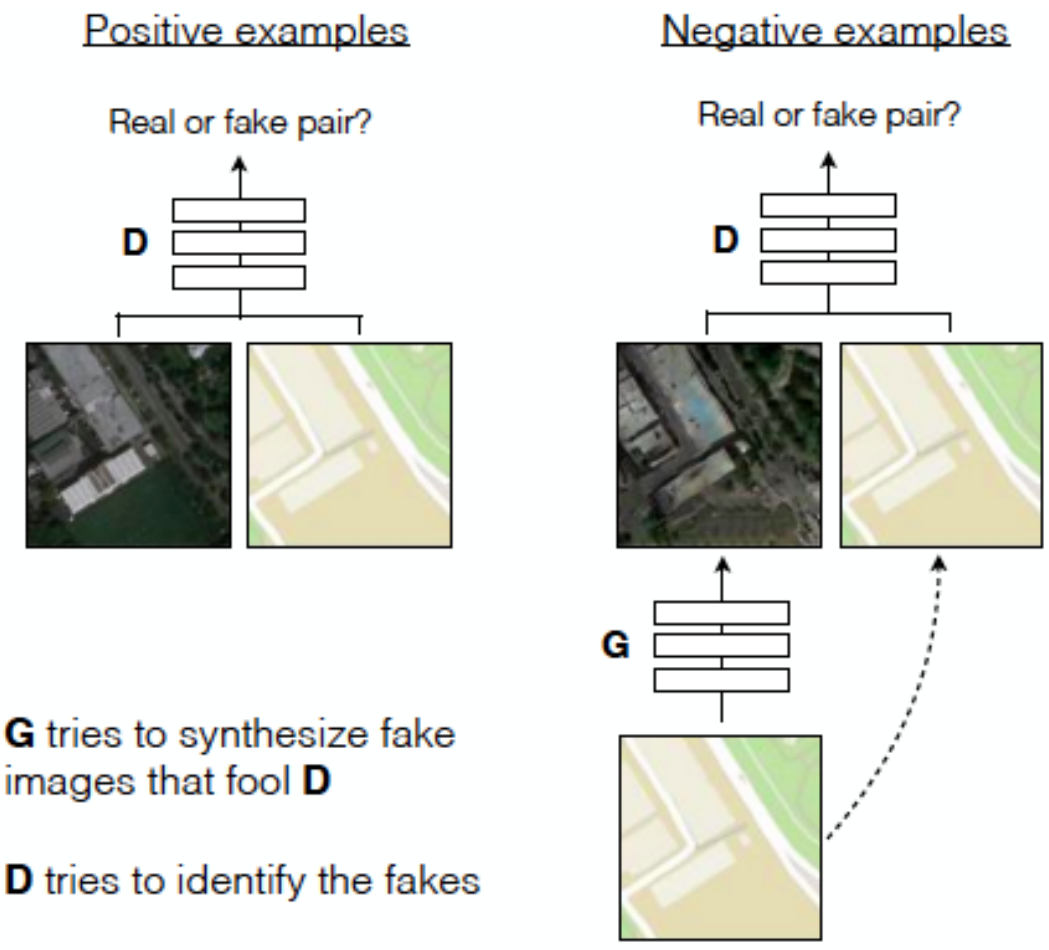
