

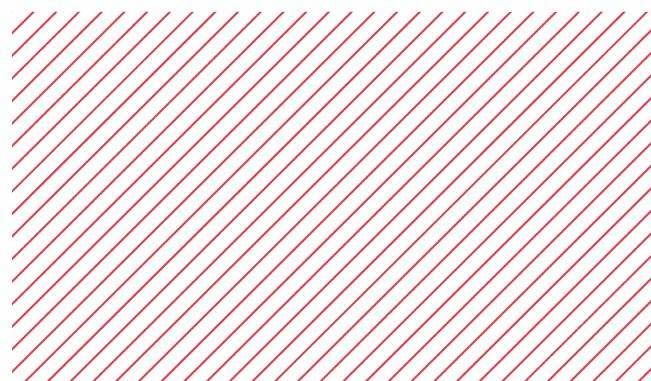
академия  
больших  
данных



# Генеративно-состязательные нейронные сети. Введение

Фёдор Киташов

Программист-исследователь в команде  
компьютерного зрения





# План лекции

---

- ❖ Мотивация и цели это лекции
  - Какие задачи решают генеративные сети
  - Эволюция генеративных сетей за последние 6 лет
- ❖ Статья “Generative Adversarial Nets” by I. Goodfellow
  - Архитектура: генератор и дискриминатор
  - Функция потерь и обучение
  - Проблемы сходимости и их решения.
  - Оптимальные дискриминатор и генератор



# План лекции

---

- ❖ Проблемы обучения ганов
- ❖ Mode Collapse
- ❖ Wasserstein GAN
  - Earth Mover's Distance
  - Непрерывность по Липшицу
  - Примеры работы



# На следующей лекции:

---

- ❖ Современные подходы к генерации изображений
  - Progressive growing of GANs
  - BigGAN
  - AdaIN слои
  - StyleGAN и StyleGAN2
- ❖ Conditional GANs
- ❖ Метрики для сравнения результатов генерации

