

Генеративно-состязательные сети

Виктор Китов
victorkitov.github.io



Содержание

1 Безусловная генерация

- Генеративно-состязательные сети
- Практические рекомендации
- Генерация изображений
- Progressive GAN
- StyleGAN

2 Применение GAN в прогнозировании

3 Условная генерация

1 Безусловная генерация

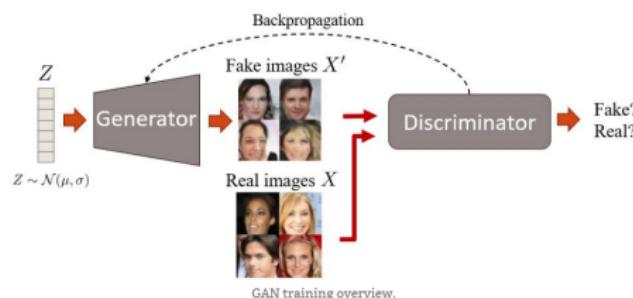
- Генеративно-состязательные сети
 - Практические рекомендации
 - Генерация изображений
 - Progressive GAN
 - StyleGAN

Решаемая задача: сэмплирование "натуральных" объектов

- Рассмотрим выборку $x_1, \dots, x_N, x_i \sim p(x)$.
- Хотим сэмплировать $\hat{x}_{N+1}, \hat{x}_{N+2}, \dots$ из похожего распределения.
- Статистический подход:
 - Оцениваем $q_\theta(x) \approx p(x)$ и сэмплируем из $q_\theta(x)$.
 - сложно оценить в многомерном пространстве
- Генеративно-состязательные сети (generative adversarial network, GAN):
 - заменим сложную задачу оценки $q_\theta(x)$ простой - бинарной классификацией
 - обучим дискриминатор, отличающий x от \hat{x}
 - дискриминатор выступает как обучаемая функция потерь для улучшения генератора $\hat{x} = g(z), z \sim p(z)$.

Аналогия GAN

GAN для изображений:



Аналогия - соревнование банка и фальшивомонетчика (имеющего шпиона в банке).

- они соревнуются, пока фальшивомонетчик не научится делать точные копии.

Основы GAN¹

- Две сети:

- генератор $G(z) : Z \rightarrow X$, $z \sim p(z)$
 - выдаёт сгенерированные объекты x
- дискриминатор $D(x) : X \rightarrow [0, 1]$
 - вероятность, что x настоящий, а не сгенерированный G .

¹[Link to paper.](#)

Основы GAN¹

- Две сети:

- генератор $G(z) : Z \rightarrow X, z \sim p(z)$
 - выдаёт сгенерированные объекты x
- дискриминатор $D(x) : X \rightarrow [0, 1]$
 - вероятность, что x настоящий, а не сгенерированный G .

- Определим:

- $p(x)$ - истинное распределение данных (из обучающей выборки)
- $q(x)$ - модельное распределение $G(z) \sim q(x), z \sim p(z)$.
- $p(z)$ - стандартное нормальное или равномерное распределение

¹[Link to paper.](#)

Соревнование

- Выходы дискриминатора

$$\begin{cases} D(x), & p(x\text{-настоящий}) \\ 1 - D(x), & p(x\text{-сгенерированный}) \end{cases}$$

Соревнование

- Выходы дискриминатора

$$\begin{cases} D(x), & p(x\text{-настоящий}) \\ 1 - D(x), & p(x\text{-сгенерированный}) \end{cases}$$

- Логарифм правдоподобия корректной классификации дискриминатором:

- считаем $p(\text{real}) = p(\text{fake}) = \frac{1}{2}$

$$V(D, G) = \mathbb{E}_{x \sim p(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))]$$

Соревнование

- Выходы дискриминатора

$$\begin{cases} D(x), & p(x\text{-настоящий}) \\ 1 - D(x), & p(x\text{-сгенерированный}) \end{cases}$$

- Логарифм правдоподобия корректной классификации дискриминатором:

- считаем $p(\text{real}) = p(\text{fake}) = \frac{1}{2}$

$$V(D, G) = \mathbb{E}_{x \sim p(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))]$$

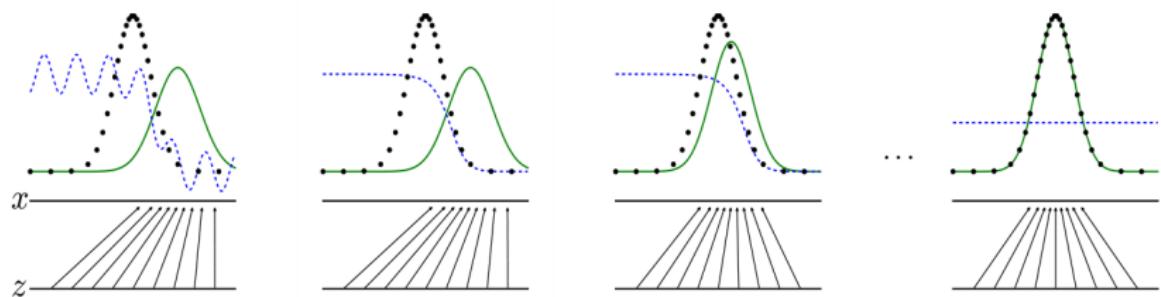
- D и G играют в антагонистическую игру с критерием $V(G, D)$:

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

- нестабильное обучение, нужно много инженерных трюков
- важно смотреть исходный код обучения

Иллюстрация соревнования в GAN

Последовательное обучение D и G :