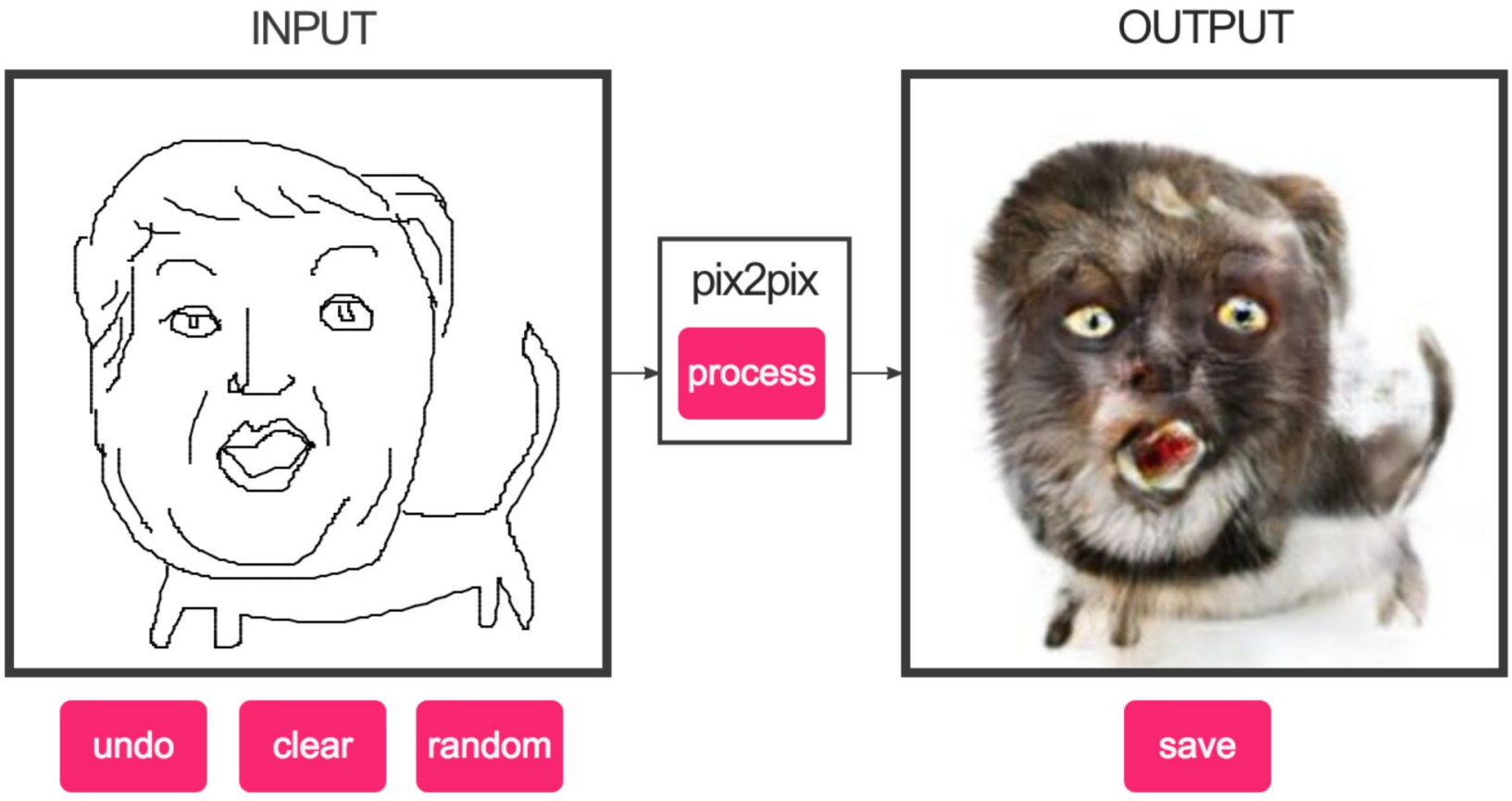