# Прогресс

---



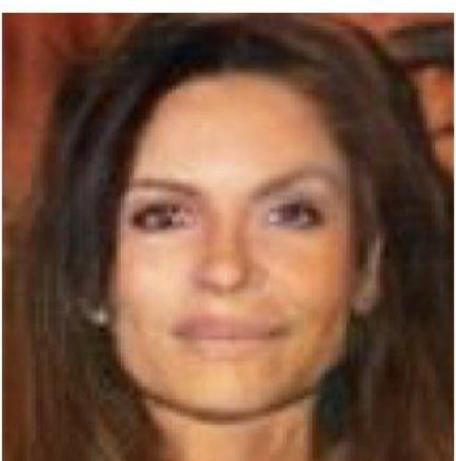
2014

GAN



2015

DCGAN



2016

CGAN



2017

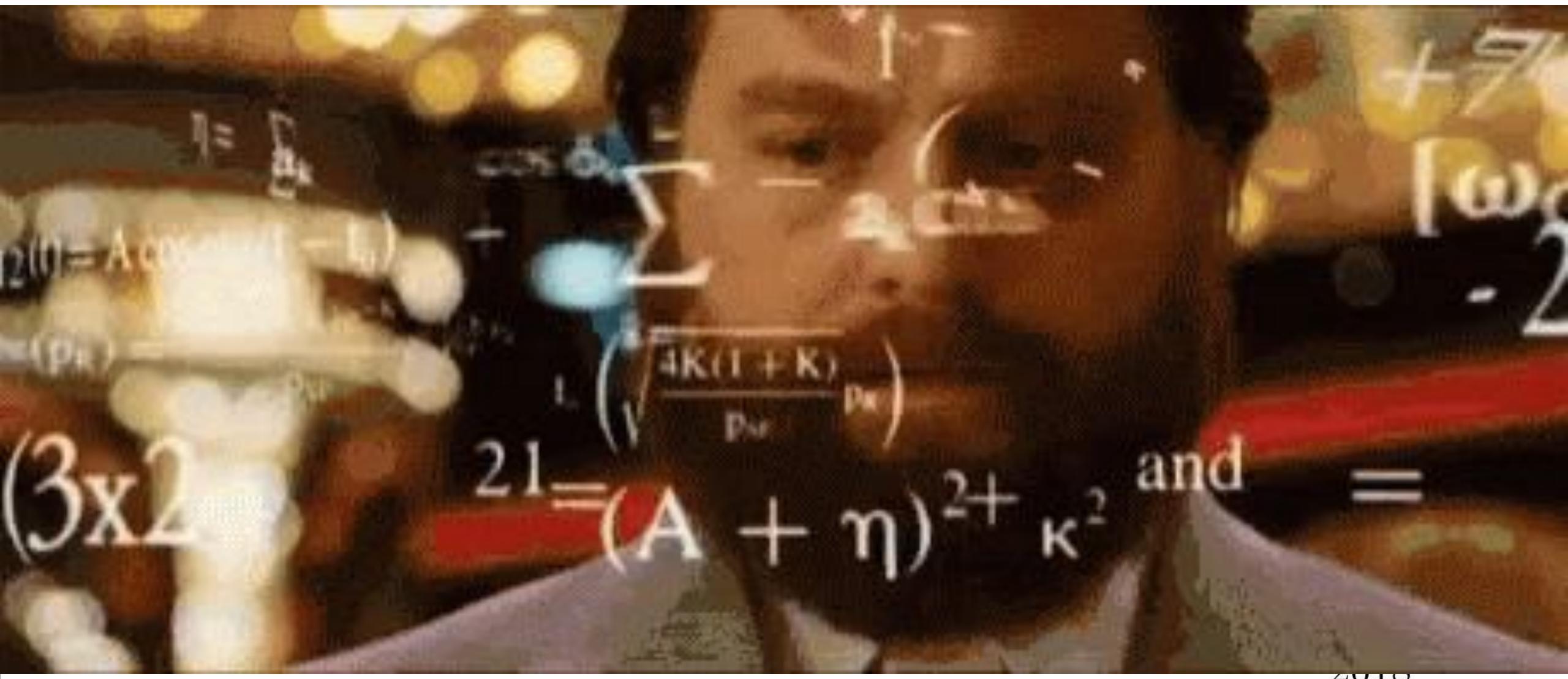
PGGAN



2018

StyleGAN

# Прогресс



2010









# Generative Adversarial Nets

---

- ❖ Статья “Generative Adversarial Nets” by I. Goodfellow
  - Архитектура: генератор и дискриминатор
  - Функция потерь и обучение
  - Оптимальные дискриминатор и генератор
  - Проблемы сходимости и их решения.



# Архитектура: генератор и дискриминатор

---

- ❖ Генератор:
  - нейронная сеть
  - генерирует изображения из случайного шума



# Архитектура: генератор и дискриминатор

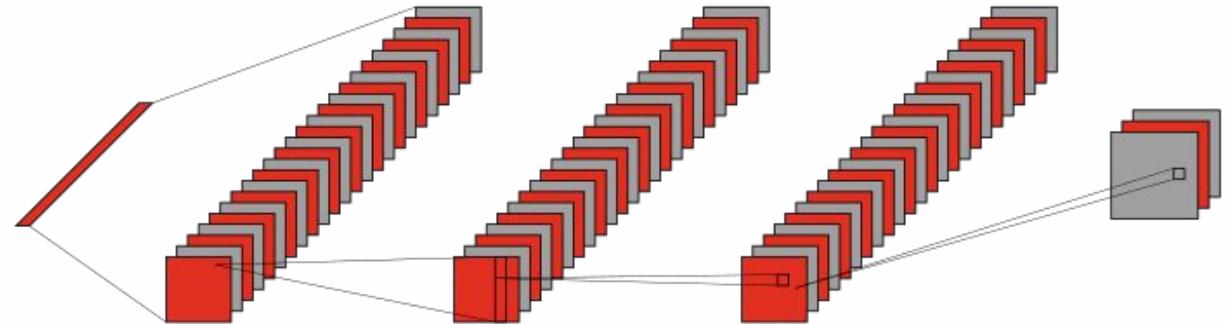
---

- ❖ Генератор:
  - нейронная сеть
  - генерирует изображения из случайного шума
  
- ❖ Дискриминатор:
  - нейронная сеть
  - учится как бинарный классификатор
  - учится отличать настоящие изображения от сгенерированных

