курс «Глубокое обучение»

GAN (часть 2)

Александр Дьяконов

План

Deep Convolutional Generative Adversarial Networks (DCGAN)

Условные состязательные сети (cGAN)

pix2pix / pix2pixHD

Проблема отсутствия выборки - CycleGAN

Аугментированная циклическая GAN (Augmented CycleGAN

BiGAN (Bidirectional)

BigGAN: Генерация изображений / интерполяция

BigBiGAN = BiGAN + BigGAN

SAGAN (Self-Attention Generative Adversarial Networks)

Semi-supervised GAN (SGAN)

AC-GAN: auxiliary classifier GAN

CAN: Creative Adversarial Networks

ProGAN (NVIDIA)

InfoGAN

Условные GANы (Conditional GANs)

Coupled GANs

Deep Convolutional Generative Adversarial Networks (DCGAN)

- **Полносвёрточная**

- **Нет пулинга**

в дискриминаторе – strided convolutions

в генераторе – fractional-strided convolutions

- **BN после каждого слоя**

кроме последнего G и первого D

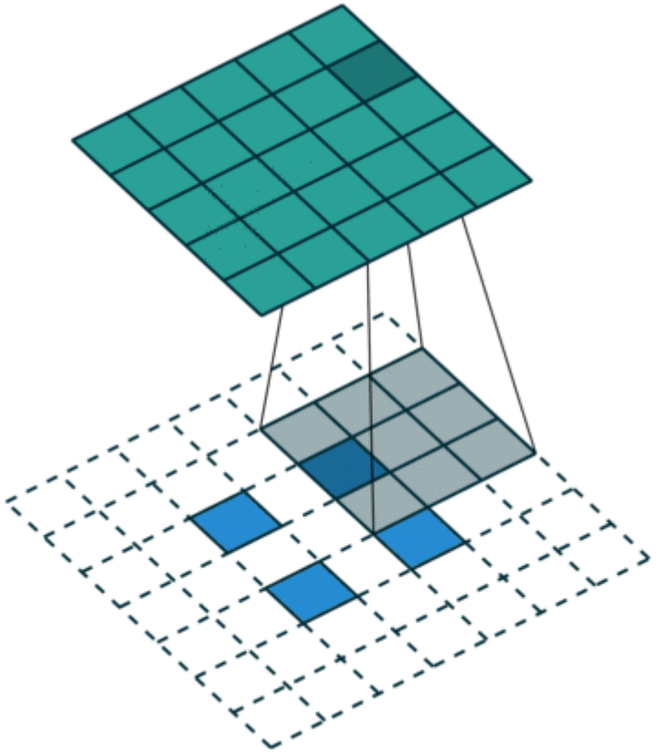
- **FC-слоёв нет (вместо этого reshape)**

сказали, что с AvrPool хуже сходимость

- **активации**

генератор: ReLU для скрытых слоёв, Tanh для выходного слоя

дискриминатор: LeakyReLU + sigmoid



fractional-strided convolutions

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)

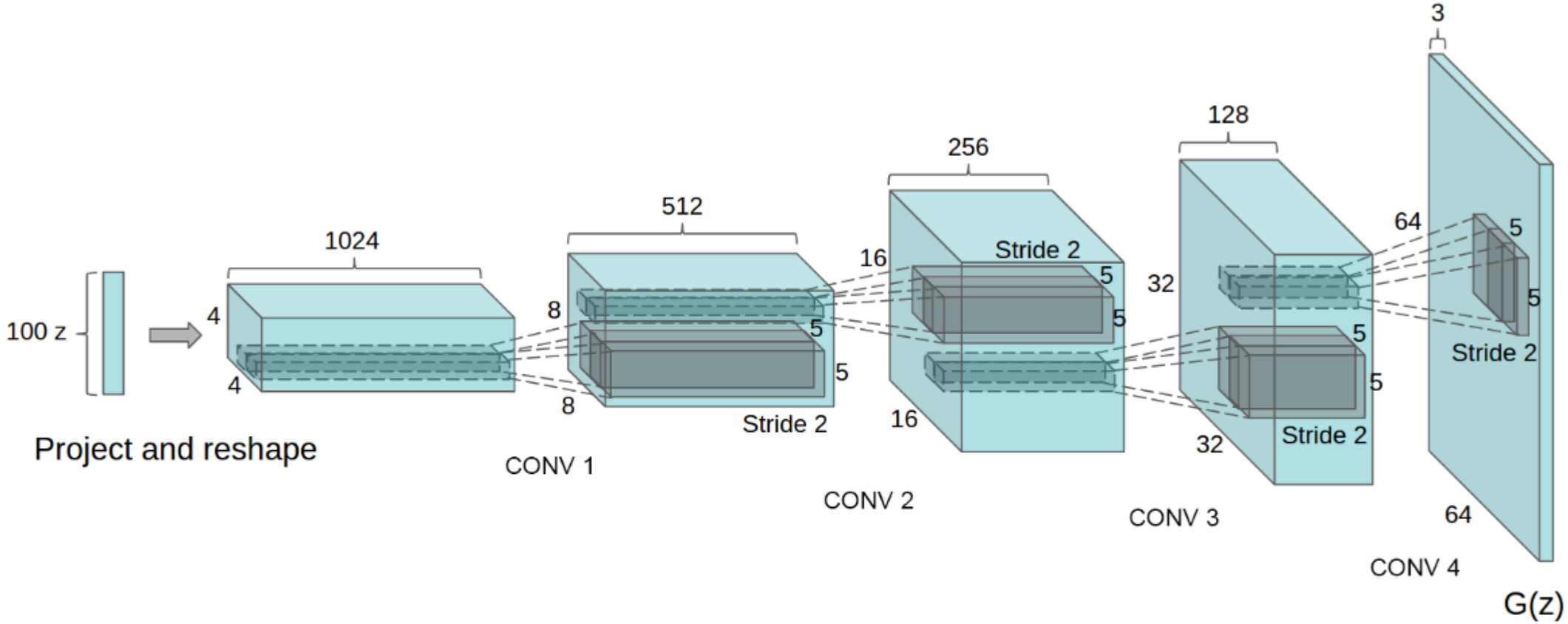


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) 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)

```
class Generator(nn.Module):
    def __init__(self, ngpu):
        super(Generator, self).__init__()
        self.ngpu = ngpu
        self.main = nn.Sequential(
            # input is Z, going into a convolution
            nn.ConvTranspose2d( nz, ngf * 8, 4, 1, 0, bias=False),
            nn.BatchNorm2d(ngf * 8),
            nn.ReLU(True),
            # state size. ``(ngf*8) x 4 x 4``
            nn.ConvTranspose2d(ngf * 8, ngf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 4),
            nn.ReLU(True),
            # state size. ``(ngf*4) x 8 x 8``
            nn.ConvTranspose2d( ngf * 4, ngf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 2),
            nn.ReLU(True),
            # state size. ``(ngf*2) x 16 x 16``
            nn.ConvTranspose2d( ngf * 2, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(True),
            # state size. ``(ngf) x 32 x 32``
            nn.ConvTranspose2d( ngf, nc, 4, 2, 1, bias=False),
            nn.Tanh()
            # state size. ``(nc) x 64 x 64``
        )
```

https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

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 .

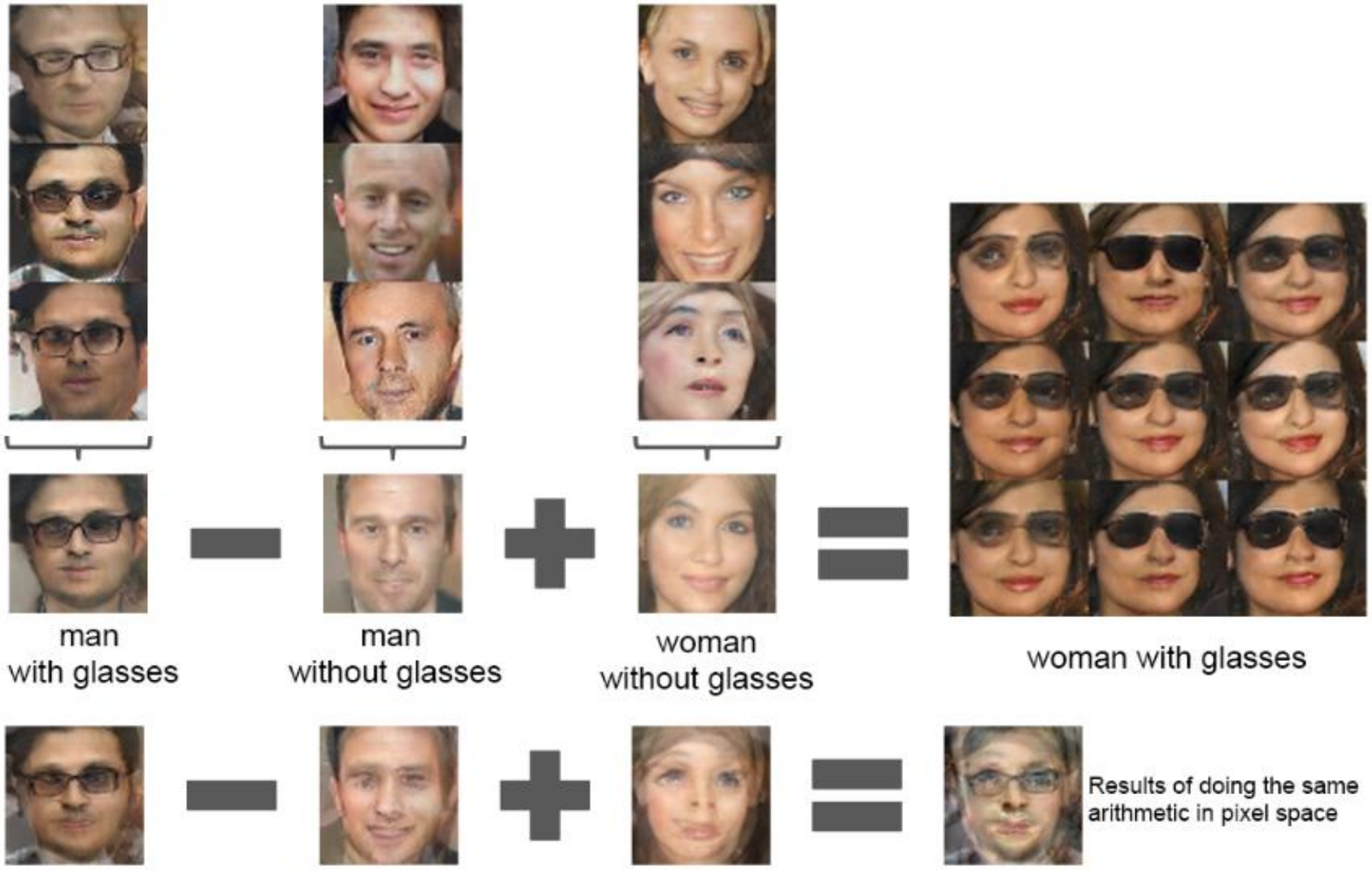


Figure 7: Vector arithmetic for visual concepts. For each column, the Z vectors of samples are averaged. Arithmetic was then performed on the mean vectors creating a new vector Y. The center sample on the right hand side is produce by feeding Y as input to the generator. To demonstrate the interpolation 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.

DCGAN: обложки дисков

Условные состязательные сети (cGAN)

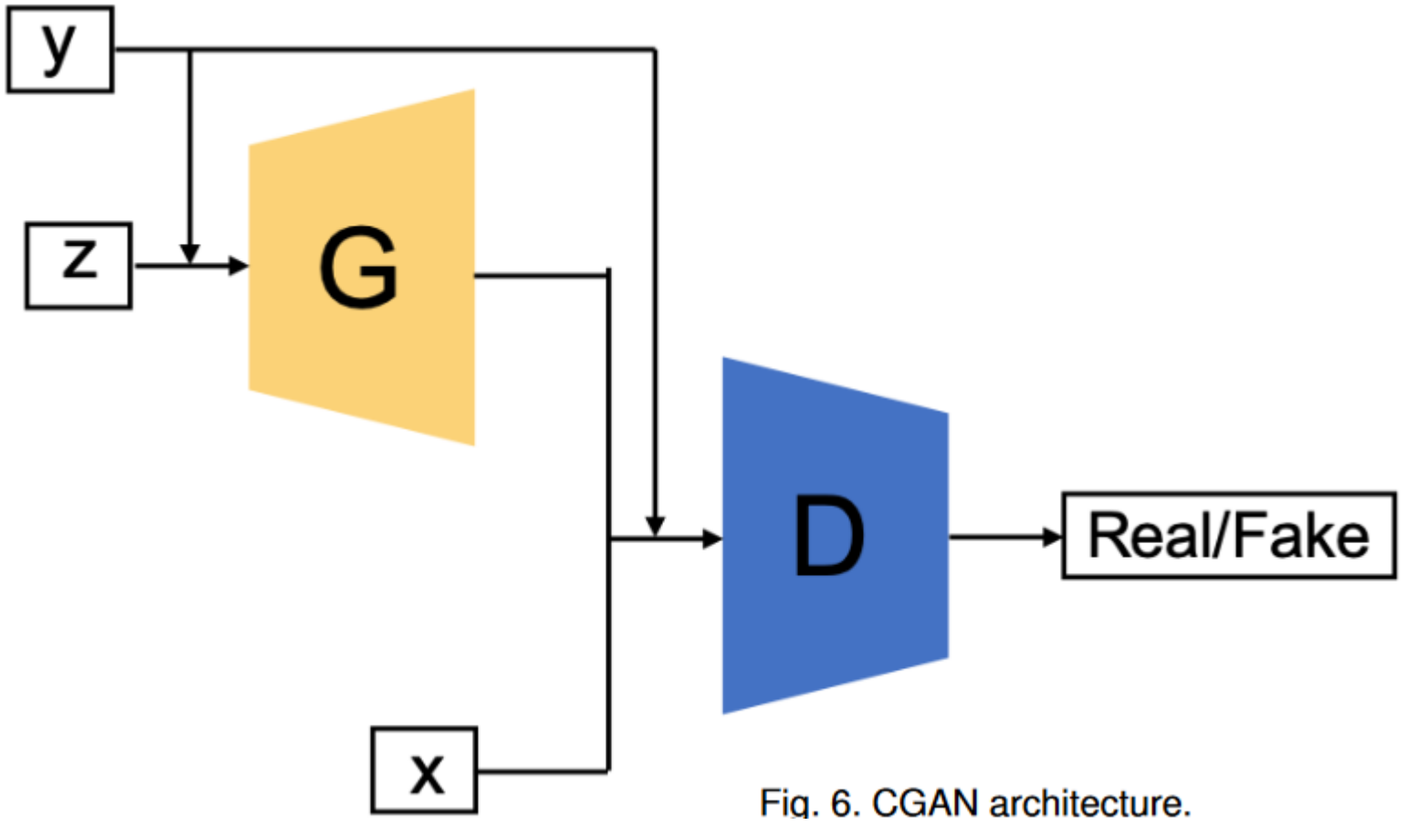
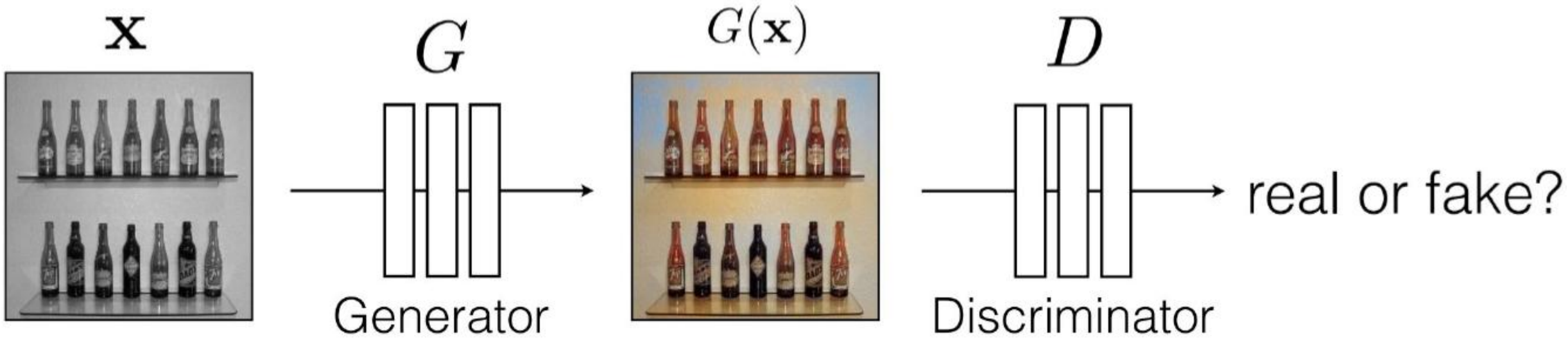


Fig. 6. CGAN architecture.

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_r} \log[D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}} \log[1 - D(G(\mathbf{z}|\mathbf{y}))]$$

https://arxiv.org/pdf/1906.01529.pdf

Условные состязательные сети (cGAN)

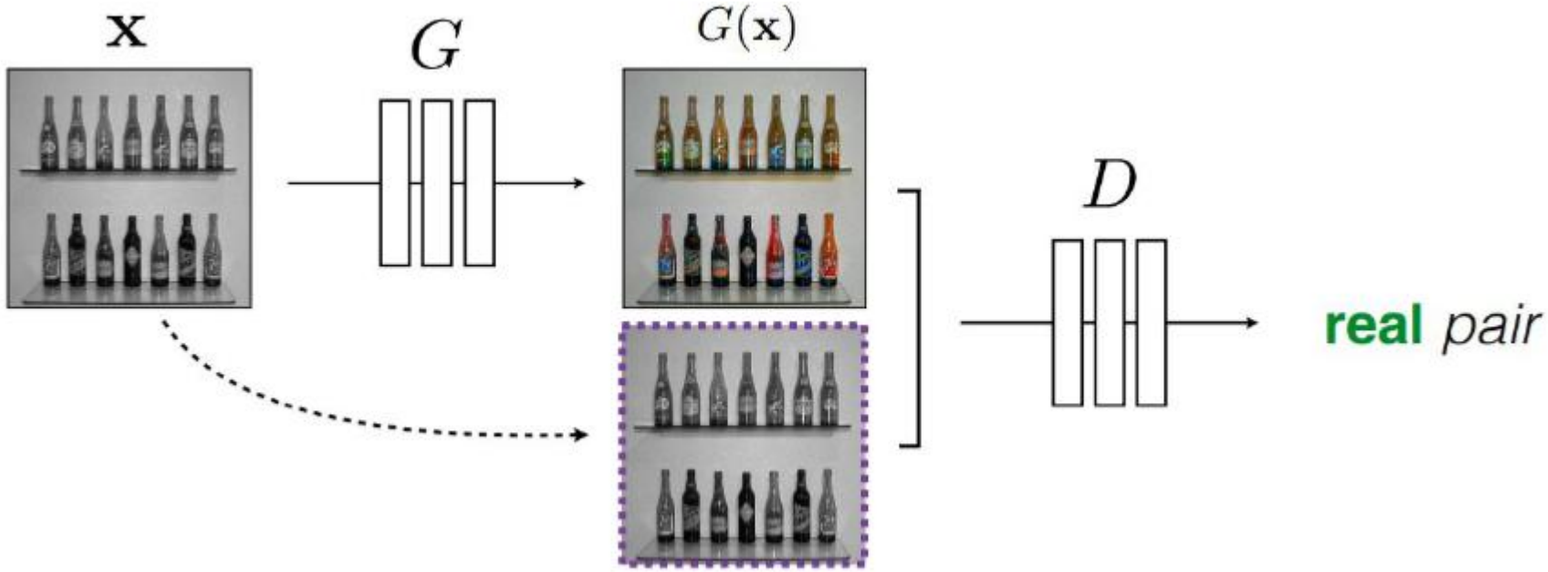


но тут проблема, что сеть начнёт генерировать что-то реальное, например «котиков» и условие будет фиктивным

т.е. тут м.б. отображение не из шума

[Phillip Isola]

Условные состязательные сети (cGAN)



лучше определять реальность/фейковость пары

«условие» подаётся в генератор и дискриминатор

Pix2pix с условными состязательными сетями (cGAN)