Figure 3: Two choices for the architecture of the generator. The "U-Net" [34] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

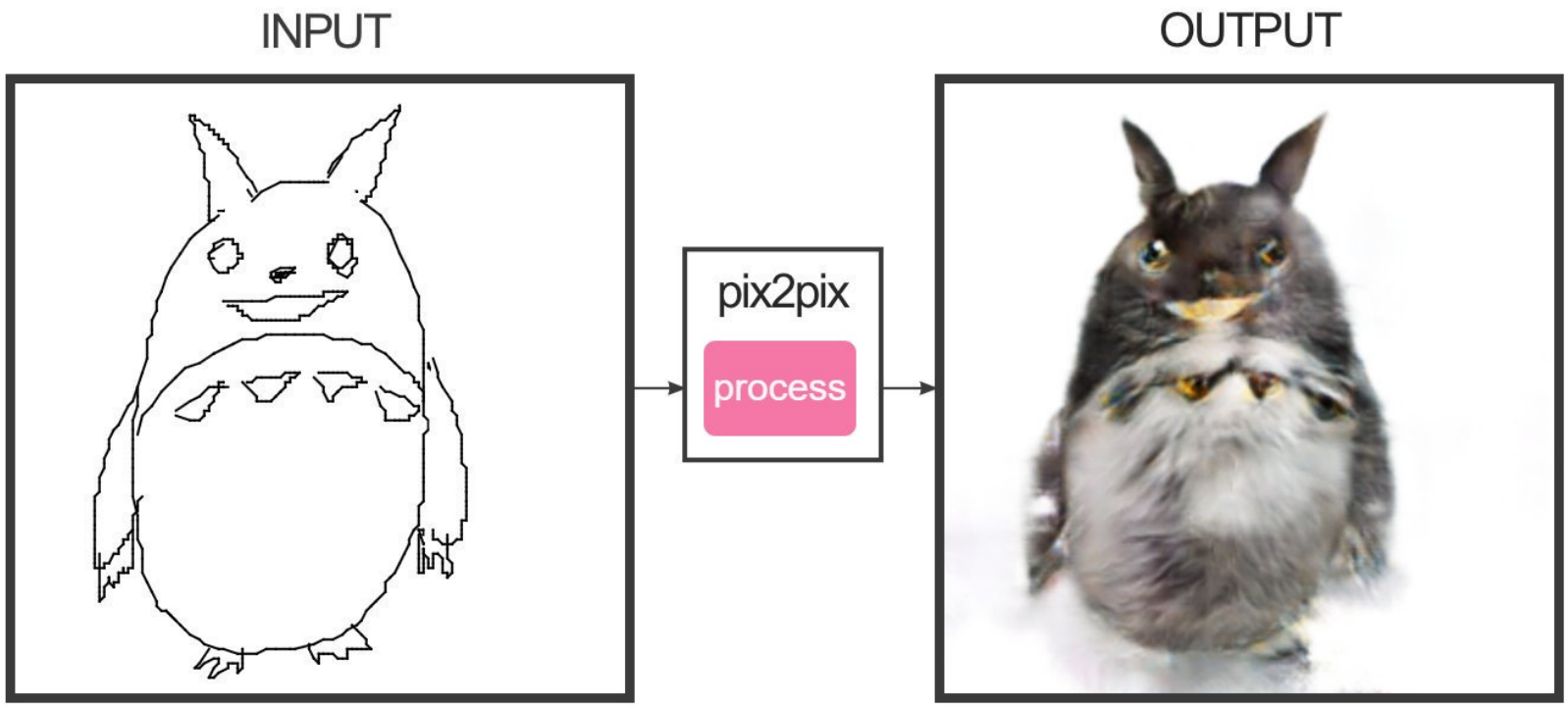
Image to image algo



MOAR IMAGES



MOAR IMAGES 2



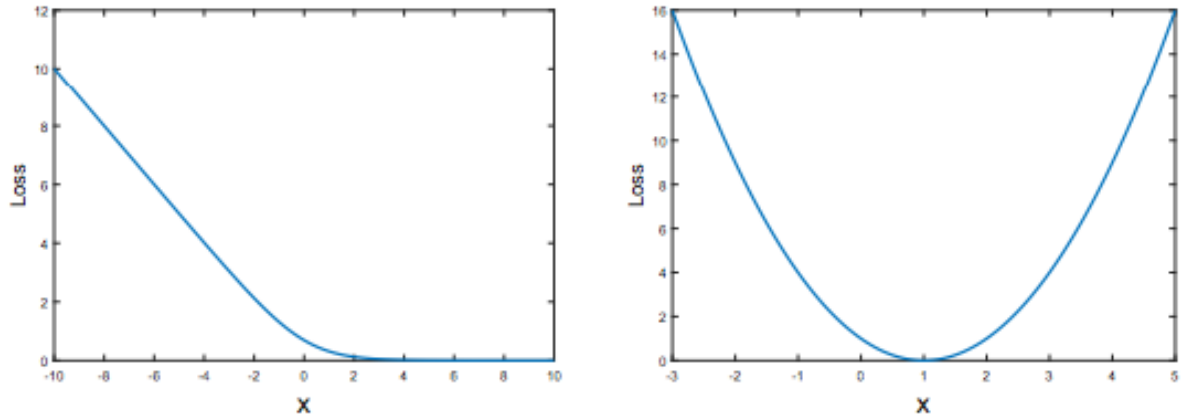
Problems

Visual

- Mode-collapsing
- Instability (long run)
- Noise generation (L1 лучше по началу)
- Bad converging (подбор гиперпараметров - боль)

Возможно, проблема в том, что мы оптимизируем что-то не то?

LSGAN



(a) (b)

Figure 2: (a): The sigmoid cross entropy loss function. (b): The least squares loss function.

$$\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x)) \tag{5}$$

LSGAN model

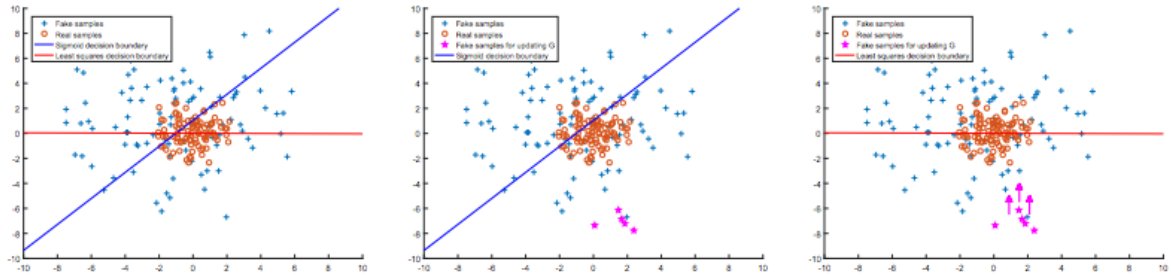


Figure 1: Illustration of different behaviors of two loss functions. (a): Decision boundaries of two loss functions. Note that the decision boundary should go across the real data distribution for a successful GANs learning. Otherwise, the learning process is saturated. (b): Decision boundary of the sigmoid cross entropy loss function. It gets very small errors for the fake samples (in magenta) for updateing G as they are on the correct side of the decision boundary. (c): Decision boundary of the least squares loss function. It penalize the fake samples (in magenta), and as a result, it forces the generator to generate samples toward decision boundary.

$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right]$$

$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right],$$

LSGAN results

(a) Church outdoor.

(b) Dining room.

(c) Kitchen.

(d) Conference room.

LSGAN problems

Model collapsing

Optimizer	BN_G Adam	BN_G RMSProp	BN_{GD} Adam	BN_{GD} RMSProp
Regular GANs	YES	NO	YES	YES
LSGANs	**NO**	NO	YES	**NO**

Wasserstein distance

- The *Jensen-Shannon* (JS) divergence

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \| \mathbb{P}_m) + KL(\mathbb{P}_g \| \mathbb{P}_m) \ ,$$

  where $\mathbb{P}_m$ is the mixture $(\mathbb{P}_r + \mathbb{P}_g)/2$. This divergence is symmetrical and always defined because we can choose $\mu = \mathbb{P}_m$.

- The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \big[\, \|x - y\| \, \big] \ , \tag{1}$$

  where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the "cost" of the optimal transport plan.