# Архитектура: генератор и дискриминатор

---

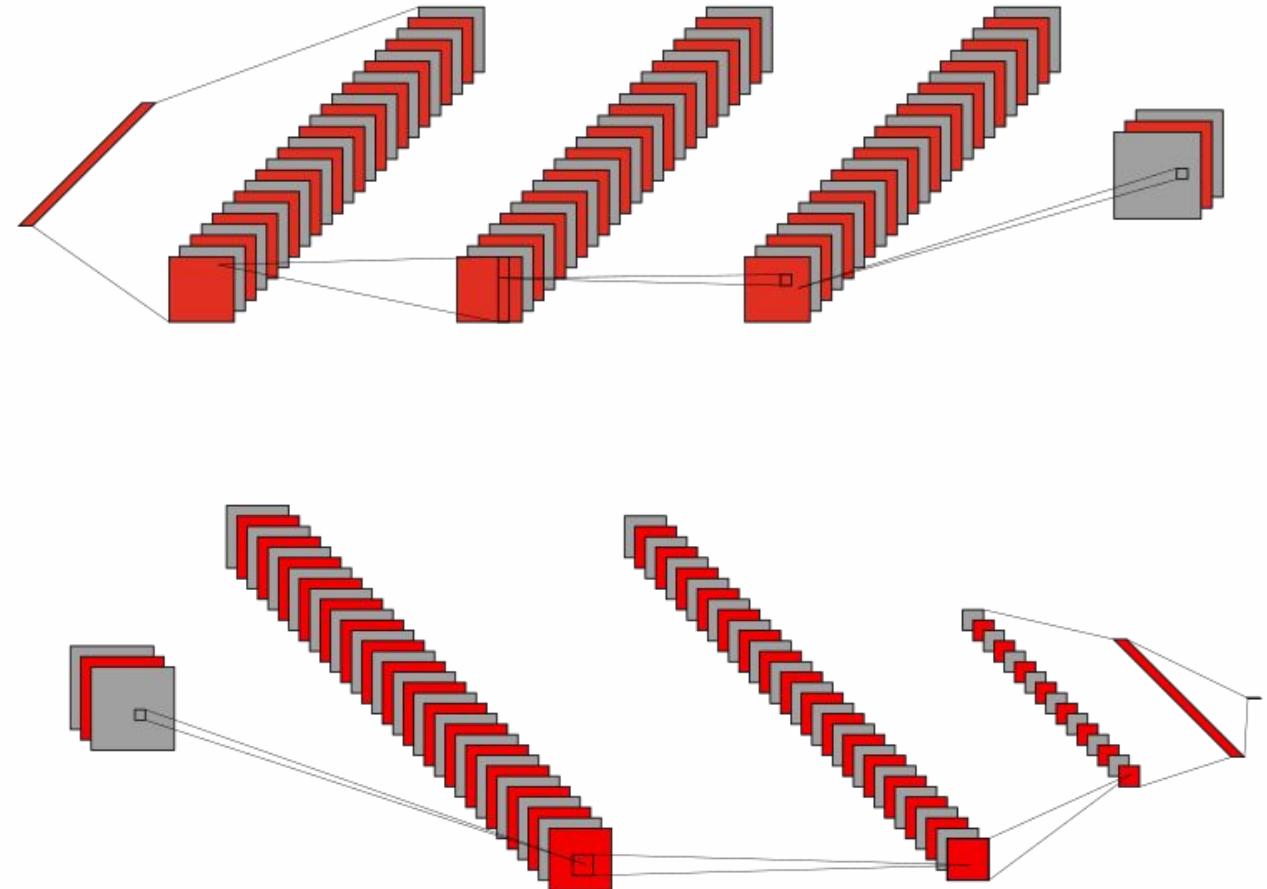
- ❖ Генератор:
  - на вход: вектор случайных чисел из равномерного или нормального распределения
  - на выход: изображение



# Архитектура: генератор и дискриминатор

---

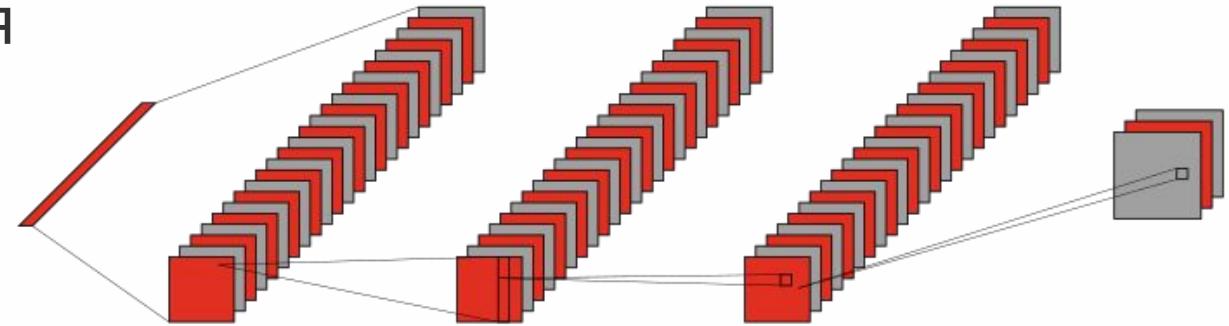
- ❖ Генератор:
  - на вход: вектор случайных чисел из равномерного или нормального распределения
  - на выход: изображение
- ❖ Дискриминатор:
  - на вход: изображение
  - на выход: вероятность того, что изображение настояще