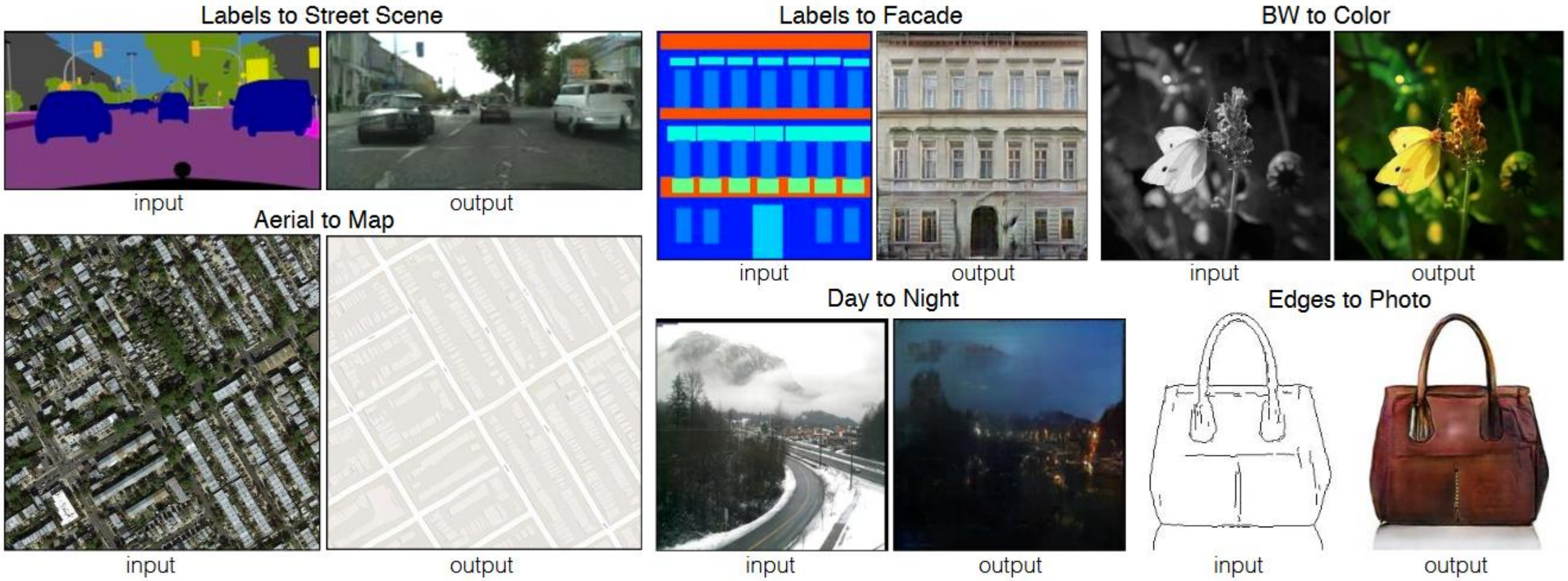


Figure 1: Many problems in image processing, graphics, and vision involve translating an input image into a corresponding output image. These problems are often treated with application-specific algorithms, even though the setting is always the same: map pixels to pixels. Conditional adversarial nets 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)

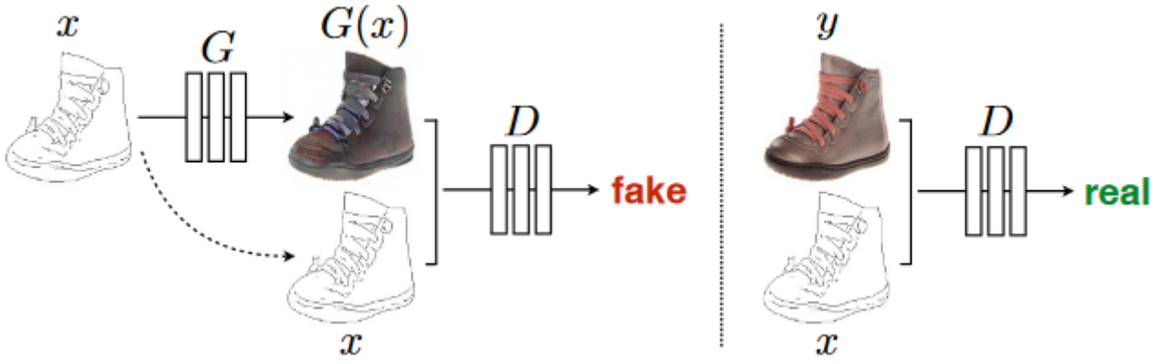


Figure 2: Training a conditional GAN to map edges→photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the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

$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x,G(x,z))],$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1] \quad G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G).$$

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)

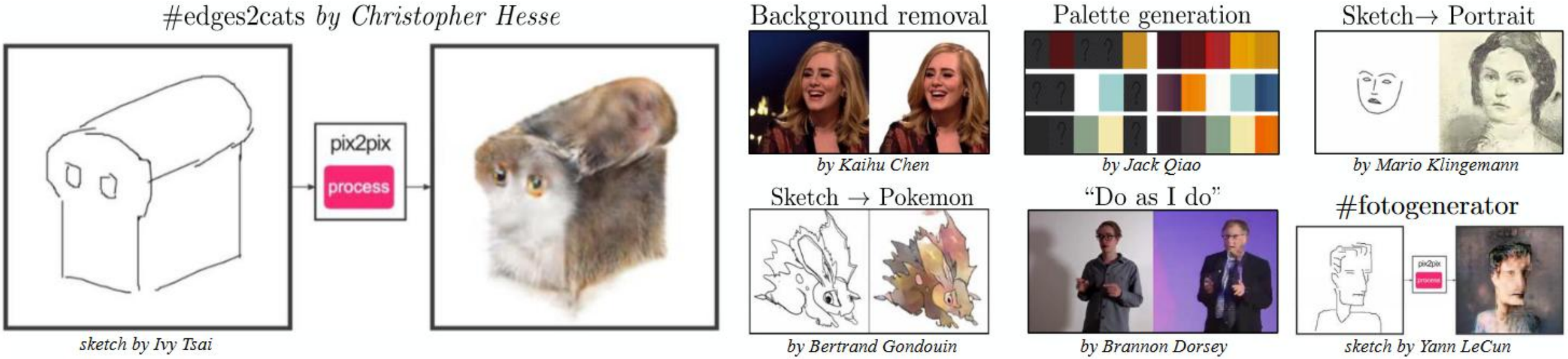


Figure 11: Example applications developed by online community based on our `pix2pix` codebase: *#edges2cats* [3] by Christopher Hesse, *Background removal* [6] by Kaihu Chen, *Palette generation* [5] by Jack Qiao, *Sketch → Portrait* [7] by Mario Klingemann, *Sketch→ Pokemon* [1] by Bertrand Gondouin,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

различные результаты сообщества с помощью разработанной библиотеки

pix2pixHD

(a) Synthesized result

Cascaded refinement network [5]

Our result

(b) Application: Change label types

(c) Application: Edit object appearance

Figure 1: We propose a generative adversarial framework for synthesizing 2048×1024 images from semantic label maps (lower left corner in (a)). Compared to previous work [5], our results express more natural textures and details. (b) We can change labels in the original label map to create new scenes, like replacing trees with buildings. (c) Our framework also allows a user to edit the appearance of individual objects in the scene, e.g. changing the color of a car or the texture of a road. Please visit our website for more side-by-side comparisons as well as interactive editing demos.

Ting-Chun Wang et al. «High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs» // https://arxiv.org/abs/1711.11585

pix2pixHD

pix2pix мог только 256×256, если больше – хуже

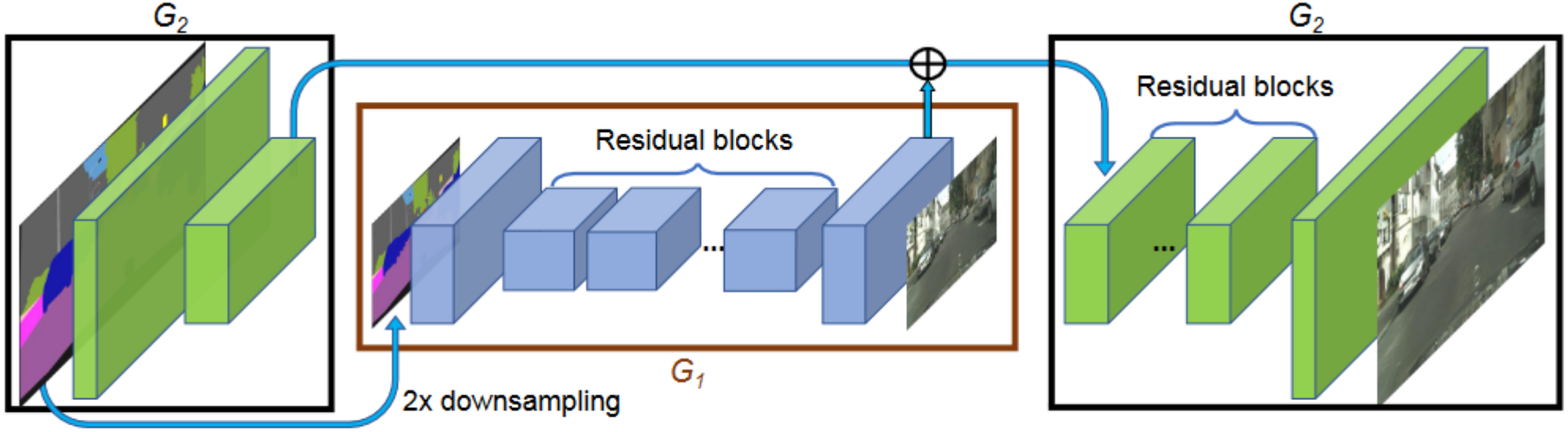


Figure 3: Network architecture of our generator. We first train a residual network G_1 on lower resolution images. Then, another residual network G_2 is appended to G_1 and the two networks are trained jointly on high resolution images. Specifically, the input to the residual blocks in G_2 is the element-wise sum of the feature map from G_2 and the last feature map from G_1.

генератор = G1 (глобальный генератор) +

оперирует на высоком разрешении 1024×512

+ G2 (локальный улучшатель)

Аналогично можно использовать три сети для разрешения 4096×2048

pix2pixHD

новшество в работе – улучшающее слагаемое (Improved adversarial loss) – feature matching loss

$$L_{FM}(G, D_k) = E_{(s,x)} \sum_{i=1}^{T} \frac{1}{N_i} \left[\left\| D_k^{(i)}(s, x) - D_k^{(i)}(s, G(s)) \right\|_1 \right]$$

суммирование по слоям дискриминатора

итоговая функция ошибки:

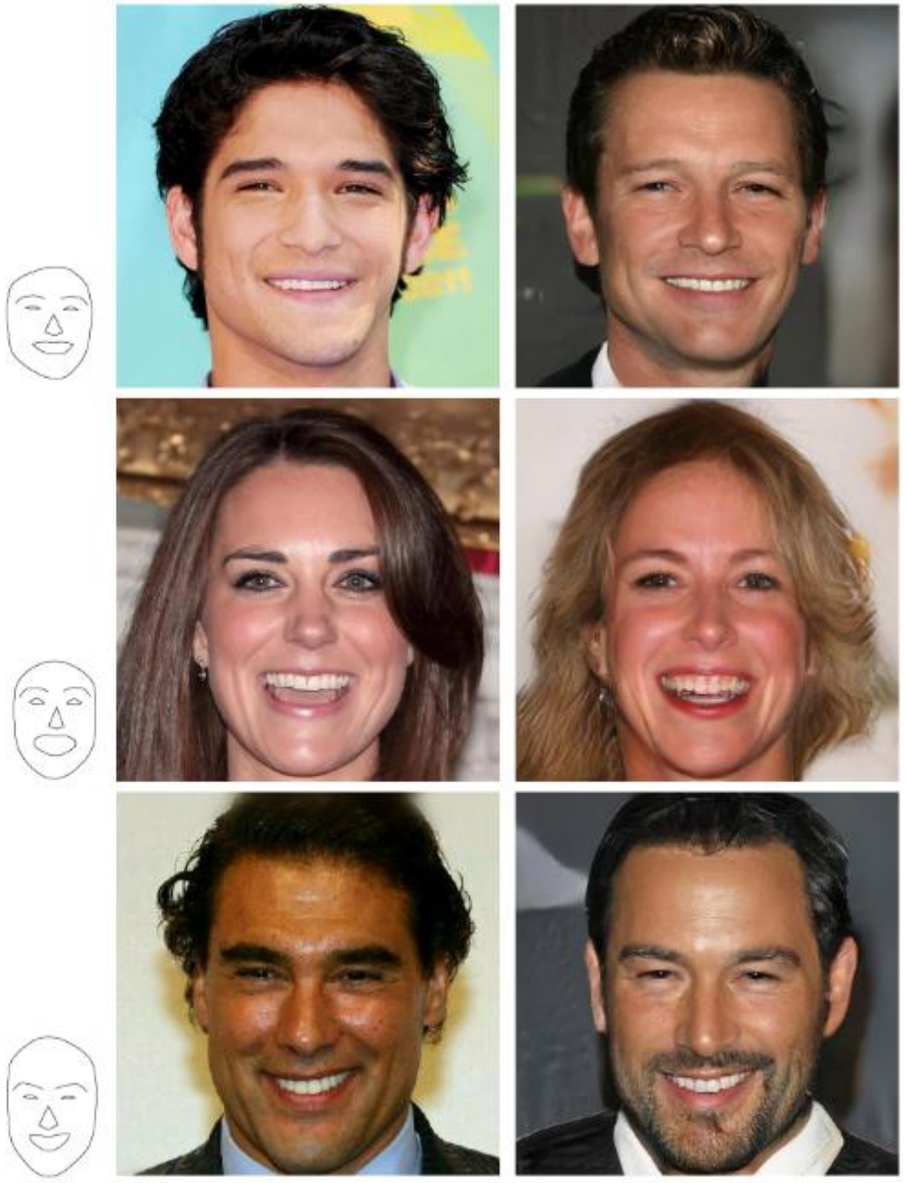
$$\min_G \max_{D_1, D_2, D_3} \sum_{k=1,2,3} \left(L_{GAN}(G, D_k) + \lambda L_{FM}(G, D_k) \right)$$

здесь 3 дискриминатора на разных масштабах, чтобы справиться с большим разрешением

pix2pixHD

(a) Original image (b) Our result

Figure 11: Results on the ADE20K dataset [63] (label maps shown at lower left corners in (a)). Our method generates images at similar level of realism as the original images.

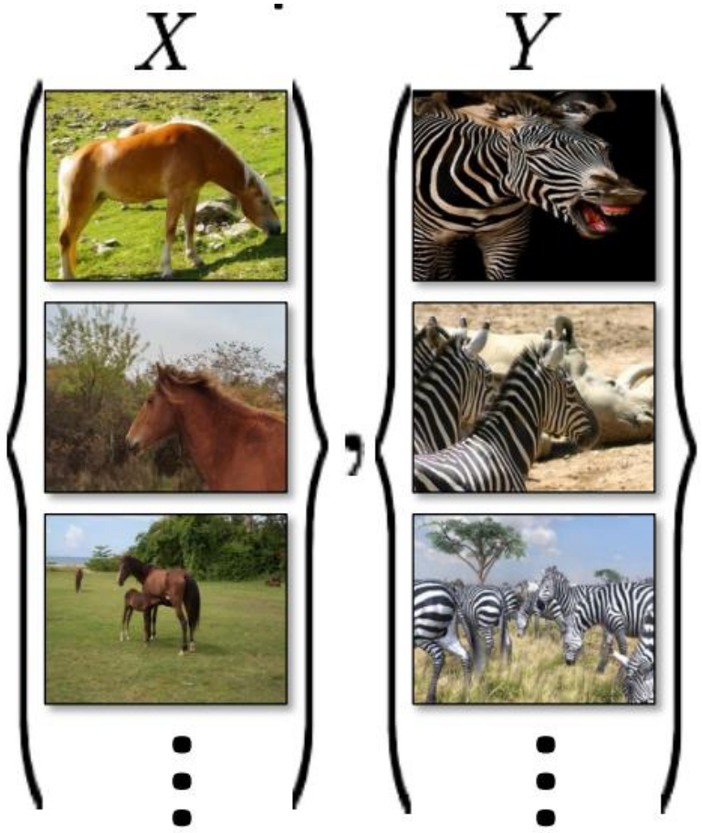


Проблема отсутствия выборки

есть размеченная выборка для pix → pix

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

есть просто представители двух множеств



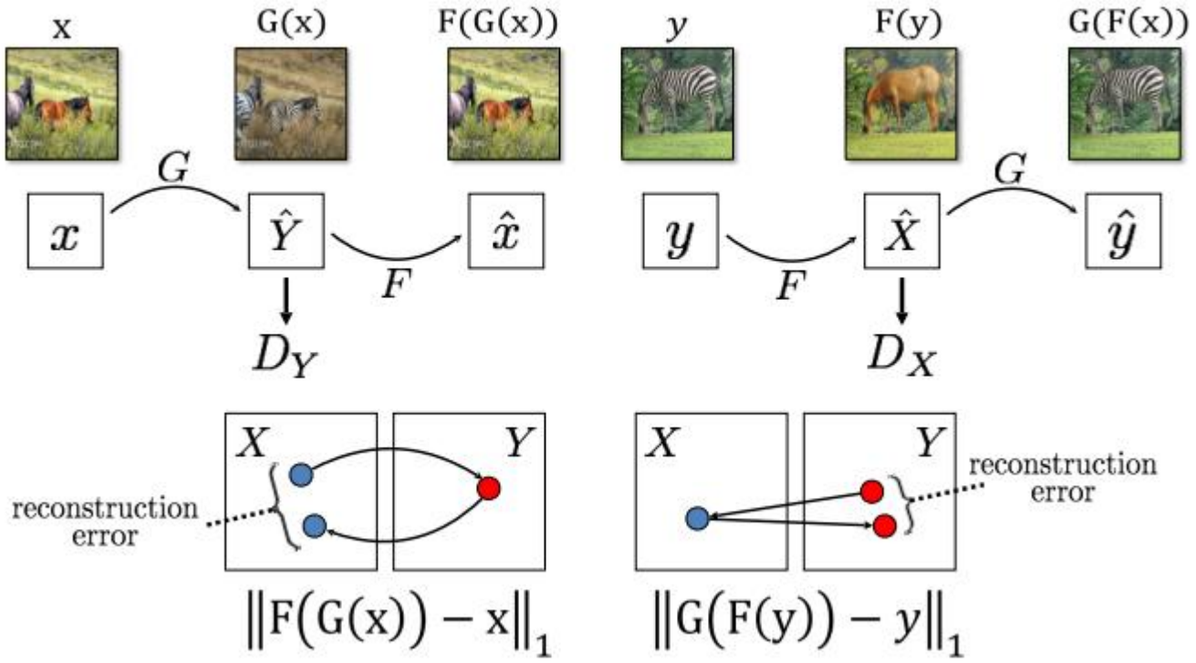
можно ли осуществить трансфер?

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

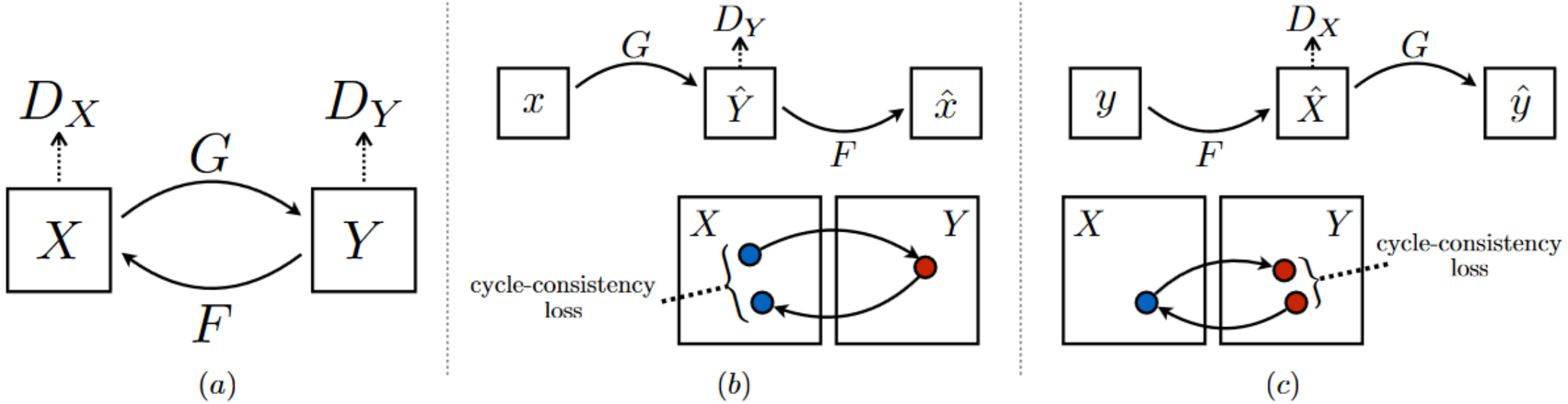


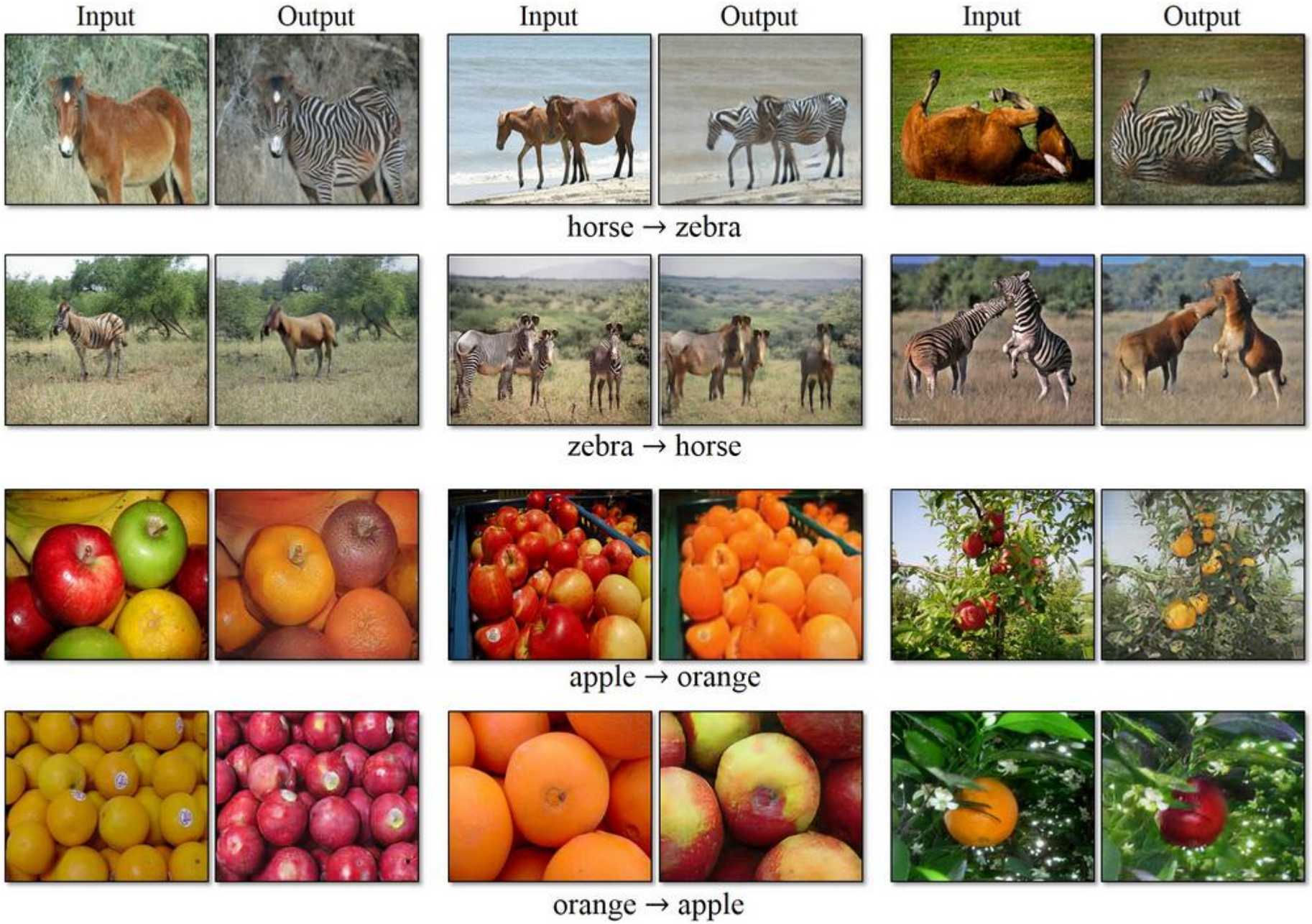
Figure 3: (a) Our model contains two mapping functions $G : X \to Y$ and $F : Y \to X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *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

