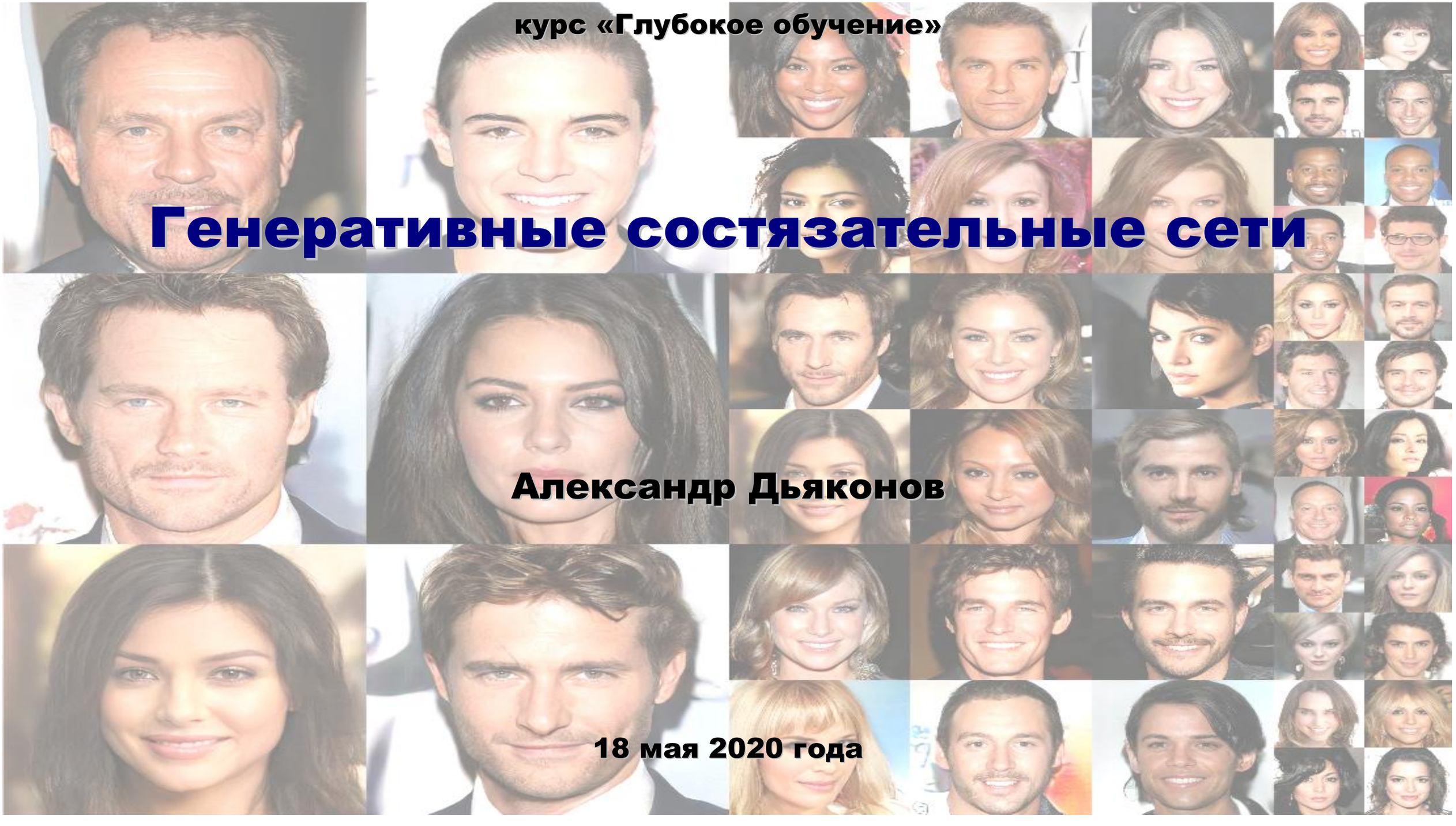
курс «Глубокое обучение»



Генеративные состязательные сети

Александр Дьяконов

18 мая 2020 года

План

Generative Adversarial Networks (GAN)

Генератор и дискриминатор

Что могут GAN

Обучение – min-max-игра

LSGAN, WGAN, WGAN-GP, EBGAN,
DCGAN, cGAN, CycleGAN, BiGAN,
BigGAN, SAGAN, CAN, ProGAN,
InfoGAN, Coupled GANs

Как оценивать качество (сгенерированные картинки)

Generative Adversarial Networks (GAN)

Есть фундаментальная проблема – учим генерировать «котиков»
– как измерить качество модели?

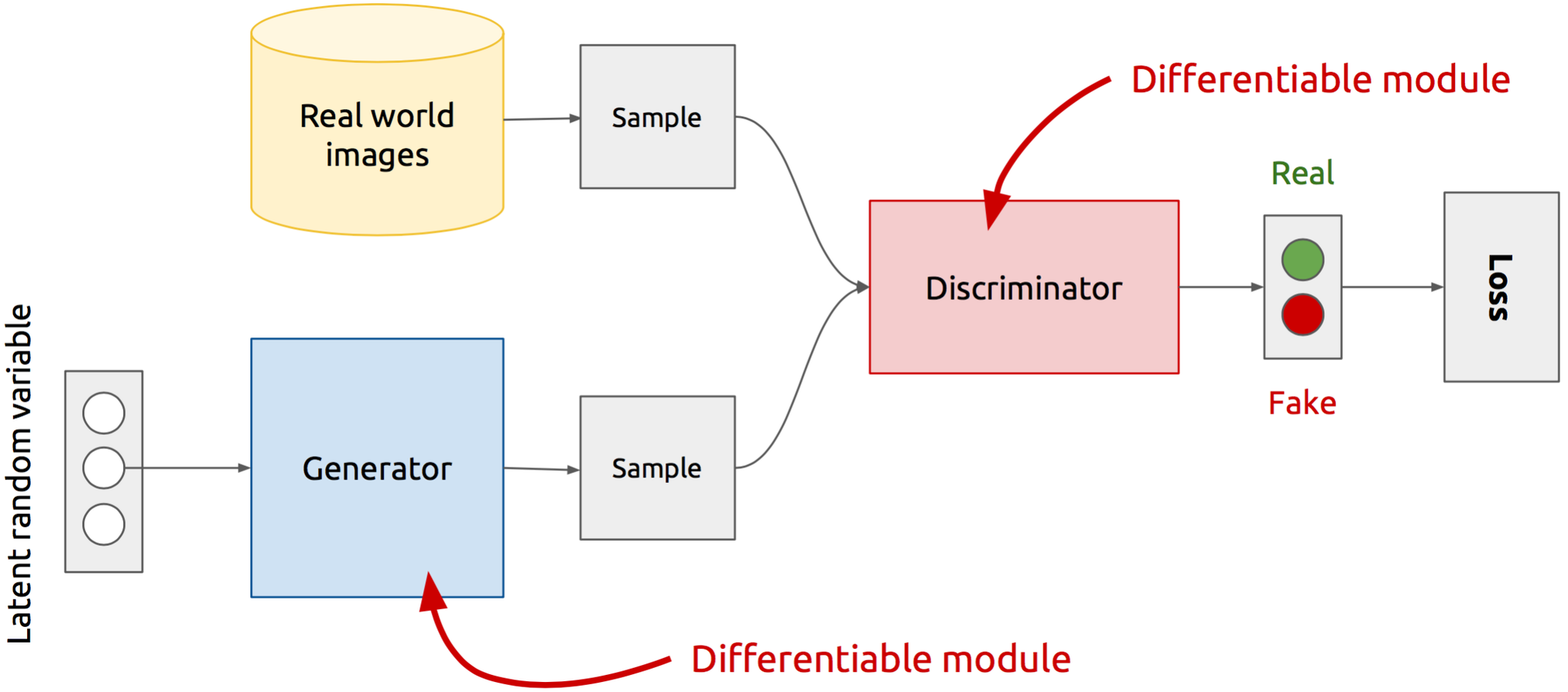
Generative ~ учим генеративную модель
Adversarial ~ в состязательной парадигме
Networks ~ DNN

Модели

Дискриминативные	Генеративные
оценивают P(X|Y)	моделируют P(X)

[Goodfellow et al., 2014]

GAN: Генератор и дискриминатор



Генератор – сеть, которая порождает объект (изображение) из шума

Дискриминатор – сеть, различающая настоящие и сгенерированные объекты

Generative Adversarial Networks (GAN)

- **главное: по сути, дискриминатор – дифференцируемая функция ошибки!**

эту идею можно использовать там, где нет подходящих функций ошибок…

- **в отличие от PixelCNN, VAE нет явной функции плотности!**

- **игровой подход!**

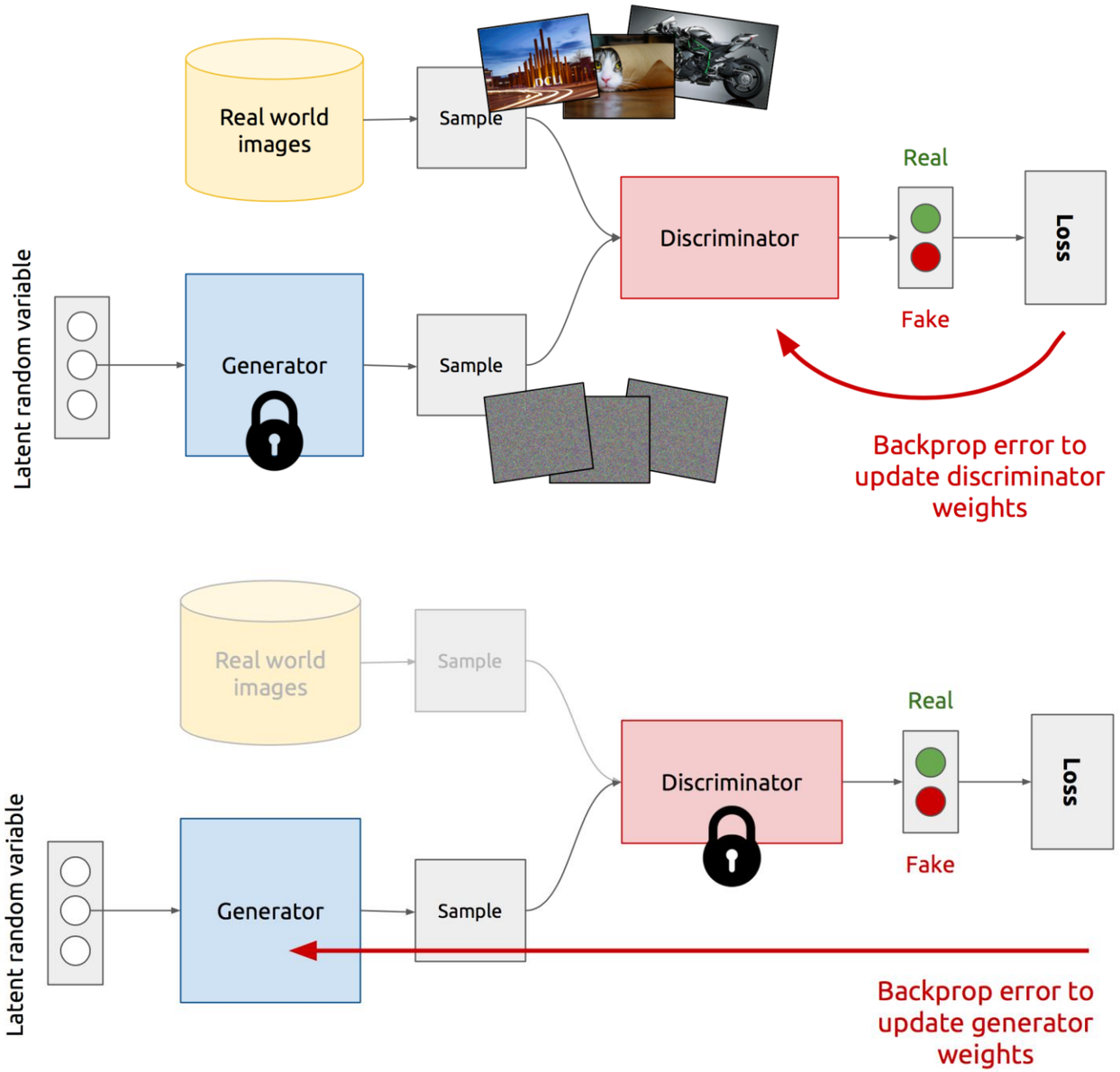
Что могут GAN

- научились работать со сложными объектами в пространствах высокой размерности
- генерация реалистичных объектов
- заполнение пропусков (и другие задачи USL)
- использование генеративных моделей-ассистентов (при редактировании)

Задачи

- улучшение изображений (Image Inpainting)
- улучшение звука (speech enhancement)
- генерация изображений (Image Generation)
- супер-разрешение (Super-resolution)
- раскраска изображений (colorization)
- творчество (artwork)
- пополнение датасета (synthetic data), моделирование (simulation)

GAN: обучение



GAN: обучение – min-max-игра

$$\min_{\theta}\max_{\varphi}\left[\mathbf{E}_{x\sim p_{\text{data}}}\log D_{\varphi}(x)+\mathbf{E}_{z\sim p(z)}\log(1-D_{\varphi}(G_{\theta}(x))\right]$$

Дискриминатор выводит правдоподобие из [0, 1]

$D_{\varphi}(x)$ **– для настоящих данных**

$D_{\varphi}(G_{\theta}(x))$ **- для сгенерированных данных**

Дискриминатор хочет $D_{\varphi}(x)\to 1,\ D_{\varphi}(G_{\theta}(x))\to 0$

Генератор хочет $D_{\varphi}(G_{\theta}(x))\to 1$

для оптимизации лучше:

$$\max_{\varphi}\left[\mathbf{E}_{x\sim p_{\text{data}}}\log D_{\varphi}(x)+\mathbf{E}_{z\sim p(z)}\log(1-D_{\varphi}(G_{\theta}(x))\right]$$

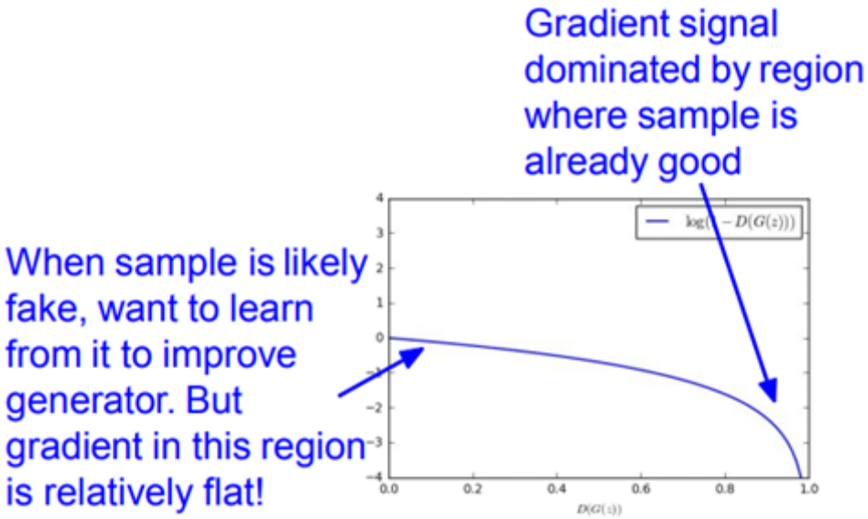
$$\max_{\theta}\mathbf{E}_{z\sim p(z)}\log(D_{\varphi}(G_{\theta}(x)))$$

кстати, оптимальный дискриминатор:

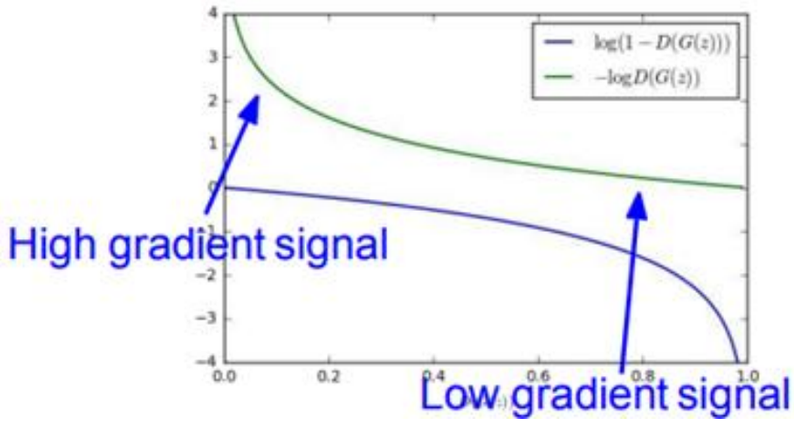
$$D^{*}(x)=\frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_{\text{G}}(x)}$$

Почему так лучше для оптимизации

$$\min_{\theta} \mathbf{E}_{z\sim p(z)} \log(1 - D_{\varphi}(G_{\theta}(x)))$$



$$\max_{\theta} \mathbf{E}_{z\sim p(z)} \log(D_{\varphi}(G_{\theta}(x)))$$



вместо min правдоподобия корректности дискриминатора – max правдоподобия того, что дискриминатор ошибался

Алгоритм настройки

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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

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 ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

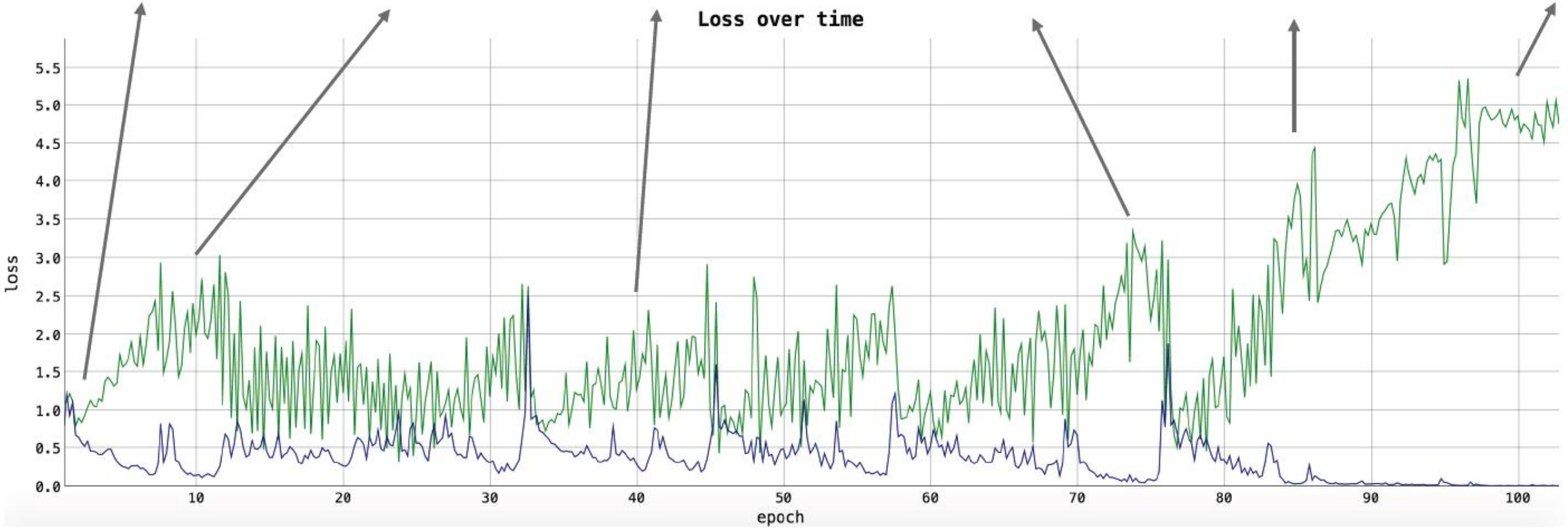
GAN: примеры

Goodfellow, Ian, et al. «Generative adversarial nets» Advances in neural information processing systems. 2014.

Обучение GAN





проблемка...