Wasserstein > KL

Example 1 (Learning parallel lines). Let $Z \sim U[0,1]$ the uniform distribution on the unit interval. Let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$ (a 0 on the x-axis and the random variable Z on the y-axis), uniform on a straight vertical line passing through the origin. Now let $g_\theta(z) = (\theta, z)$ with θ a single real parameter. It is easy to see that in this case,

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|$,
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$

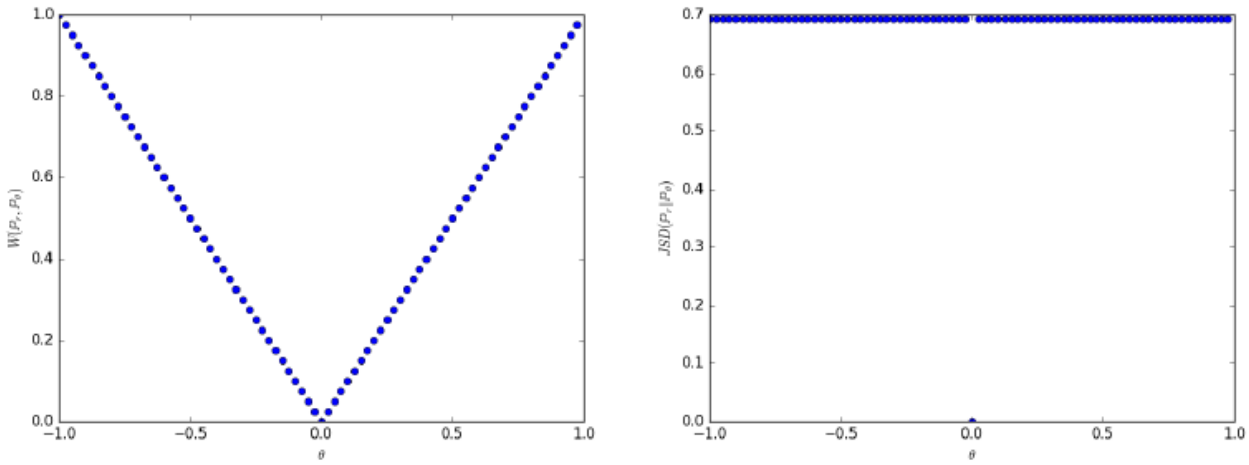


Figure 1: These plots show $\rho(\mathbb{P}_\theta, \mathbb{P}_0)$ as a function of θ when ρ is the EM distance (left plot) or the JS divergence (right plot). The EM plot is continuous and provides a usable gradient everywhere. The JS plot is not continuous and does not provide a usable gradient.