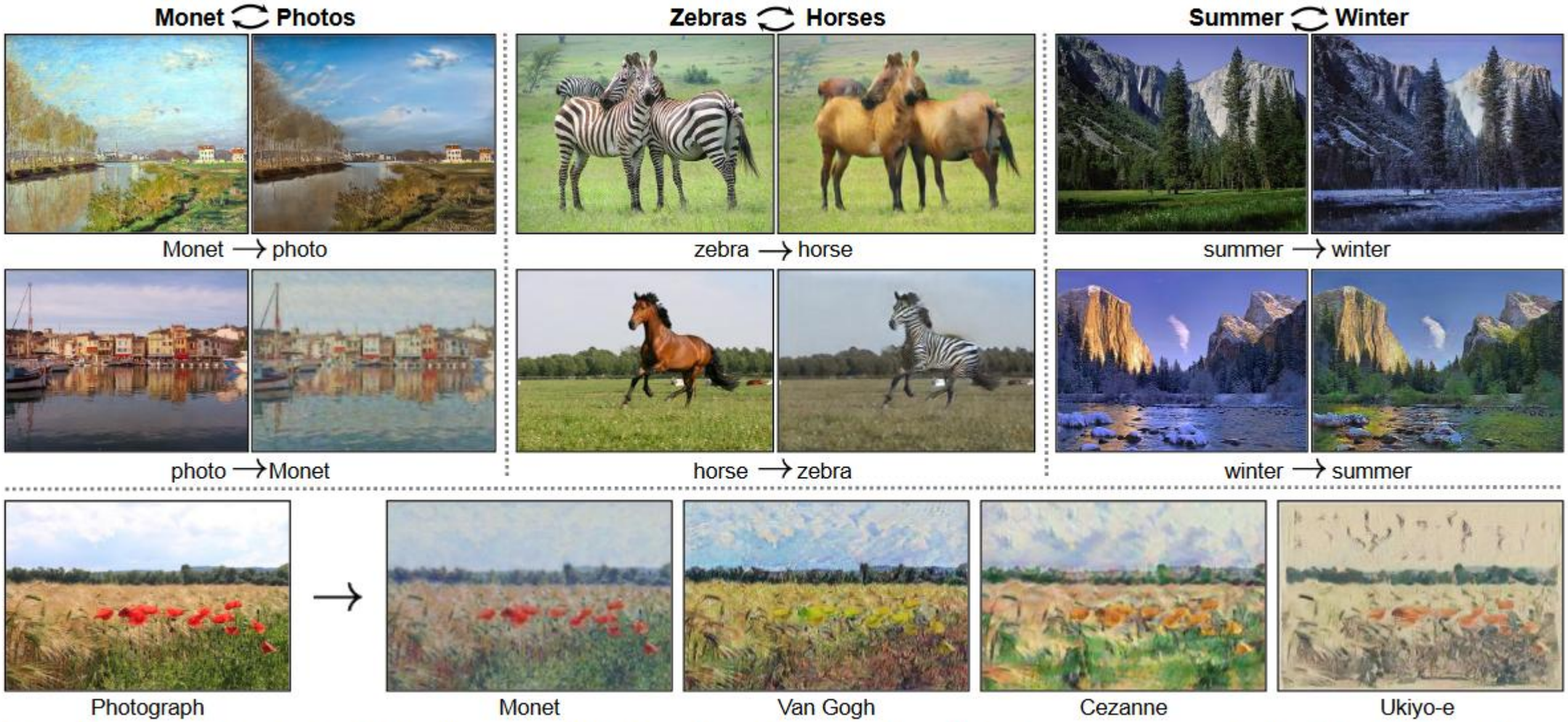


Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos 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

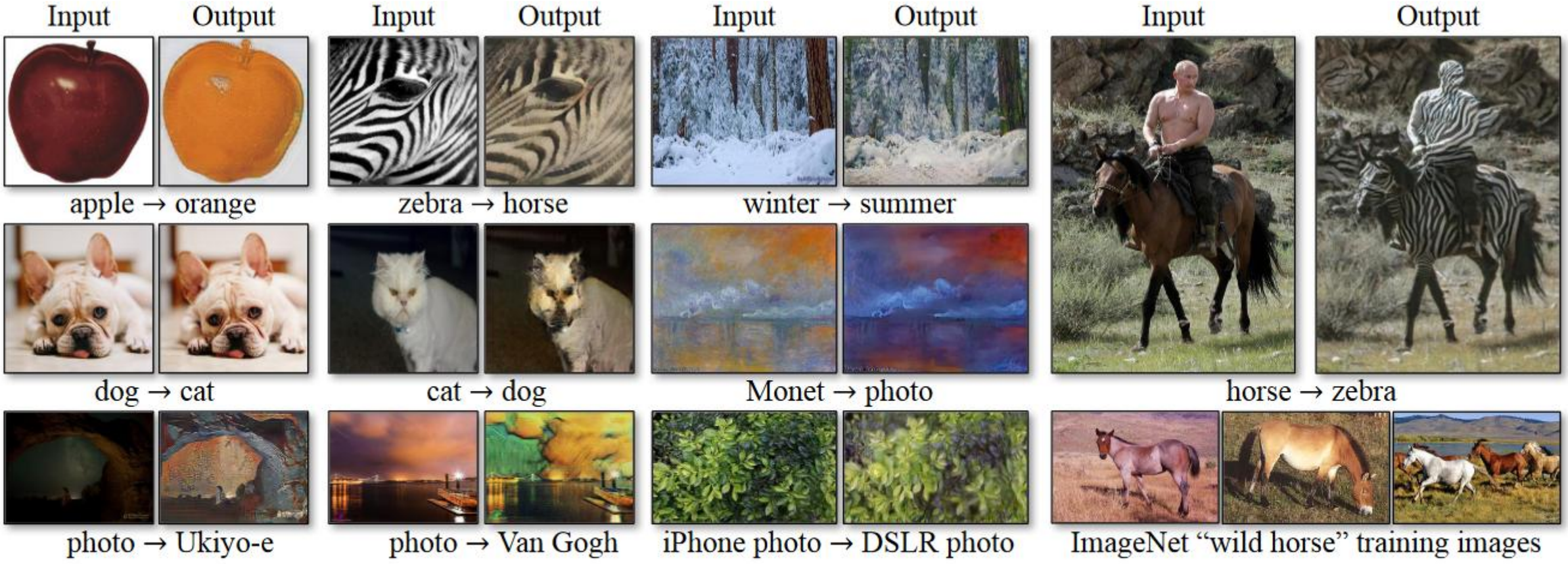
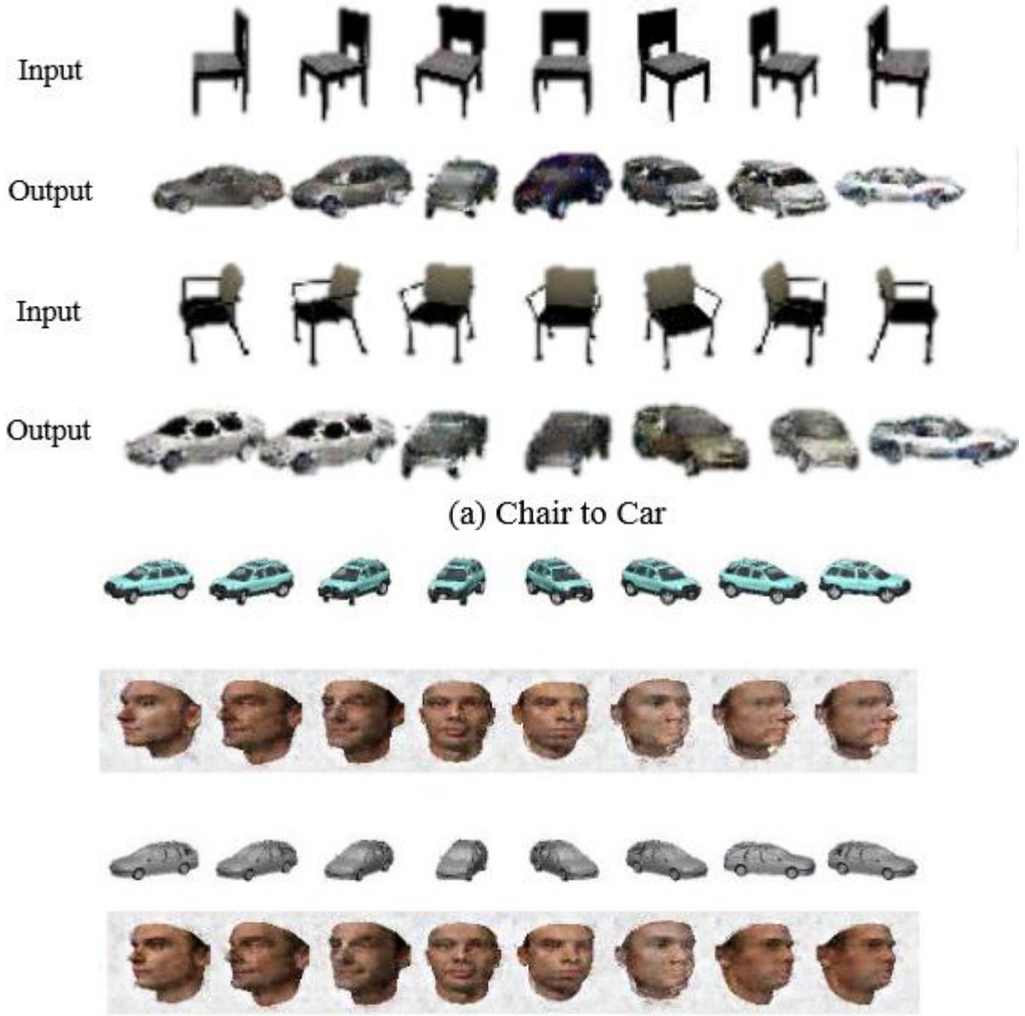


Figure 17: Typical failure cases of our method. Left: in the task of dog→cat transfiguration, CycleGAN can only make minimal changes to the input. Right: CycleGAN also fails in this horse → zebra example as our model has not seen images of horseback riding during training. Please see our website for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

Идеи аналогичные CycleGAN

DiscoGAN



(a) Chair to Car

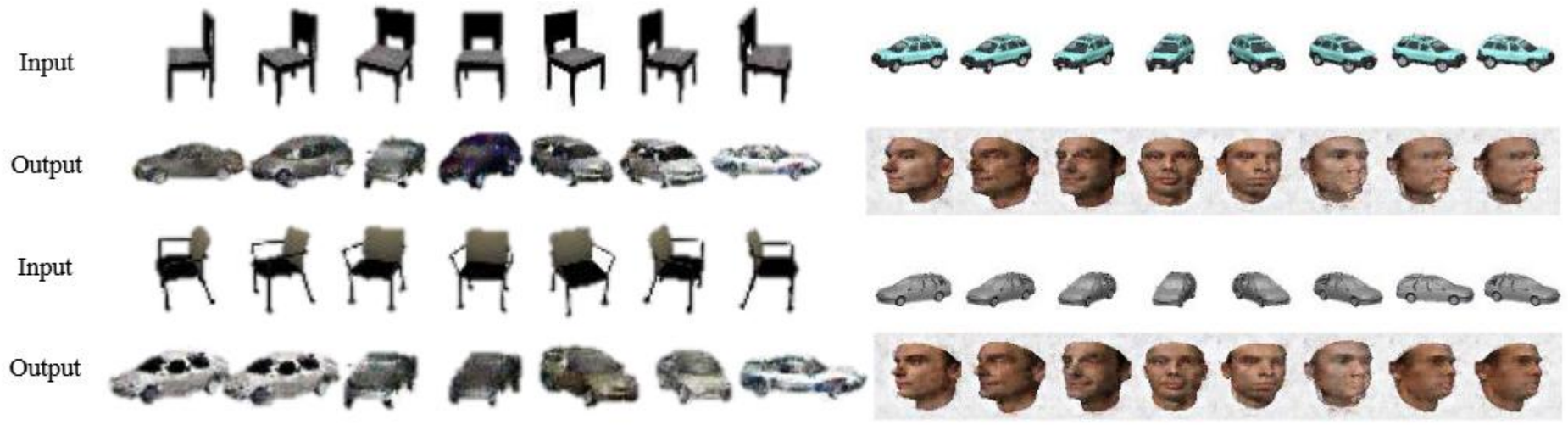
(b) Car to Face

https://arxiv.org/pdf/1703.05192.pdf

DualGAN

но тут Wasserstein GAN

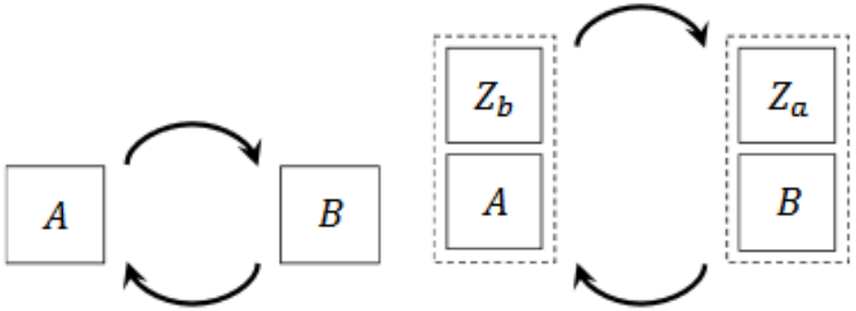
https://arxiv.org/pdf/1704.02510.pdf



(a) Chair to Car (b) Car to Face

Figure 8. Discovering relations of images from visually very different object classes. (a) chair to car translation. DiscoGAN is trained on chair and car images (b) car to face translation. DiscoGAN is trained on car and face images. Our model successfully pairs images with similar orientation.

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

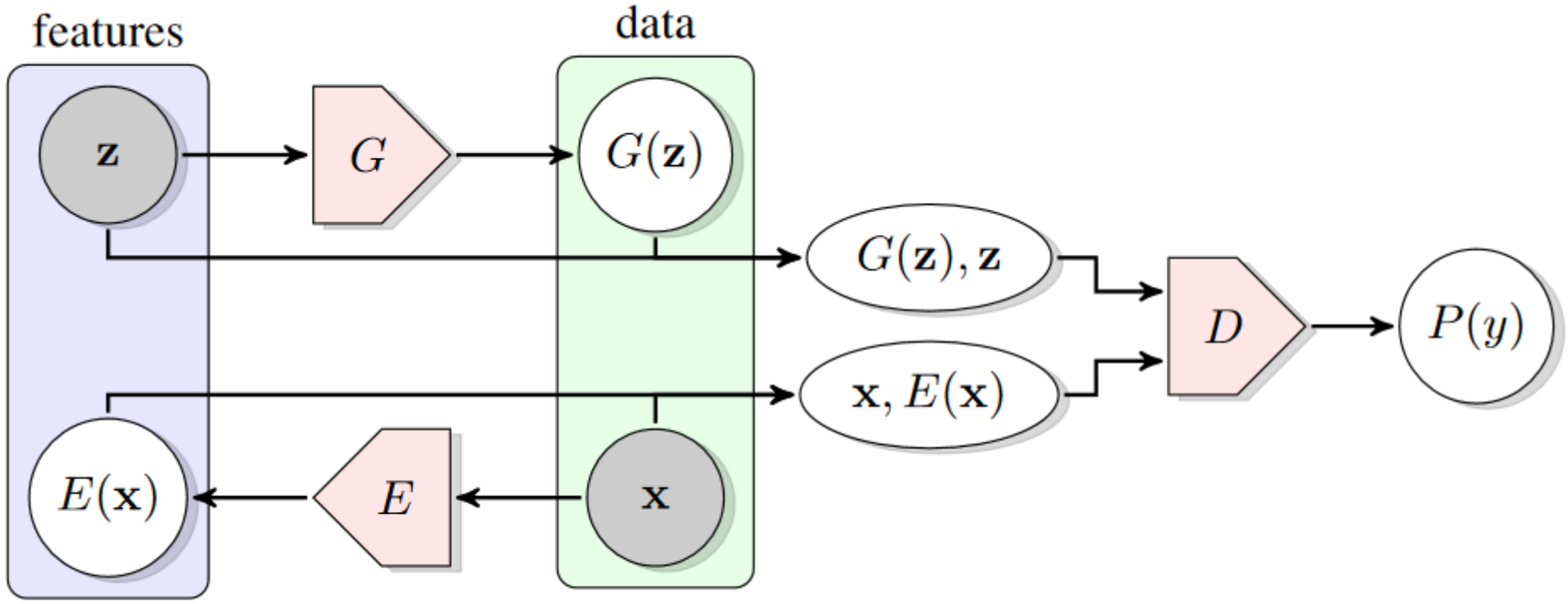


Figure 1: The structure of Bidirectional Generative Adversarial Networks (BiGAN).