чёрный поточечный: $p(x)$ - истинное распределение

зелёный: $q(x)$ - модельное распределение \hat{x}

голубой пунктирный: $D(x) = p(x\text{-настоящий}|x)$

Критерии оптимизации

Задача D (для фикс. G):

$$\mathbb{E}_{x \sim p(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \rightarrow \max_D$$

Задача G (для фикс. D):

$$\mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \rightarrow \min_G$$

Критерии оптимизации

Задача D (для фикс. G):

$$\mathbb{E}_{x \sim p(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \rightarrow \max_D$$

Задача G (для фикс. D):

$$\mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \rightarrow \min_G$$

- Т.е. $G(z)$ учится генерировать реалистичные \hat{x} .
- D и G должны учиться синхронно:
 - хорошо обученный D легко отличит x от \hat{x} , $\nabla_G \mathcal{L} \approx 0$, нет обучения.
 - хорошо обученный G будет сэмплировать $\hat{x} = \arg \max_x D(x)$ (mode collapse)

Алгоритм настройки

for number of training iterations **do**

for k steps **do**

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

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

end for

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

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

end for

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

Сгенерированные изображения



a)



b)



Интерполяция в пространстве Z

Линейная интерполяция в пространстве Z приводит к



Оптимальный D^* при фиксированном G

Теорема: Для фикс. G оптимальный дискриминатор равен

$$D^*(x|G) = \frac{p(x)}{p(x) + q(x)}$$

Оптимальный D^* при фиксированном G

Теорема: Для фикс. G оптимальный дискриминатор равен

$$D^*(x|G) = \frac{p(x)}{p(x) + q(x)}$$

Доказательство:

$$\begin{aligned} V(G, D) &= \int_x p(x) \log(D(x)) dx + \int_z p(g(z)) \log(1 - D(g(z))) dz = \\ &= \int_x p(x) \log(D(x)) dx + q(x) \log(1 - D(x)) dx \end{aligned}$$

Применяя для $\forall x \in X$ (при достаточно гибком $D(\cdot)$)

$\arg \max_{d \in \mathbb{R}} \{p \log(d) + q \log(1 - d)\} = \frac{p}{p+q}$ $\forall p, q \in \mathbb{R}$, получим

$$\arg \max_D V(G, D) = \frac{p(x)}{p(x) + q(x)}$$

Оптимальный G^* при условии D^*

Теорема: при оптимальном D^* , оптимальный G^* выдаёт $q(x) = p(x)$.

Оптимальный G^* при условии D^*

Теорема: при оптимальном D^* , оптимальный G^* выдаёт $q(x) = p(x)$.

Доказательство:

Определим G^* как генератор, выдающий $\hat{x} \sim q^*(x) = p(x)$.

Тогда

$$\begin{aligned}
 V(G^*, D^*(G)) &= \\
 &= \int_x p(x) \log \frac{p(x)}{p(x) + q^*(x)} dx + \int_x q(x) \log \frac{q(x)}{p(x) + q^*(x)} dx \\
 &= \int_x p(x) \log \frac{1}{2} dx + \int_x q(x) \log \frac{1}{2} dx = 2 \log \frac{1}{2} = -2 \log 2
 \end{aligned}$$

Оптимальный G^* при условии D^*

$V(G^*, D^*(G)) = \min_G V(G, D^*(G))$, поскольку

$$\begin{aligned}
 & V(G, D^*) - V(G^*, D^*) \\
 &= \int_x p(x) \log \frac{p(x)}{p(x) + q(x)} dx + \int_x q(x) \log \frac{q(x)}{p(x) + q(x)} dx \\
 &\quad + \int_x p(x) \log 2 dx + \int_x q(x) \log 2 dx \\
 &= \int_x p(x) \log \frac{p(x)}{\frac{p(x)+q(x)}{2}} dx + \int_x q(x) \log \frac{q(x)}{\frac{p(x)+q(x)}{2}} dx \\
 &= KL\left(p(x) \parallel \frac{p(x) + q(x)}{2}\right) + KL\left(q(x) \parallel \frac{p(x) + q(x)}{2}\right) \geq 0
 \end{aligned}$$

Использовали $KL(p_1(x) \parallel p_2(x)) \geq 0 \ \forall p_1(\cdot), p_2(\cdot)$.

- на практике: $D \neq D^*$, G ищется не при условии $D^*(G)$, настройка $G \rightarrow$ лишь лок. минимум $V(G, D^*(G))$.

1 Безусловная генерация

- Генеративно-состязательные сети
- Практические рекомендации
- Генерация изображений
- Progressive GAN
- StyleGAN

Проблема слишком точного генератора

Проблема:

- На ранних итерациях \hat{x} совсем не похожи на x .
- Приводят к однозначной дискриминации D .
- $D(G(z)) \approx 0 \Rightarrow \nabla_{\theta_G} \log(1 - D(G(z))) \approx 0$
 $\Rightarrow G$ не обучается.

Проблема диагностируется, если $\log P(D) \approx 0$.

Проблема слишком точного генератора

Решение - заменить критерий для G :

$$\mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \rightarrow \min_G$$

на

$$\mathbb{E}_{z \sim p(z)} [\log(D(G(z)))] \rightarrow \max_G$$

- G по-прежнему учится $\uparrow D(G(z))$
- $\log(\cdot)$ усиливает $D(G(z))$ около нуля.
 - $\nabla \log(D(G(z)))$ не вырождается.