Настройка GAN

Очень сложно обучать

Есть проблемы со стабильностью

плюс дальше опишем сложности

$\Rightarrow$ **много трюков и приёмов**

например для стабильности добавляем шум,

но тогда изображения размываются

в дискриминативной постановке решаем невыпуклую задачу оптимизации – можем сойтись в локальный, а не глобальный минимум

В состояние равновесия можем вообще не попасть

(SGD не для этого, плохой пример – $\min\limits_x \max\limits_y xy$**)**

Функция ошибки

$$\min_{\theta}\max_{\varphi}\left[\mathbf{E}_{x\sim p_{\text{data}}}\log D_{\varphi}(x)+\mathbf{E}_{z\sim p(z)}\log(1-D_{\varphi}(G_{\theta}(x))\right]$$

$$D_{\varphi}(x)\ ,\ D_{\varphi}(G_{\theta}(x))$$

можно показать, что минимизируется расстояние Йенсена-Шеннона

JSD (Jensen-Shannon Divergence)

$$\text{JSP}(p\,\|\,q)=\frac{\text{KL}(p\,\|\,(p+q)/2)+\text{KL}(q\,\|\,(p+q)/2)}{2}$$

проблемы, если носители распределений не пересекаются

градиенты затухают, когда D обучился

Советы по настройке GAN

1. Нормализация входа (изображения → [–1, +1], выход - tanh)
2. Вместо min(log(1-D(G))) лучше max(log(D(G)))
   приём: меняем метки местами real ↔ fake
3. z сэмплируется не из равномерного, а гауссовского распределения, см. также https://arxiv.org/abs/1609.04468
4. Минибатчи лучше делать чистыми (все real или все fake)
5. Не использовать Sparse Gradients (ReLU, MapPool)
лучше LeakyReLU,
Downsampling: Average Pooling, Conv2d + stride
Upsampling: PixelShuffle, ConvTranspose2d + stride
6. Лучше размывать метки: 0 → [0, 0.3], 1 → [0.7, 1.2]
   чаще только 1 размывают
7. Используйте DCGAN (или гибридные: KL + GAN или VAE + GAN)
8. Используйте трюки из RL (например, Experience Replay)
9. Adam для генератора, SGD для дискриминатора
10. Мониторьте ошибки (ex: loss(D)~0 failure mode – проверяйте градиенты)

Советы по настройке GAN

11. Добавляйте шум ко входу / к слоям генератора / меткам

12. Дискретные переменные в условных GANax:
используйте Embedding layer, добавляйте как новый канал в изображениях, поддерживайте низкой embedding dimensionality

13. Используйте Dropouts в G

14. Virtual batch normalization (VBN)

статистика считается на референсном батче (но его приходится каждый раз прогонять дополнительно через НС)

15. Регуляризация нормы градиента

16. Новые loss-функции

17. Пусть дискриминатор определяет не только фейк / не фейк, а ещё и класс

Salimans T., et al «Improved Techniques for Training GANs» 2016
https://arxiv.org/pdf/1606.03498.pdf

https://github.com/soumith/ganhacks

Пример настройки GAN https://poloclub.github.io/ganlab/

GAN в зависимости от функции ошибки в дискриминаторе

ф-я ошибки	вид GAN
Binary cross-entropy	Vanilla GAN
Least squares (L2)	LSGAN
EM-distance / Wasserstein-1	WGAN
Wasserstein GAN + Gradient penalty	WGAN-GP

Least Square GAN

квадратичная функция ошибки

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

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

Mao et al «Least Squares Generative Adversarial Networks» //
https://arxiv.org/pdf/1611.04076.pdf

Wasserstein GAN (WGAN)

Earth-Mover (EM) distance (Wasserstein-1)

$$W(p, p') = \inf \mathbf{E}_{(x,y)\sim\Pi(p,p')} \| x - y \|$$

интерпретация – как «выровнять» распределения

если мы как бы переносим кучи песка…

нет проблемы с пустыми пересечениями носителей

Двойственность Монжа-Канторовича (Kantorovich-Rubinstein duality)

$$W(p, p') = \sup_{\|f\|_L \le 1} \mathbf{E}_{x\sim p} f(x) - \mathbf{E}_{x\sim p'} f(x)$$

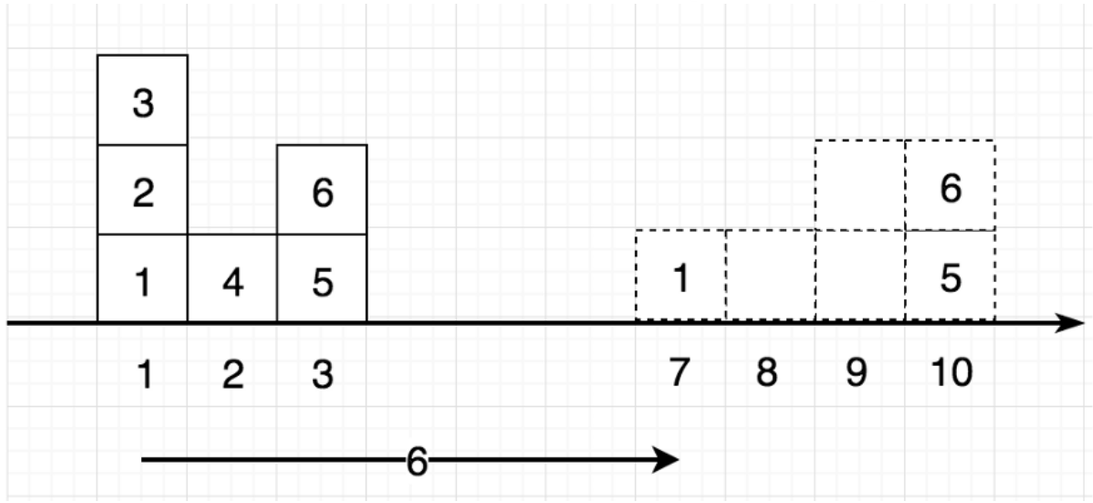
Для $\| f \|_L \le 1$ **обрезаем градиенты (и работает!)**

Дискриминатор ~ средство для оценки расстояния между распределениями

Martin Arjovsky, Soumith Chintala, Léon Bottou «Wasserstein GAN»

https://arxiv.org/abs/1701.07875

Wasserstein GAN (WGAN)



стоимость переноса = вес·расстояние

	Discriminator/Critic	Generator
GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$
WGAN	$\nabla_w \frac{1}{m} \sum_{i=1}^{m} \left[f\left(x^{(i)}\right) - f\left(G\left(z^{(i)}\right)\right) \right]$	$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f\left(G\left(z^{(i)}\right)\right)$

https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

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

Wasserstein GAN

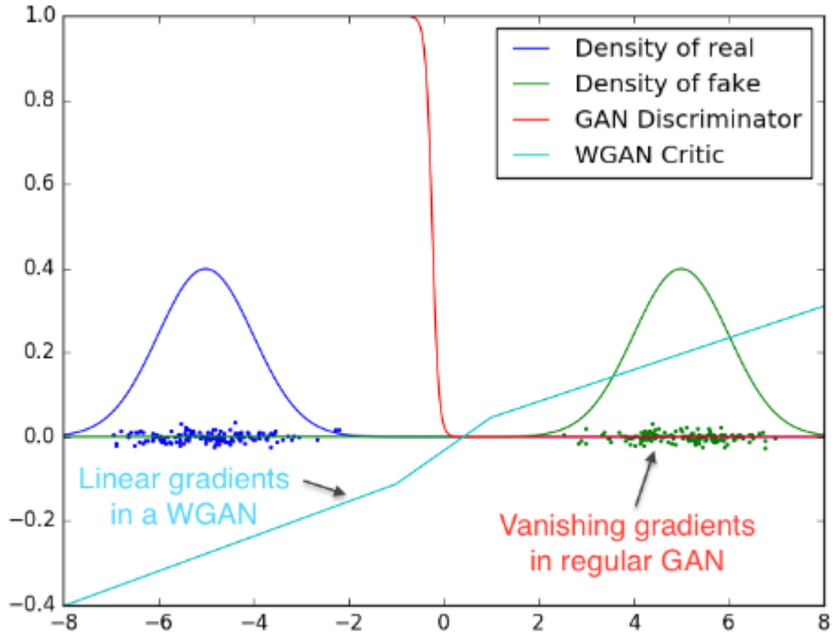


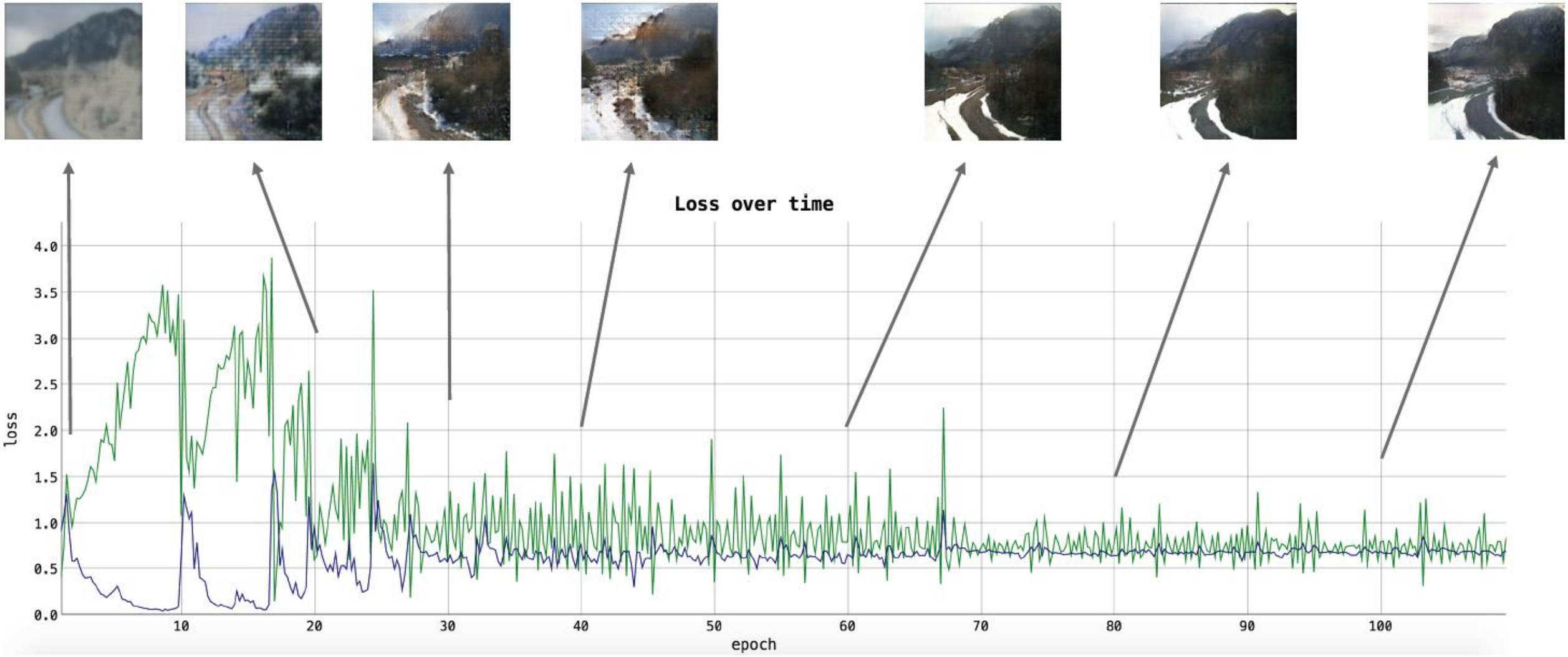
Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

здесь дискриминатор переименовали в критика

нет эффекта коллапса см. дальше

генератор продолжает обучаться, когда критик работает хорошо

Wasserstein GAN



ошибка прямо отражает качество

Figure 5: Algorithms trained with a DCGAN generator. Left: WGAN algorithm. Right: standard GAN formulation. Both algorithms produce high quality samples.

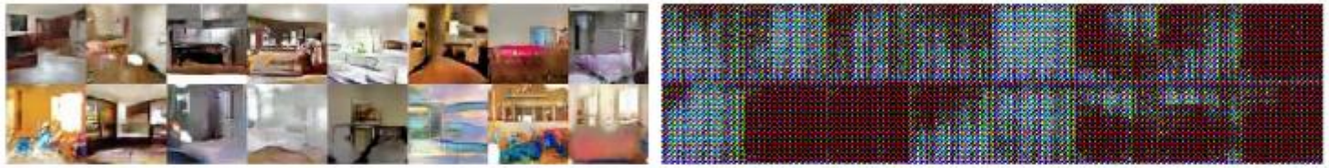


Figure 6: Algorithms trained with a generator without batch normalization and constant number of filters at every layer (as opposed to duplicating them every time as in [18]). Aside from taking out batch normalization, the number of parameters is therefore reduced by a bit more than an order of magnitude. Left: WGAN algorithm. Right: standard GAN formulation. As we can see the standard GAN failed to learn while the WGAN still was able to produce samples.

Figure 7: Algorithms trained with an MLP generator with 4 layers and 512 units with ReLU nonlinearities. The number of parameters is similar to that of a DCGAN, but it lacks a strong inductive bias for image generation. Left: WGAN algorithm. Right: standard GAN formulation. The WGAN method still was able to produce samples, lower quality than the DCGAN, and of higher quality than the MLP of the standard GAN. Note the significant degree of mode collapse in the GAN MLP.

WGAN-GP

вместо обрезания – мягкий штраф на градиенты – «Gradient Penalty»

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}\left[D(\tilde{\boldsymbol{x}})\right] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}\left[D(\boldsymbol{x})\right]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{\boldsymbol{x}}}}\left[\left(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1\right)^2\right]}_{\text{Our gradient penalty}}.$$

нет critic batch normalization

устойчивость к выбору архитектуры

потом ипользовался в SOTA-моделях (Prog GAN, Style GAN и т.п.)

Gulrajani et al «Improved Training of Wasserstein GANs» // https://arxiv.org/pdf/1704.00028.pdf

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			

Figure 2: Different GAN architectures trained with different methods. We only succeeded in training every architecture with a shared set of hyperparameters using WGAN-GP.

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 $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **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

GAN: проблемы – Mode-Collapse

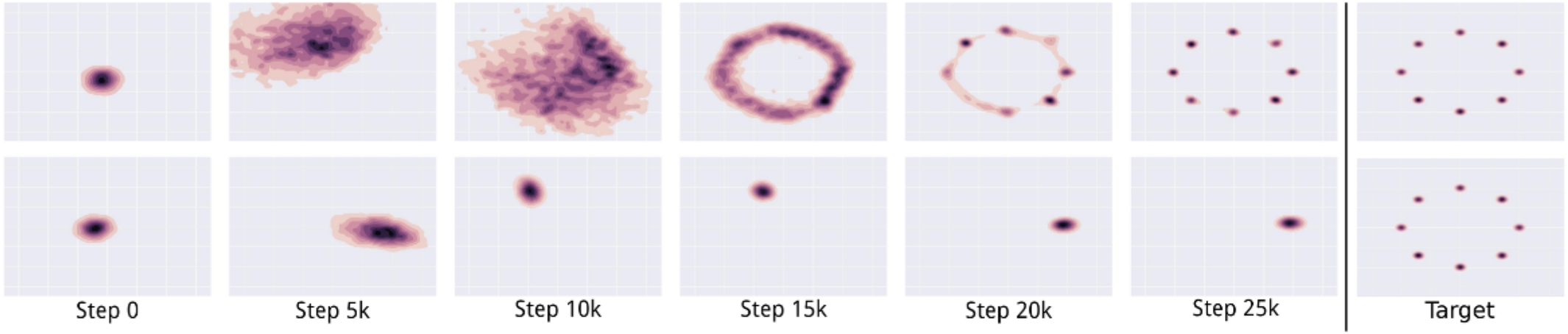


Figure 2: Unrolling the discriminator stabilizes GAN training on a toy 2D mixture of Gaussians dataset. Columns show a heatmap of the generator distribution after increasing numbers of training steps. The final column shows the data distribution. The top row shows training for a GAN with 10 unrolling steps. Its generator quickly spreads out and converges to the target distribution. The bottom row shows standard GAN training. The generator rotates through the modes of the data distribution. It never converges to a fixed distribution, and only ever assigns significant probability mass to a single data mode at once.

нижний ряд – «типичный GAN», нет разнообразия в ответах (sample diversity) см ниже

Metz, Luke, et al. «Unrolled Generative Adversarial Networks» //

https://arxiv.org/pdf/1611.02163.pdf

GAN: проблемы – Mode-Collapse

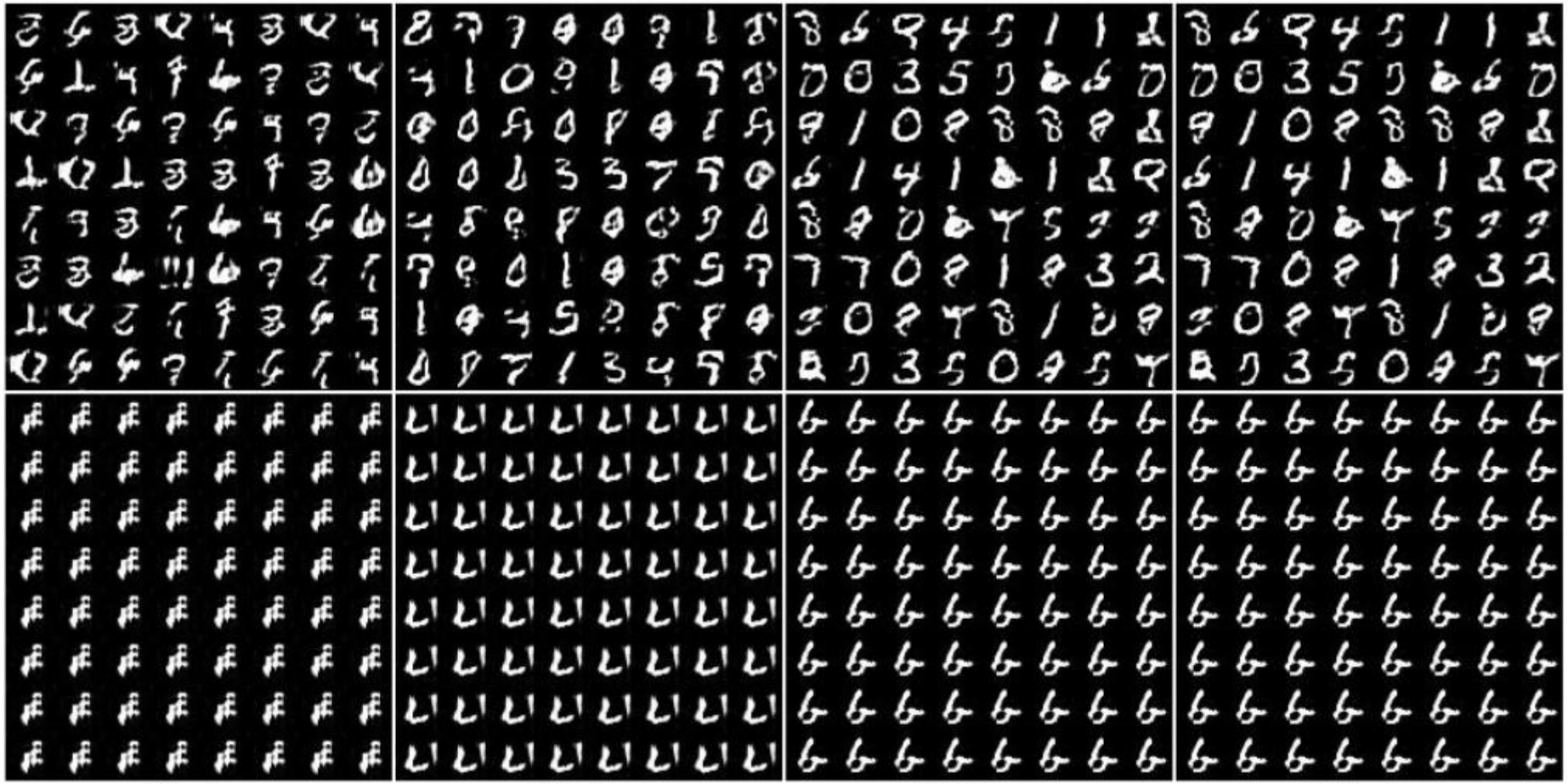


Figure 3: Unrolled GAN training increases stability for an RNN generator and convolutional discriminator trained on MNIST. The top row was run with 20 unrolling steps. The bottom row is a standard GAN, with 0 unrolling steps. Images are samples from the generator after the indicated number of training steps.