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

BiGAN (Bidirectional)

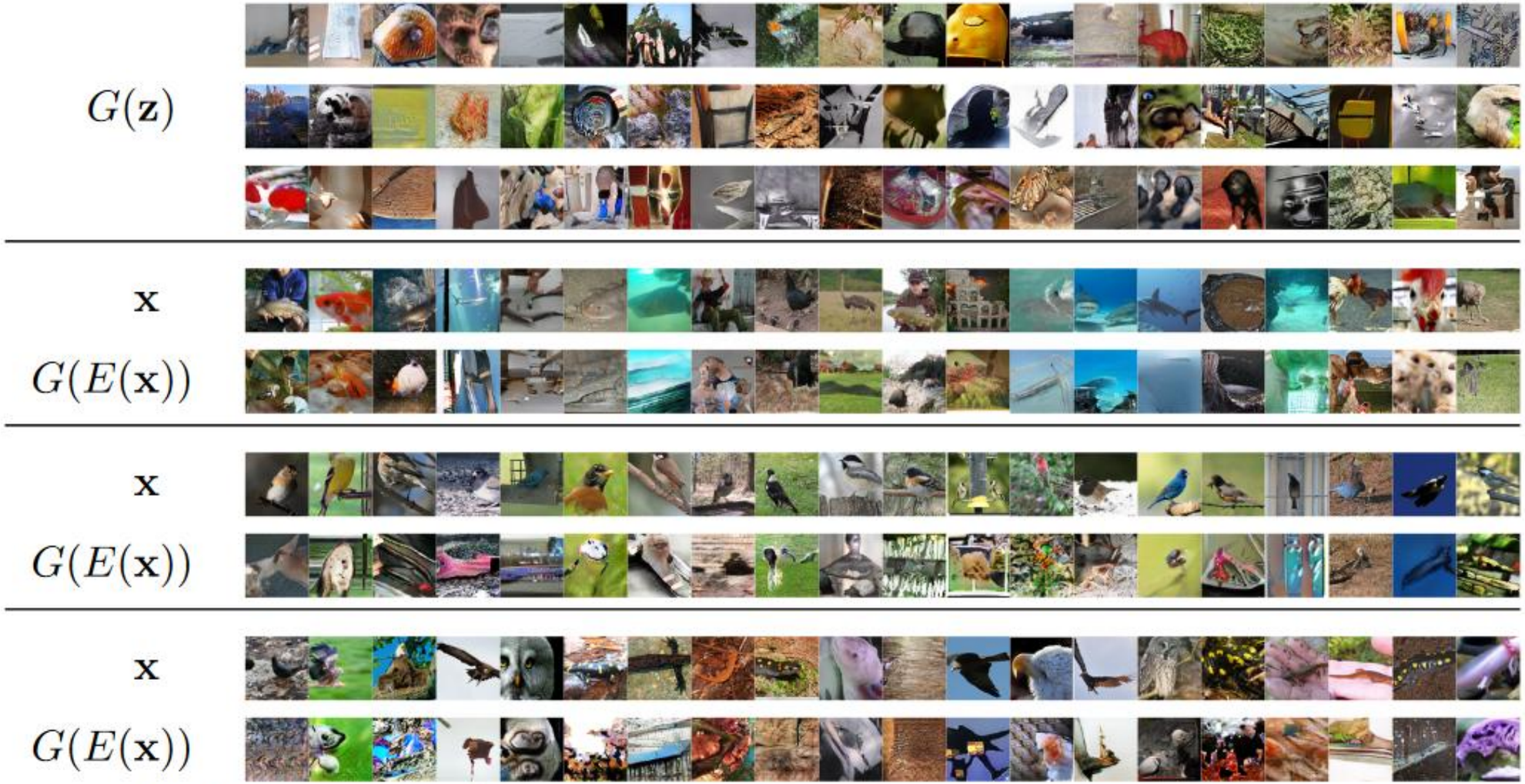


Figure 4: Qualitative results for ImageNet BiGAN training, including generator samples $G(\mathbf{z})$, real data $\mathbf{x}$, and corresponding reconstructions $G(E(\mathbf{x}))$.

BiGAN (Bidirectional)

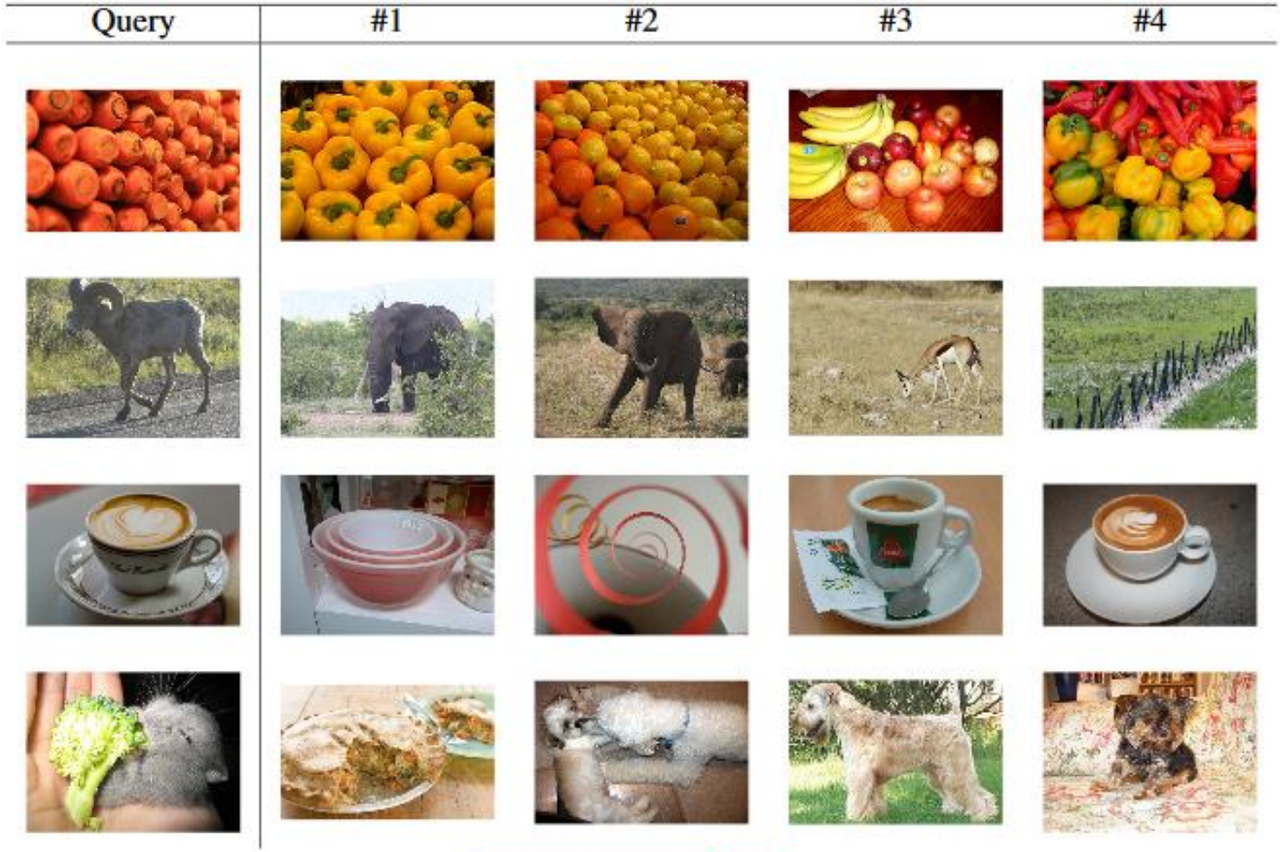


Figure 5: For the query images used in Krähenbühl et al. (2016) (left), nearest neighbors (by minimum cosine distance) from the ImageNet LSVRC (Russakovsky et al., 2015) training set in the *fc6* feature space of the ImageNet-trained BiGAN encoder E. (The *fc6* weights are set randomly; this space is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

с помощью BiGAN поиск ближайших соседей в признаковом пространстве

BiGAN – получаются хорошие признаковые пространства

	trained layers	Classification (% mAP) fc8	fc6-8	all	*FRCN* Detection (% mAP) all	*FCN* Segmentation (% mIU) all
sup.	ImageNet (Krizhevsky et al., 2012)	**77.0**	**78.8**	**78.3**	**56.8**	**48.0**
self-sup.	Agrawal et al. (2015)	31.2	31.0	54.2	43.9	-
	Pathak et al. (2016)	30.5	34.6	56.5	44.5	**30.0**
	Wang & Gupta (2015)	28.4	**55.6**	63.1	47.4	-
	Doersch et al. (2015)	**44.7**	55.1	**65.3**	**51.1**	-
unsup.	k-means (Krähenbühl et al., 2016)	32.0	39.2	56.6	45.6	32.6
	Discriminator (D)	30.7	40.5	56.4	-	-
	Latent Regressor (LR)	36.9	47.9	57.1	-	-
	Joint LR	37.1	47.9	56.5	-	-
	Autoencoder (ℓ_2)	24.8	16.0	53.8	41.9	-
	BiGAN (ours)	37.5	48.7	58.9	46.2	34.9
	BiGAN, 112×112 E (ours)	**41.7**	**52.5**	**60.3**	**46.9**	**35.2**

Table 3: Classification and Fast R-CNN (Girshick, 2015) detection results for the PASCAL VOC 2007 (Everingham et al., 2014) test set, and FCN (Long et al., 2015) segmentation results on the PASCAL VOC 2012 validation set, under the standard mean average precision (mAP) or mean intersection over union (mIU) metrics for each task. Classification models are trained with various portions of the *AlexNet* (Krizhevsky et al., 2012) model frozen. In the *fc8* column, only the linear classifier (a multinomial logistic regression) is learned – in the case of BiGAN, on top of randomly initialized fully connected (FC) layers *fc6* and *fc7*. In the *fc6-8* column, all three FC layers are trained fully supervised with all convolution layers frozen. Finally, in the *all* column, the entire network is "fine-tuned". BiGAN outperforms other unsupervised (*unsup.*) feature learning approaches, including the GAN-based baselines described in Section 4.1, and despite its generality, is competitive with contemporary self-supervised (*self-sup.*) feature learning approaches specific to the visual domain.

Adversarially learned inference (ALI) / Adversarial Generator-Encoder (AGE)

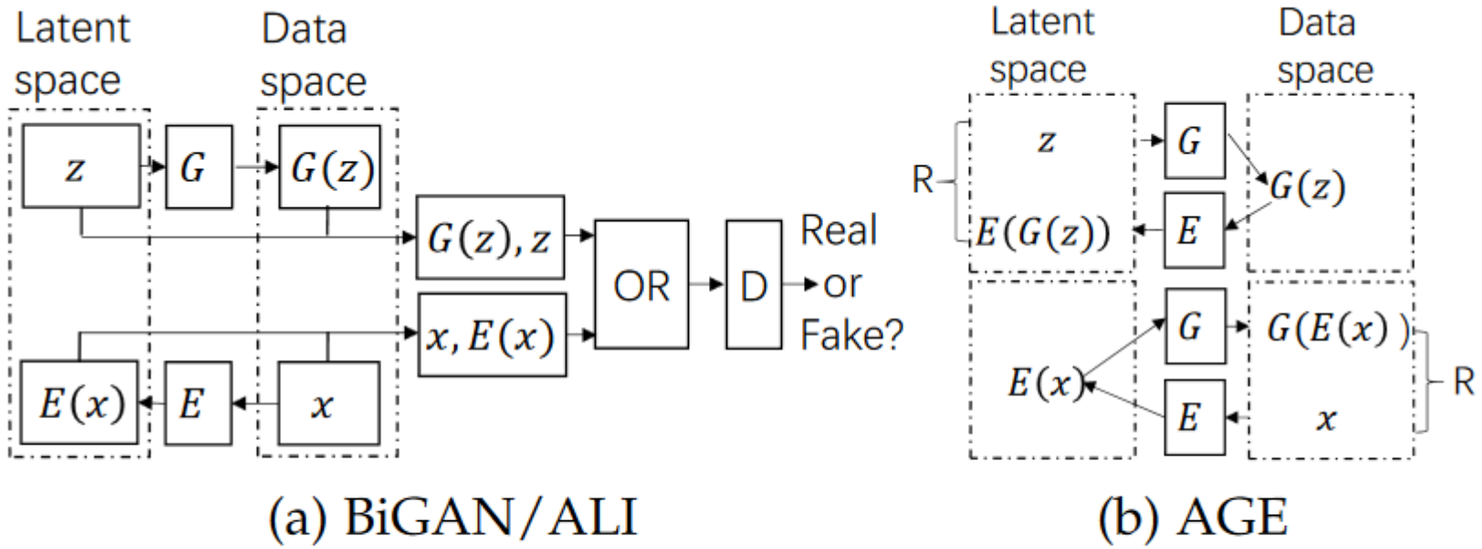


Fig. 3: The structures of (a) BiGAN and ALI and (b) AGE.

AGE ~ смесь BiGAN/ALI и pix2pix

D. Ulyanov, A. Vedaldi, and V. Lempitsky «It takes (only) two: Adversarial generator-encoder networks» in AAAI Conference on Artificial Intelligence, pp. 1250–1257, 2018.
https://arxiv.org/pdf/1704.02304.pdf

(a) Colorizations – AGE network (top rows) vs. pix2pix [11] (bottom rows) (b) Color transfer

Figure 4: (a) Each pane shows colorizations of the input grayscale image (emphasized) using conditional AGE networks (top rows) and pix2pix [11] with added noise maps (bottom rows). AGE networks produce diverse colorizations, which are hard to obtain using pix2pix. (b) In each row we show the result of color transfer using the conditional AGE network. The color scheme from the first

раскраска: Lab-кодировка и обуславливание по L (подаётся в генератор и дискриминатор)

$\Rightarrow$ z соответствует разным раскраскам

SAGAN (Self-Attention Generative Adversarial Networks)

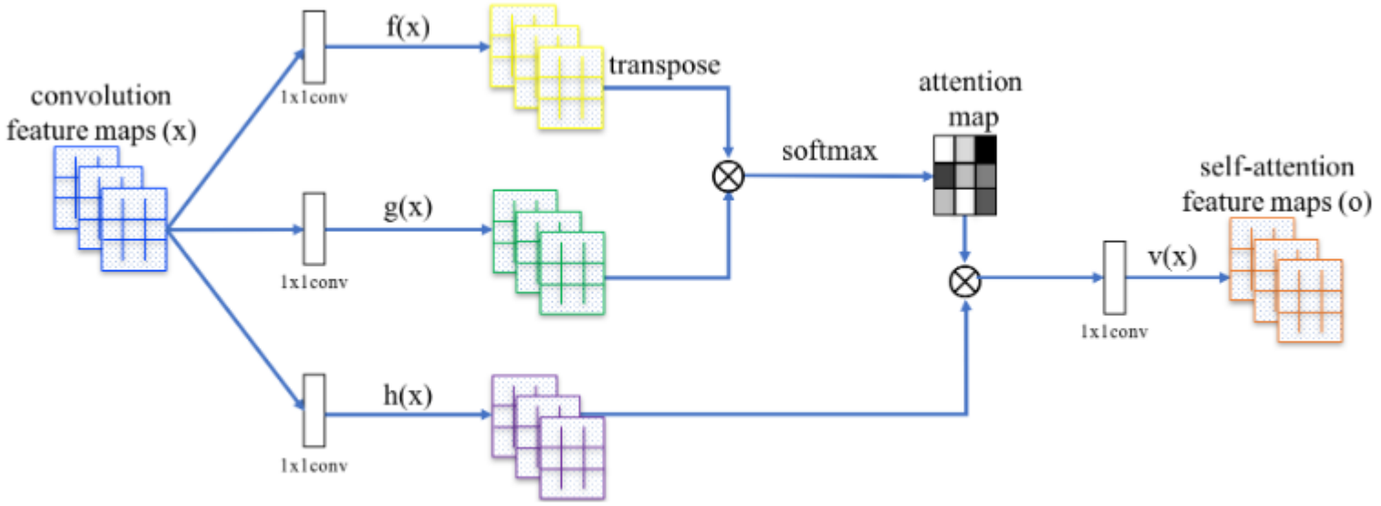


Figure 2. The proposed self-attention module for the SAGAN. The ⊗ denotes matrix multiplication. The softmax operation is performed on each row.

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

впервые безусловная хорошая генерация примеров по ImageNet

1×1-свёртки – создание Q,K,V, применение self-A, 1×1-свёртки – получение ответа

не очень понятно по рисункам, но тут «естественное» внимание по C-описаниям пикселей (точнее, размерность C сначала уменьшается 1-свёртками)

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

```
class Self_Attn(nn.Module):

    """ Self attention Layer"""
    def __init__(self,in_dim,activation):
        super(Self_Attn,self).__init__()
        self.chanel_in = in_dim
        self.activation = activation

        self.query_conv = nn.Conv2d(in_channels = in_dim , out_channels = in_dim//8 , kernel_size= 1)
        self.key_conv = nn.Conv2d(in_channels = in_dim , out_channels = in_dim//8 , kernel_size= 1)
        self.value_conv = nn.Conv2d(in_channels = in_dim , out_channels = in_dim , kernel_size= 1)
        self.gamma = nn.Parameter(torch.zeros(1))

        self.softmax  = nn.Softmax(dim=-1) #
    def forward(self,x):
        """
            inputs :
                x : input feature maps( B X C X W X H)
            returns :
                out : self attention value + input feature
                attention: B X N X N (N is Width*Height)
        """
        m_batchsize,C,width ,height = x.size()
        proj_query  = self.query_conv(x).view(m_batchsize,-1,width*height).permute(0,2,1) # B X CX(N)
        proj_key =  self.key_conv(x).view(m_batchsize,-1,width*height) # B X C x (*W*H)
        energy =  torch.bmm(proj_query,proj_key) # transpose check
        attention = self.softmax(energy) # BX (N) X (N)
        proj_value = self.value_conv(x).view(m_batchsize,-1,width*height) # B X C X N

        out = torch.bmm(proj_value,attention.permute(0,2,1) )
        out = out.view(m_batchsize,C,width,height)

        out = self.gamma*out + x
        return out,attention
```

https://github.com/heykeetae/Self-Attention-GAN/blob/master/sagan_models.py

SAGAN (Self-Attention Generative Adversarial Networks)

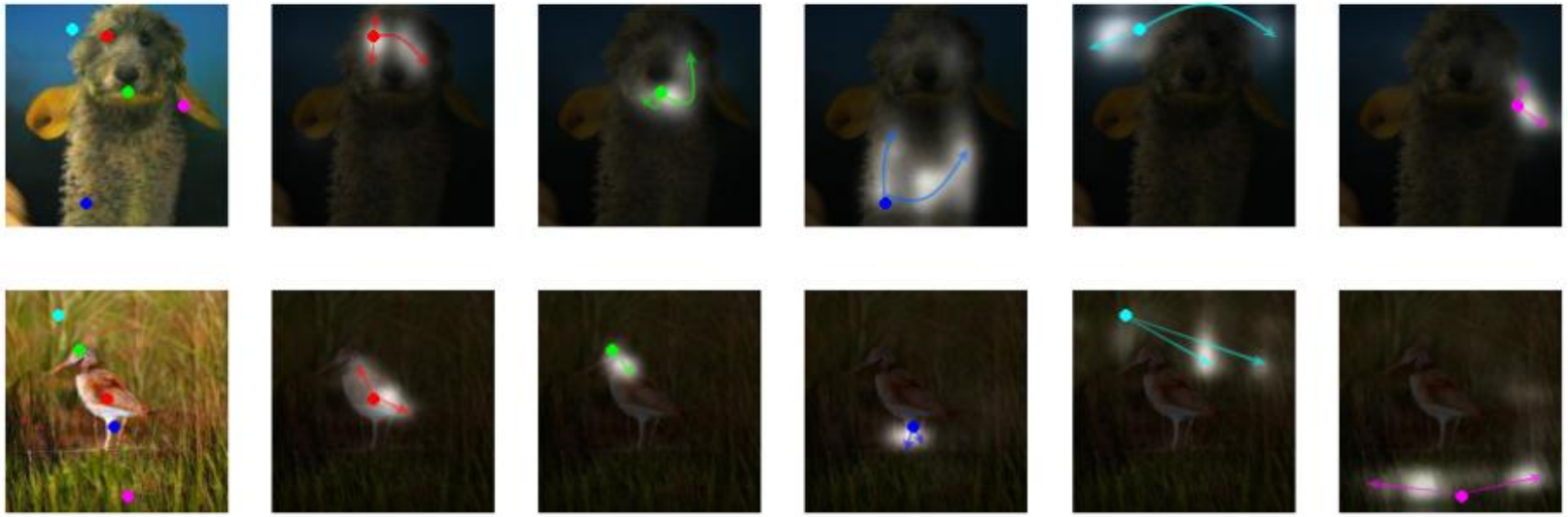


Figure 1: The proposed SAGAN generates images by leveraging complementary features in distant portions of the image rather than local regions of fixed shape to generate consistent objects/scenarios. In each row, the first image shows five representative query locations with color coded dots. The other 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)

матрица весов каждого слоя делится на спектральную норму

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

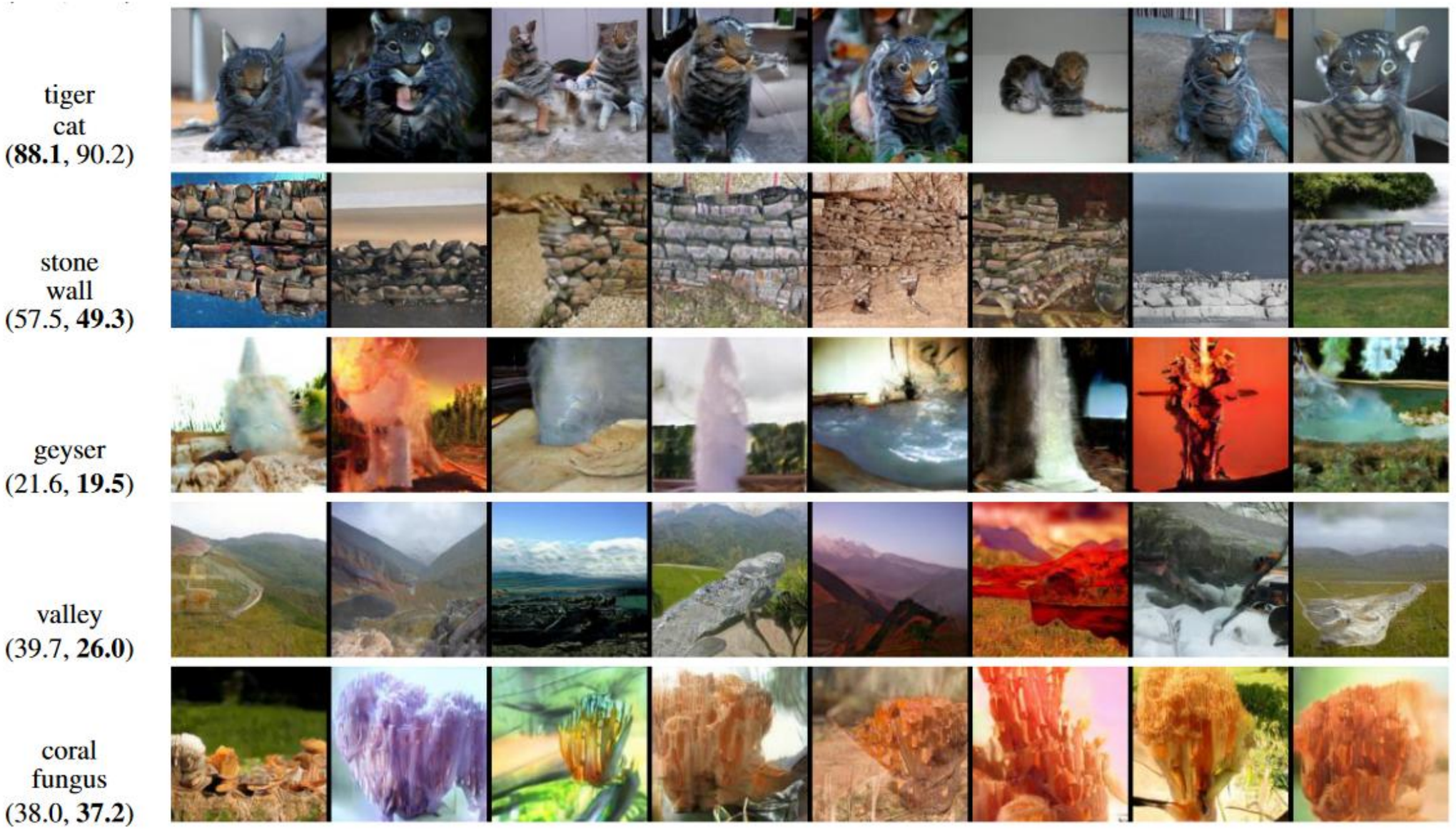


Figure 6. 128x128 example images generated by SAGAN for different classes. Each row shows examples from one class. In the leftmost column, the intra FID of our SAGAN (*left*) and the state-of-the-art method (Miyato & Koyama, 2018)) (*right*) are listed.