Проблема слишком точного генератора

- Критерий дискриминатора:

$$\sum_n \alpha_n \log D(x_n) + (1 - \alpha_n) \log (1 - D(x_n)) \rightarrow \max_D$$

$$\alpha_n = \mathbb{I}[x_n\text{-настоящий}]$$

- Для более неоднозначной классификации дискриминатора можно

- случайно изменять метки с малой p :

$$\alpha'_n = \begin{cases} \alpha_n & \text{с вероятностью } 1 - p \\ 1 - \alpha_n & \text{с вероятностью } p \end{cases}$$

- использовать мягкую разметку:

$$\alpha'_n = \begin{cases} \text{Uniform}[0, 0.2] & \text{если } \alpha_n = 0 \\ \text{Uniform}[0.8, 1] & \text{если } \alpha_n = 1 \end{cases}$$

- вычислять $D(x + \text{шум})$

Проблема mode collapse

- При слишком тщательной настройке G он будет выдавать $const \equiv \arg \max_x D(x)$
 - англ: mode collapse
- Решения проблемы:
 - сделать более рандомизированную генерацию:
 - добавлять (sum(concat²) шум в более поздние слои G
 - использовать дропаут на этапе обучения и применения
 - при обучении минибатчами штрафовать $G(z_1), G(z_2), \dots G(z_K)$ за сходство, например

$$\log P(D, G) - \lambda \sum_{i < j} \|G(z_i) - G(z_j)\|_2^2 \rightarrow \min_G$$

²<https://arxiv.org/pdf/1609.03126.pdf>

1 Безусловная генерация

- Генеративно-состязательные сети
- Практические рекомендации
- Генерация изображений
- Progressive GAN
- StyleGAN

Deep convolutional GAN (DCGAN)³

Deep convolutional GAN (DCGAN) - архитектура G и D , специально для изображений:



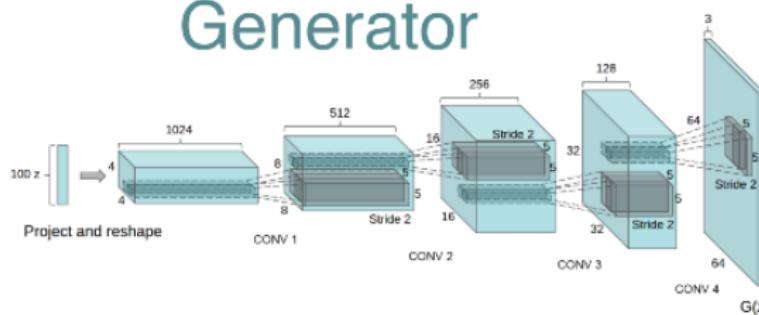
³<https://arxiv.org/pdf/1511.06434.pdf>

Архитектура DCGAN

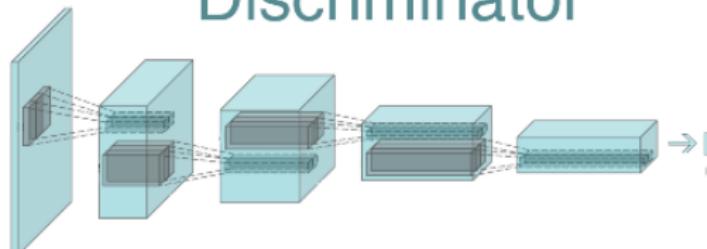
Для G и D используется полносвёрточная архитектура:

- D - развёрнутый G

Generator



Discriminator



Основные идеи DCGAN

- ➊ Нет полносвязных слоёв.
- ➋ Upsampling в G : транспонированные свёртки.
- ➌ Downsampling в D : свёртки со $\text{stride} > 1$.
 - более полный градиент для G , чем при max-pooling
- ➍ LeakyReLU активации на всех слоях дискриминатора (кроме последнего).
 - более полный градиент для G , чем при ReLU
- ➎ BatchNorm в G и D для ускорения обучения.

Математические операции в пространстве Z



man
with glasses



man
without glasses



woman
without glasses



woman with glasses

Эволюция качества генерации изображений



1 Безусловная генерация

- Генеративно-состязательные сети
- Практические рекомендации
- Генерация изображений
- **Progressive GAN**
- StyleGAN

Progressive GAN⁴

- Progressive GAN использовали поэтапное обучение для генерации HD изображений.

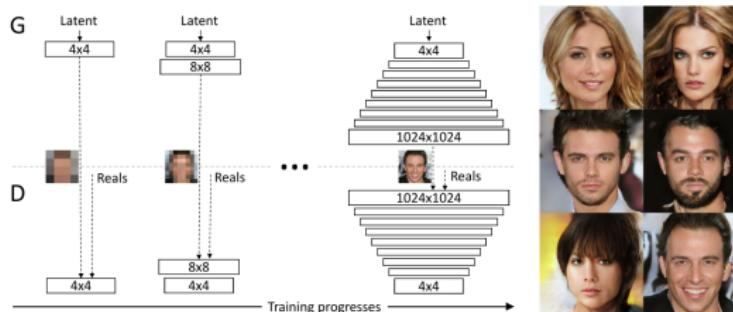
Сгенерированные изображения (обучение-датасет celebreties):



⁴Karras et al. 2017.

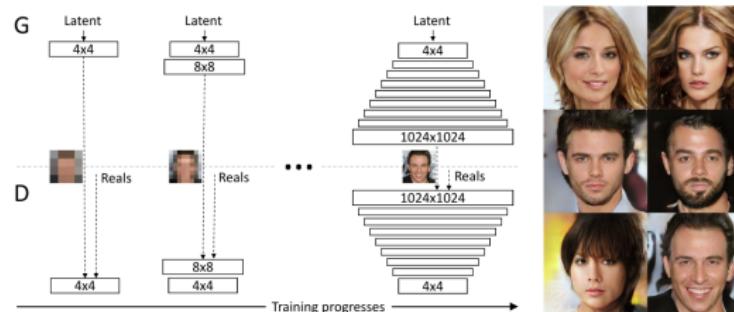
Поэтапное обучение

G, D сначала учатся на низком разрешении, потом к ним синхронно добавляются слои для \uparrow разрешения:



Поэтапное обучение

G, D сначала учатся на низком разрешении, потом к ним синхронно добавляются слои для \uparrow разрешения:



При появлении нового слоя создаются 2 ветви (как в ResNet):

- просто перемасштабирование с весом $1 - \alpha$
- перемасштабирование+всёртка+нелинейность с весом α

Во время обучения α постепенно изменяется от 0 к 1.