# Архитектура: генератор и дискриминатор

---

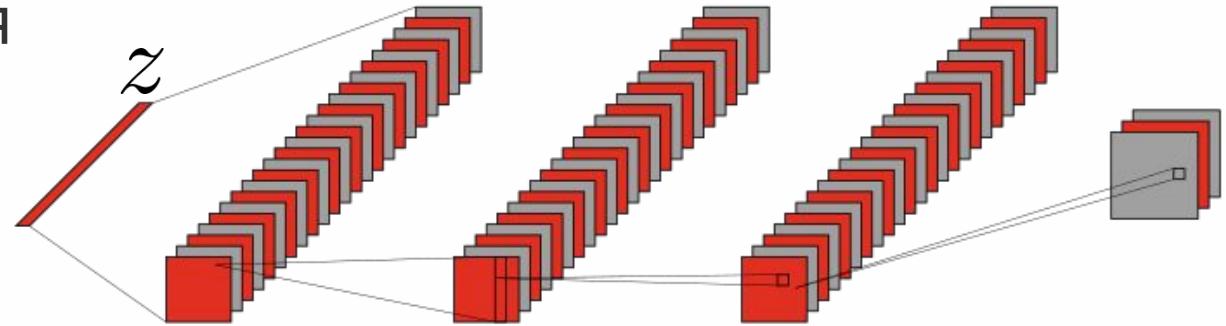
- ❖ Генератор (дифференцируемая функция  $G$ )



# Архитектура: генератор и дискриминатор

---

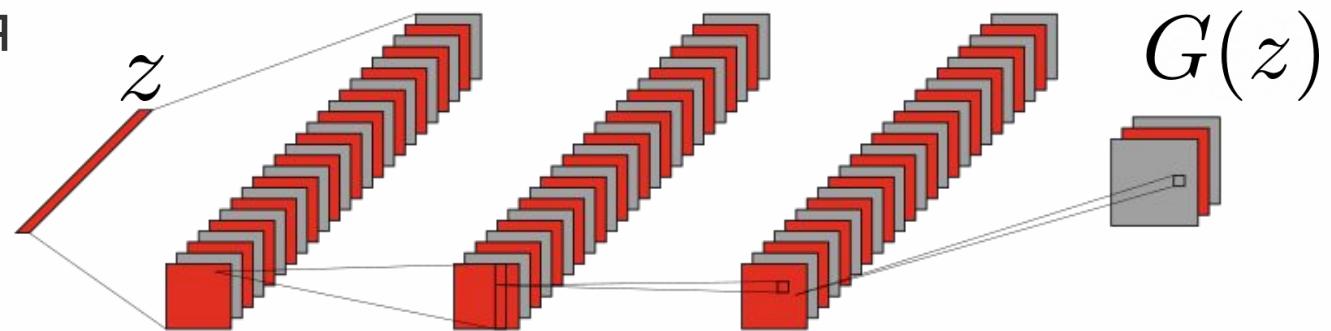
- ❖ Генератор (дифференцируемая функция  $G$ )
  - на вход: вектор шума  $z$



# Архитектура: генератор и дискrimинатор

---

- ❖ Генератор (дифференцируемая функция  $G$ )
  - на вход: вектор шума  $\mathbf{z}$
  - на выход: изображение  $\mathbf{G}(\mathbf{z})$