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) – условный по классам GAN,

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

Важные модификации

Self-attention module + Hinge loss

Class-conditional batch normalization

Update discriminator more than generator

Moving average of model weights

перед генерацией усредняются веса

orthogonal weight initialization

можно такой регуляризатор – $R_\beta(W) = \beta\|W^\top W - I\|_{\mathrm{F}}^2$

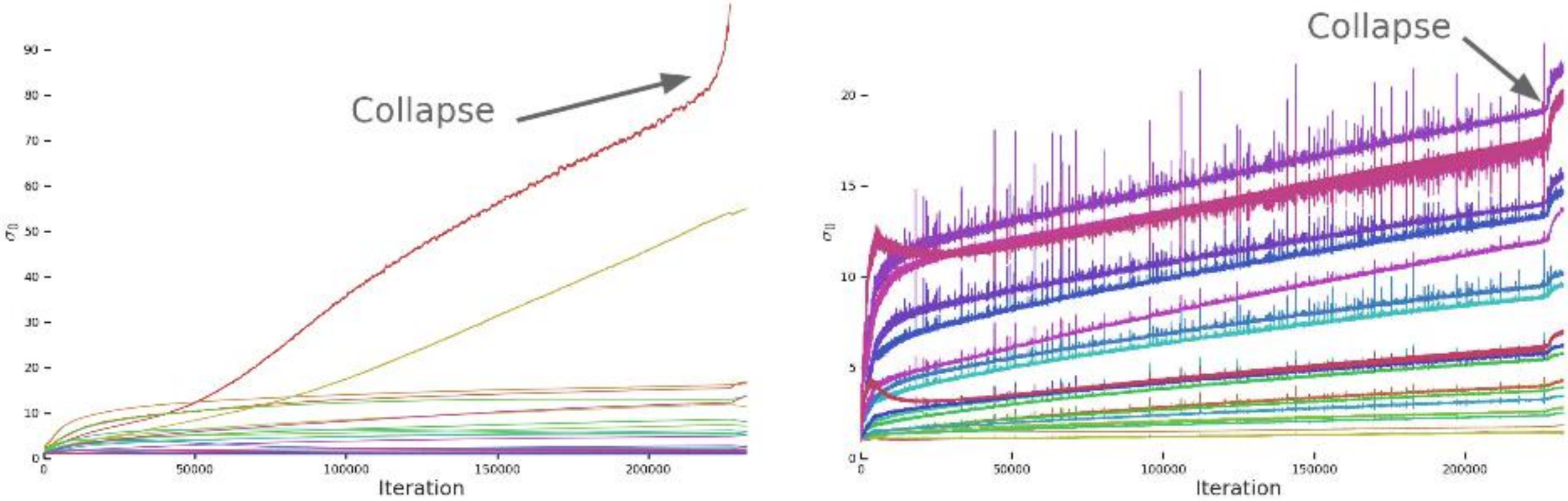
лучше такой – $R_\beta(W) = \beta\|W^\top W \odot (\mathbf{1} - I)\|_{\mathrm{F}}^2$

larger batch size

skip-z connections(z $\to$ *)

shared class embeddings (for conditional BN)

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.

BigGAN: архитектура

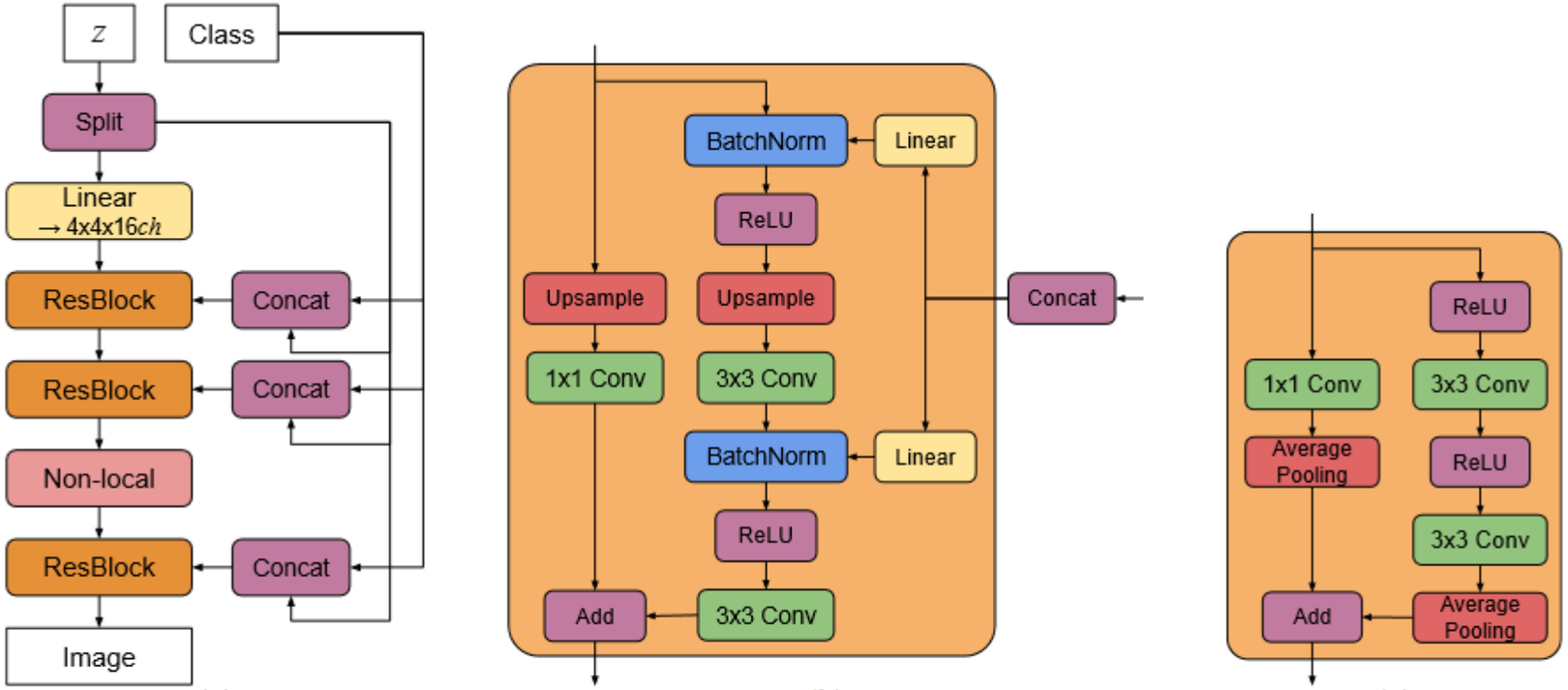


Figure 15: (a) A typical architectural layout for BigGAN's G; details are in the following tables. (b) A Residual Block (*ResBlock up*) in BigGAN's G. (c) A Residual Block (*ResBlock down*) in BigGAN's D.

есть ещё более сложная «глубокая версия»

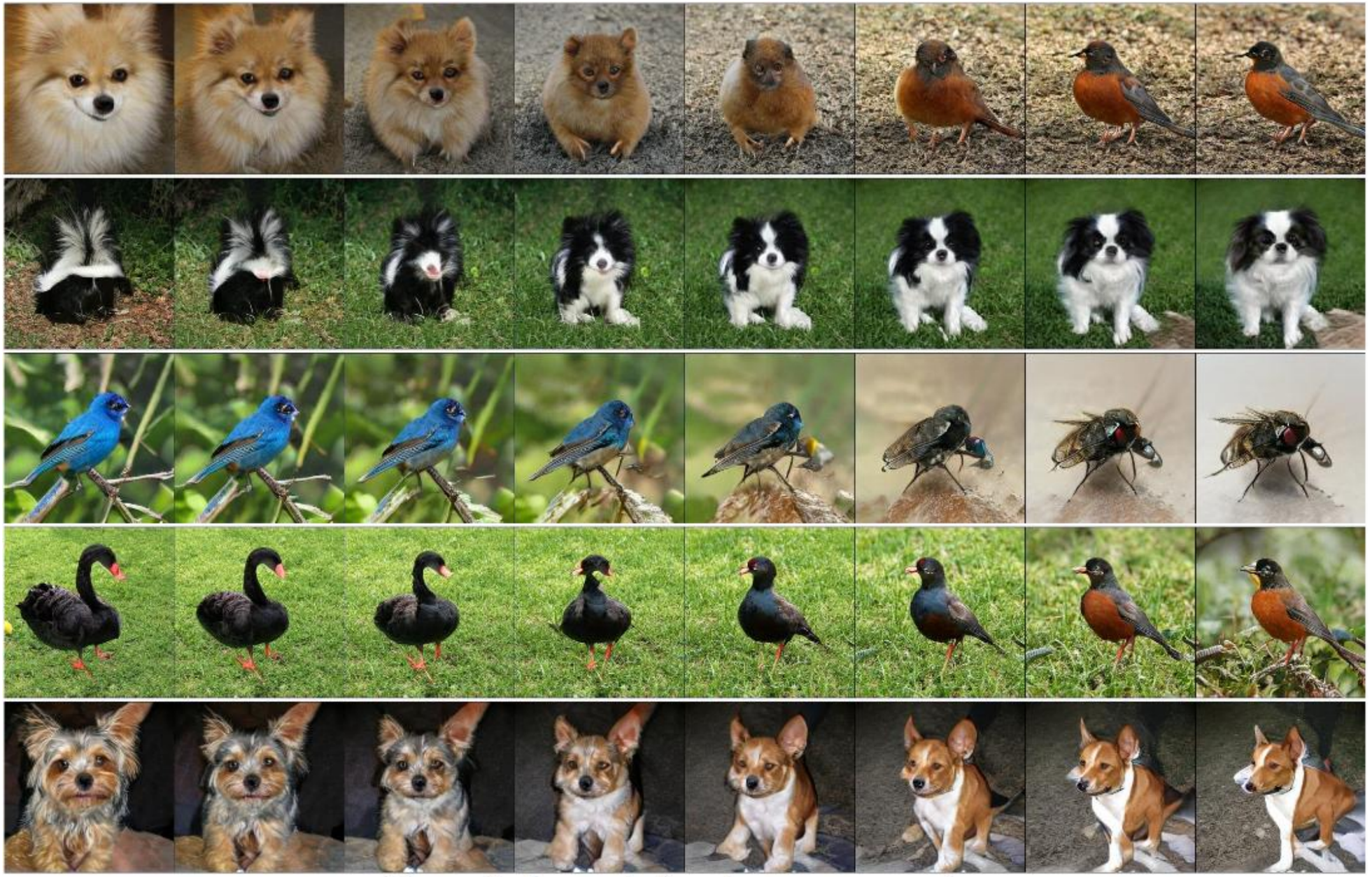


Figure 8: Interpolations between z, c pairs.

BigGAN: Генерация изображений / интерполяция

(a) 128×128 (b) 256×256 (c) 512×512 (d)

Figure 4: Samples from our BigGAN model with truncation threshold 0.5 (a-c) and an example of class leakage in a partially trained model (d).

«class leakage» = images from one class contain properties of another

BigBiGAN = BiGAN + BigGAN

Figure 2: Selected reconstructions from an unsupervised BigBiGAN model (Section 3.3). Top row images are real data $\mathbf{x} \sim P_{\mathbf{x}}$; bottom row images are generated reconstructions of the above image $\mathbf{x}$ computed by $\mathcal{G}(\mathcal{E}(\mathbf{x}))$. Unlike most explicit reconstruction costs (e.g., pixel-wise), the reconstruction cost implicitly minimized by a (Big)BiGAN [7, 10] tends to emphasize more semantic, high-level details. Additional reconstructions are presented in Appendix B.

Jeff Donahue, Karen Simonyan «Large Scale Adversarial Representation Learning» //

https://arxiv.org/pdf/1907.02544.pdf

BigBiGAN = BiGAN + BigGAN

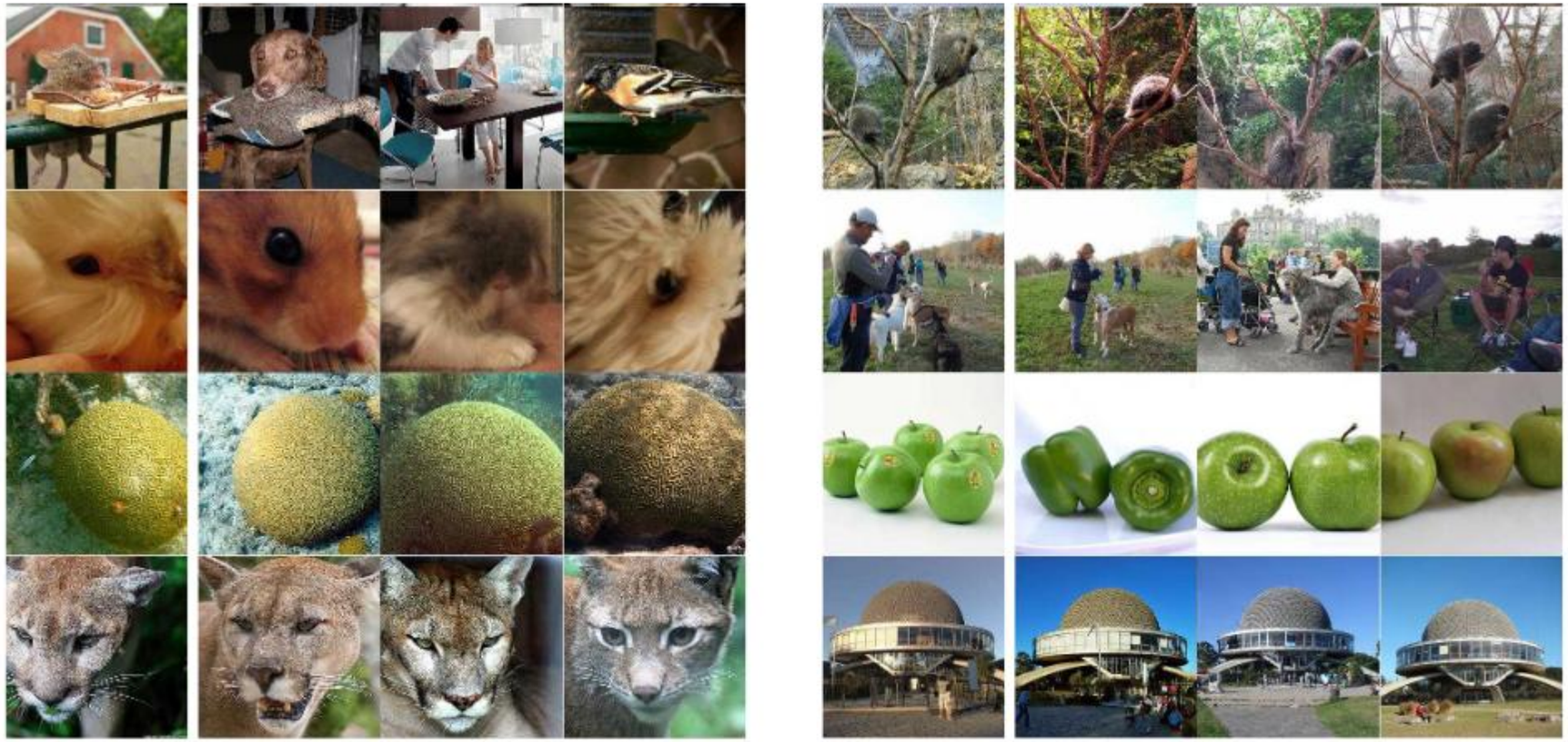


Figure 13: Nearest neighbors in BigBiGAN $\mathcal{E}$ feature space, from our best performing model (*RevNet* $\times 4$, $\uparrow \mathcal{E}$ *LR*). In each row, the first (left) column is a query image, and the remaining columns are its three nearest neighbors from the training set (the leftmost being the nearest, next being the second nearest, etc.). The query images above are the first 24 images in the ImageNet validation set.

SOTA: unsupervised representation learning on ImageNet / unconditional image generation

Semi-supervised GAN (SGAN)

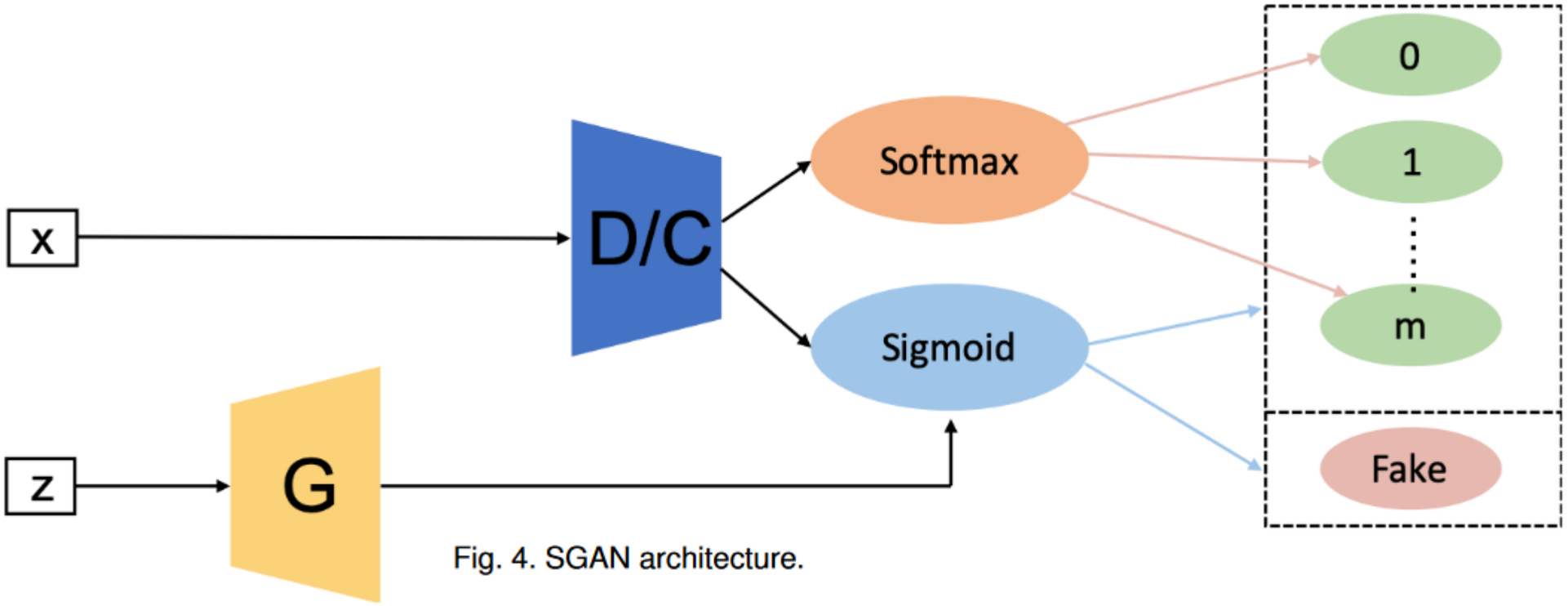


Fig. 4. SGAN architecture.