GAN: проблемы – Mode-Collapse

решение из статьи – Unrolled GAN

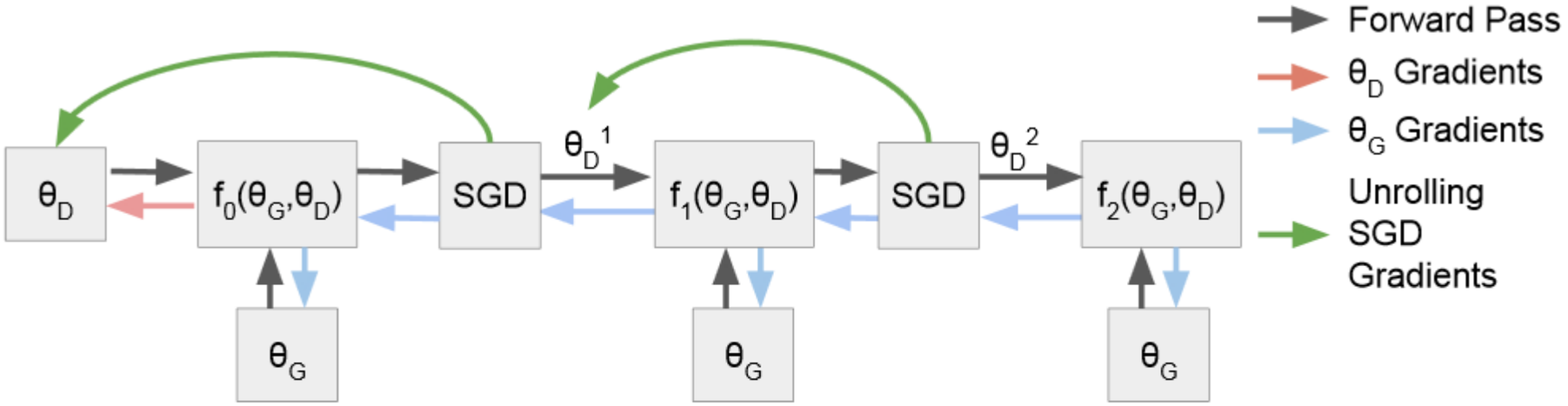


Figure 1: An illustration of the computation graph for an unrolled GAN with 3 unrolling steps. The generator update in Equation 10 involves backpropagating the generator gradient (blue arrows) through the unrolled optimization. Each step k in the unrolled optimization uses the gradients of f_k with respect to θ_D^k, as described in Equation 7 and indicated by the green arrows. The discriminator update in Equation 11 does not depend on the unrolled optimization (red arrow).

ещё решения: организация батчей, W-GANs, LSGANs (было выше) и т.п.

если батч примеров, то дискриминатор видит из однообразность

можно помочь – добавить специальные признаки, ex:

L2-норму разности каждой пары в батче

GAN: проблемы первых моделей






Figure 29: GANs on 128 × 128 ImageNet seem to have trouble with counting, often generating animals with the wrong number of body parts.

Ian Goodfellow «NIPS 2016 Tutorial: Generative Adversarial Networks» https://arxiv.org/abs/1701.00160

Energy-Based GAN (EBGAN)

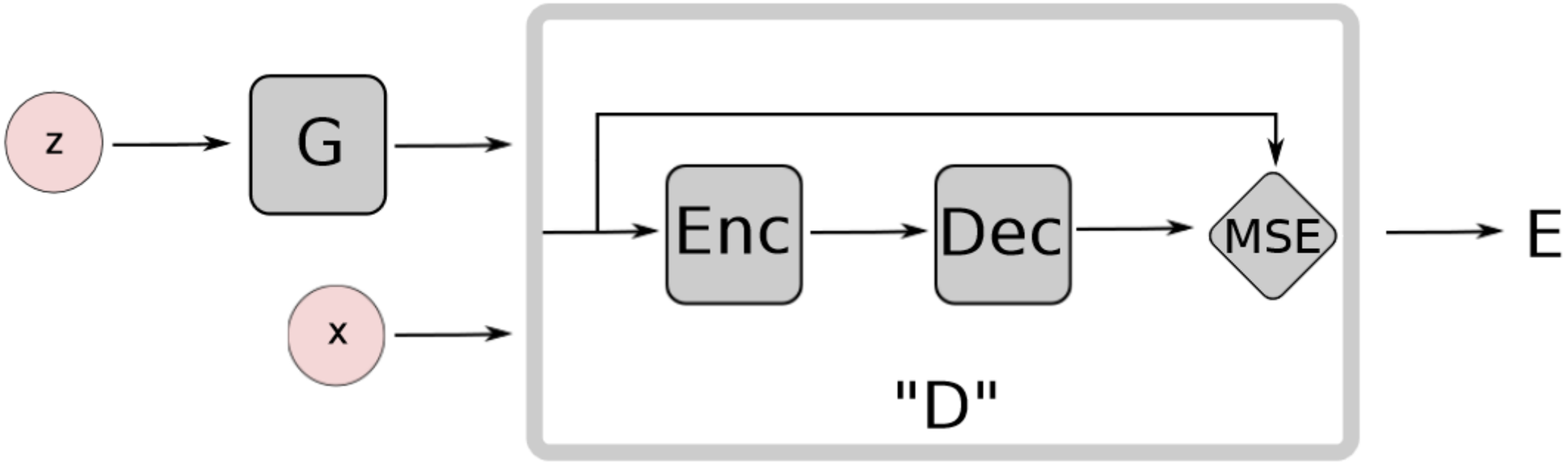


Figure 1: EBGAN architecture with an auto-encoder discriminator.

Пусть дискриминатор – автокодировщик, учим его на реальных данных

**Если он хорошо восстанавливает сгенерированные данные,
то данные хорошие**

Есть развитие идеи – Boundary Equilibrium GAN

Deep Convolutional Generative Adversarial Networks (DCGAN)

- **Полносвёрточная**

- **Нет пулинга**

в дискриминаторе – strided convolutions

в генераторе – fractional-strided convolutions

- **BN после каждого слоя**

- **FC скрытые слои → свёртки**

- **активации**

генератор: ReLU для скрытых слоёв, Tanh для выходного слоя

дискриминатор: LeakyReLU

Впервые удалось использовать CNN для порождения реалистичных картинок

Alec Radford, Luke Metz, Soumith Chintala «Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks» https://arxiv.org/abs/1511.06434

Deep Convolutional Generative Adversarial Networks (DCGAN)

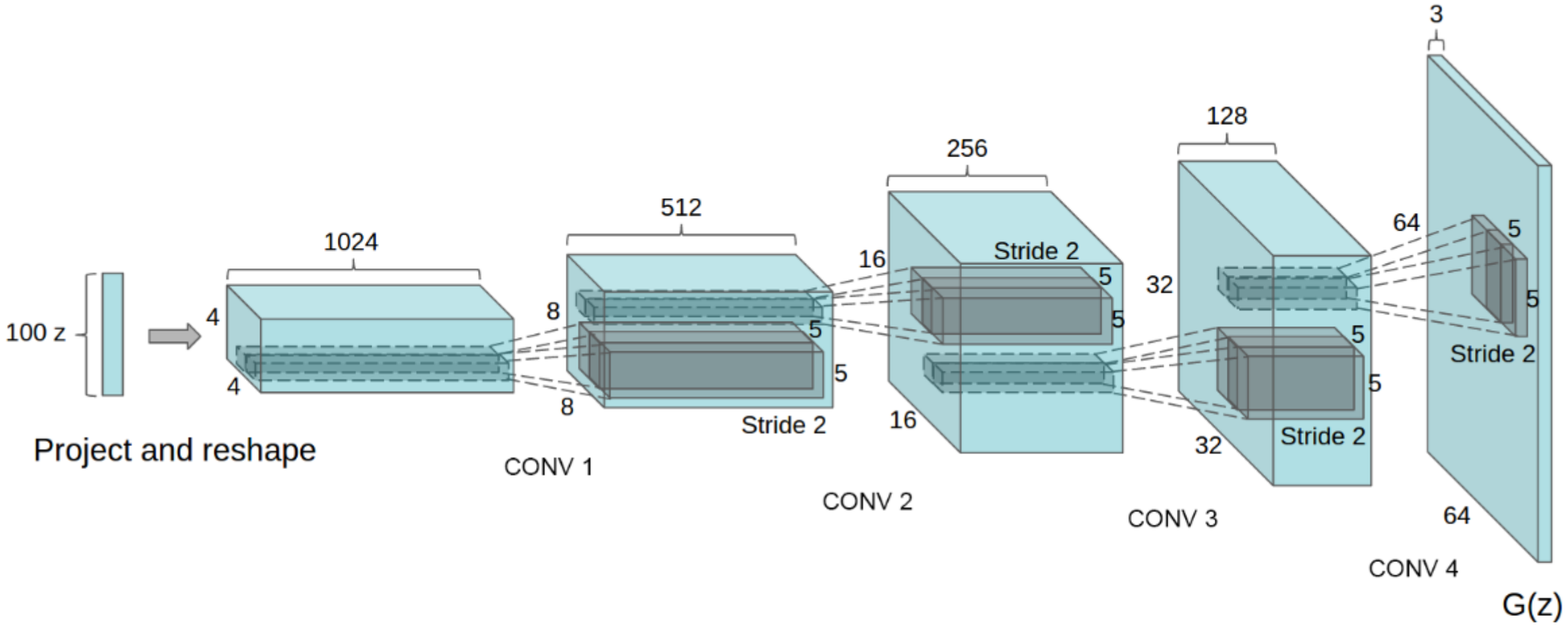


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions)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

Deep Convolutional Generative Adversarial Networks (DCGAN)

Figure 9: Side-by-side illustration of (from left-to-right) the MNIST dataset, generations from a baseline GAN, and generations from our DCGAN .

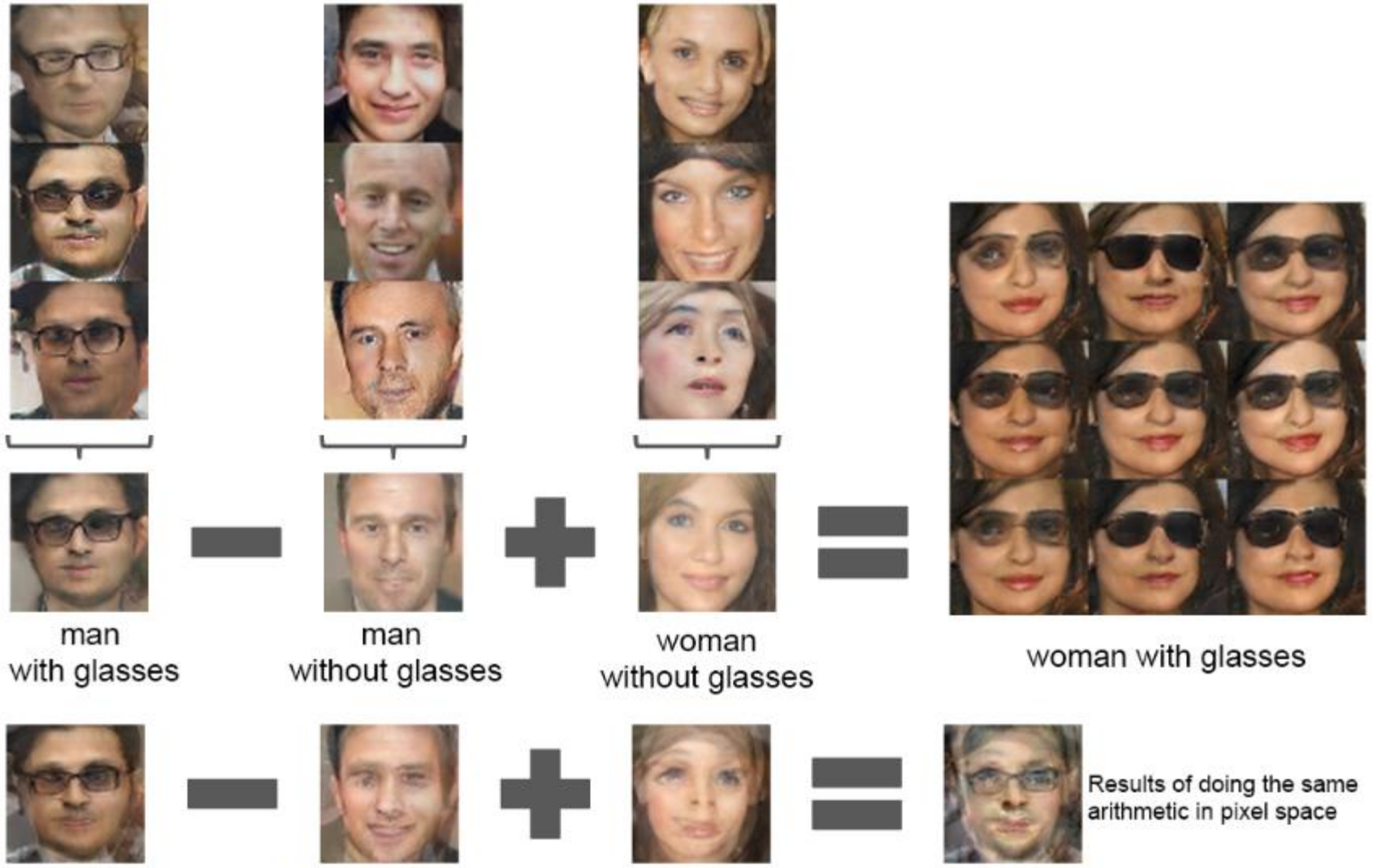
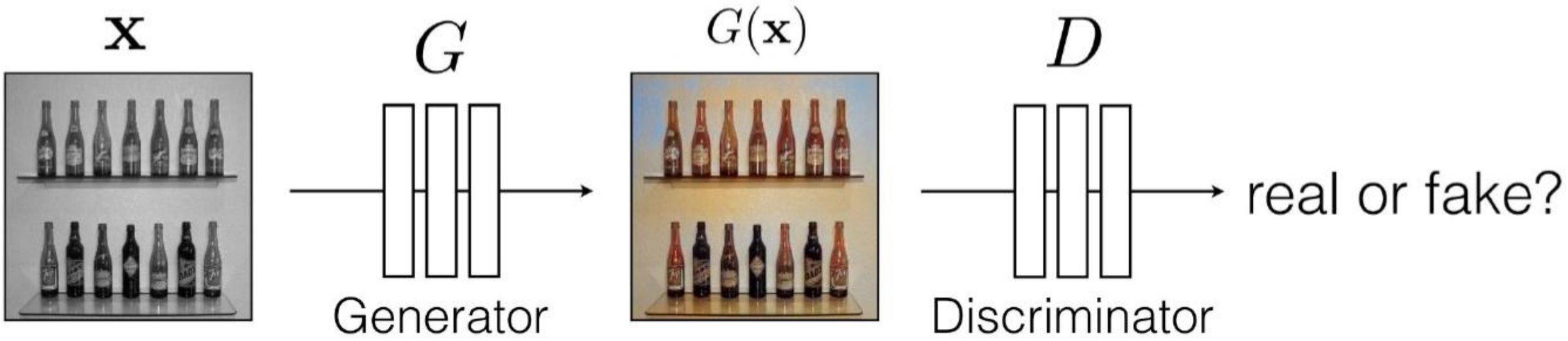


Figure 7: Vector arithmetic for visual concepts. For each column, the Z vectors of samples are averaged. Arithmetic was then performed on the mean vectors creating a new vector Y. The center sample on the right hand side is produce by feeding Y as input to the generator. To demonstrate the interpolation capabilities of the generator, uniform noise sampled with scale +-0.25 was added to Y to produce the 8 other samples. Applying arithmetic in the input space (bottom two examples) results in noisy overlap due to misalignment.

Figure 8: A ”turn” vector was created from four averaged samples of faces looking left vs looking right. By adding interpolations along this axis to random samples we were able to reliably transform their pose.

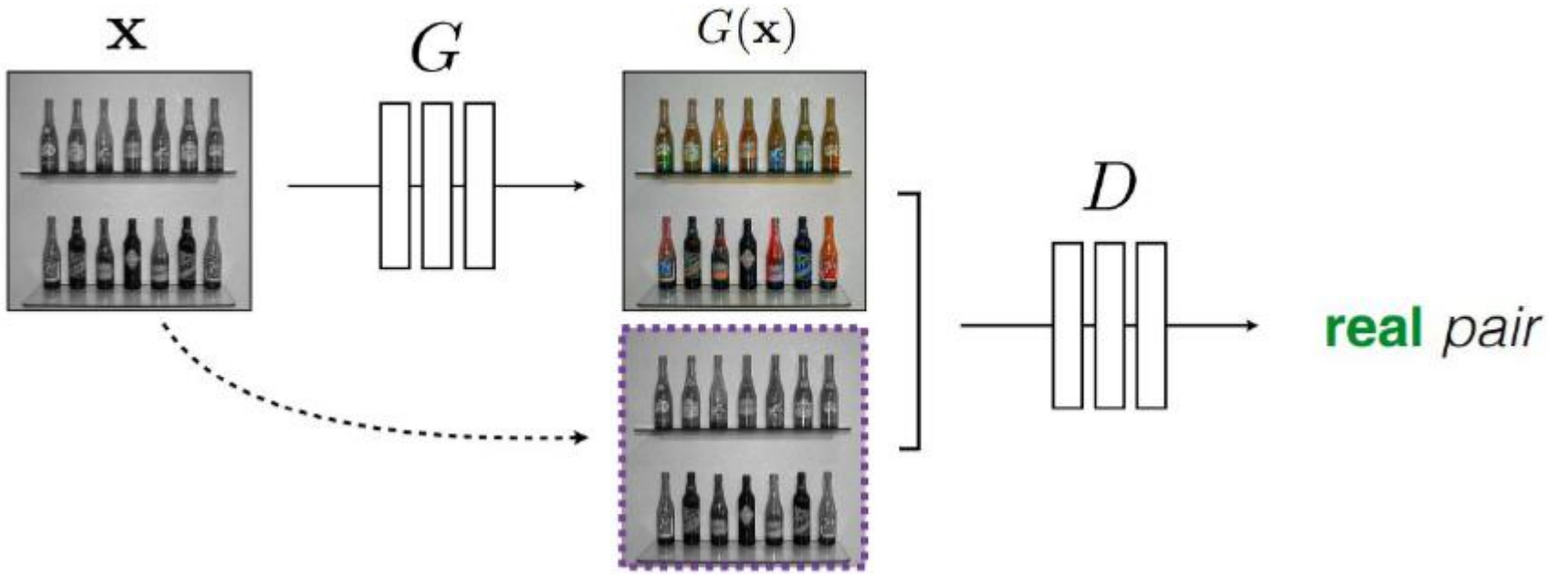
Условные состязательные сети (cGAN)



но тут проблема, что сеть начнёт генерировать что-то реальное, например «котиков» и условие будет фиктивным

[Phillip Isola]

Условные состязательные сети (cGAN)



лучше определять реальность/фейковость пары

Pix2pix с условными состязательными сетями (cGAN)