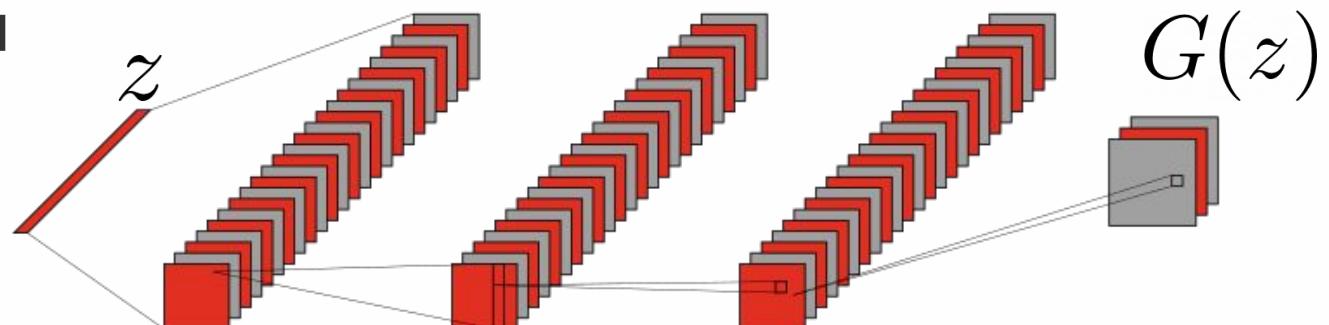


# Архитектура: генератор и дискриминатор

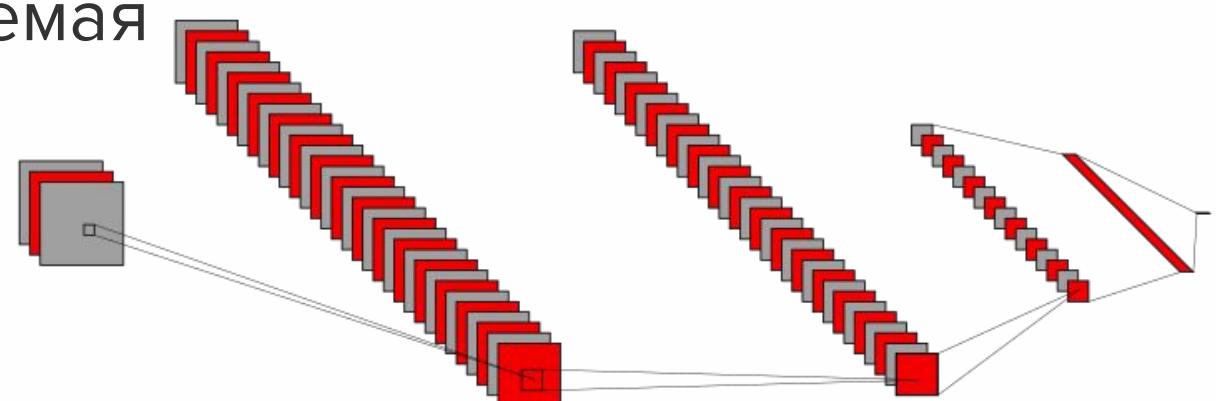
---

- ❖ Генератор (дифференцируемая функция  $G$ )

- на вход: вектор шума  $z$
- на выход: изображение  $G(z)$



- ❖ Дискриминатор (дифференцируемая функция  $D$ )

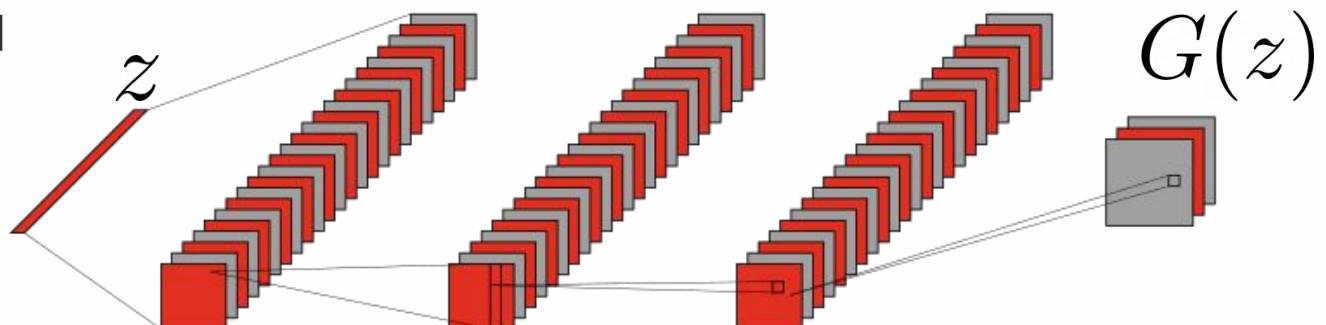


# Архитектура: генератор и дискриминатор

---

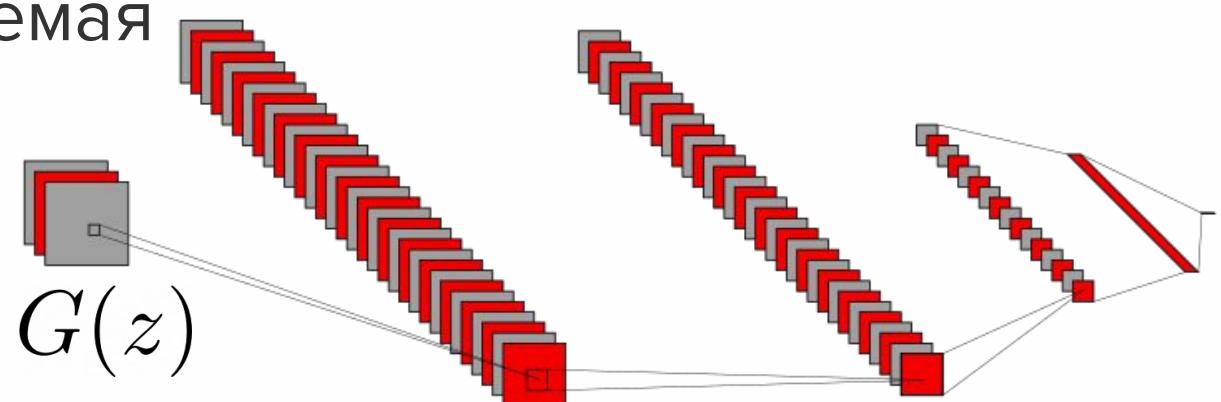
- ❖ Генератор (дифференцируемая функция  $G$ )