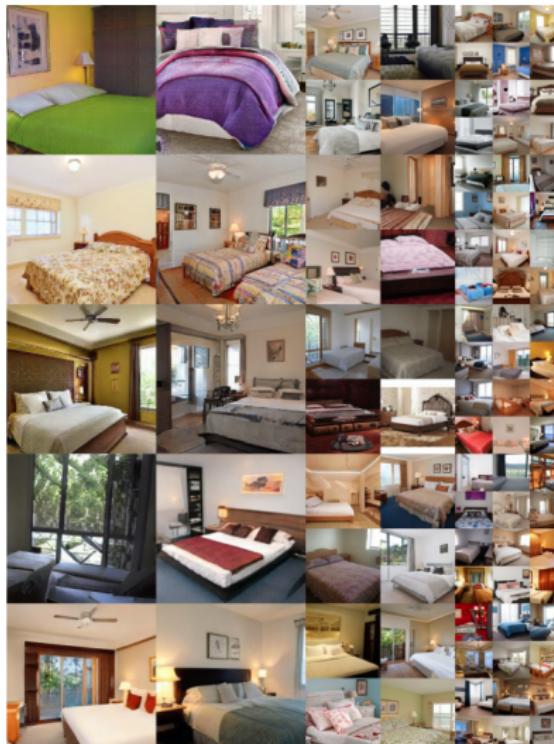
1 Безусловная генерация

- Генеративно-состязательные сети
- Практические рекомендации
- Генерация изображений
- Progressive GAN
- StyleGAN