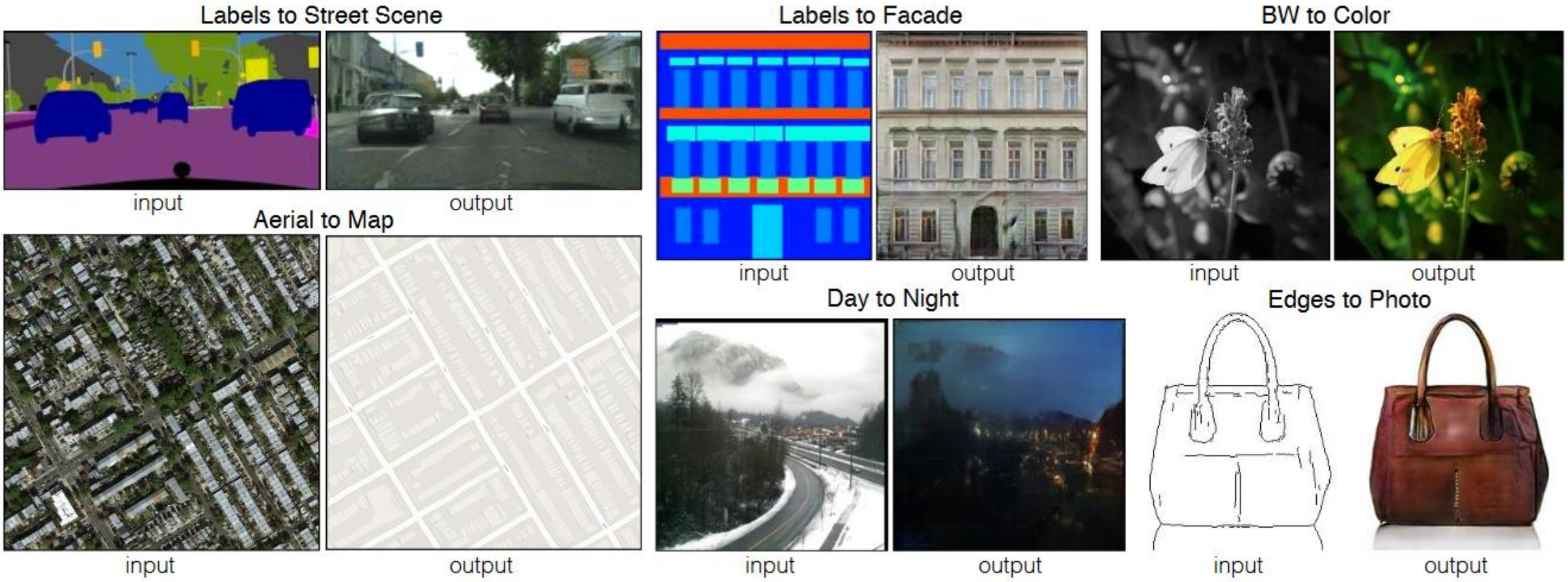


Figure 1: Many problems in image processing, graphics, and vision involve translating an input image into a corresponding output image. These problems are often treated with application-specific algorithms, even though the setting is always the same: map pixels to pixels. Conditional adversarial nets are a general-purpose solution that appears to work well on a wide variety of these problems. Here we show results of the method on several. In each case we use the same architecture and objective, and simply train on different data.

Pix2pix с условными состязательными сетями (cGAN)

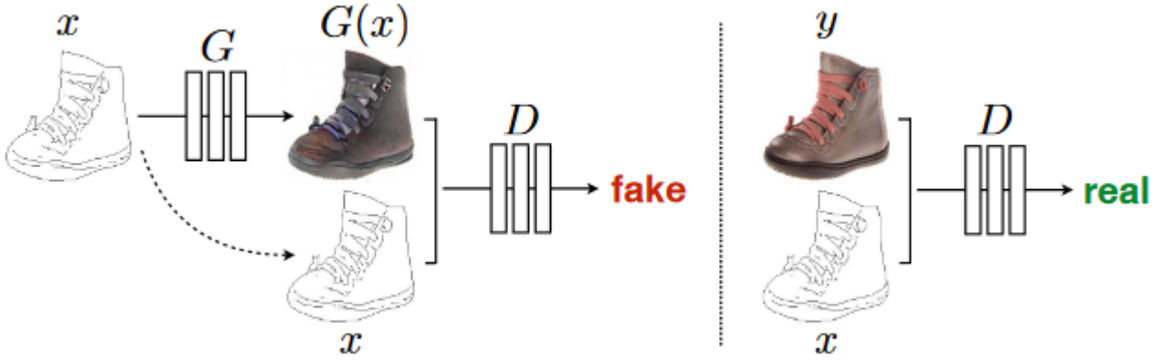


Figure 2: Training a conditional GAN to map edges→photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

нужна размеченная выборка
(пары изображений),
поэтому такие примеры

- **силуэт**
- **сегментация**
- **стилизация**

U-net для генератора
PatchGAN для дискриминатора –
классифицирует кусочки,
на всём изображении L1-ошибка

вместо определения фейковости всего изображения смотреть на патчи
– быстрее, можно применять к большим изображениям

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros «Image-to-Image Translation with Conditional Adversarial Nets» // CVPR 2017, https://phillipi.github.io/pix2pix/

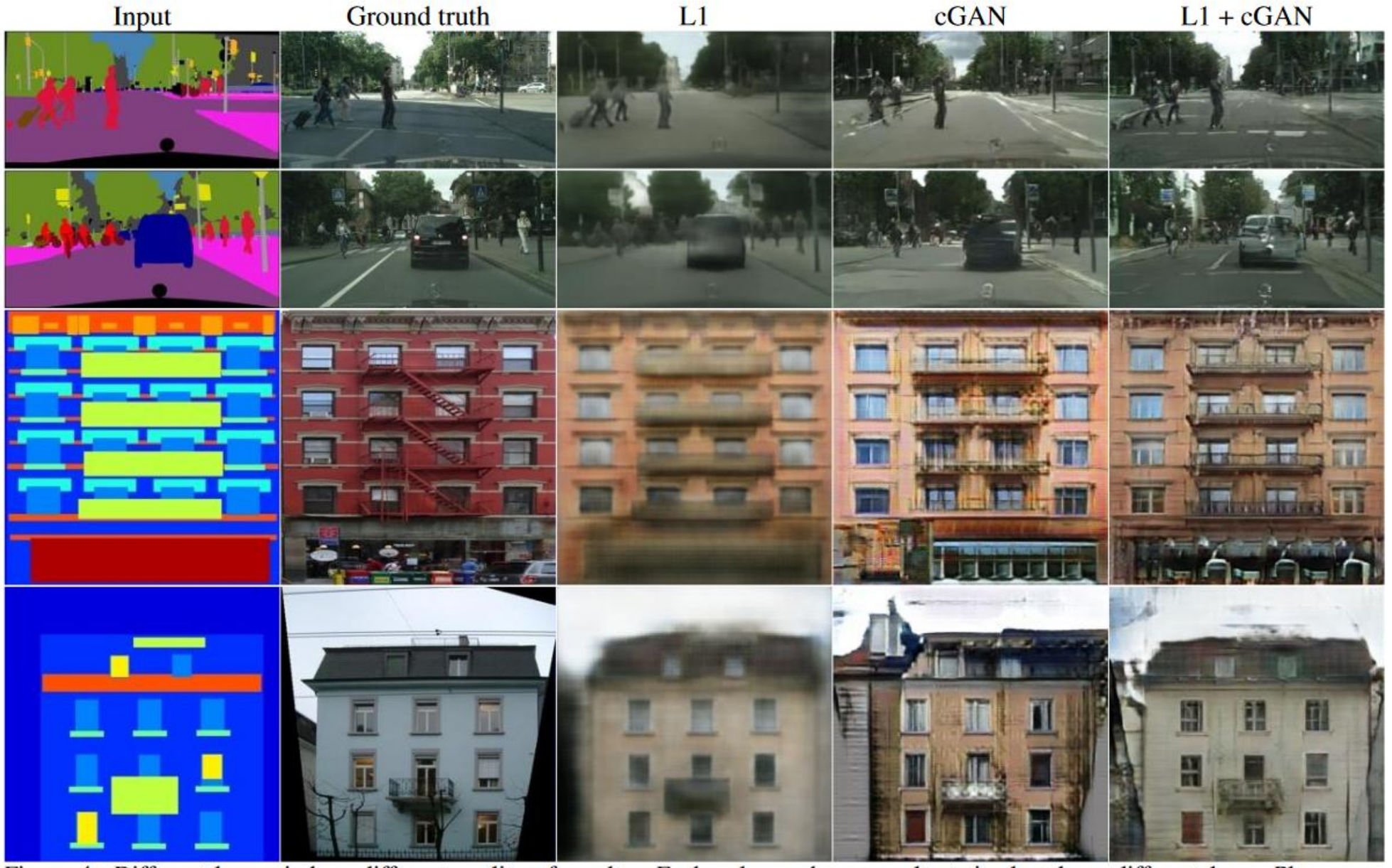


Figure 4: Different losses induce different quality of results. Each column shows results trained under a different loss. Please see https://phillipi.github.io/pix2pix/ for additional examples.

Figure 8: Example results on Google Maps at 512x512 resolution (model was trained on images at 256 × 256 resolution, and run convolutionally on the larger images at test time). Contrast adjusted for clarity.

Figure 16: Example results of our method on automatically detected edges→handbags, compared to ground truth.

Pix2pix с условными состязательными сетями (cGAN)

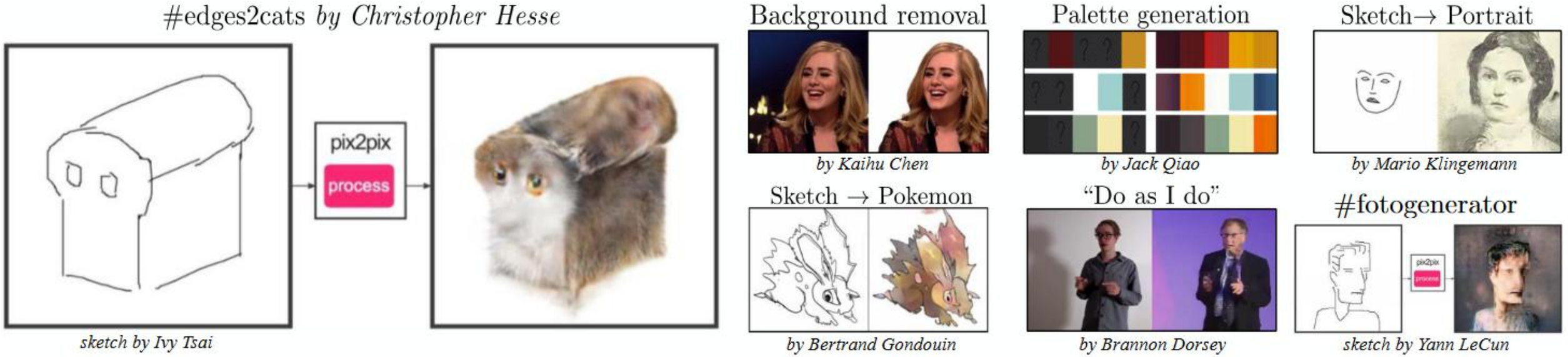


Figure 11: Example applications developed by online community based on our `pix2pix` codebase: *#edges2cats* [3] by Christopher Hesse, *Background removal* [6] by Kaihu Chen, *Palette generation* [5] by Jack Qiao, *Sketch → Portrait* [7] by Mario Klingemann, *Sketch→ Pokemon* [1] by Bertrand Gondouin, *"Do As I Do" pose transfer* [2] by Brannon Dorsey, and *#fotogenerator* by Bosman et al. [4].

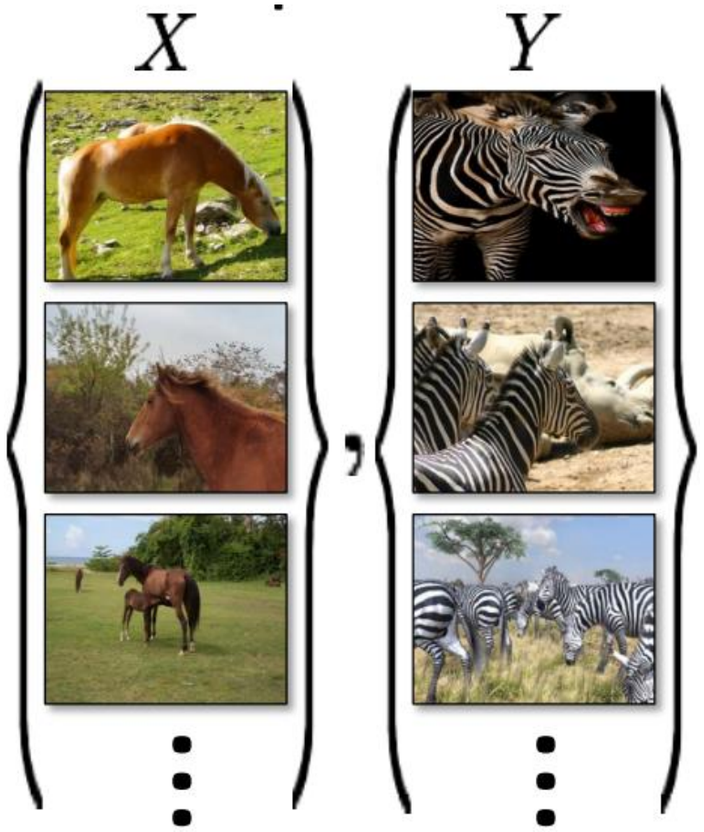
различные результаты сообщества с помощью разработанной библиотеки

Проблема отсутствия выборки

есть размеченная выборка для pix → pix

или может быть дёшево получена, как в рассмотренном примере (получение контуров)

есть просто представители двух множест



можно ли осуществить трансфер?

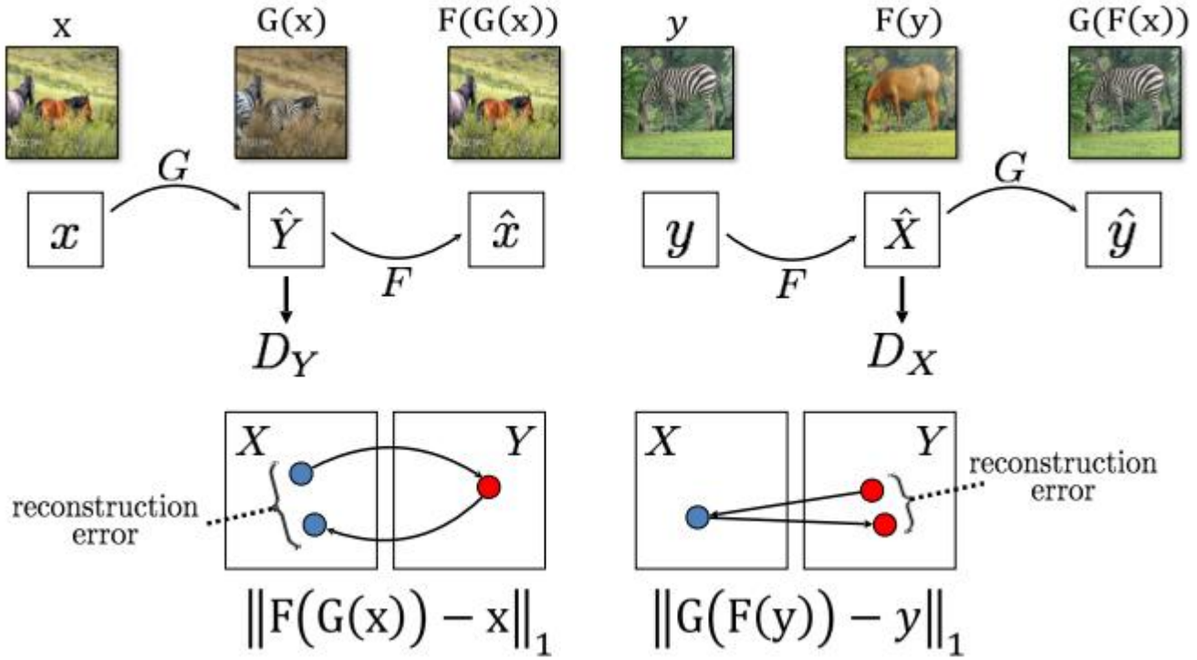
CycleGAN

Перенос стиля без размеченных данных (пар картинок)

Идея: два генератора (в разные стороны)

Надо, чтобы после работы их подряд получалась исходная картинка

Cycle Consistency Loss



Zhu et al., «Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks», ICCV 2017 https://junyanz.github.io/CycleGAN/

CycleGAN

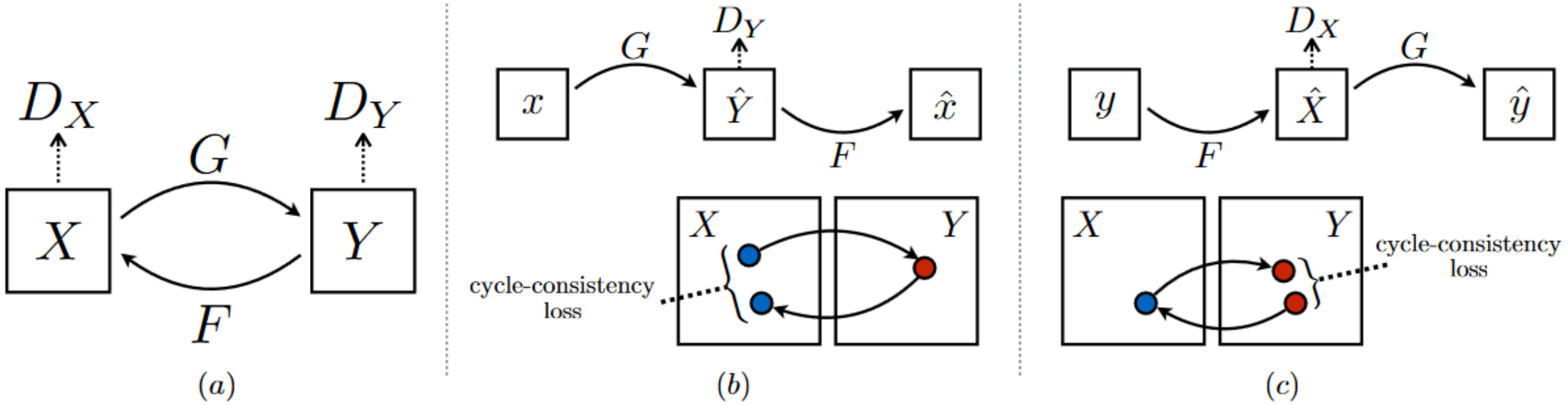


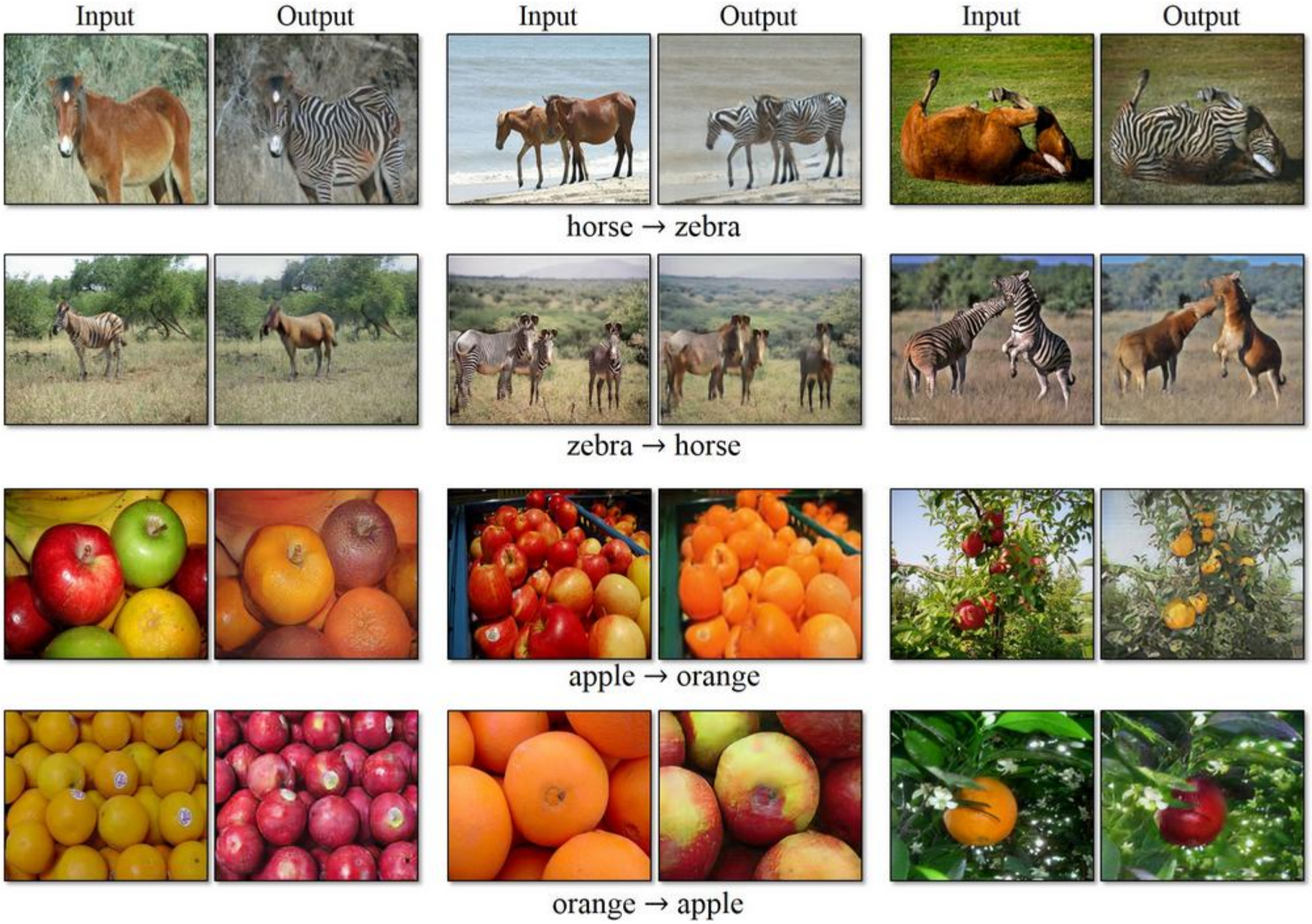
Figure 3: (a) Our model contains two mapping functions $G : X \to Y$ and $F : Y \to X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \to G(x) \to F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \to F(y) \to G(F(y)) \approx y$