рис https://arxiv.org/pdf/1906.01529.pdf

есть ещё приём: классы = метки ∪ {Fake}

A. Odena, «Semi-supervised learning with generative adversarial networks» // https://arxiv.org/abs/1606.01583

AC-GAN: auxiliary classifier GAN

в обычном GAN фактически дискриминатор максимизирует

$$L_S = E\left[\log P\left(S = real \middle| X_{real}\right)\right] + E\left[\log\left(P\left(S = fake \middle| X_{fake}\right)\right)\right]$$

здесь пусть

$$L_C = E\left[\log P\left(C = c \middle| X_{real}\right)\right] + E\left[\log\left(P\left(C = c \middle| X_{fake}\right)\right)\right]$$

дискриминатор максимизирует

$$L_C + L_S$$

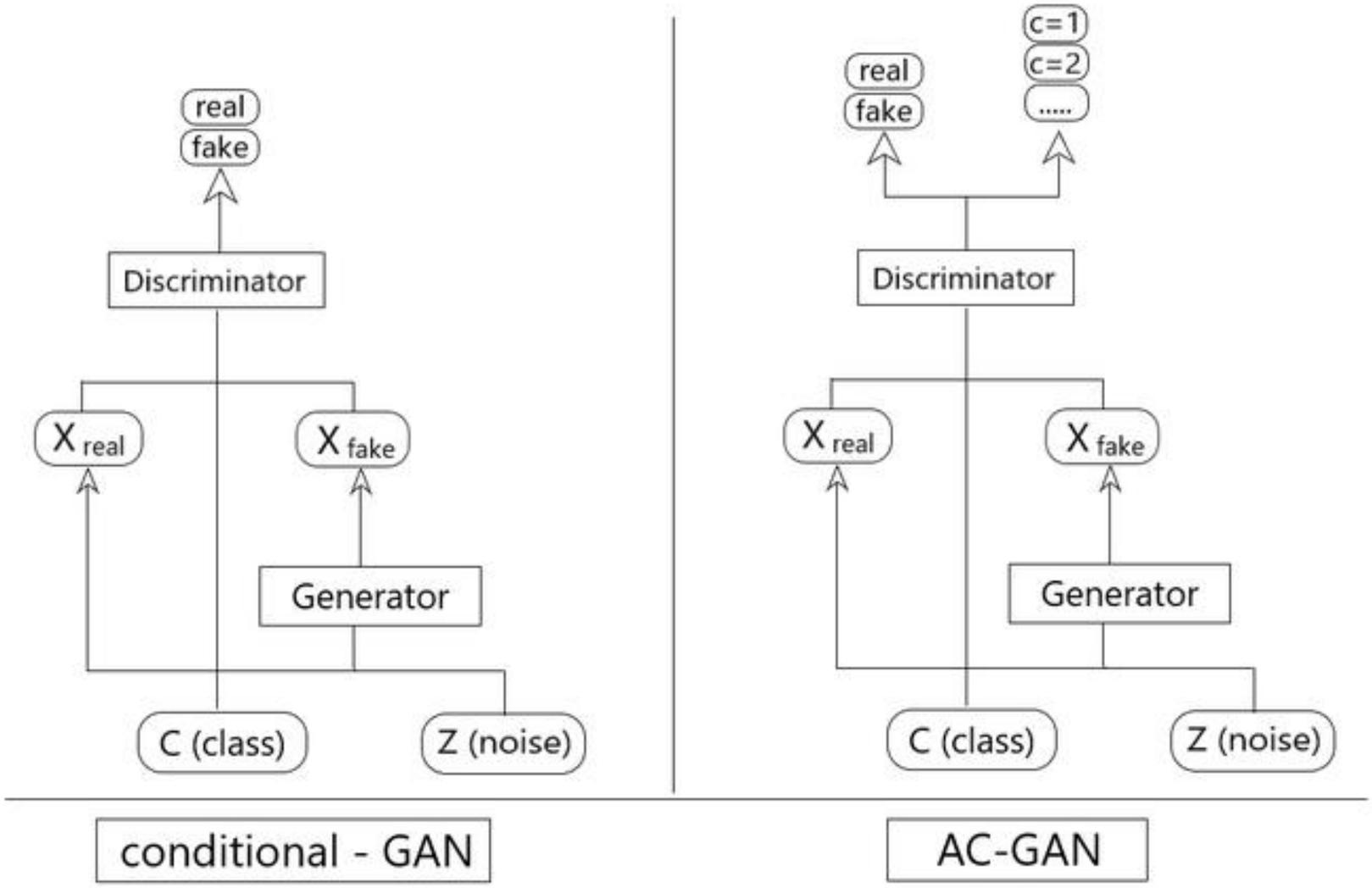
генератор максимизирует

$$L_C - L_S$$

дискриминатор не просто отличает подделку, но и определяет класс

Augustus Odena «Conditional Image Synthesis With Auxiliary Classifier GANs» // https://arxiv.org/abs/1610.09585

AC-GAN: auxiliary classifier GAN



в отличие от SGAN метка класса подаётся на вход генератору

https://www.geeksforgeeks.org/understanding-auxiliary-classifier-gan/

AC-GAN: auxiliary classifier GAN

Figure 1. 128 × 128 resolution samples from 5 classes taken from an AC-GAN trained on the ImageNet dataset. Note that the classes shown have been selected to highlight the success of the model and are not representative. Samples from all ImageNet classes are linked later in the text.

первый GAN, воспроизводящий похожее на ImageNet

CAN: Creative Adversarial Networks

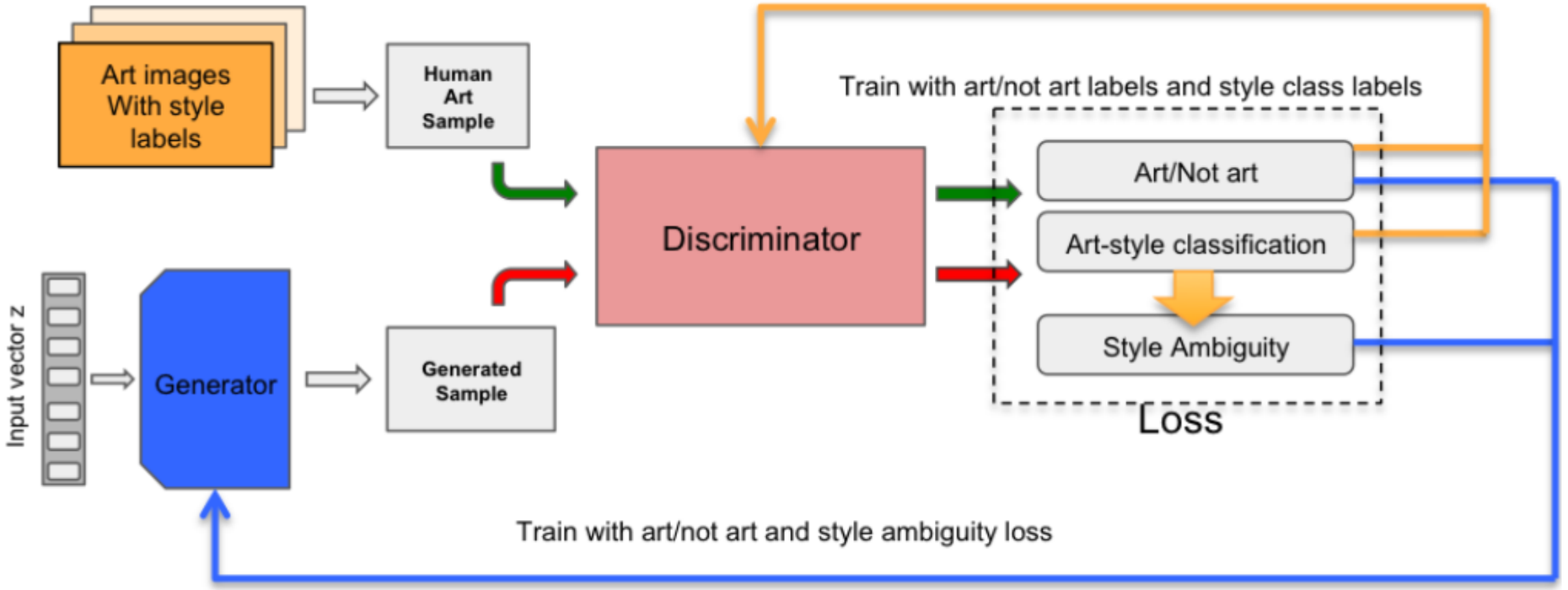


Figure 2: Block diagram of the CAN system.

+ 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

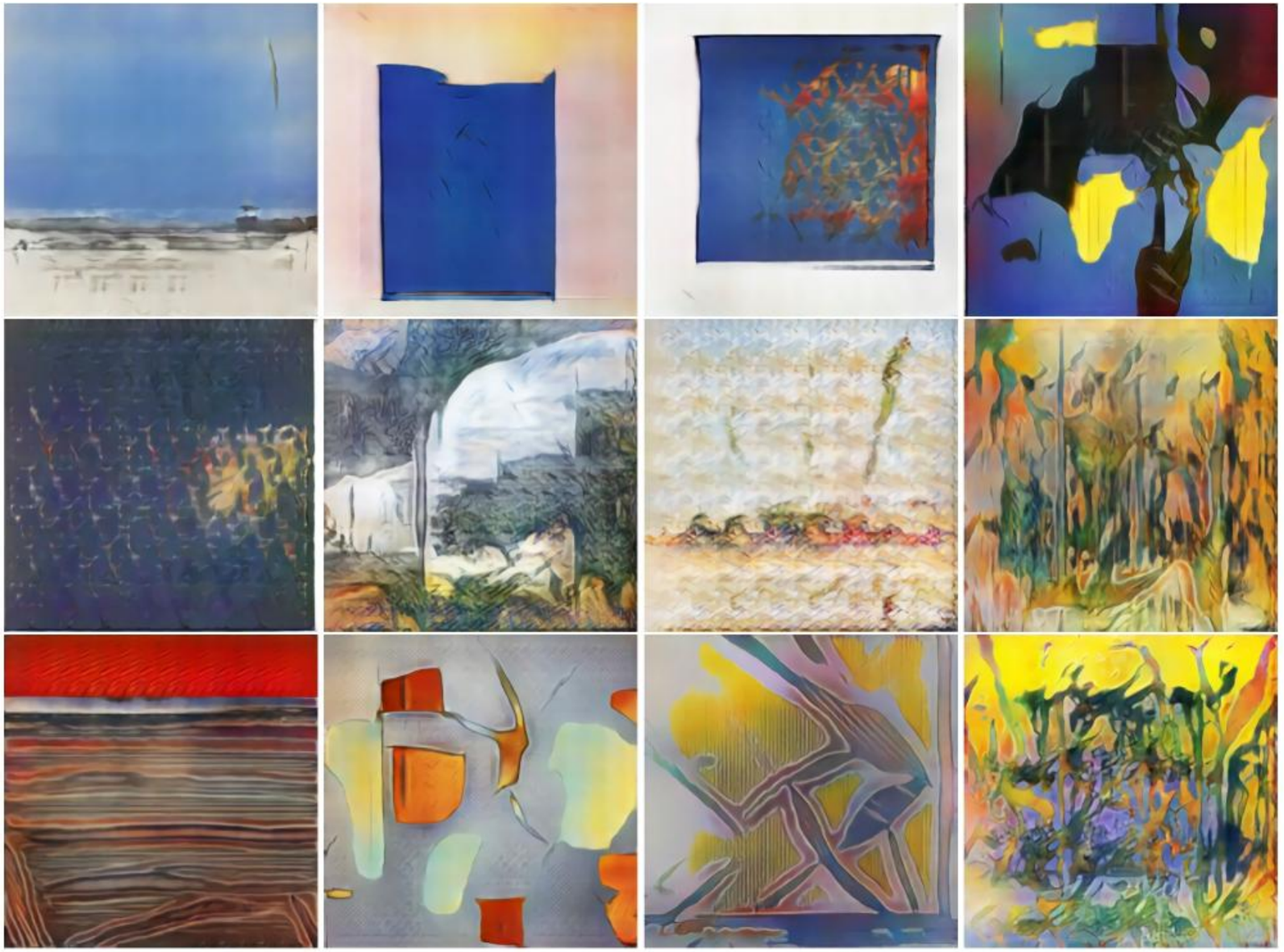


Figure 1: Example of images generated by CAN. The generated images vary from simple abstract ones to complex textures and compositions. More examples are shown in Figures 4.2 and 7

Triple GAN

three-player game

$$\min_{G,C}\max_{D} V(D,C,G) := \mathbb{E}_{(\boldsymbol{x},y)\sim p_{\text{data}}(\boldsymbol{x},y)}\big[\log(D(\boldsymbol{x},y))\big]$$
$$+\alpha\mathbb{E}_{(\boldsymbol{x},y)\sim p_c(\boldsymbol{x},y)}\big[\log(1-D(\boldsymbol{x},y))\big]$$
$$+(1-\alpha)\mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z}),y\sim p(y)}\Big[\log\Big(1-D\big(G(\boldsymbol{z},y),y\big)\Big)\Big]$$
$$+\mathbb{E}_{(\boldsymbol{x},y)\sim p_{\text{data}}(\boldsymbol{x},y)}\big[-\log(p_c(y|\boldsymbol{x}))\big]$$

тоже есть классификатор $p_c(y|x)$

$p_c(x, y) = p_c(x|y)p(y)$

последнее слагаемое ~ $KL(p_c(x, y), p_{data}(x, y))$

ProGAN (NVIDIA)

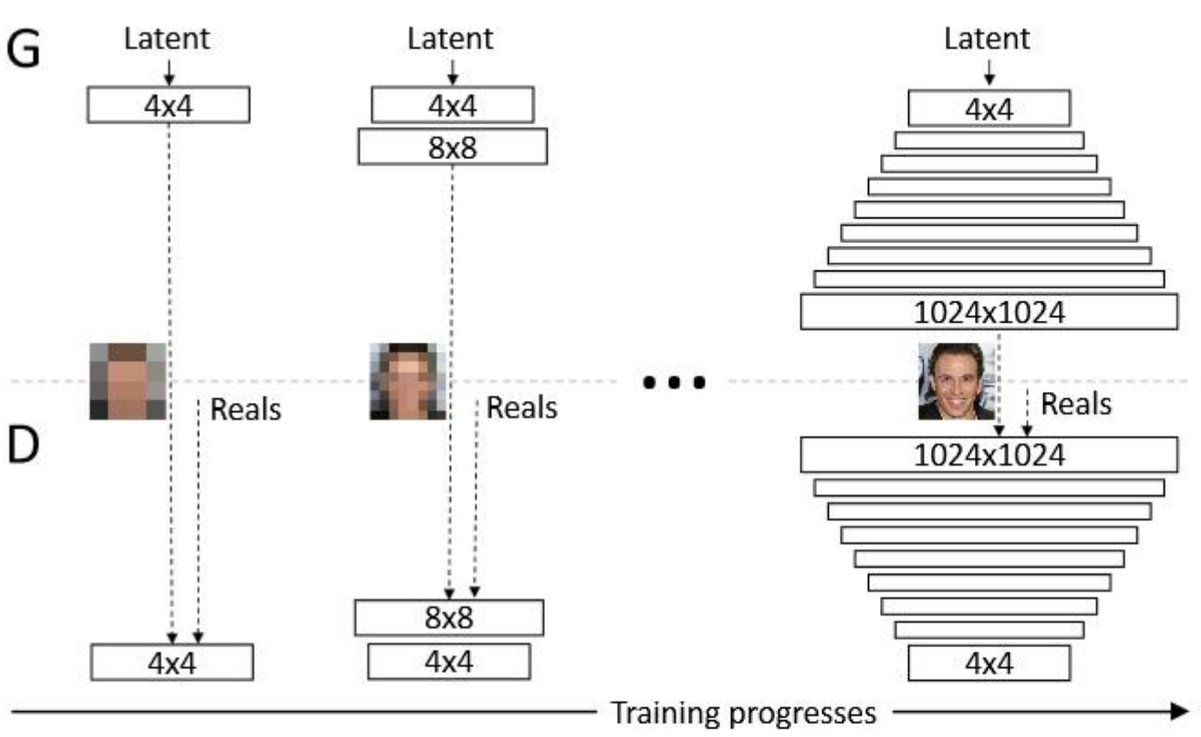


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout 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)

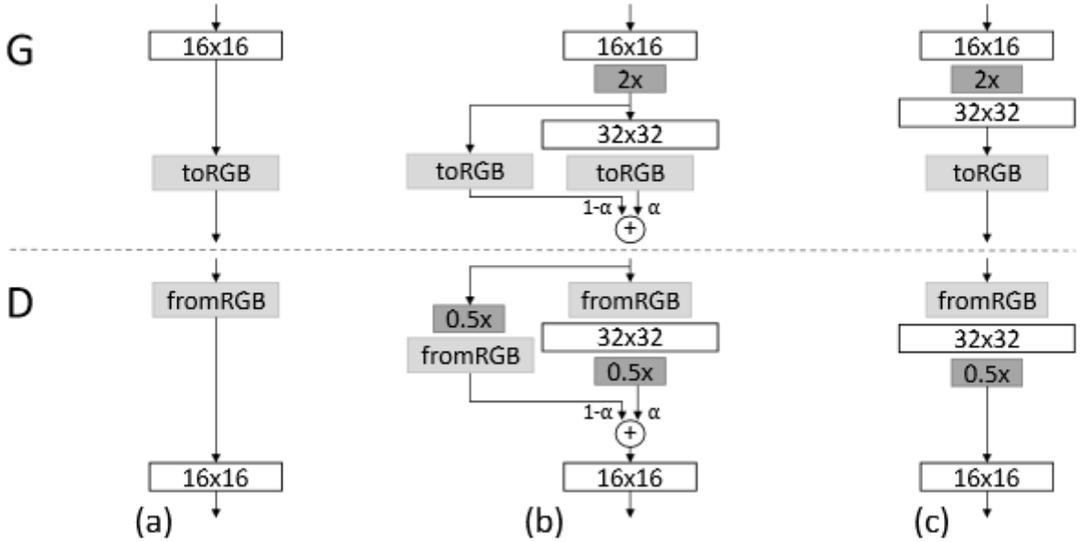


Figure 2: When doubling the resolution of the generator (G) and discriminator (D) we fade in the new layers smoothly. This example illustrates the transition from 16×16 images (a) to 32×32 images (c). During the transition (b) we treat the layers that operate on the higher resolution like a residual 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.

прокидывание связи с меняющимся весом → лучше доучиваться до большего размера

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

ProGAN (NVIDIA)

Generator	Act.	Output shape	Params
Latent vector	–	512 × 1 × 1	–
Conv 4 × 4	LReLU	512 × 4 × 4	4.2M
Conv 3 × 3	LReLU	512 × 4 × 4	2.4M
Upsample	–	512 × 8 × 8	–
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Upsample	–	512 × 16 × 16	–
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Upsample	–	512 × 32 × 32	–
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Upsample	–	512 × 64 × 64	–
Conv 3 × 3	LReLU	256 × 64 × 64	1.2M
Conv 3 × 3	LReLU	256 × 64 × 64	590k
Upsample	–	256 × 128 × 128	–
Conv 3 × 3	LReLU	128 × 128 × 128	295k
Conv 3 × 3	LReLU	128 × 128 × 128	148k
Upsample	–	128 × 256 × 256	–
Conv 3 × 3	LReLU	64 × 256 × 256	74k
Conv 3 × 3	LReLU	64 × 256 × 256	37k
Upsample	–	64 × 512 × 512	–
Conv 3 × 3	LReLU	32 × 512 × 512	18k
Conv 3 × 3	LReLU	32 × 512 × 512	9.2k
Upsample	–	32 × 1024 × 1024	–
Conv 3 × 3	LReLU	16 × 1024 × 1024	4.6k
Conv 3 × 3	LReLU	16 × 1024 × 1024	2.3k
Conv 1 × 1	linear	3 × 1024 × 1024	51
Total trainable parameters			**23.1M**

Discriminator	Act.	Output shape	Params
Input image	–	3 × 1024 × 1024	–
Conv 1 × 1	LReLU	16 × 1024 × 1024	64
Conv 3 × 3	LReLU	16 × 1024 × 1024	2.3k
Conv 3 × 3	LReLU	32 × 1024 × 1024	4.6k
Downsample	–	32 × 512 × 512	–
Conv 3 × 3	LReLU	32 × 512 × 512	9.2k
Conv 3 × 3	LReLU	64 × 512 × 512	18k
Downsample	–	64 × 256 × 256	–
Conv 3 × 3	LReLU	64 × 256 × 256	37k
Conv 3 × 3	LReLU	128 × 256 × 256	74k
Downsample	–	128 × 128 × 128	–
Conv 3 × 3	LReLU	128 × 128 × 128	148k
Conv 3 × 3	LReLU	256 × 128 × 128	295k
Downsample	–	256 × 64 × 64	–
Conv 3 × 3	LReLU	256 × 64 × 64	590k
Conv 3 × 3	LReLU	512 × 64 × 64	1.2M
Downsample	–	512 × 32 × 32	–
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Downsample	–	512 × 16 × 16	–
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Downsample	–	512 × 8 × 8	–
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Downsample	–	512 × 4 × 4	–
Minibatch stddev	–	513 × 4 × 4	–
Conv 3 × 3	LReLU	512 × 4 × 4	2.4M
Conv 4 × 4	LReLU	512 × 1 × 1	4.2M
Fully-connected	linear	1 × 1 × 1	513
Total trainable parameters			**23.1M**

Table 2: Generator and discriminator that we use with CELEBA-HQ to generate 1024×1024 images.