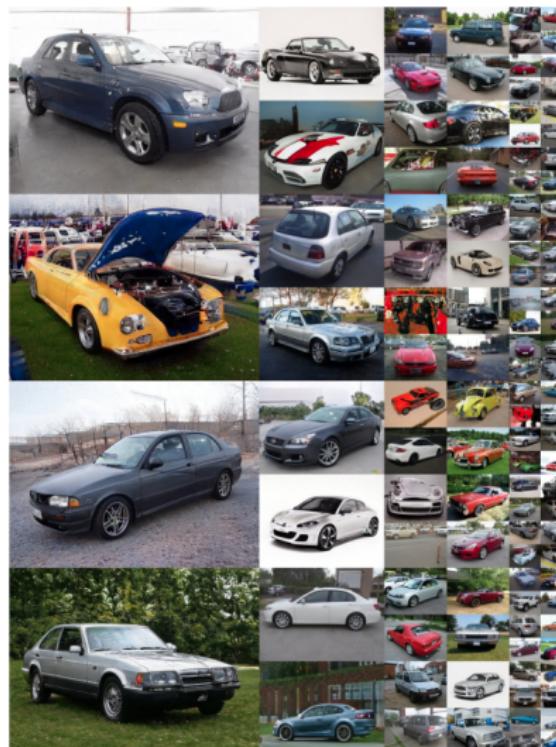
StyleGAN



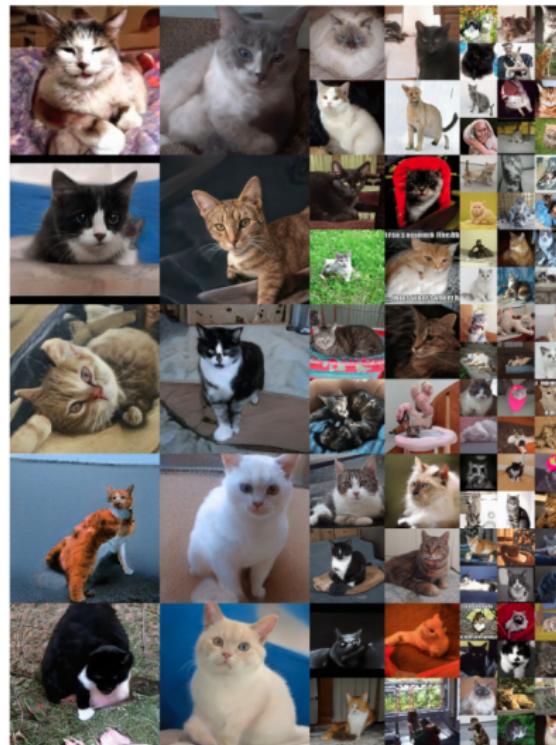
StyleGAN



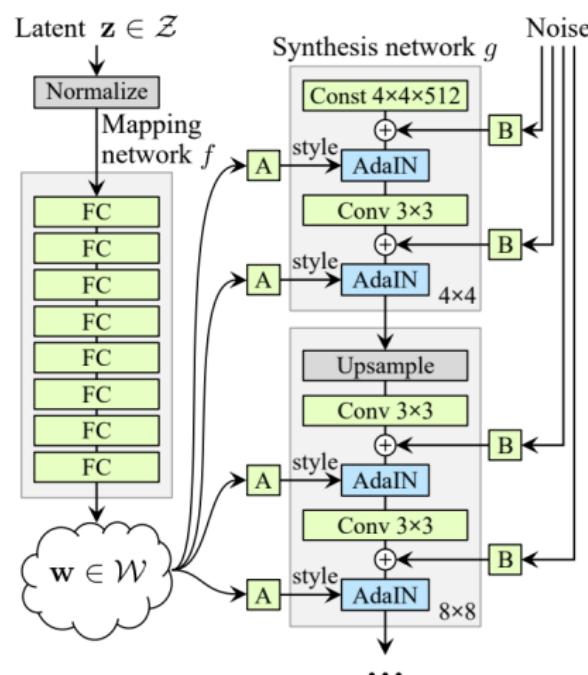
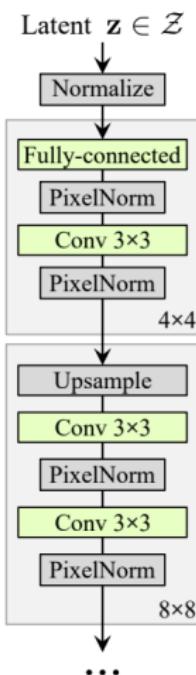
StyleGAN



StyleGAN



Архитектура



Основные идеи архитектуры

- Генерация - с выучиваемой константы $4 \times 4 \times 512$.
- Шум B - контент на глобальном и локальном уровне.
- Шум z преобразуется в глобальные характеристики (стиль) - общие и тонкие:

$$w = [\alpha_1^{coarse}, \beta_1^{coarse}, \alpha_2^{coarse}, \beta_2^{coarse}, \dots, \alpha_1^{fine}, \beta_1^{fine}, \alpha_2^{fine}, \beta_2^{fine}, \dots]$$

накладываемые для каждого канала i :

$$AdaIn(x_i, \alpha, \beta) = \alpha_i \frac{x_i - \mu(x_i)}{\sigma(x_i)} + \beta_i$$

Смешивание стилей на разных слоях

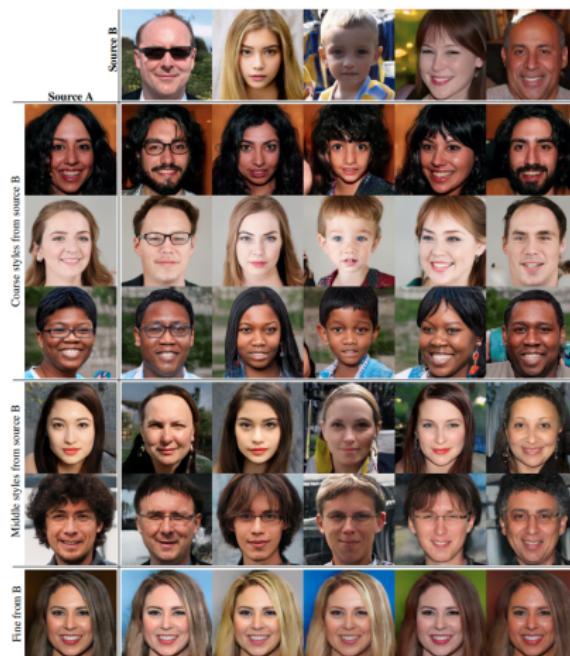


Figure 3. Two sets of images were generated from their respective latent codes (sources A and B); the rest of the images were generated by copying a specified subset of styles from source B and taking the rest from source A. Copying the styles corresponding to coarse spatial resolutions ($4^2 - 8^2$) brings high-level aspects such as pose, general hair style, face shape, and eyeglasses from source B, while all colors (eyes, hair, lighting) and finer facial features resemble A. If we instead copy the styles of middle resolutions ($16^2 - 32^2$) from B, we inherit smaller scale facial features, hair style, eyes open/closed from B, while the pose, general face shape, and eyeglasses from A are preserved. Finally, copying the fine styles ($64^2 - 1024^2$) from B brings mainly the color scheme and microstructure.

Разделение контента и стиля по слоям (отвечающим разному разрешению) позволяют смешивать грубые и тонкие характеристики результата.

Содержание

- 1 Безусловная генерация
- 2 Применение GAN в прогнозировании
 - Частичное обучение с GAN
- 3 Условная генерация

2 Применение GAN в прогнозировании

- Частичное обучение с GAN

Частичное обучение

- Есть мало размеченных данных:

$$(x_1, y_1), \dots (x_N, y_N)$$

- и много неразмеченных:

$$x_{N+1}, x_{N+2}, \dots x_{N+M}, M \gg N$$

- Частичное обучение (semi-supervised learning) - использование в том числе неразмеченных данных для обучения предиктора y .

Частичное обучение

- Есть мало размеченных данных:

$$(x_1, y_1), \dots (x_N, y_N)$$

- и много неразмеченных:

$$x_{N+1}, x_{N+2}, \dots x_{N+M}, M \gg N$$

- Частичное обучение (semi-supervised learning) - использование в том числе неразмеченных данных для обучения предиктора y .
- GAN - обучается на неразмеченной выборке.
 - извлекает признаки, характеризующие реальные изображения
 - может помочь при классификации

Частичное обучение с GAN⁵

Частичное обучение с GAN (semi-supervised GAN, SGAN):

- классификатор и генератор объединены и учатся совместно
- их общность параметров (для разных, но связанных задач) помогает каждую задачу решать точнее
 - обучая общие внутренние представления: характеризующие как класс, так и общую правдоподобность изображений

Классификатор-генератор всегда выдаёт $C + 1$ вероятностей:

$$[p(x\text{-real}, y = 1|x), \dots p(x\text{-real}, y = C|x), p(x\text{-fake}|x)]$$

⁵<https://arxiv.org/pdf/1606.01583.pdf>

Обучение SGAN

Algorithm 1 SGAN Training Algorithm

Input: I : number of total iterations

for $i = 1$ **to** I **do**

 Draw m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.