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))], \quad (1)$$

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]. \quad (2)$$

Our full objective is:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F), \quad (3)$$


Input
Output
Input
Output
Input
Output
horse → zebra
zebra → horse
apple → orange
orange → apple

CycleGAN

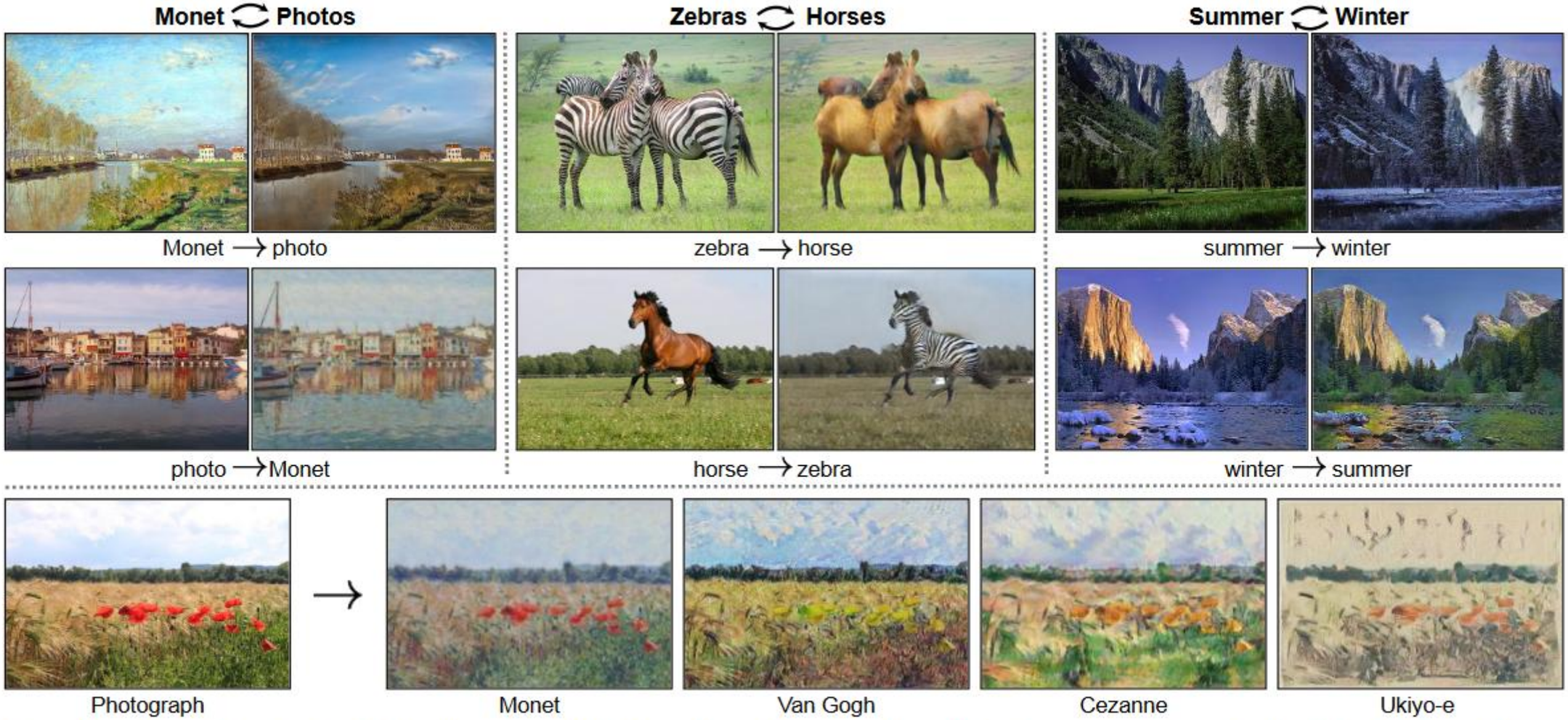


Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos from Flickr. Example application *(bottom)*: using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

CycleGAN: неудачные примеры

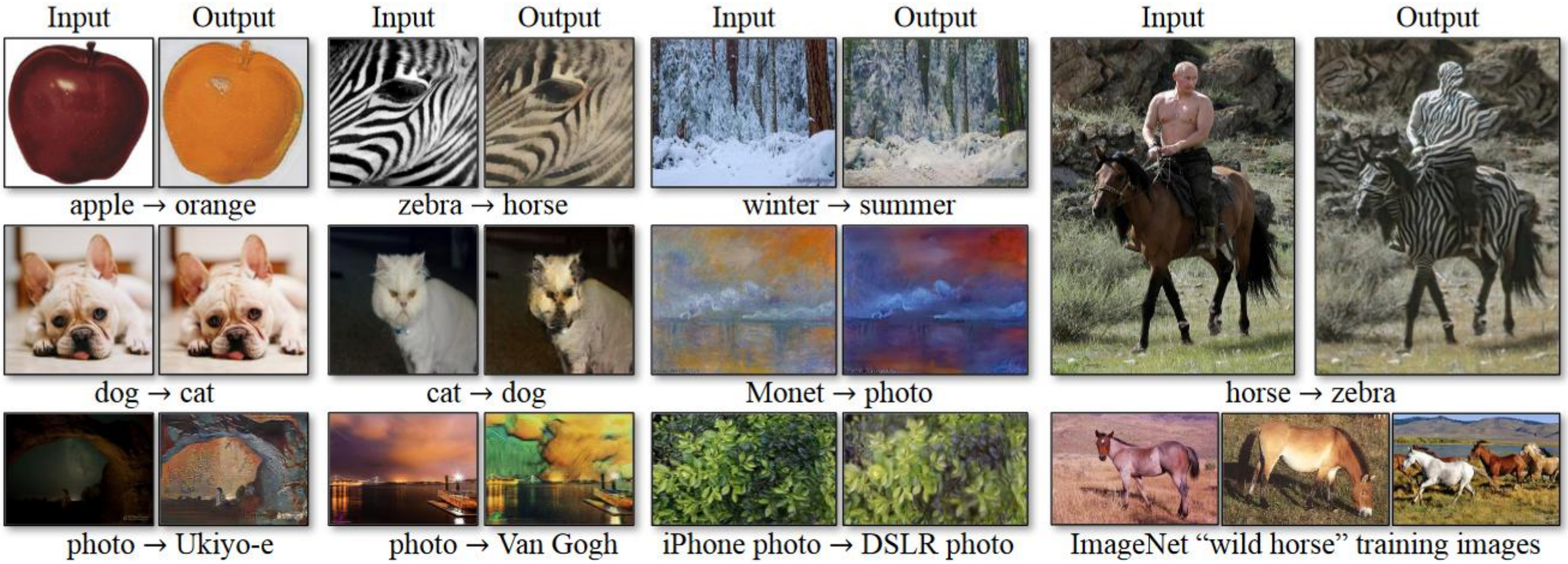
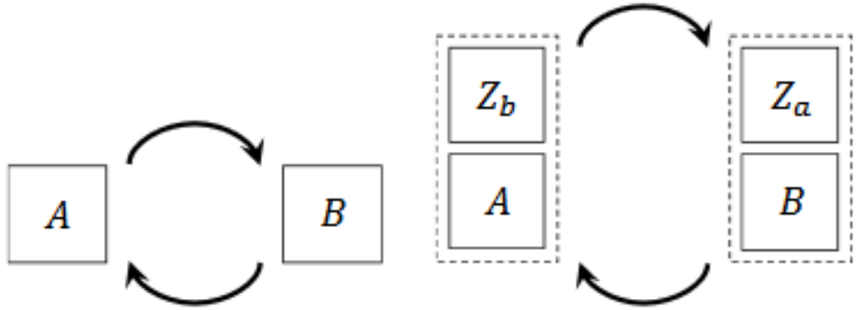


Figure 17: Typical failure cases of our method. Left: in the task of dog→cat transfiguration, CycleGAN can only make minimal changes to the input. Right: CycleGAN also fails in this horse → zebra example as our model has not seen images of horseback riding during training. Please see our website for more comprehensive results.

Аугментированная циклическая GAN (Augmented CycleGAN)



(a) CycleGAN (b) Augmented CycleGAN

Figure 1: (a) Original CycleGAN model. (b) We propose to learn many-to-many mappings by cycling over the original domains augmented with auxiliary latent spaces. By marginalizing out auxiliary variables, we can model many-to-many mappings in between the domains.

Решение проблемы, когда данных мало

Раньше ограничение – учим однозначные отображения

Теперь многозначное отображение

Добавляем латентную переменную!
В ф-ии ошибки – матожидание по ней!

Amjad Almahairi et al «Augmented CycleGAN: Learning Many-to-Many Mappings from Unpaired Data» // https://arxiv.org/pdf/1802.10151.pdf

BiGAN (Bidirectional)

В классике учим отображение из латентного пр-ва в объекты – а как обратное?
Здесь: учим одновременно и обратную функцию к генератору
Получается хорошая арифметика в латентном пространстве

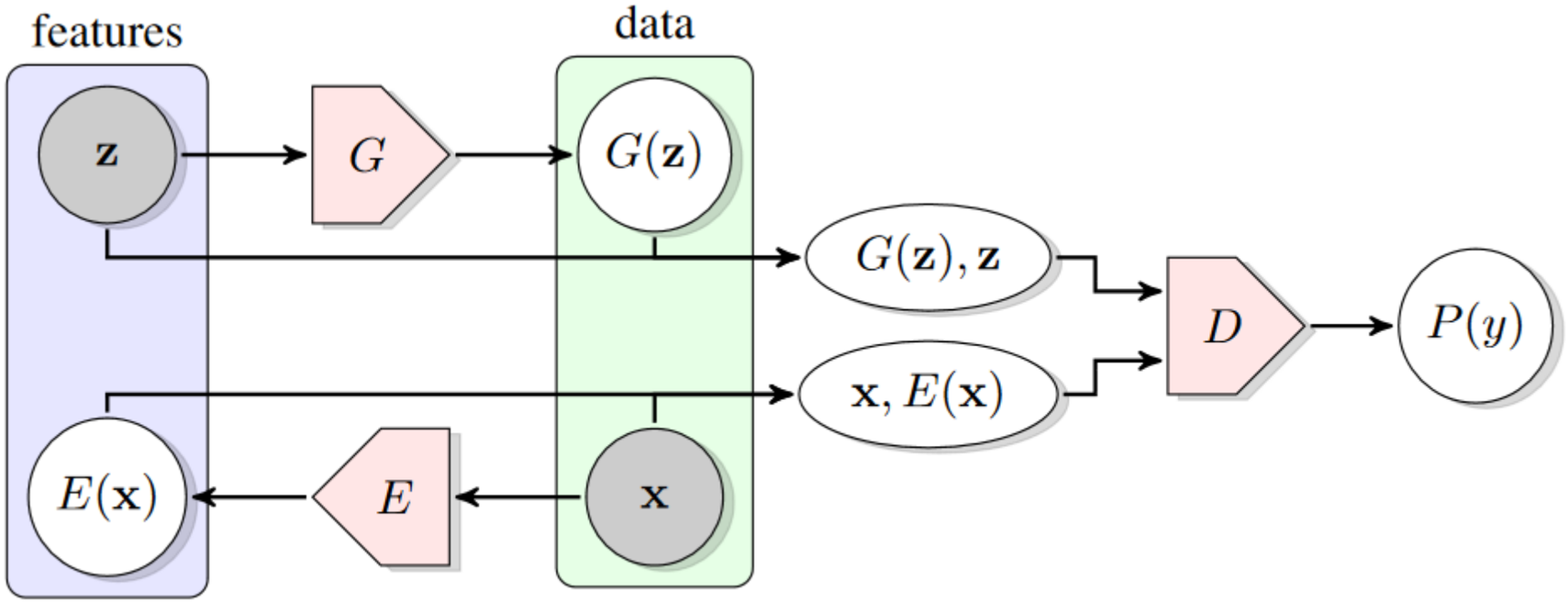


Figure 1: The structure of Bidirectional Generative Adversarial Networks (BiGAN).

Jeff Donahue, Philipp Krähenbühl, Trevor Darrell «Adversarial Feature Learning» https://arxiv.org/abs/1605.09782

BiGAN (Bidirectional)

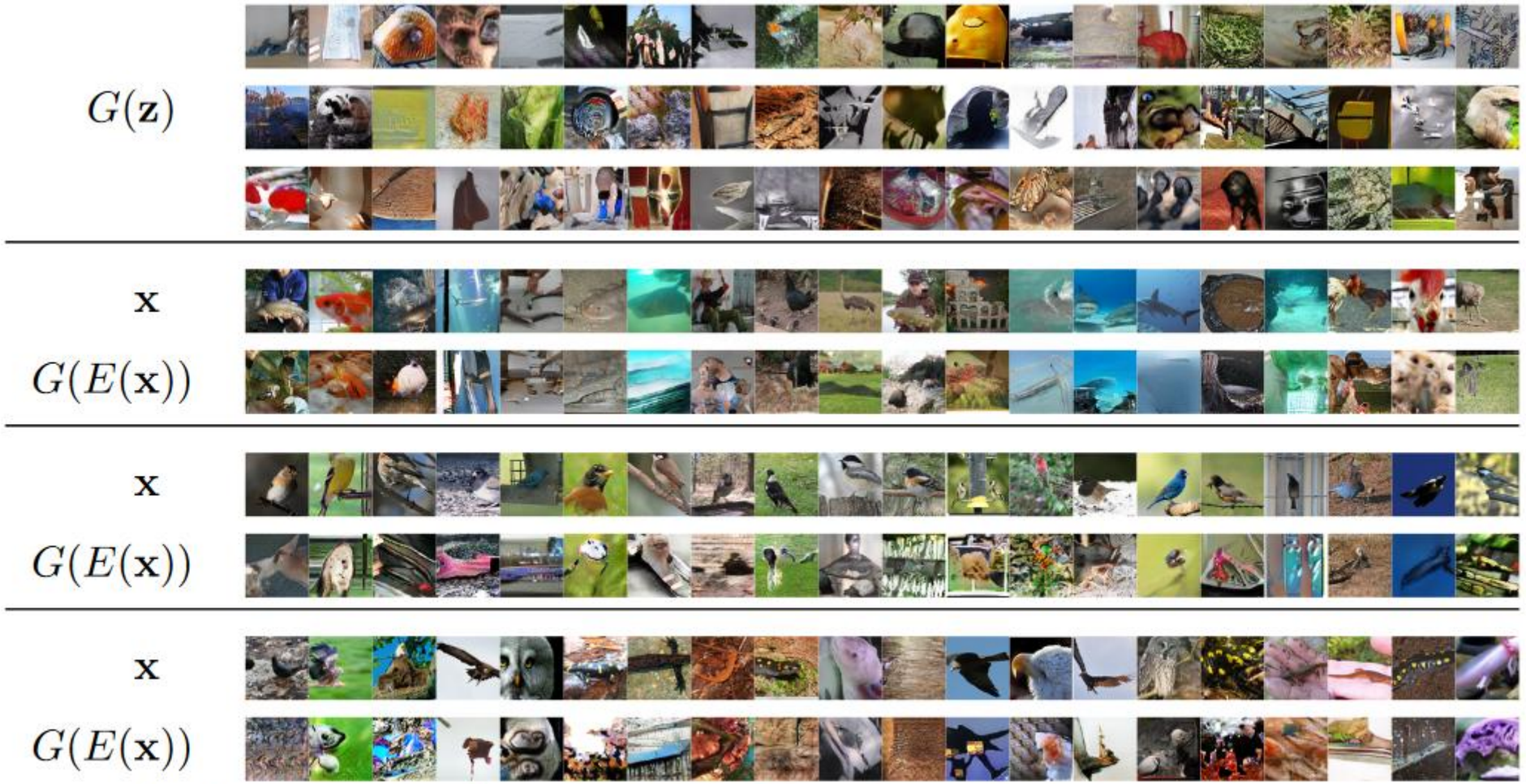


Figure 4: Qualitative results for ImageNet BiGAN training, including generator samples $G(\mathbf{z})$, real data $\mathbf{x}$, and corresponding reconstructions $G(E(\mathbf{x}))$.