Wasserstein training

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

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

Счастье, радость, сходимость

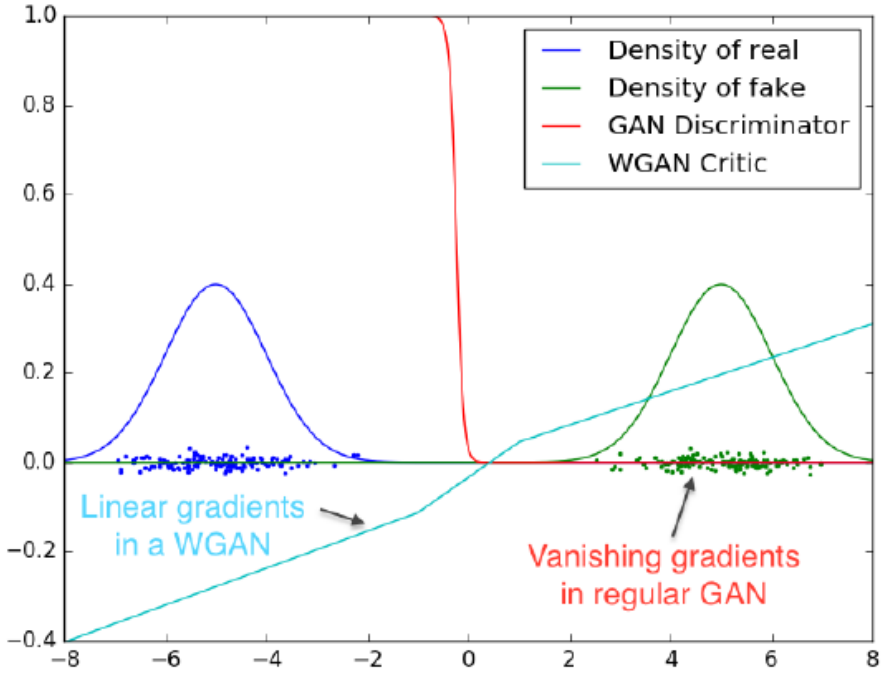


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the traditional GAN discriminator saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

Счастье, радость, сходимость 2

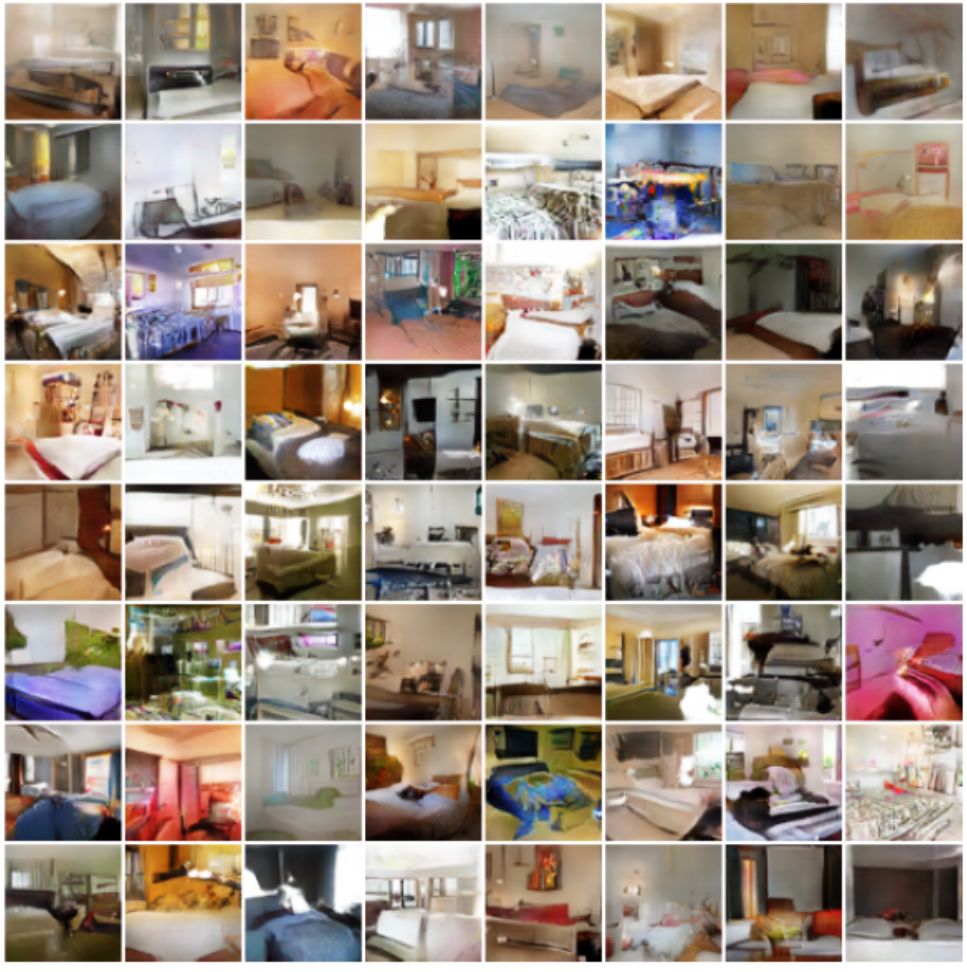


Figure 9: WGAN algorithm: generator and critic are DCGANs.

Вопросы