 Draw m examples $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ from data generating distribution $p_d(x)$.

 Perform gradient descent on the parameters of D w.r.t. the NLL of D/C's outputs on the combined minibatch of size $2m$.

 Draw m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.

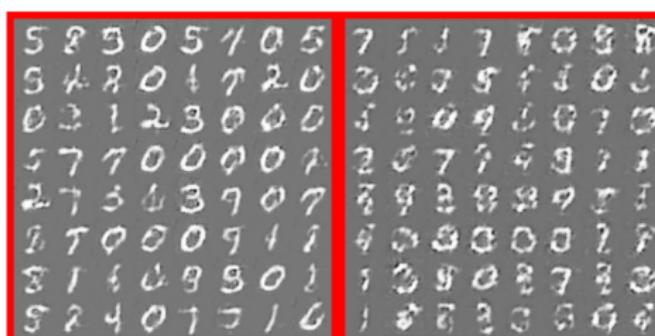
 Perform gradient descent on the parameters of G w.r.t. the NLL of D/C's outputs on the minibatch of size m .

end for

Генерация SGAN

SGAN сходится быстрее за счет классификатора:

Generated MNIST images by SGAN (left) and GAN (right) after 2 MNIST epochs



Классификация SGAN

Классификатор работает точнее за счёт GAN:

Сравнение точности CNN и SGAN для обучающих выборок разного размера MNIST:

EXAMPLES	CNN	SGAN
1000	0.965	0.964
100	0.895	0.928
50	0.859	0.883
25	0.750	0.802

Содержание

- 1 Безусловная генерация
- 2 Применение GAN в прогнозировании
- 3 Условная генерация
 - Условная генерация pix2pix
 - Условная генерация CycleGAN
 - Другие способы условной генерации

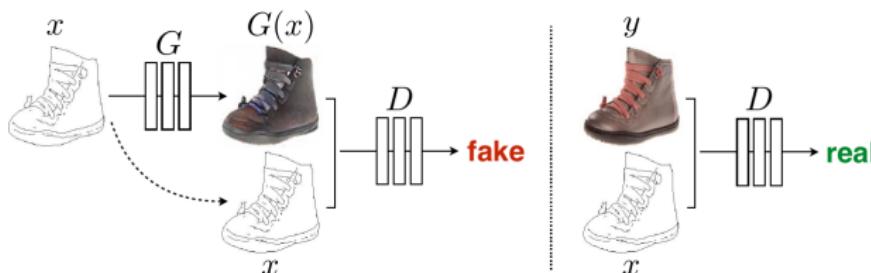
3 Условная генерация

- Условная генерация pix2pix
- Условная генерация CycleGAN
- Другие способы условной генерации

Условная генерация pix2pix⁶

- $z \sim p(z)$ - случайная инициализация
- x - исходная версия объекта
- y - целевая версия объекта
- $\hat{y} = G(x, z)$ - условная реконструкция объекта

Conditional GAN:



⁶<https://arxiv.org/pdf/1611.07004.pdf>

Критерии обучения генерации

- Потери реконструкции:

$$\mathcal{L}_{rec}(G) = \mathbb{E}_{(x,y),z} \|y - G(x, z)\|_1$$

- L_2 приводит к размытости, L_1 - чётче результат

Критерии обучения генерации

- Потери реконструкции:

$$\mathcal{L}_{rec}(G) = \mathbb{E}_{(x,y),z} \|y - G(x, z)\|_1$$

- L_2 приводит к размытости, L_1 - чётче результат
- Генеративно-состязательные потери:

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

- выигрыш для D , потери для G
- поощряет реалистичность $\hat{y}|x$

Критерии обучения генерации

- Потери реконструкции:

$$\mathcal{L}_{rec}(G) = \mathbb{E}_{(x,y),z} \|y - G(x, z)\|_1$$

- L_2 приводит к размытости, L_1 - чётче результат
- Генеративно-состязательные потери:

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

- выигрыш для D , потери для G
- поощряет реалистичность $\hat{y}|x$

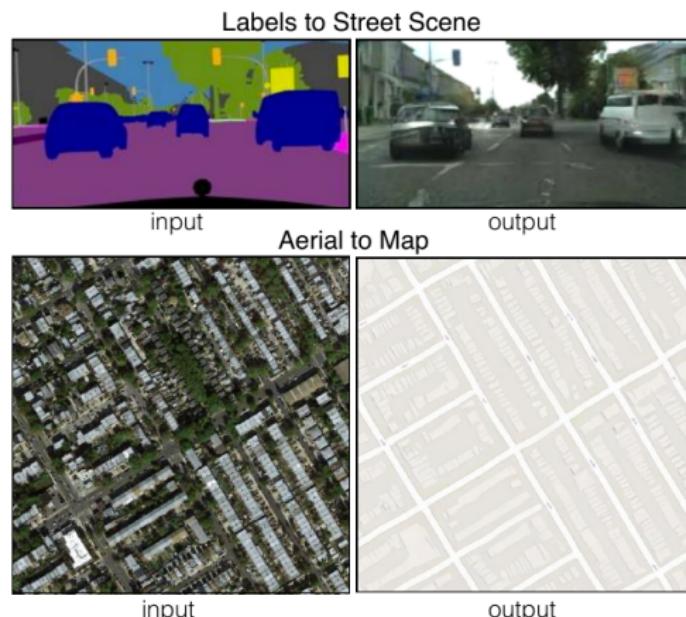
Настройка G :

$$G^* = \arg \min_G \max_D \{\mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{rec}(G)\}$$