BiGAN (Bidirectional)

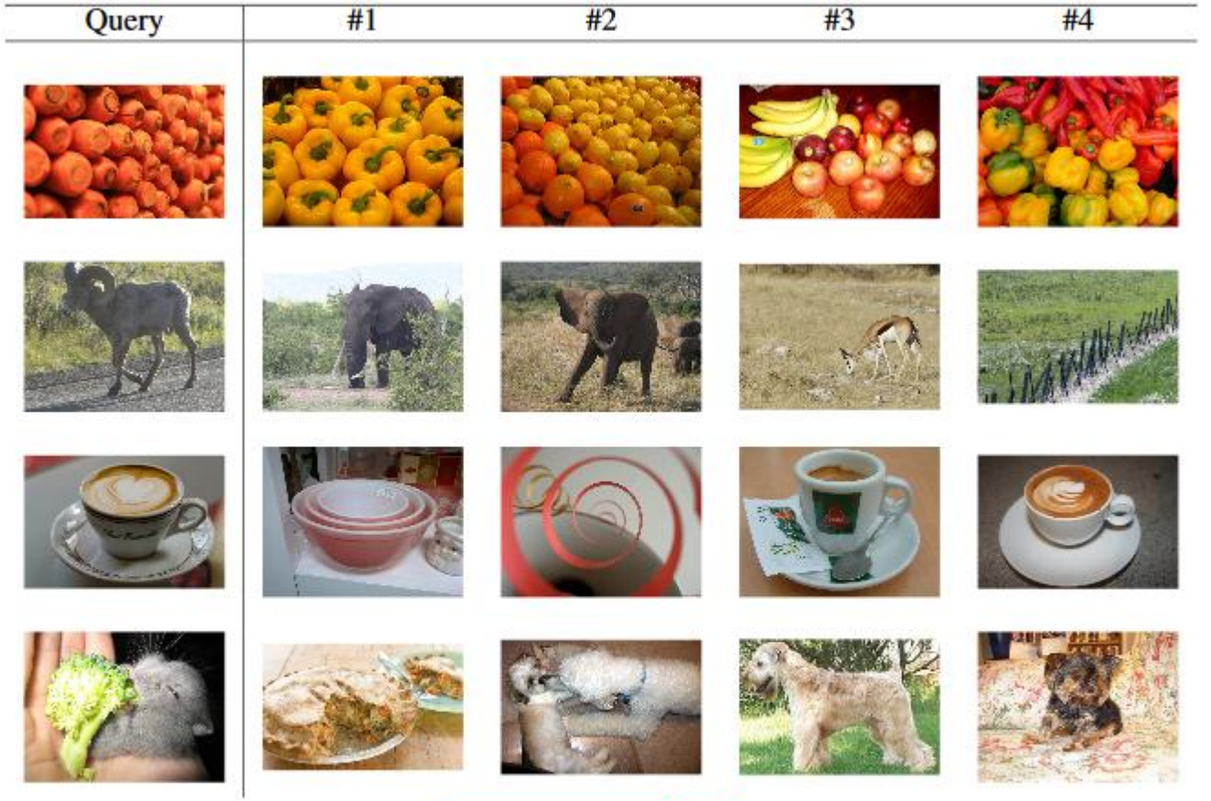


Figure 5: For the query images used in Krähenbühl et al. (2016) (left), nearest neighbors (by minimum cosine distance) from the ImageNet LSVRC (Russakovsky et al., 2015) training set in the *fc6* feature space of the ImageNet-trained BiGAN encoder E. (The *fc6* weights are set randomly; this space is a random projection of the learned *conv5* feature space.)

с помощью BiGAN поиск ближайших соседей в признаковом пространстве

BigGAN: Генерация изображений / интерполяция

Figure 6: Samples generated by our BigGAN model at 512×512 resolution.

BigGAN: Генерация изображений / интерполяция

«Truncation trick»

– если сгенерированная латентная переменная далеко от центра – проецируем

(a) (b)

Figure 2: (a) The effects of increasing truncation. From left to right, the threshold is set to 2, 1, 0.5, 0.04. (b) Saturation artifacts from applying truncation to a poorly conditioned model.

Здесь разные классы (и их много), 512×512

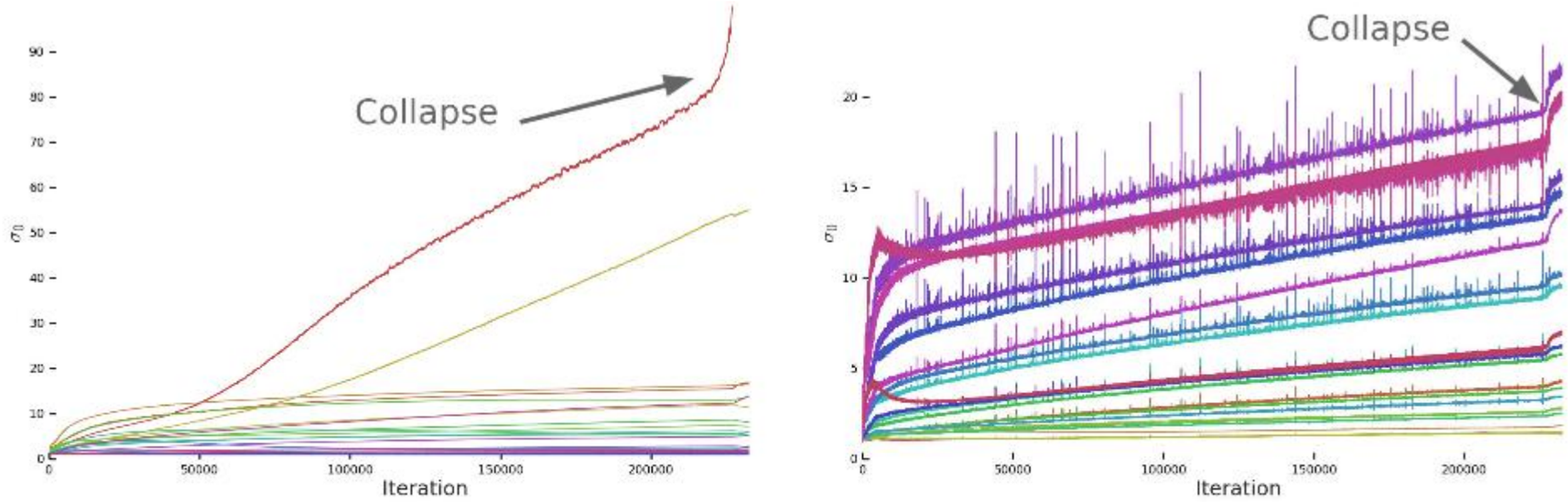
~350M параметров, на больших мощностях

разумные интерполяции (хотя примеры есть и плохие)

ResNet, на основе SA-GAN

Andrew Brock «Large scale GAN training for high fidelity natural image synthesis» // https://arxiv.org/pdf/1809.11096.pdf

BigGAN: Генерация изображений / интерполяция



(a) **G** (b) **D**

Figure 3: A typical plot of the first singular value σ_0 in the layers of **G** (a) and **D** (b) before Spectral Normalization. Most layers in **G** have well-behaved spectra, but without constraints a small subset grow throughout training and explode at collapse. **D**'s spectra are noisier but otherwise better-behaved. Colors from red to violet indicate increasing depth.

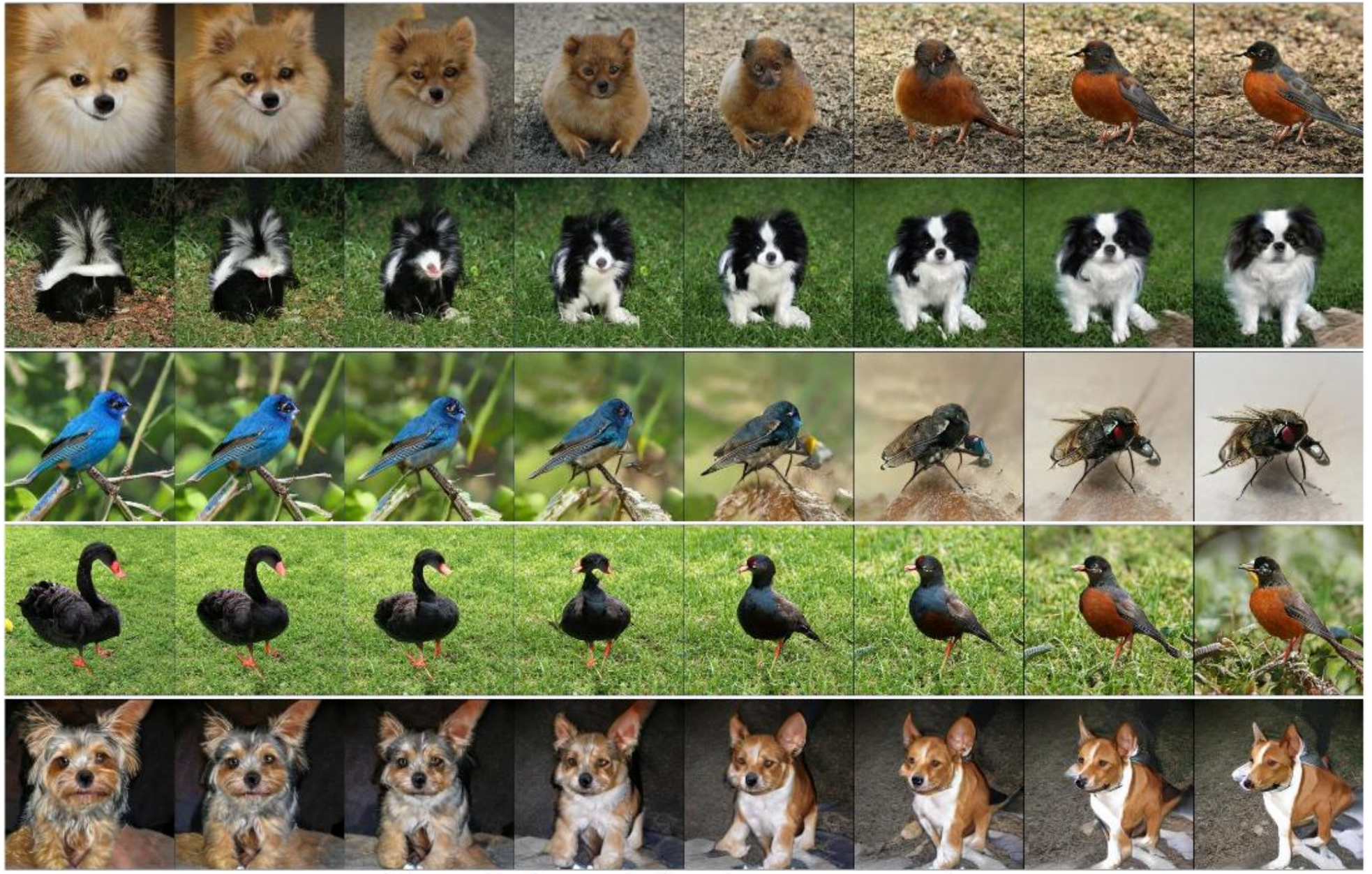


Figure 8: Interpolations between z, c pairs.

SAGAN (Self-Attention Generative Adversarial Networks)

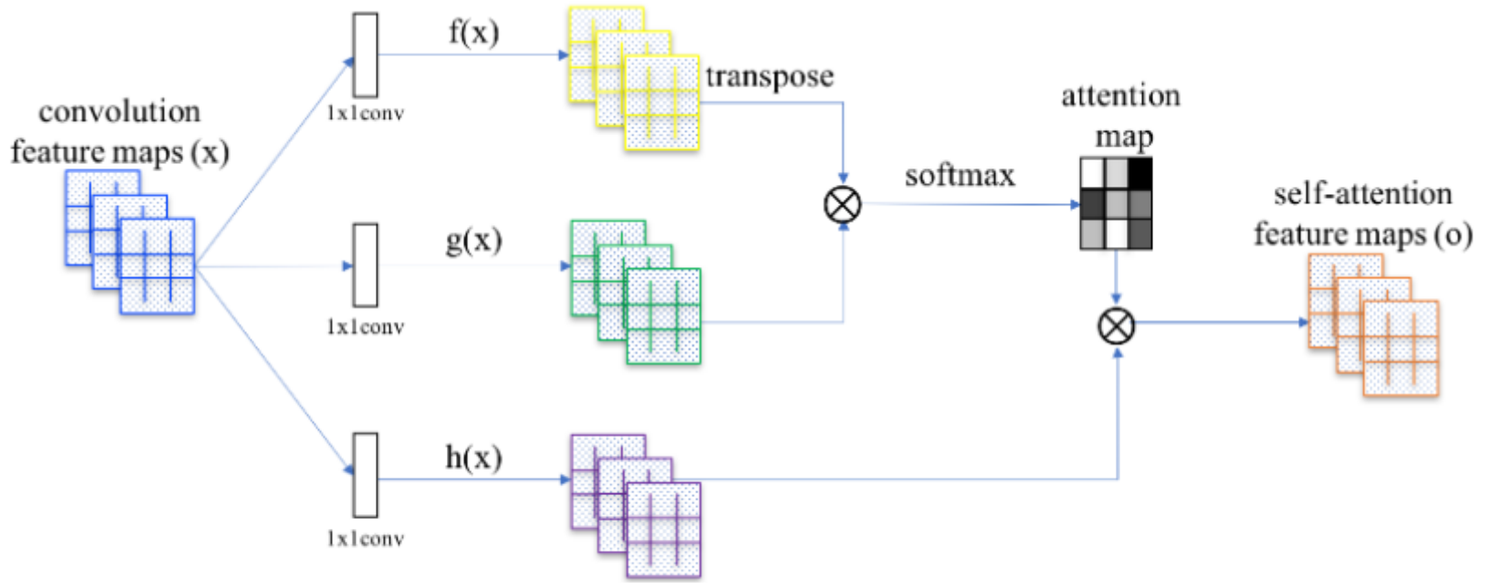


Figure 2: The proposed self-attention mechanism. The ⊗ denotes matrix multiplication. The softmax operation is performed on each row.

внимание (self-attention в генераторе и дискриминаторе) + GAN

впервые безусловная хорошая генерация примеров по ImageNet

Han Zhang, Ian Goodfellow, Dimitris Metaxas, Augustus Odena «Self-Attention Generative Adversarial Networks», 2018 https://arxiv.org/abs/1805.08318

SAGAN (Self-Attention Generative Adversarial Networks)

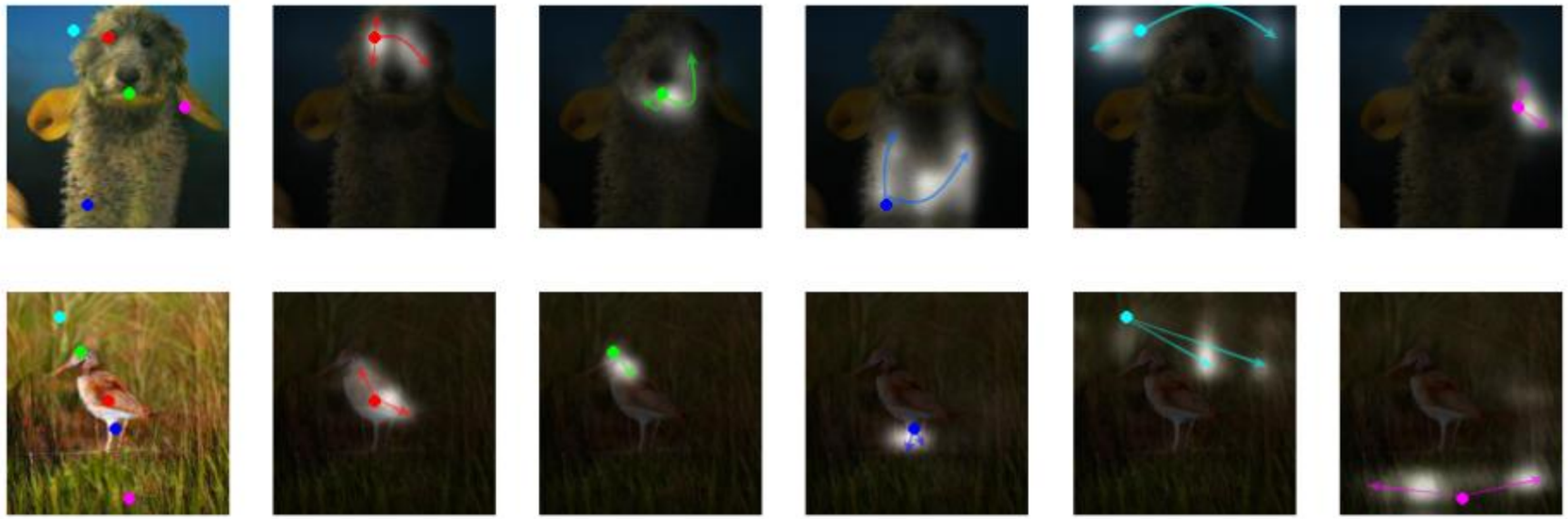


Figure 1: The proposed SAGAN generates images by leveraging complementary features in distant portions of the image rather than local regions of fixed shape to generate consistent objects/scenarios. In each row, the first image shows five representative query locations with color coded dots. The other five images are attention maps for those query locations, with corresponding color coded arrows summarizing the most-attended regions.

внимание $\Rightarrow$ можем учитывать контекст (далёкие пиксели)

для стабилизации:

+ spectral normalization (в генераторе и дискриминаторе)

+ separate learning rates (TTUR) для G и D (быстрее будет обучение)

CAN: Creative Adversarial Networks

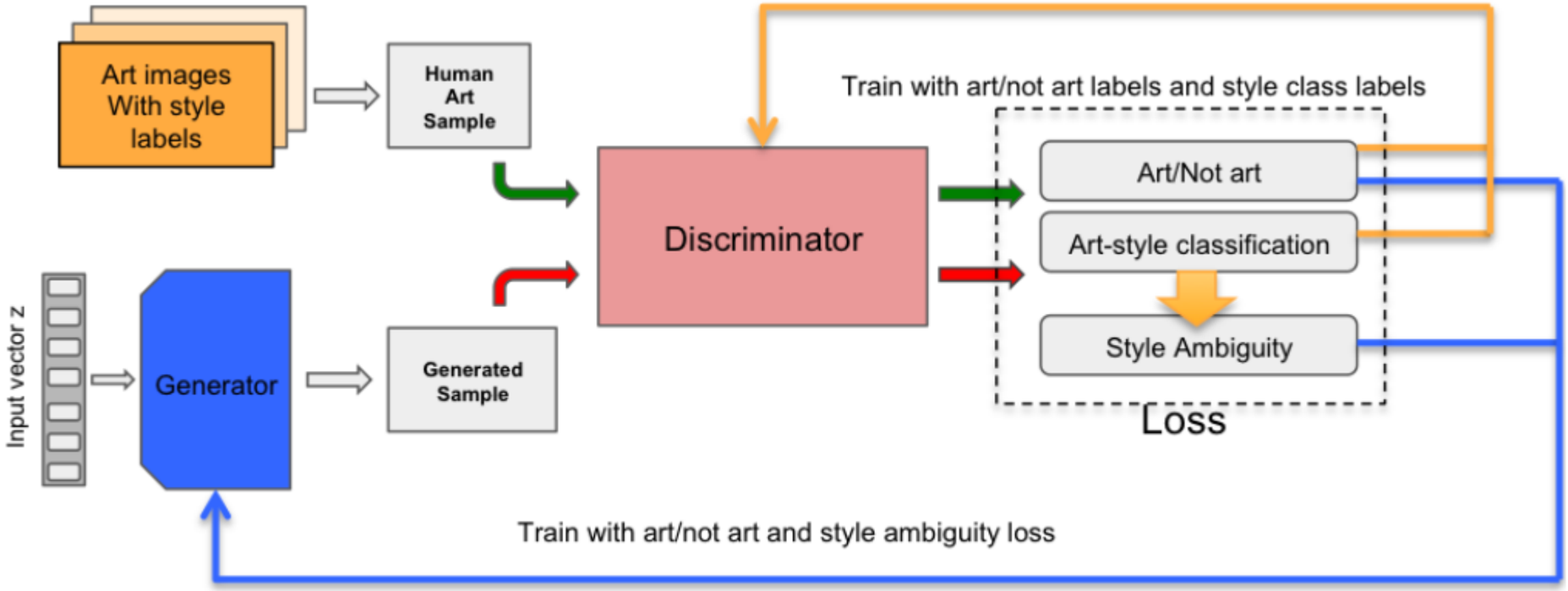


Figure 2: Block diagram of the CAN system.