- на вход: вектор шума  $z$
- на выход: изображение  $\mathbf{G}(z)$



- ❖ Дискриминатор (дифференцируемая функция  $D$ )

- на вход: изображение  $\mathbf{G}(z)$

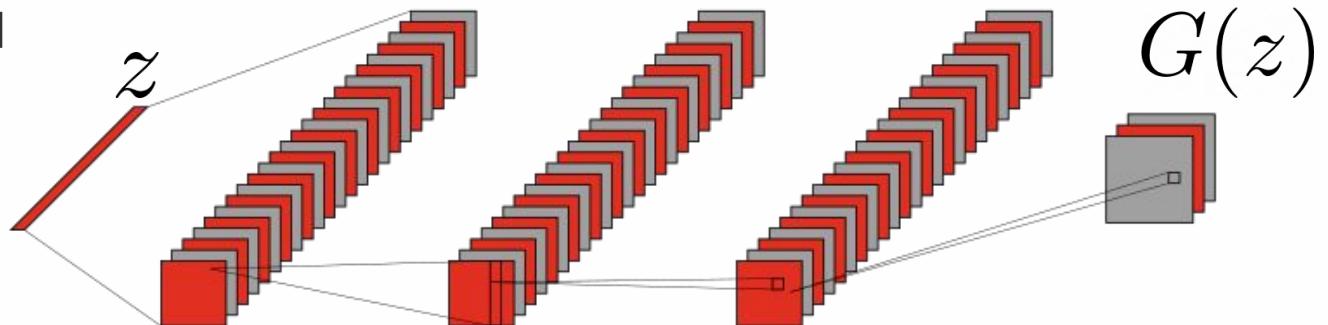


# Архитектура: генератор и дискриминатор

---

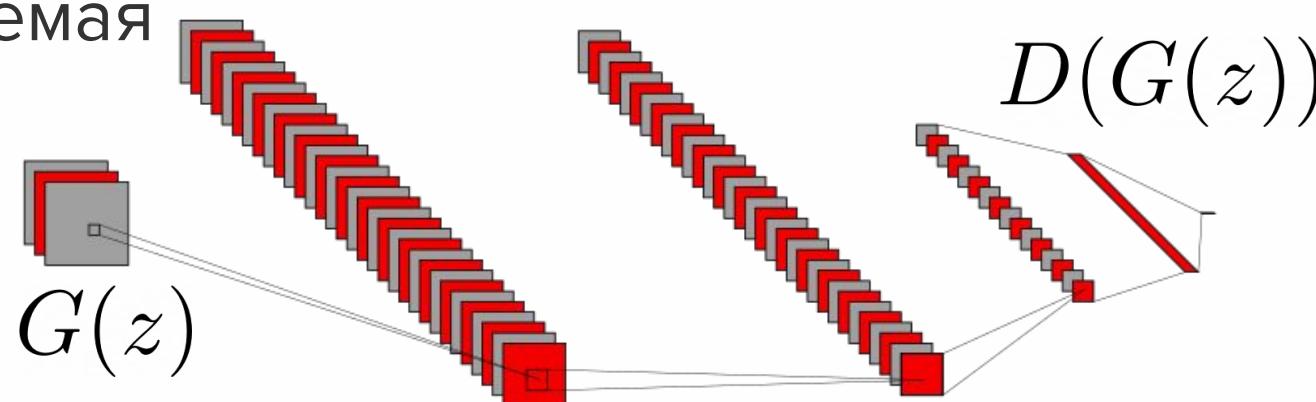
- ❖ Генератор (дифференцируемая функция  $G$ )

- на вход: вектор шума  $z$
- на выход: изображение  $\mathbf{G}(z)$



- ❖ Дискриминатор (дифференцируемая функция  $D$ )