Условная генерация

Условная генерация pix2pix

Результаты для разных X, Y

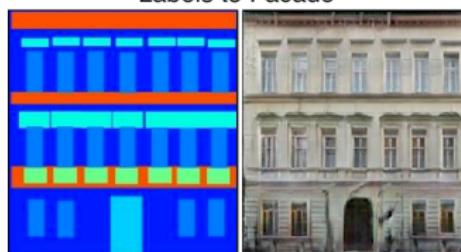


Условная генерация

Условная генерация pix2pix

Результаты для разных X, Y

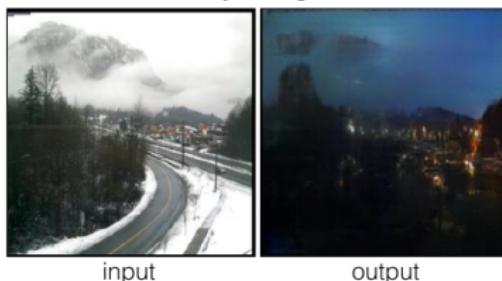
Labels to Facade



input

output

Day to Night



input

output

Результаты для разных X,Y

BW to Color



input

output

Edges to Photo



input

output

Условная генерация

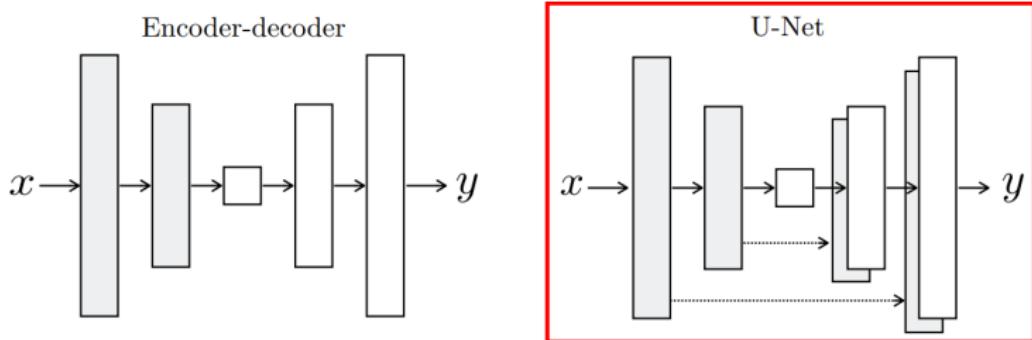
Условная генерация pix2pix

Результаты для разных X, Y



Комментарии

- Генератор использует архитектуру U-net:



- Дискриминатор применяется для пересекающихся патчей $N \times N$
 - выход - усреднённый результат
 - может применяться для изображений \forall размера
 - минимально-достаточное $N = 70$

И U-net, и потери сGAN важны

L1+cGAN better than L1 only, Unet better than encoder-decoder architecture.



Эффект выбора разных N

Потери L1 vs. потери L1+потери сGAN с разным N :

L1



1×1



16×16



70×70



286×286



3 Условная генерация

- Условная генерация pix2pix
- Условная генерация CycleGAN
- Другие способы условной генерации

CycleGAN⁷

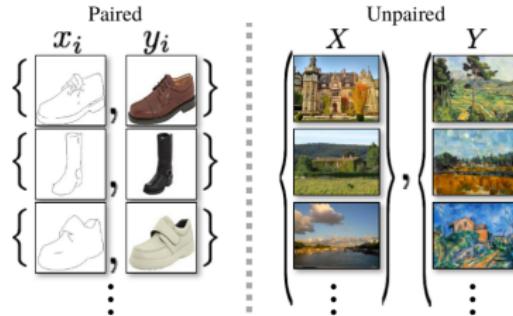
- Два представления объектов: X, Y .
- pix2pix обучался на парной выборке: $\{(x_n, y_n)\}_{n=1}^N$

⁷<https://arxiv.org/pdf/1703.10593.pdf>

CycleGAN⁷

- Два представления объектов: X, Y .
- pix2pix обучался на парной выборке: $\{(x_n, y_n)\}_{n=1}^N$
- CycleGAN обучается на непарных выборках:

$$\{x_1, x_2, \dots x_N\}, \{y_1, y_2, \dots y_M\}$$



- Задача CycleGAN: выучить $G_{XY} : X \rightarrow Y$, $G_{YX} : Y \rightarrow X$.

⁷<https://arxiv.org/pdf/1703.10593.pdf>

Потери

$$\begin{aligned}\mathcal{L}_{GAN}(G_{XY}, D_Y) = & \mathbb{E}_{y \sim p(y)} [\log(D_Y(y))] \\ & + \mathbb{E}_{x \sim p(x)} [\log(1 - D_Y(G_{XY}(x)))]\end{aligned}$$

$$\begin{aligned}\mathcal{L}_{GAN}(G_{YX}, D_X) = & \mathbb{E}_{x \sim p(x)} [\log(D_X(x))] \\ & + \mathbb{E}_{y \sim p(y)} [\log(1 - D_X(G_{YX}(y)))]\end{aligned}$$

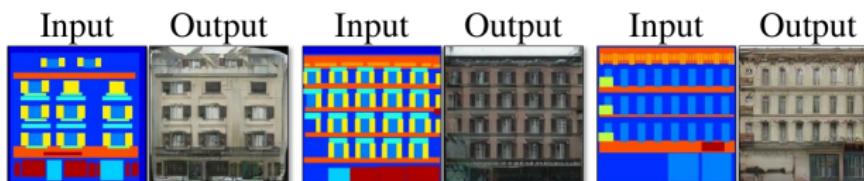
$$\begin{aligned}\mathcal{L}_{cyc}(G_{XY}, G_{YX}) = & \mathbb{E}_{x \sim p(x)} \|G_{YX}(G_{XY}(x)) - x\|_1 \\ & + \mathbb{E}_{y \sim p(y)} \|G_{XY}(G_{YX}(y)) - y\|_1\end{aligned}$$