+ классификация стилей

Ahmed Elgammal et al «CAN: Creative Adversarial Networks, Generating "Art" by Learning About Styles and Deviating from Style Norms» https://arxiv.org/abs/1706.07068

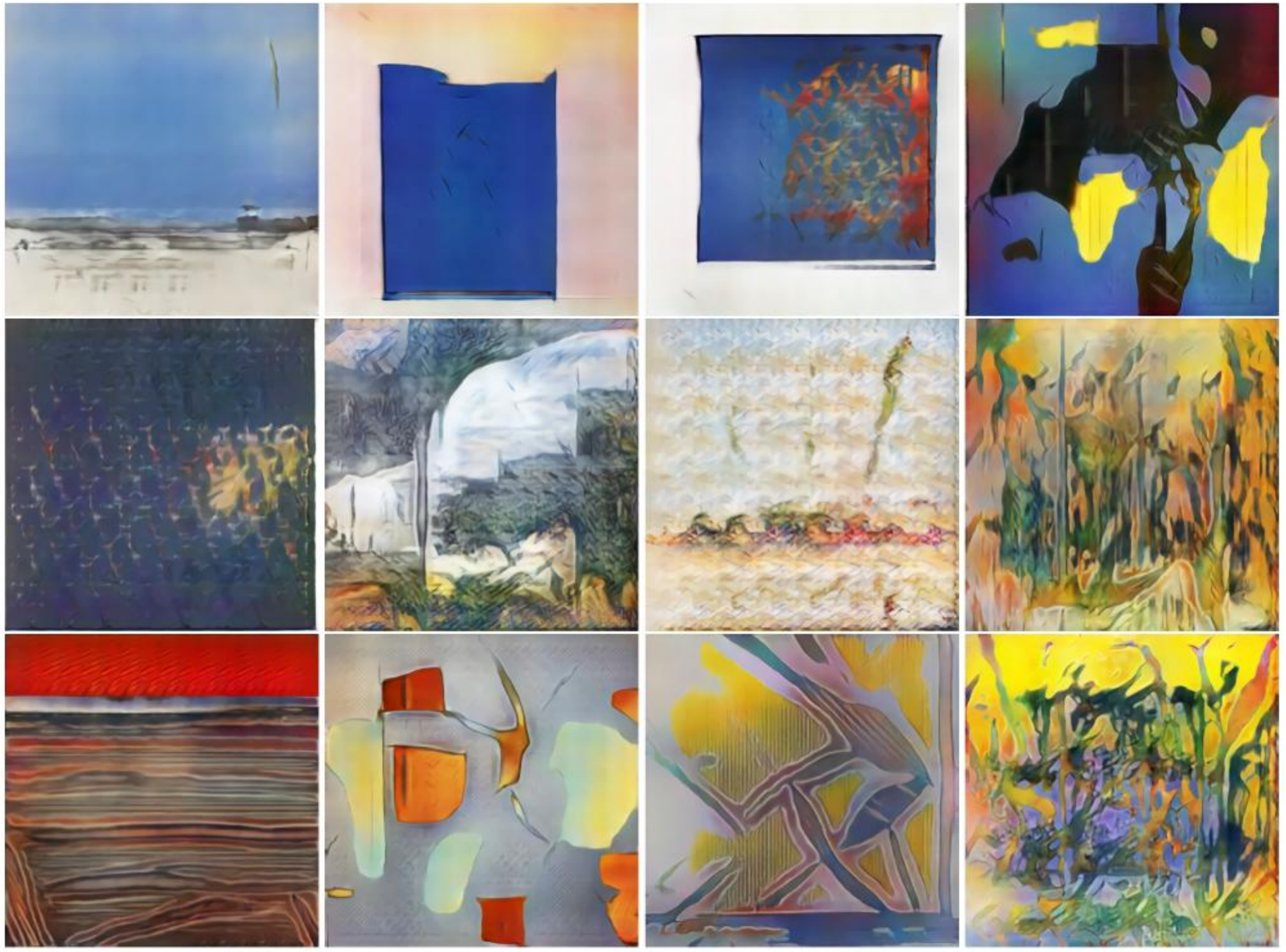


Figure 1: Example of images generated by CAN. The generated images vary from simple abstract ones to complex textures and compositions. More examples are shown in Figures 4.2 and 7

ProGAN (NVIDIA)

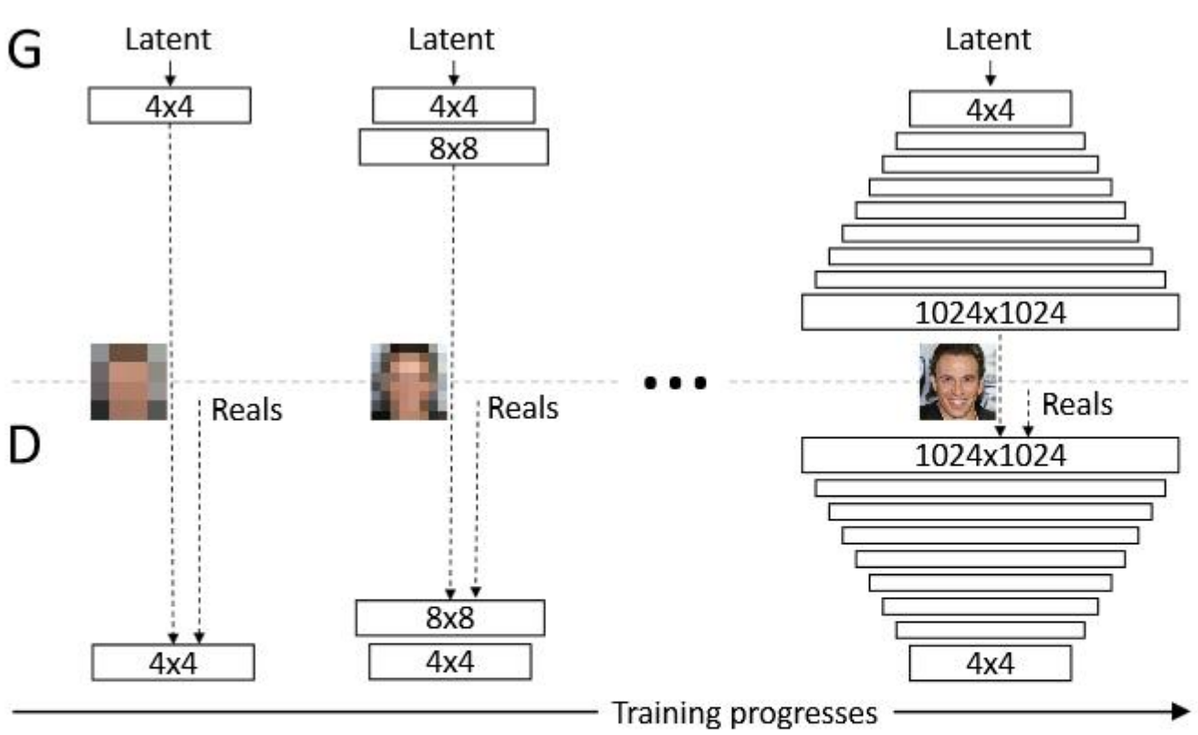


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $N \times N$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. One the right we show six example images generated using progressive growing at 1024×1024.

Karras et al., «Progressive Growing of GANs for Improved Quality, Stability, and Variation» // ICLR 2018 https://arxiv.org/abs/1710.10196

ProGAN (NVIDIA)

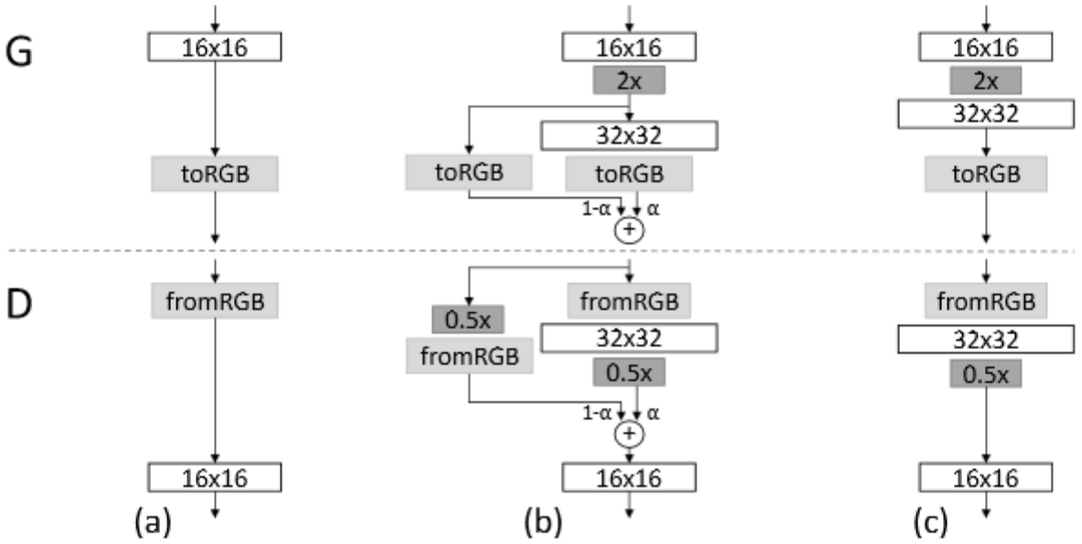


Figure 2: When doubling the resolution of the generator (G) and discriminator (D) we fade in the new layers smoothly. This example illustrates the transition from 16×16 images (a) to 32×32 images (c). During the transition (b) we treat the layers that operate on the higher resolution like a residual block, whose weight α increases linearly from 0 to 1. Here $2\times$ and $0.5\times$ refer to doubling and halving the image resolution using nearest neighbor filtering and average pooling, respectively. The toRGB represents a layer that projects feature vectors to RGB colors and fromRGB does the reverse; both use 1×1 convolutions. When training the discriminator, we feed in real images that are downscaled to match the current resolution of the network. During a resolution transition, we interpolate between two resolutions of the real images, similarly to how the generator output combines two resolutions.

Поэтапно учимся создавать картинки в 4 раза больше до размера 1024×1024

Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

ProGAN (NVIDIA)

Mao et al. (2016b) (128 × 128) Gulrajani et al. (2017) (128 × 128) Our (256 × 256)

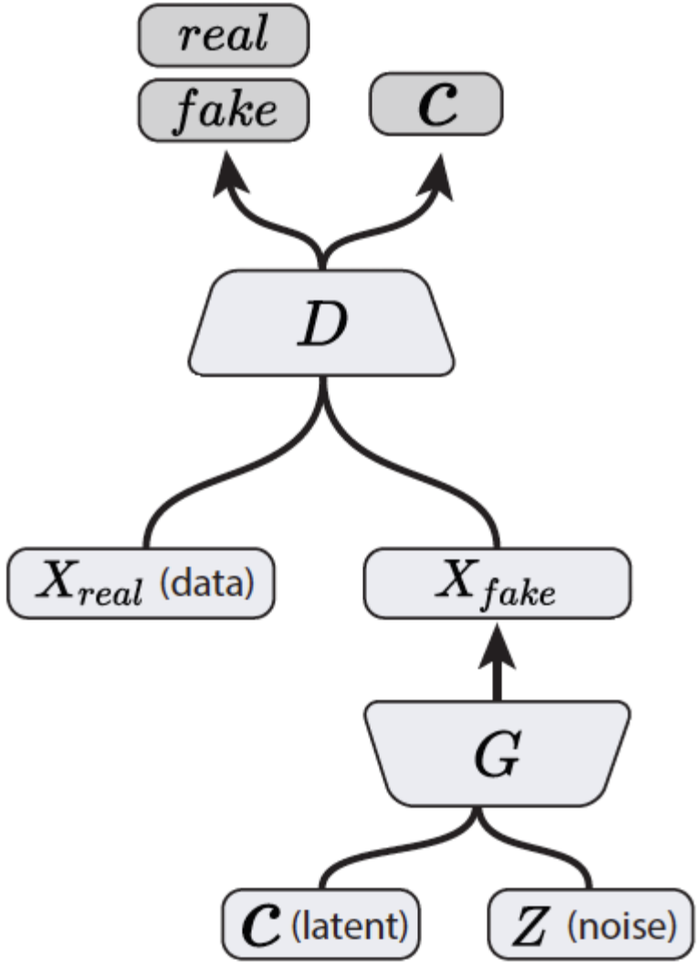
Figure 6: Visual quality comparison in LSUN BEDROOM; pictures copied from the cited articles.

InfoGAN

Пусть
Z – шум
C – важная информация
(класс, угол, толщина)

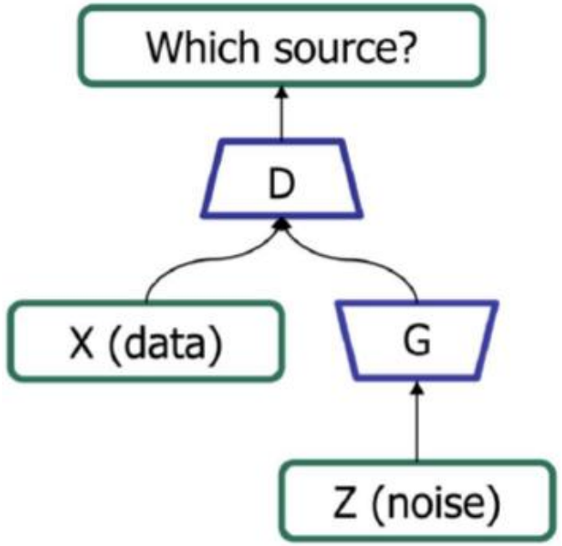
Хотим параллельно максимизировать взаимною информацию между c и x:

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$

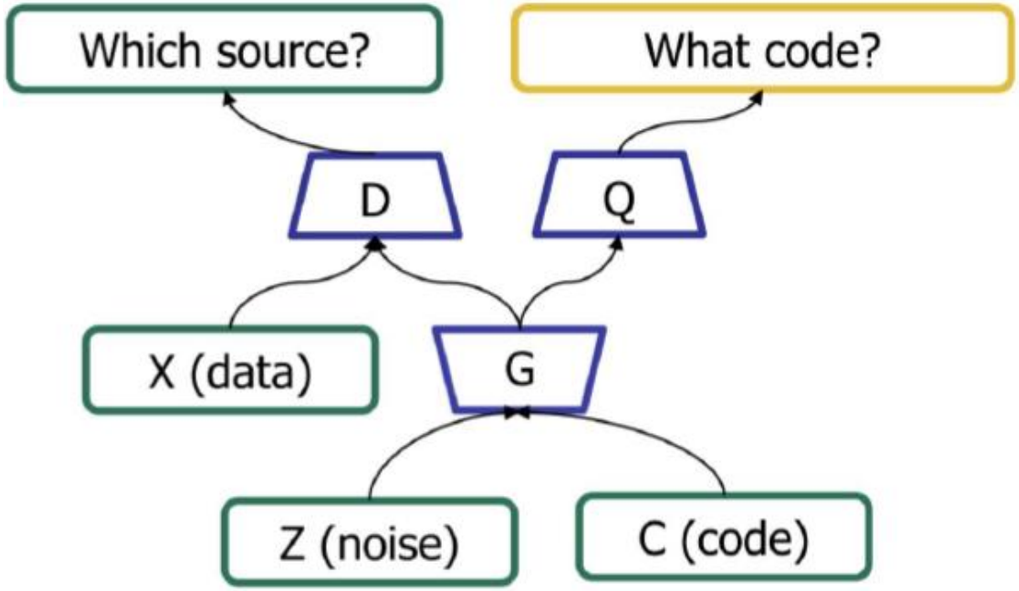


Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. «InfoGAN: Interpretable Representation Learning by Information Maximization Generative Adversarial Nets» // https://papers.nips.cc/paper/6399-infogan-interpretable-representation-learning-by-information-maximizing-generative-adversarial-nets.pdf

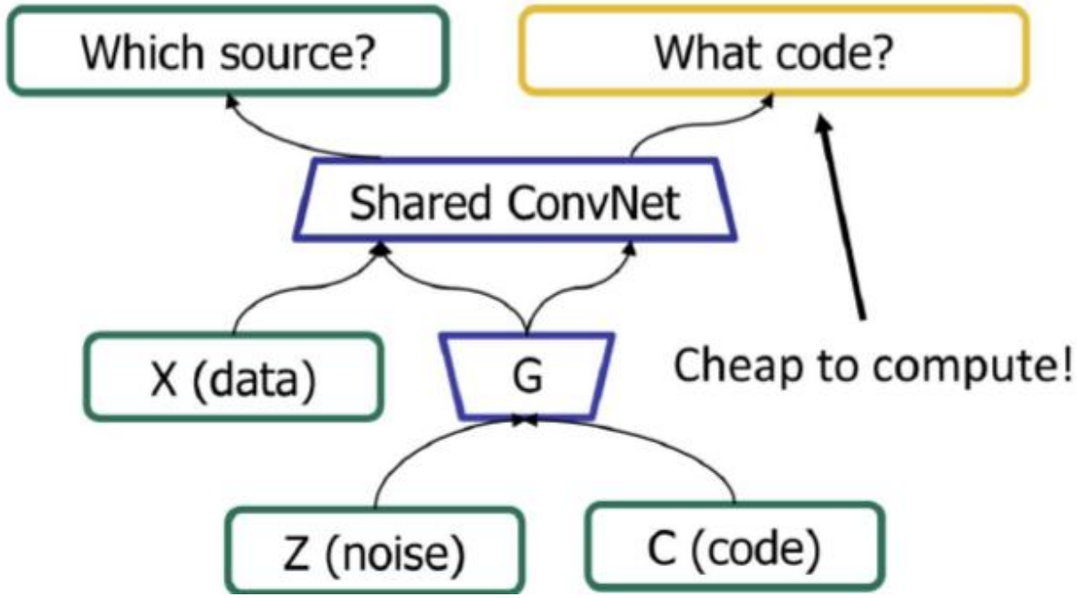
InfoGAN



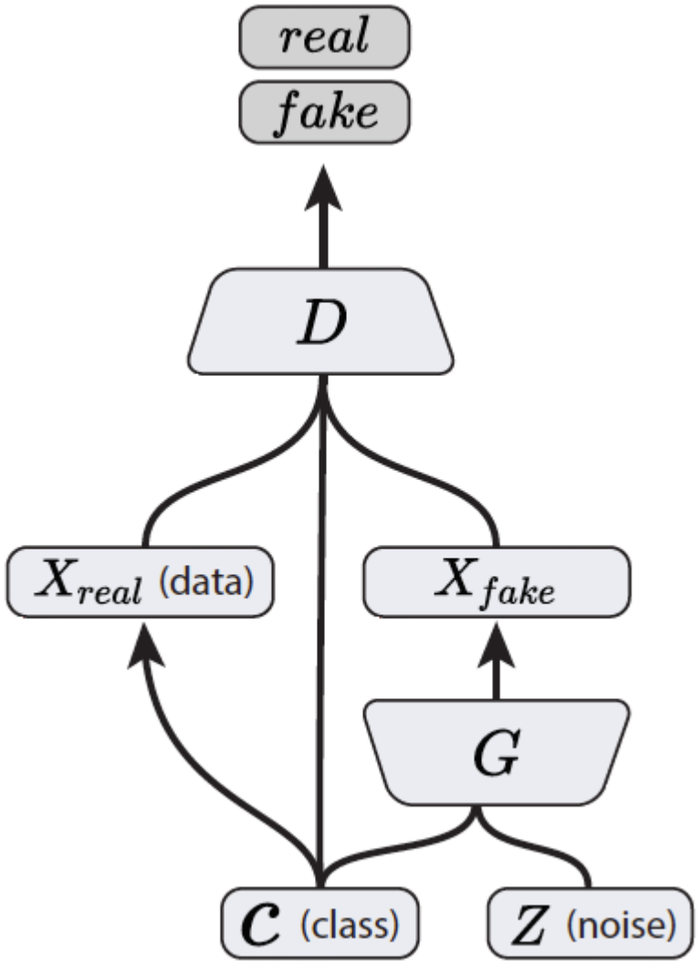
GAN



InfoGAN



Условные GANы (Conditional GANs)