ProGAN: хаки, которые применялись

Training configuration	CELEBA						LSUN BEDROOM					
	Sliced Wasserstein distance $\times 10^3$					MS-SSIM	Sliced Wasserstein distance $\times 10^3$					MS-SSIM
	128	64	32	16	Avg		128	64	32	16	Avg	
(a) Gulrajani et al. (2017)	12.99	7.79	7.62	8.73	9.28	0.2854	11.97	10.51	8.03	14.48	11.25	**0.0587**
(b) + Progressive growing	4.62	**2.64**	3.78	6.06	4.28	**0.2838**	7.09	6.27	7.40	9.64	7.60	0.0615
(c) + Small minibatch	75.42	41.33	41.62	26.57	46.23	0.4065	72.73	40.16	42.75	42.46	49.52	0.1061
(d) + Revised training parameters	9.20	6.53	4.71	11.84	8.07	0.3027	7.39	5.51	3.65	9.63	6.54	0.0662
(e*) + Minibatch discrimination	10.76	6.28	6.04	16.29	9.84	0.3057	10.29	6.22	5.32	11.88	8.43	0.0648
(e) Minibatch stddev	13.94	5.67	2.82	5.71	7.04	0.2950	7.77	5.23	3.27	9.64	6.48	0.0671
(f) + Equalized learning rate	4.42	3.28	2.32	7.52	4.39	0.2902	**3.61**	3.32	**2.71**	6.44	4.02	0.0668
(g) + Pixelwise normalization	**4.06**	3.04	**2.02**	**5.13**	**3.56**	0.2845	3.89	**3.05**	3.24	**5.87**	**4.01**	0.0640
(h) Converged	2.42	2.17	2.24	4.99	2.96	0.2828	3.47	2.60	2.30	4.87	3.31	0.0636

Table 1: Sliced Wasserstein distance (SWD) between the generated and training images (Section 5) and multi-scale structural similarity (MS-SSIM) among the generated images for several training setups at 128×128. For SWD, each column represents one level of the Laplacian pyramid, and the last one gives an average of the four distances.

(a) (b) (c) (d) (e*) (e) (f) (g) (h) Converged

Figure 3: (a) – (g) CELEBA examples corresponding to rows in Table 1. These are intentionally non-converged. (h) Our converged result. Notice that some images show aliasing and some are not sharp – this is a flaw of the dataset, which the model learns to replicate faithfully.

Оценивать разнообразие в батчах

Нормализация весов

тут хитрая техника...

Нормализация по пикселям в генераторе

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.

Идея Progressive появилась в стекинге GAN-ов

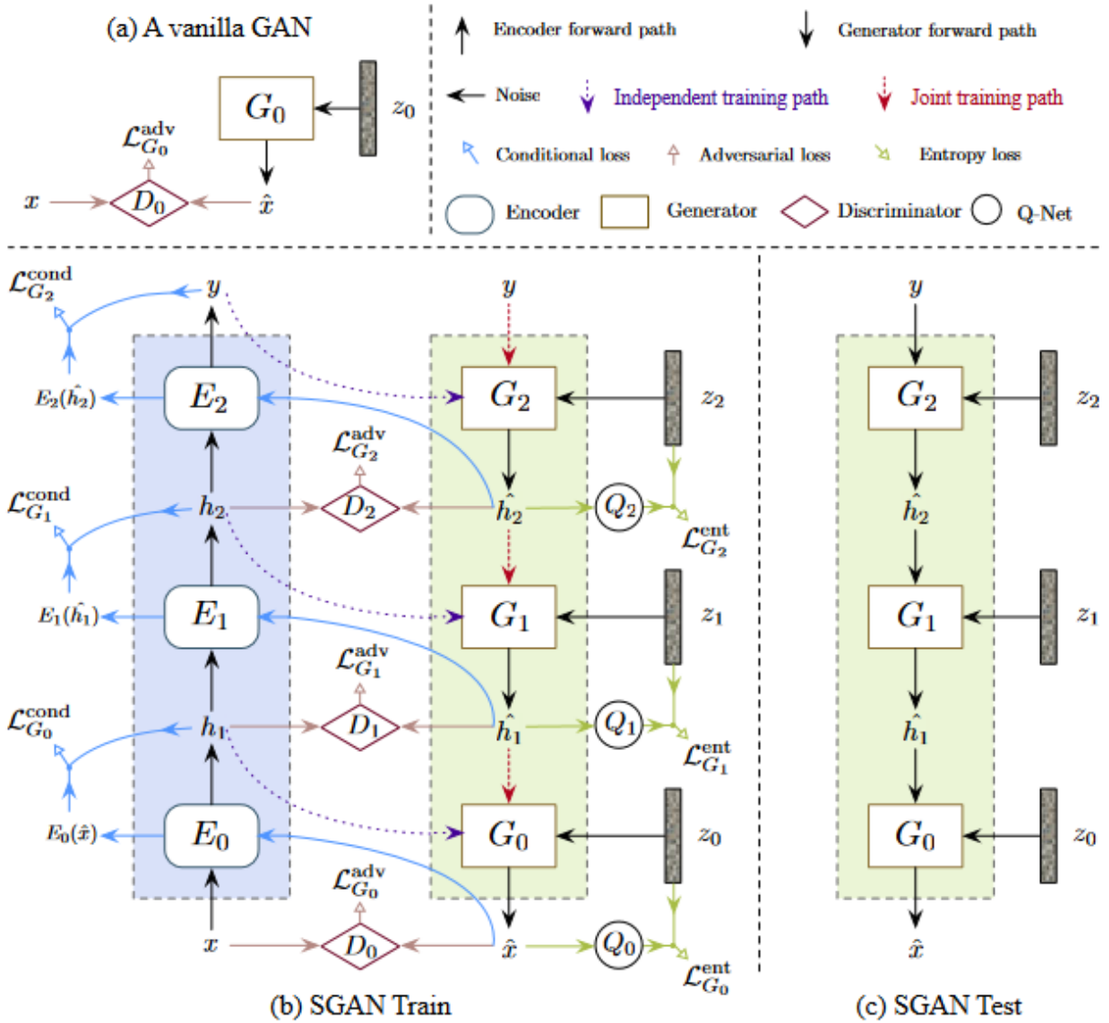


Figure 1: **An overview of SGAN.** (a) The original GAN in [17]. (b) The workflow of training SGAN, where each generator G_i tries to generate plausible features that can fool the corresponding representation discriminator D_i. Each generator receives conditional input from encoders in the independent training stage, and from the upper generators in the joint training stage. (c) New images can be sampled from SGAN (during test time) by feeding random noise to each generator G_i.

«Stacked Generative Adversarial Networks» // https://arxiv.org/pdf/1612.04357.pdf

InfoGAN

Пусть

Z – шум

C – важная информация (класс, угол, толщина) – неизвестная информация, определяется в результате обучения

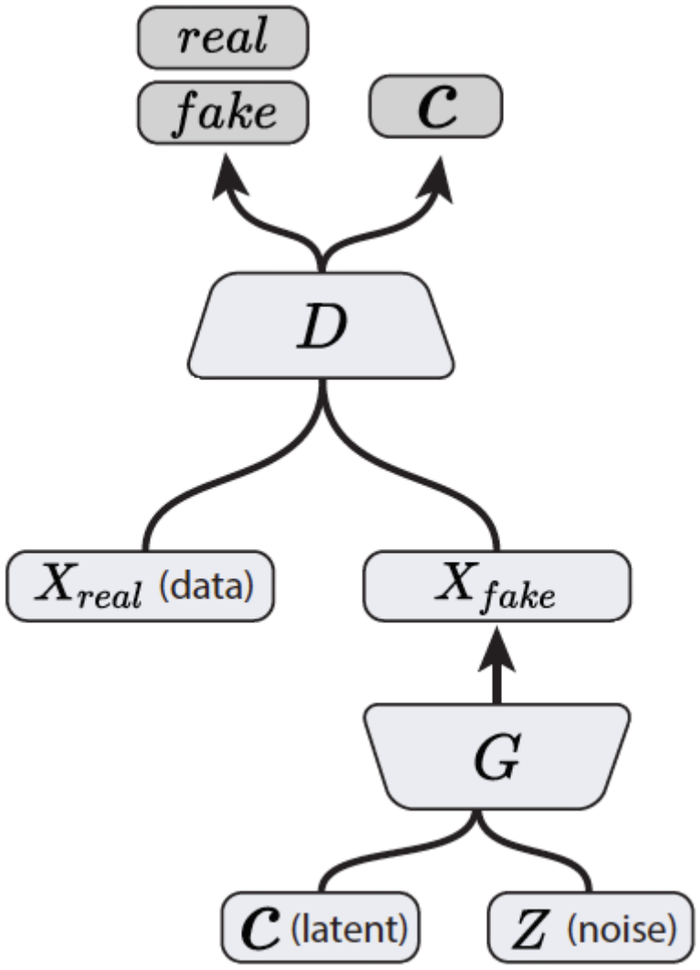
Хотим параллельно максимизировать взаимною информацию между c и x:

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

V – ошибка обычного GAN

чтобы больше «смотрели» на c

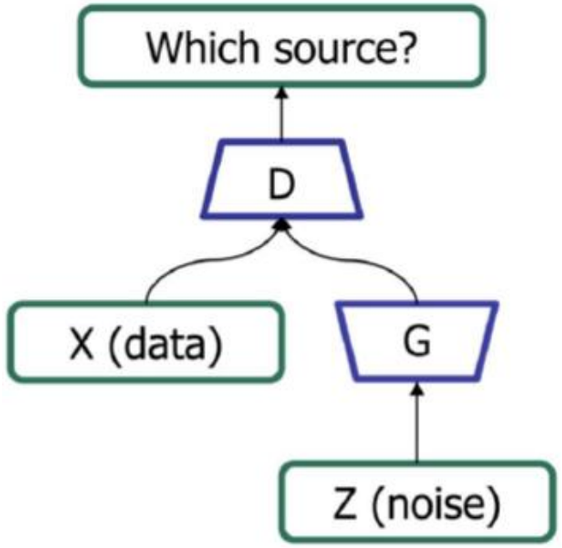
на практике немного другая формула



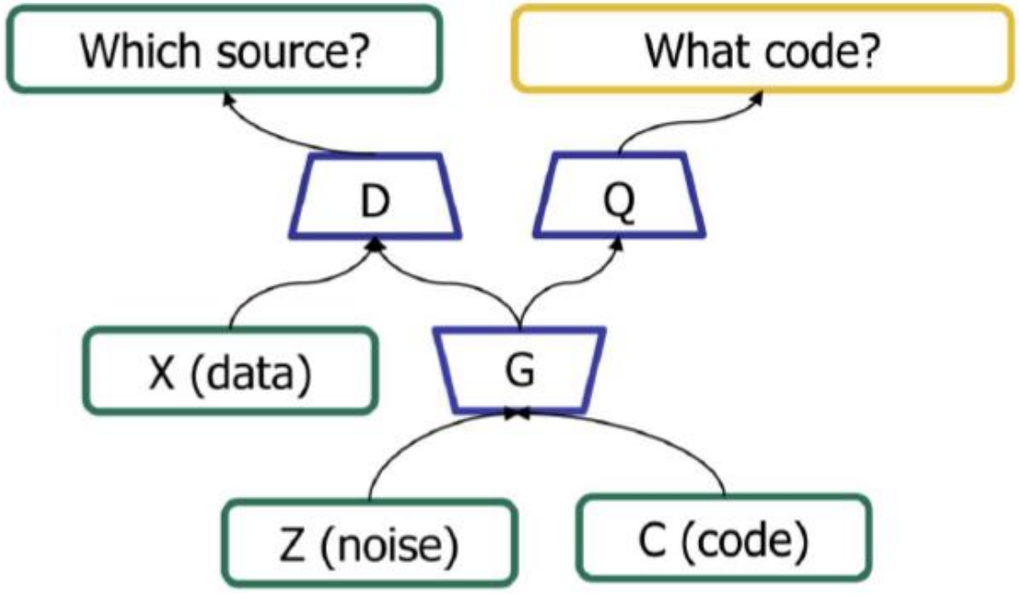
Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. «InfoGAN: Interpretable Representation Learning by Information Maximization Generative Adversarial Nets» // https://papers.nips.cc/paper/6399-infogan-interpretable-representation-learning-by-information-maximizing-generative-adversarial-nets.pdf

InfoGAN

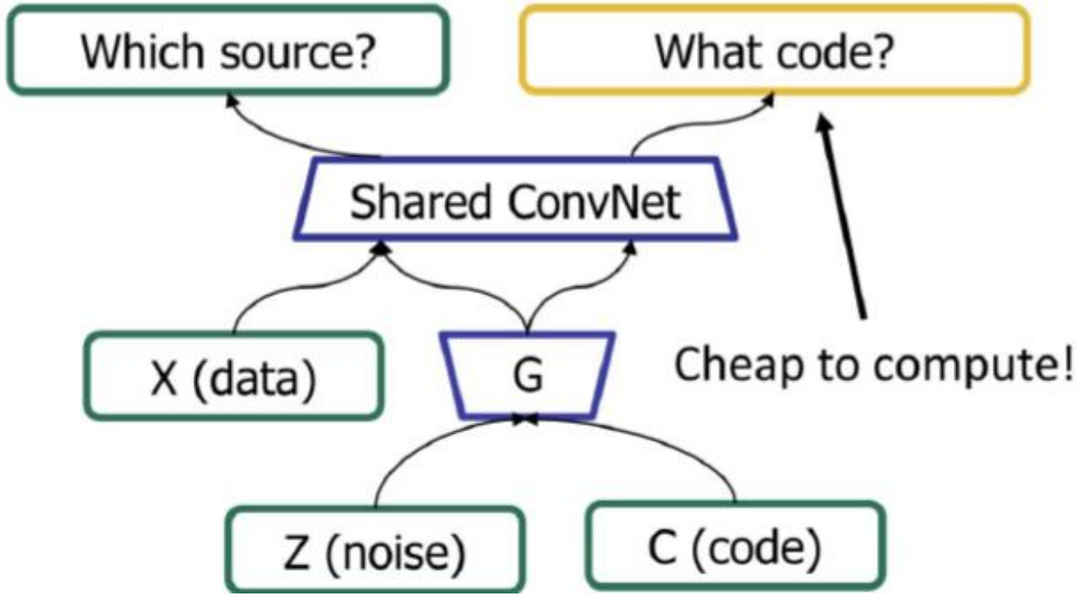
Раньше мы подавали шум на вход, сейчас хотим различить шум и что-то в нём, что отвечает за признаки изображения (поза, длина волос и т.п.)

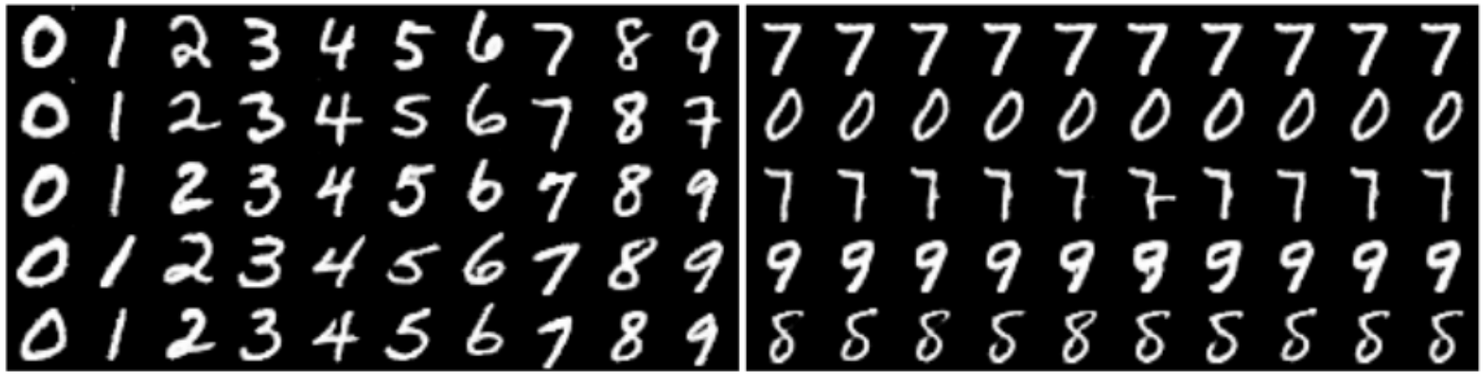


GAN



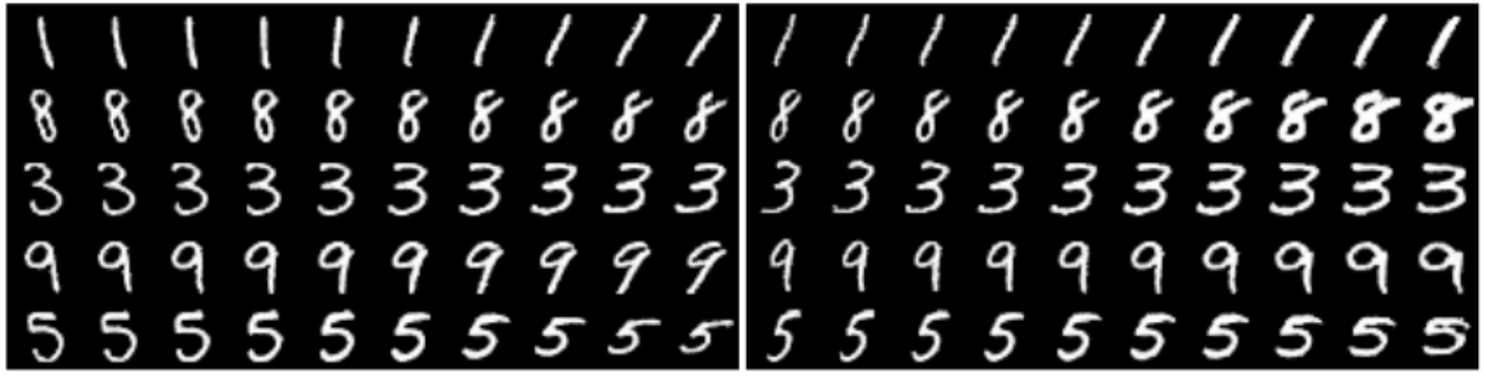
InfoGAN





(a) Varying c_1 on InfoGAN (Digit type)

(b) Varying c_1 on regular GAN (No clear meaning)

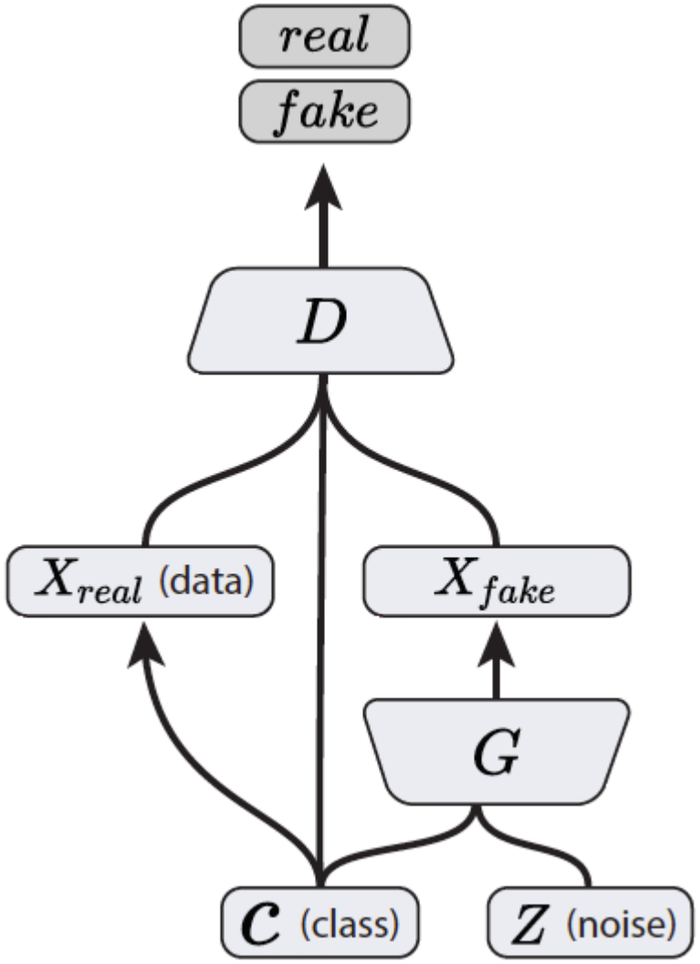


(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Figure 2: **Manipulating latent codes on MNIST:** *In all figures of latent code manipulation, we will use the convention that in each one latent code varies from left to right while the other latent codes and noise are fixed. The different rows correspond to different random samples of fixed latent codes and noise. For instance, in (a), one column contains five samples from the same category in c_1, and a row shows the generated images for* 10 *possible categories in c_1 with other noise fixed.* In (a), each category in c_1 largely corresponds to one digit type; in (b), varying c_1 on a GAN trained without information regularization results in non-interpretable variations; in (c), a small value of c_2 denotes left leaning digit whereas a high value corresponds to right leaning digit; in (d), c_3 smoothly controls the width. We reorder (a) for visualization purpose, as the categorical code is inherently unordered.