Потери и задача оптимизации

$$\begin{aligned}\mathcal{L}(G_{XY}, G_{YX}, D_X, D_Y) = & \mathcal{L}_{GAN}(G_{XY}, D_Y) + \mathcal{L}_{GAN}(G_{YX}, D_X) \\ & + \mathcal{L}_{cyc}(G_{XY}, G_{YX})\end{aligned}$$

$$G_{XY}^*, G_{YX}^* = \arg \min_{G_{XY}, G_{YX}} \max_{D_X, D_Y} \mathcal{L}(G_{XY}, G_{YX}, D_X, D_Y)$$

Примеры работы



label → facade



facade → label



edges → shoes



shoes → edges

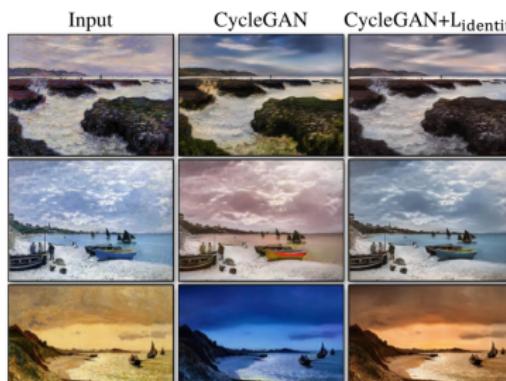
Регуляризатор тождественного преобразования

- Поскольку цели: $G_{YX}(G_{XY}(x)) \approx x$, $G_{XY}(G_{YX}(y)) \approx y$, то G_{XY} и G_{YX} определяются с точностью до константы

$$G'_{XY}(x) = G_{XY}(x) + \alpha, \quad G'_{YX}(x) = G_{YX}(y) - \alpha \quad \forall \alpha$$

- Для однозначности вводится identity loss:
 - "ничего не делать, если объект уже целевого домена"

$$\mathcal{L}_{identity} = \mathbb{E}_{x \sim p(x)} \|G_{YX}(x) - x\|_1 + \mathbb{E}_{y \sim p(y)} \|G_{XY}(y) - y\|_1$$



Условная генерация

Другие способы условной генерации

3 Условная генерация

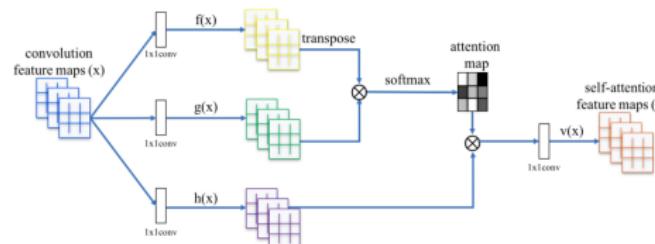
- Условная генерация pix2pix
- Условная генерация CycleGAN
- Другие способы условной генерации

Self-Attention GAN (SAGAN)⁸

- SAGAN (Self-Attention GAN) - используется self-attention
 - для учёта удалённых зависимостей в пространстве
- Дальнейшее развитие: BigGAN.



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.



⁸<https://arxiv.org/pdf/1805.08318.pdf>

Text2Image⁹

Text2Image генерирует изображения при условии их текстового описания:

this small bird has a pink breast and crown, and black primaries and secondaries.



this magnificent fellow is almost all black with a red crest, and white cheek patch



the flower has petals that are bright pinkish purple with white stigma



this white and yellow flower have thin white petals and a round yellow stamen

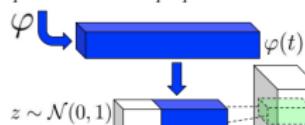


⁹<https://arxiv.org/pdf/1605.05396.pdf>

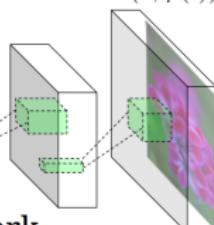
Text2Image

Описание кодируется в виде эмбеддинга, конкатенирующегося к случайной инициализации:

This flower has small, round violet petals with a dark purple center

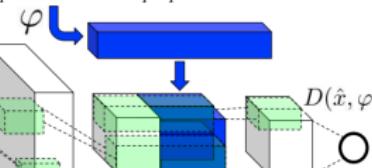


$$\hat{x} := G(z, \varphi(t))$$



Generator Network

This flower has small, round violet petals with a dark purple center



$$D(\hat{x}, \varphi(t))$$

Discriminator Network

Заключение

- GAN: $x_1, \dots, x_N \sim p(x) \longrightarrow \hat{x} \sim q(x) \approx p(x)$
- Проблемы настройки¹⁰:
 - слишком точный $D \Rightarrow \nabla_G \mathcal{L} \approx 0$
 - изменение $\mathcal{L}(G)$, зашумление либо сглаживание меток
 - слишком точный G : mode collapse $G(z) = \arg \max_x D(x)$
 - больше рандомизации в G , штраф за близкие $G(z)$ и $G(z')$
- Применения: частичное обучение, обучение RNN, условная генерация.
 - pix2pix учится на парных, а cycleGAN - на непарных объектах
- GAN могут применяться и для других доменов: текст, речь, графы и т.д.

¹⁰Обе проблемы также могут решаться через Waserstein GAN