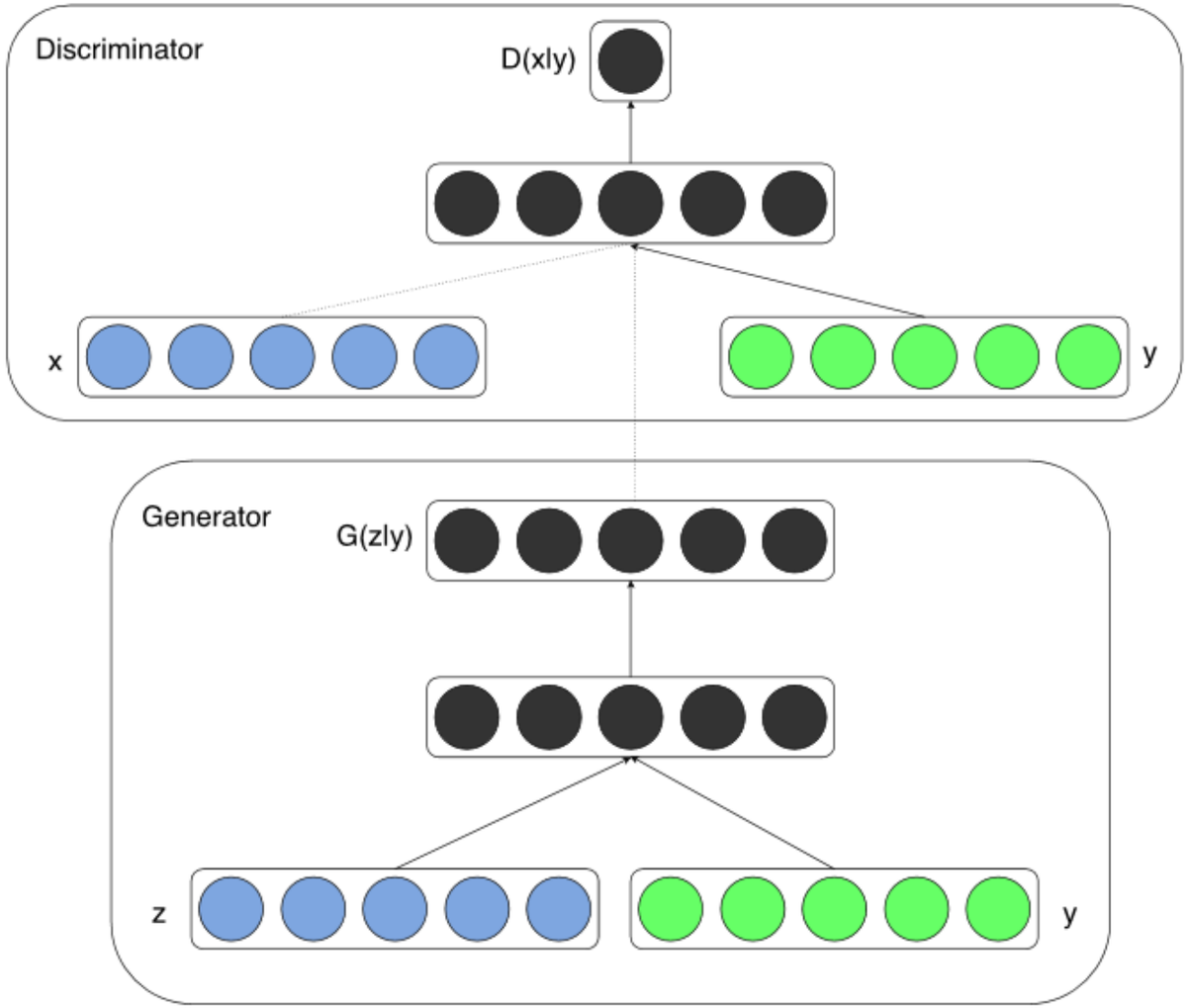


Figure 1: Conditional adversarial net

Mirza, Mehdi, and Simon Osindero «Conditional generative adversarial nets» // https://arxiv.org/pdf/1411.1784.pdf

Условные GANы для изменения возраста: Age-cGAN

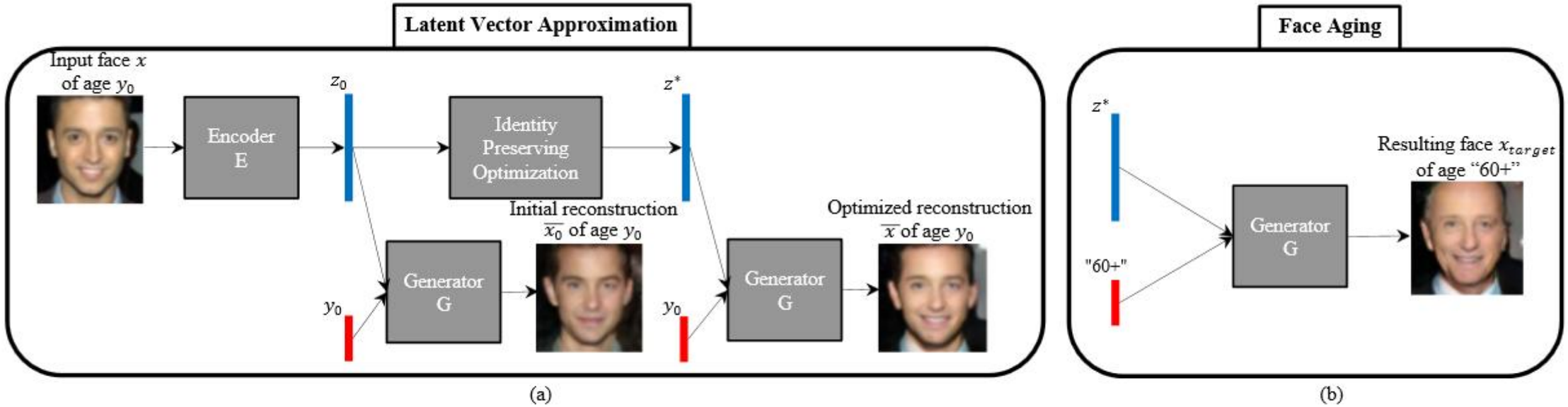


Fig. 1. Our face aging method. (a) approximation of the latent vector to reconstruct the input image; (b) switching the age condition at the input of the generator G to perform face aging.

есть выборка с помеченным возрастом – и это делаем доп. латентной переменной

потом переменную можно менять – «подменять возраст»

тут возраст категориальный – 6 категорий

Antipov, G., Baccouche, M., & Dugelay, J. L. «Face Aging With Conditional Generative Adversarial Networks» // https://arxiv.org/abs/1702.01983

Условные GANы для изменения возраста: Age-cGAN

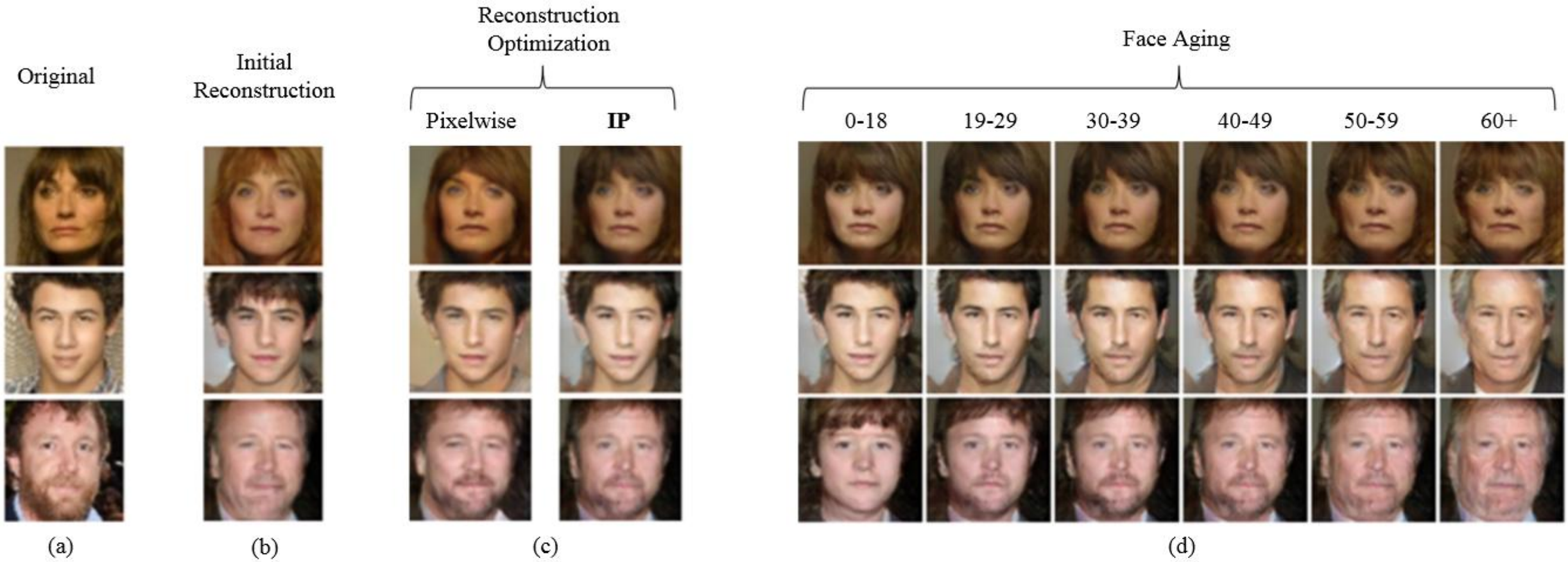


Fig. 3. Examples of face reconstruction and aging. (a) original test images, (b) reconstructed images generated using the initial latent approximations: z_0, (c) reconstructed images generated using the "Pixelwise" and "Identity-Preserving" optimized latent approximations: z^*_{pixel} and z^*_{IP}, and (d) aging of the reconstructed images generated using the identity-preserving z^*_{IP} latent approximations and conditioned on the respective age categories y (one per column).

Coupled GANs

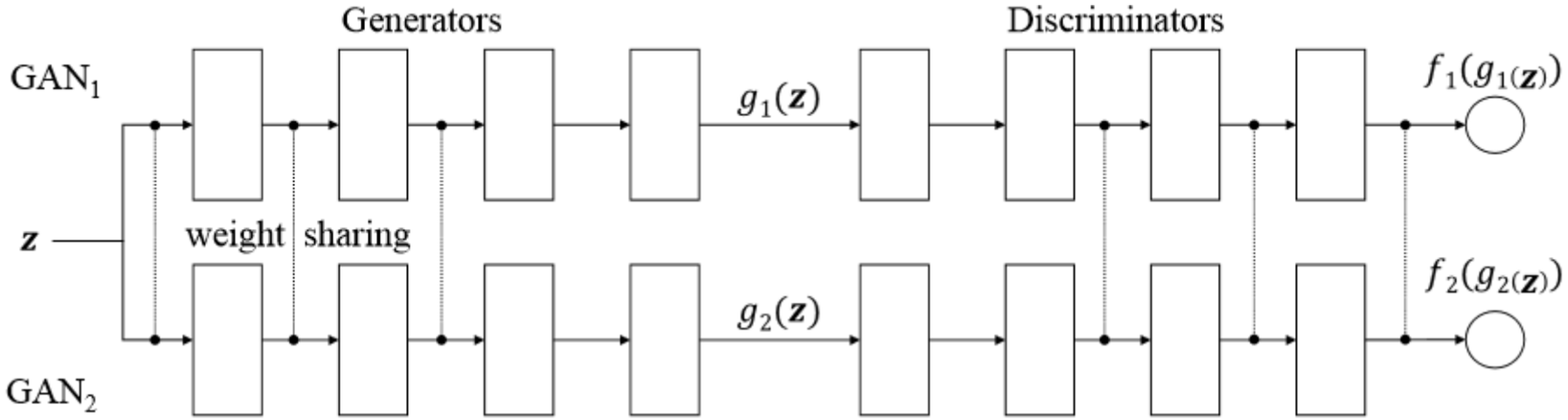


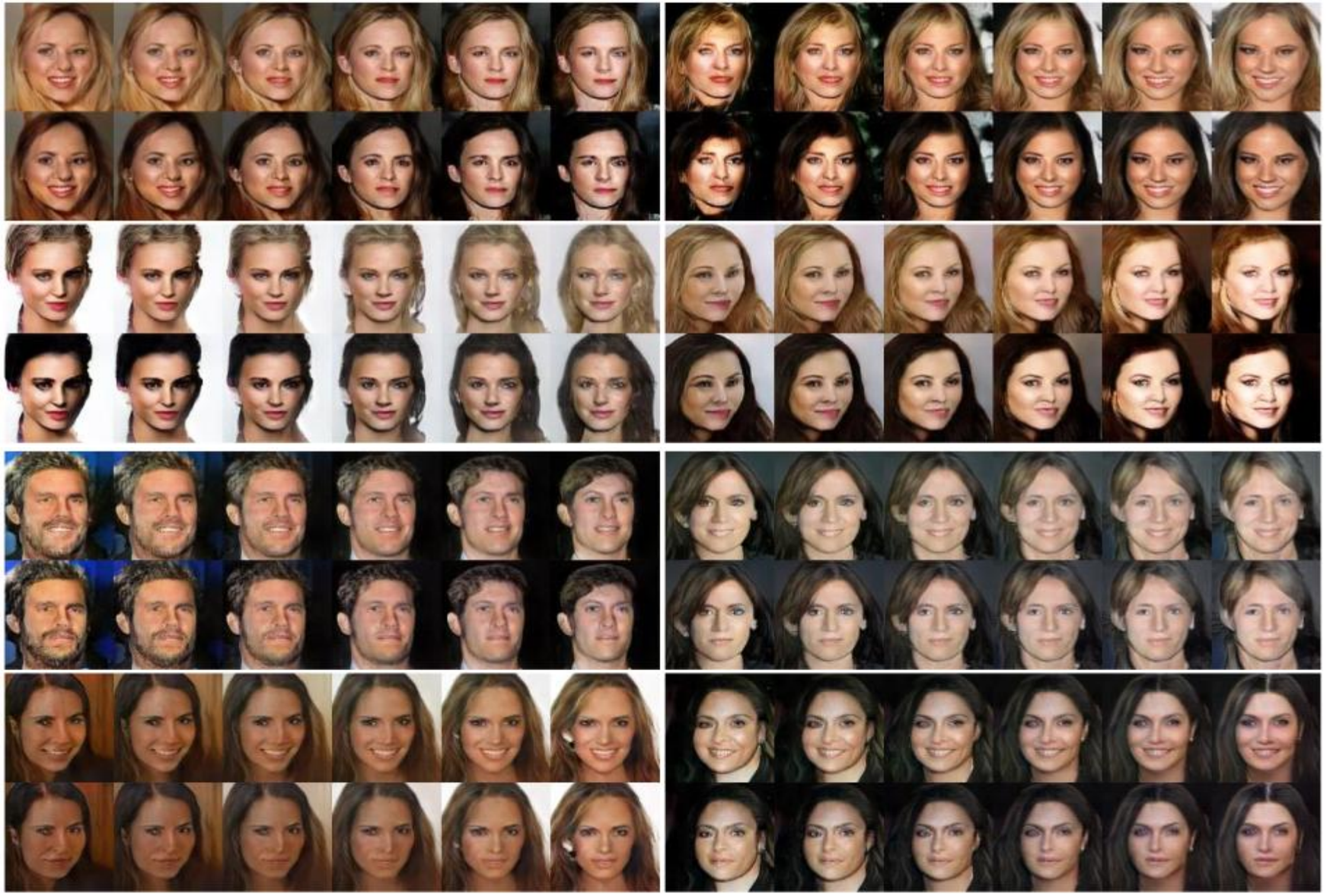
Figure 1: CoGAN consists of a pair of GANs: GAN_1 and GAN_2. Each has a generative model for synthesizing realistic images in one domain and a discriminative model for classifying whether an image is real or synthesized. We tie the weights of the first few layers (responsible for decoding high-level semantics) of the generative models, g_1 and g_2. We also tie the weights of the last few layers (responsible for encoding high-level semantics) of the discriminative models, f_1 and f_2. This weight-sharing constraint allows CoGAN to learn a joint distribution of images without correspondence supervision. A trained CoGAN can be used to synthesize pairs of corresponding images—pairs of images sharing the same high-level abstraction but having different low-level realizations.

Liu, Ming-Yu, and Oncel Tuzel. «Coupled generative adversarial networks». NIPS (2016).

Coupled GANs: генерация из одного латентного кода z

цвет волос

выражение лица



Coupled GANs: генерация из одного латентного кода z

очки

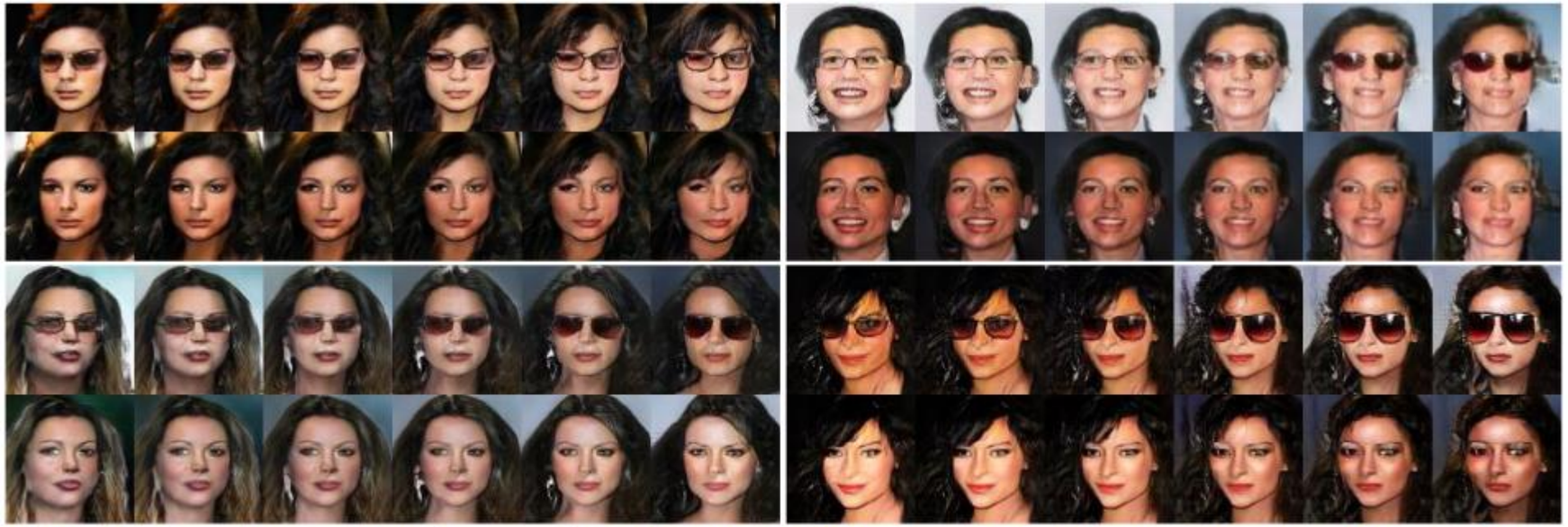


Figure 4: Generation of face images with different attributes using CoGAN. From top to bottom, the figure shows pair face generation results for the blond-hair, smiling, and eyeglasses attributes. For each pair, the 1st row contains faces with the attribute, while the 2nd row contains corresponding faces without the attribute.

Как оценивать качество (сгенерированные картинки)

Inception score (Salimans et al 2016) !!!

порождённые картинки → SOTA-классификатор

1) если уверенная классификация, то похожи на настоящие
2) если распределения в слоях похожи на настоящие

Frechet Inception Distance (FID) –
сравнение активаций предобученной сети

+ можно померить разнообразие

Различные виды GAN

перечень видов со ссылками на первоисточники

https://github.com/wiseodd/generative-models

https://github.com/hwalsuklee/tensorflow-generative-model-collections

ProGAN https://www.youtube.com/watch?v=G06dEcZ-QTg

BigGAN https://www.youtube.com/watch?v=YY6LrQSxIbc

Scribbles2Photos https://www.youtube.com/watch?v=5jfViPdYLic

Ian Goodfellow «NIPS 2016 Tutorial:Generative Adversarial Networks»

https://arxiv.org/pdf/1701.00160.pdf