- на вход: изображение  $\mathbf{G}(z)$
- на выход: число  $D(\mathbf{G}(z))$

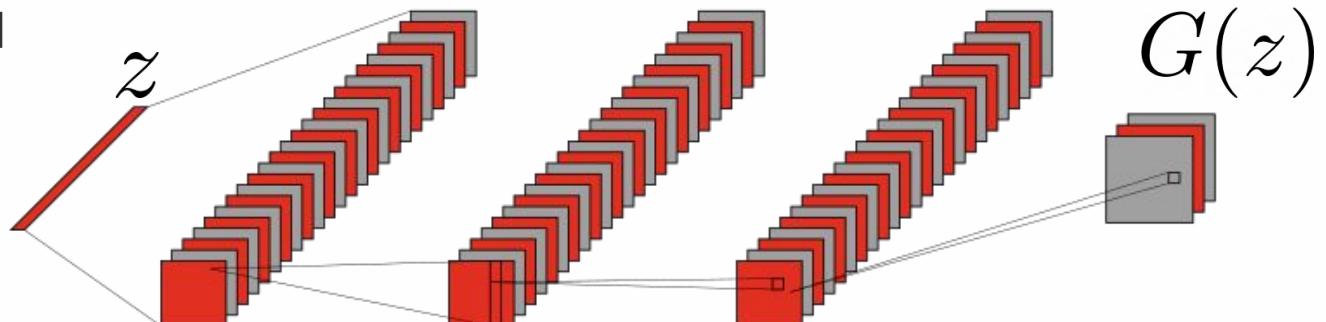


# Архитектура: генератор и дискриминатор

---

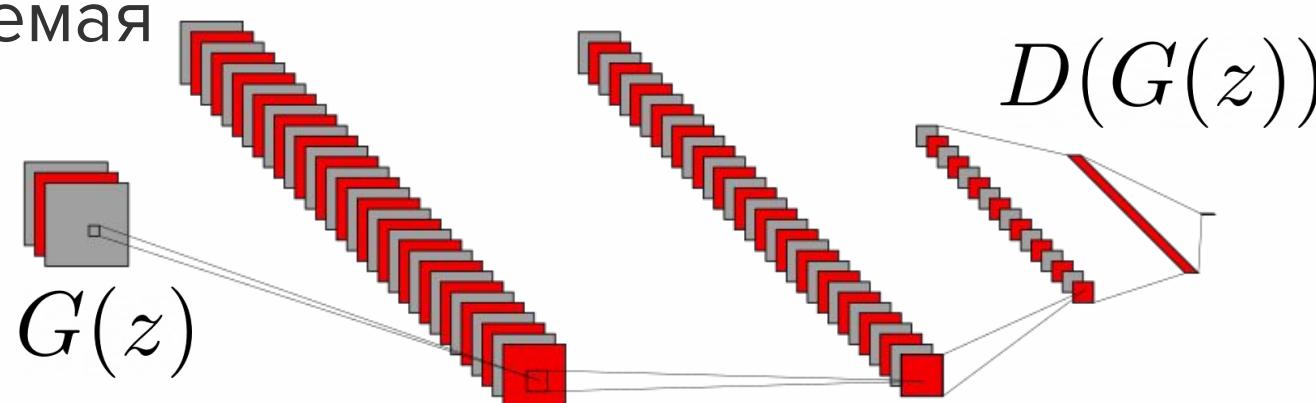
- ❖ Генератор (дифференцируемая функция  $G$ )

- на вход: вектор шума  $z$
- на выход: изображение  $\mathbf{G}(z)$



- ❖ Дискриминатор (дифференцируемая функция  $D$ )

- на вход: изображение  $\mathbf{G}(z)$
- на выход: число  $D(\mathbf{G}(z))$
- Дискриминатор  $D$  пытается сделать  $D(\mathbf{G}(z)) = 0$