Условные GANы (Conditional GANs)



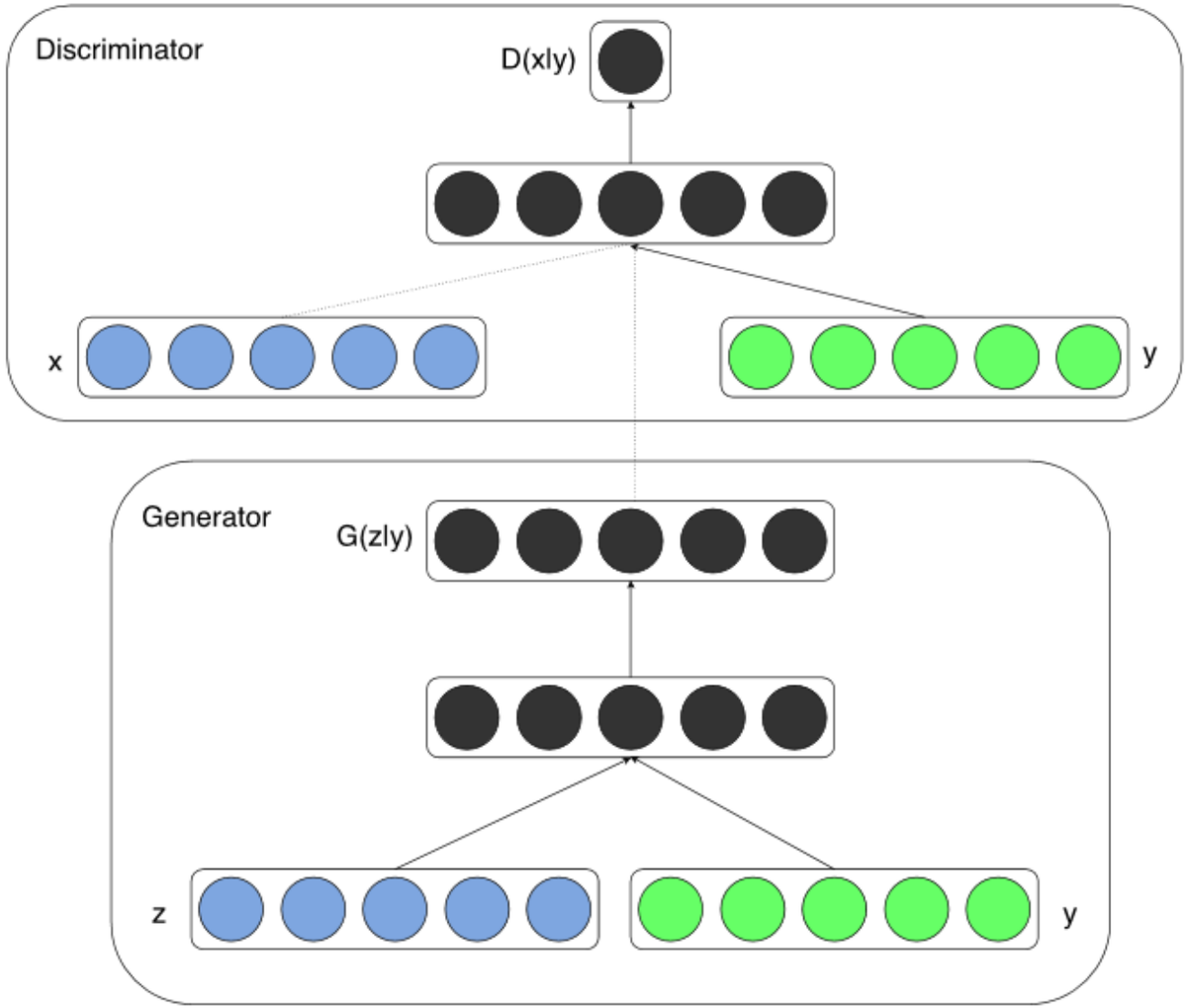


Figure 1: Conditional adversarial net

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

Условные GANы для изменения возраста: Age-cGAN

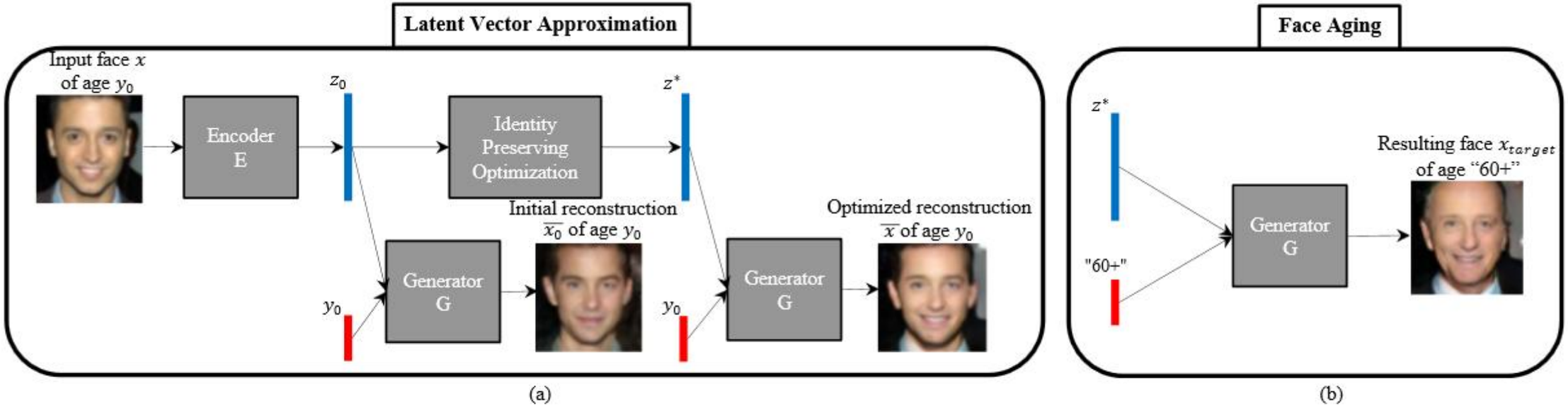


Fig. 1. Our face aging method. (a) approximation of the latent vector to reconstruct the input image; (b) switching the age condition at the input of the generator G to perform face aging.

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

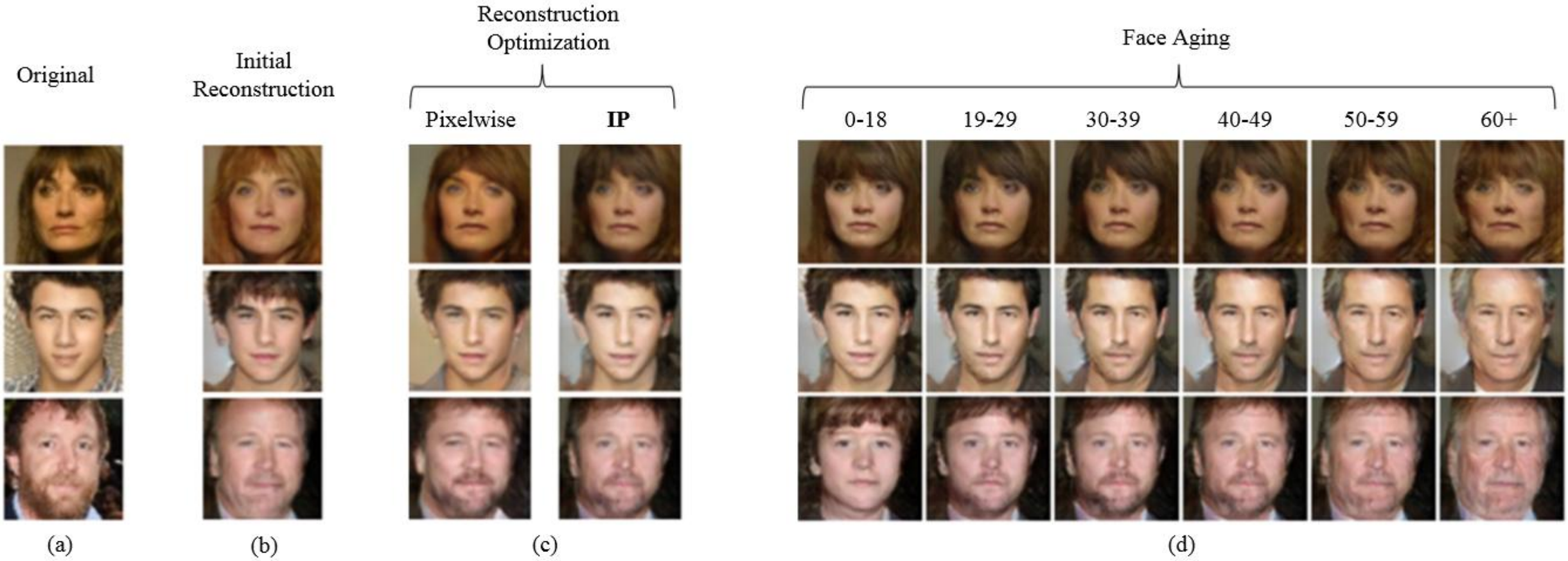


Fig. 3. Examples of face reconstruction and aging. (a) original test images, (b) reconstructed images generated using the initial latent approximations: z_0, (c) reconstructed images generated using the "Pixelwise" and "Identity-Preserving" optimized latent approximations: z^*_{pixel} and z^*_{IP}, and (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 GAN (CoGAN)

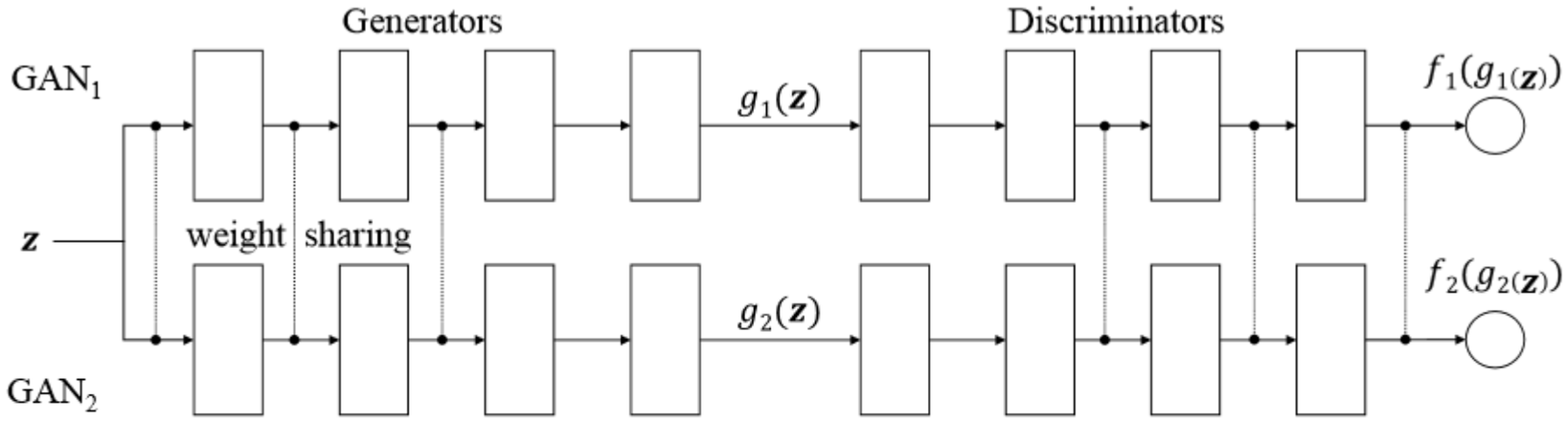


Figure 1: CoGAN consists of a pair of GANs: GAN_1 and GAN_2. Each has a generative model for synthesizing realistic images in one domain and a discriminative model for classifying whether an image is real or synthesized. We tie the weights of the first few layers (responsible for decoding high-level 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.

для синтеза объектов в двух доменах

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

Coupled GAN (CoGAN)

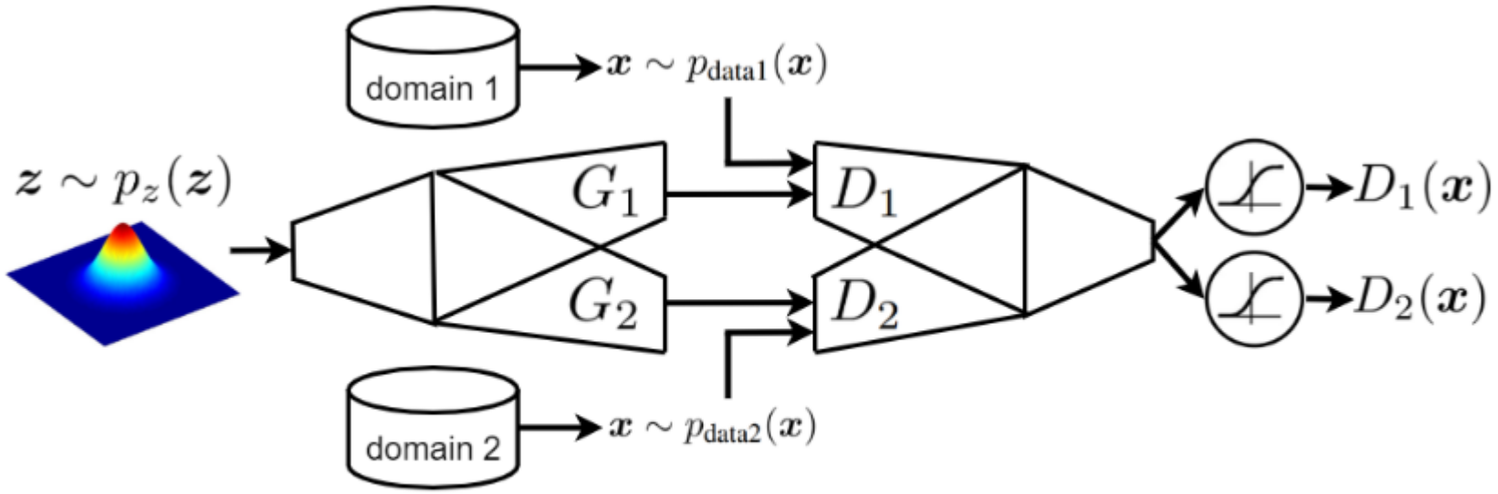


Figure 6. The structure of CoGAN.

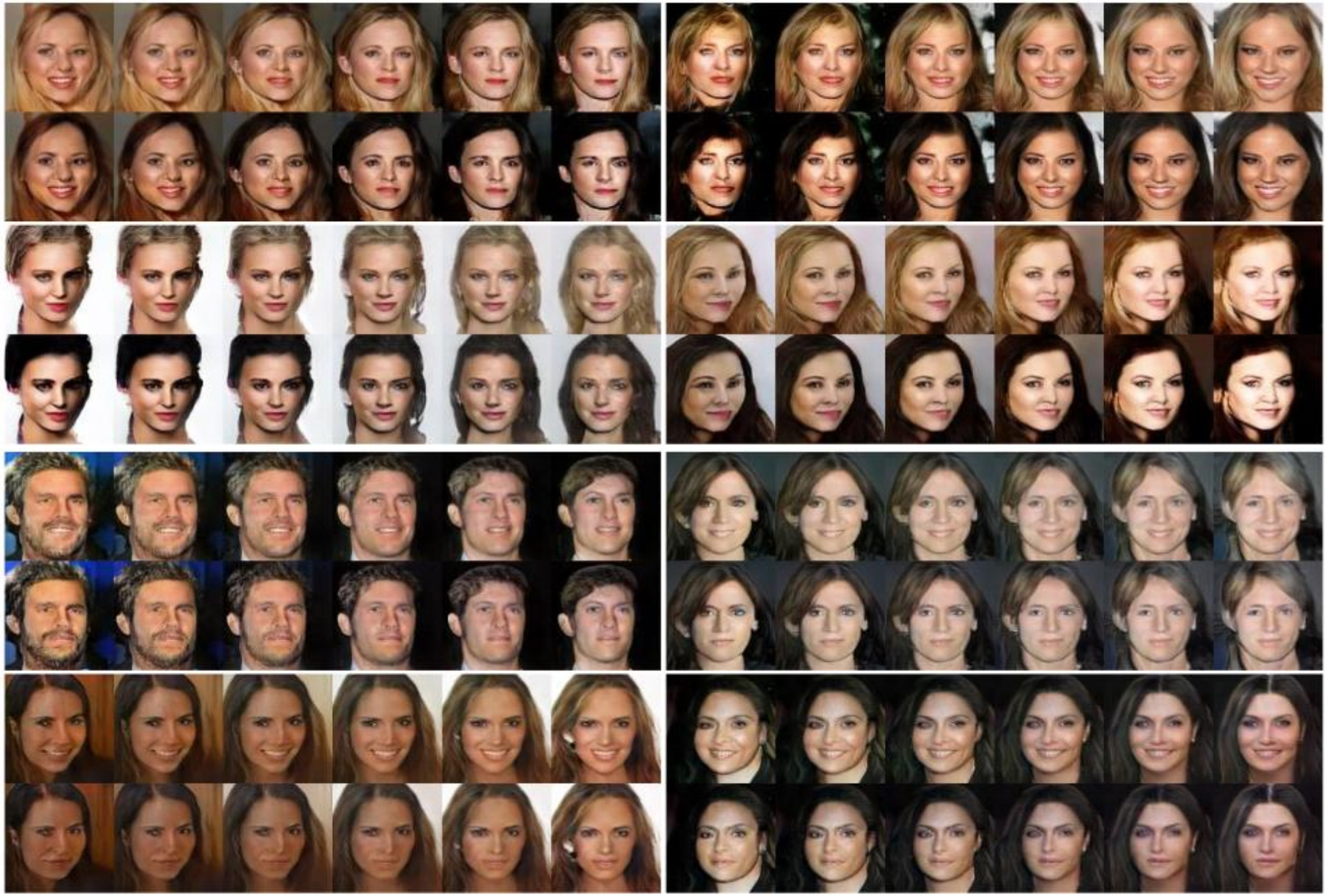
ещё одна картинка

https://arxiv.org/pdf/2111.13282.pdf

Coupled GANs: генерация из одного латентного кода z

цвет волос

выражение лица



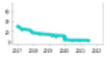
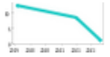
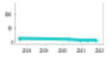
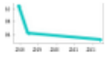
Coupled GANs: генерация из одного латентного кода z

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.

Image Generation

1113 papers with code • 81 benchmarks • 62 datasets

Trend	Dataset	Best Model	Paper	Code	Compare
	CIFAR-10	StyleGAN-XL			See all
	ImageNet 64x64	CDM			See all
	FFHQ 256 x 256	StyleNAT			See all
	STL-10	Diffusion ProjectedGAN			See all
	ImageNet 32x32	StyleGAN-XL			See all
	LSUN Churches 256 x 256	Projected GAN			See all
	CelebA 64x64	Diffusion StyleGAN2			See all
	FFHQ 1024 x 1024	StyleGAN-XL			See all
	LSUN Bedroom 256 x 256	Diffusion ProjectedGAN			See all
	CelebA 256x256	Efficient-VDVAE			See all

декабрь 2022 года https://paperswithcode.com/task/image-generation

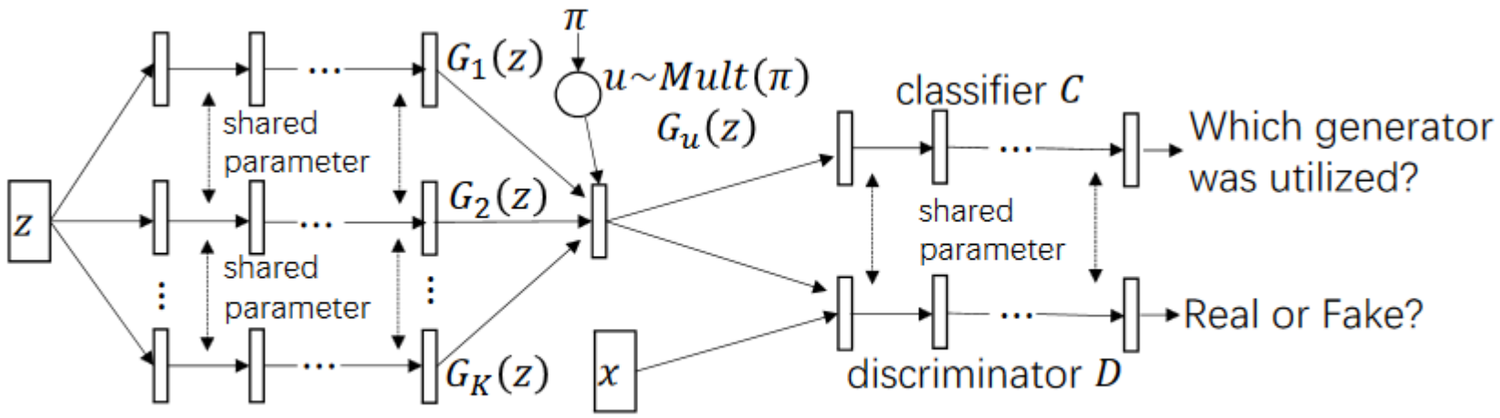
Итог

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

Дополнение: MGAN



(c) MGAN