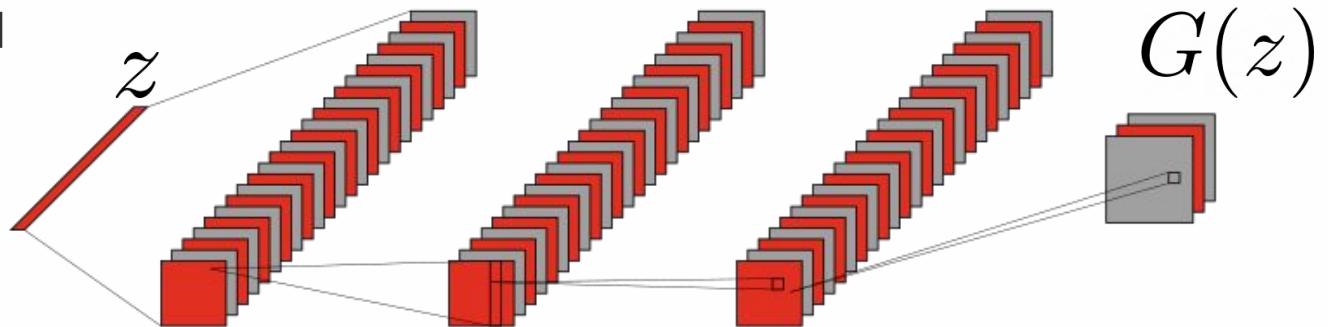


# Архитектура: генератор и дискриминатор

---

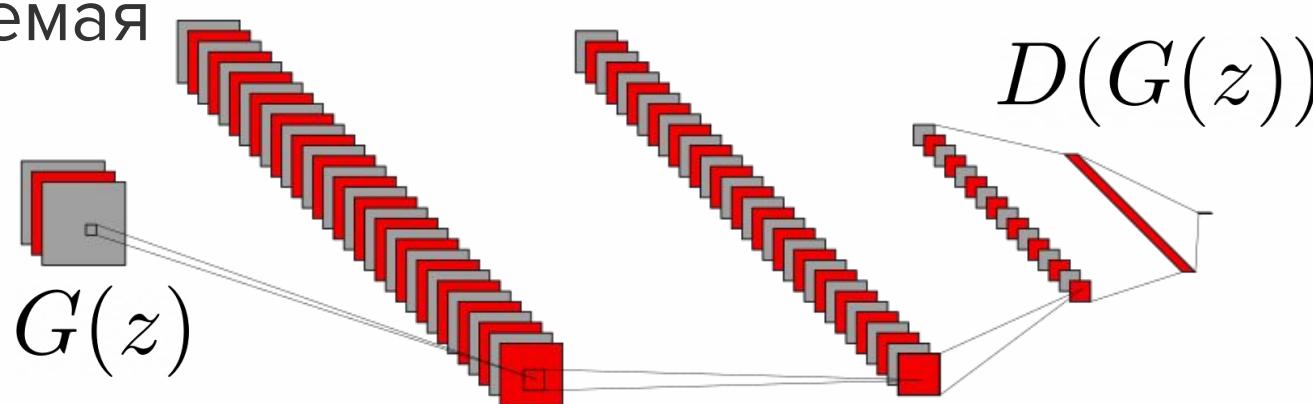
- ❖ Генератор (дифференцируемая функция  $G$ )

- на вход: вектор шума  $z$
- на выход: изображение  $\mathbf{G}(z)$
- Генератор  $G$  пытается сделать  $D(\mathbf{G}(z)) = 1$



- ❖ Дискриминатор (дифференцируемая функция  $D$ )

- на вход: изображение  $\mathbf{G}(z)$
- на выход: число  $D(\mathbf{G}(z))$
- Дискриминатор  $D$  пытается сделать  $D(\mathbf{G}(z)) = 0$





# Обучение: minimax game

---

- ❖ Генератор G пытается сделать ❖ Дискриминатор D пытается сделать  
 $D(G(z)) = 1$                                $D(G(z)) = 0$



# Обучение: minimax game

---

- ❖ Генератор G пытается сделать ❖ Дискриминатор D пытается сделать  
 $D(G(z)) = 1$                                     $D(G(z)) = 0$

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

# Обучение: minimax game

---

- ❖ Генератор G пытается сделать ❖ Дискриминатор D пытается сделать  
 $D(G(z)) = 1$                                     $D(G(z)) = 0$

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

- ❖ Учим дискриминатор правильно отличать примеры из датасета и из G(z)

# Обучение: minimax game

---

- ❖ Генератор G пытается сделать ❖ Дискриминатор D пытается сделать  
 $D(G(z)) = 1$   $D(G(z)) = 0$

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

- ❖ Учим дискриминатор правильно отличать примеры из датасета и из G(z)
- ❖ Дискриминатор максимизирует выражение V(D, G)

# Обучение: minimax game

---

- ❖ Генератор G пытается сделать ❖ Дискриминатор D пытается сделать  
 $D(G(z)) = 1$   $D(G(z)) = 0$

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

- ❖ Учим дискриминатор правильно отличать примеры из датасета и из  $G(z)$
- ❖ Дискриминатор максимизирует выражение  $V(D, G)$
- ❖ Учим генератор обманывать дискриминатор. Генератор минимизирует  $\log(1 - D(G(z)))$

# Обучение: minimax game

- ❖ Генератор G пытается сделать  $D(G(z)) = 1$
- ❖ Дискриминатор D пытается сделать  $D(G(z)) = 0$

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

- ❖ Учим дискриминатор правильно отличать примеры из датасета и из  $G(z)$
- ❖ Дискриминатор максимизирует выражение  $V(D, G)$
- ❖ Учим генератор обманывать дискриминатор. Генератор минимизирует  $\log(1 - D(G(z)))$



# Обучение : minimax game

---

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

- ❖ Обучение проходит в два шага
- ❖ Обучение дискриминатора:
  - сэмплируем настоящие картинки, считаем ошибку
  - генерируем сэмплы, считаем ошибку
  - обновляем веса дискриминатора



# Обучение : minimax game

---

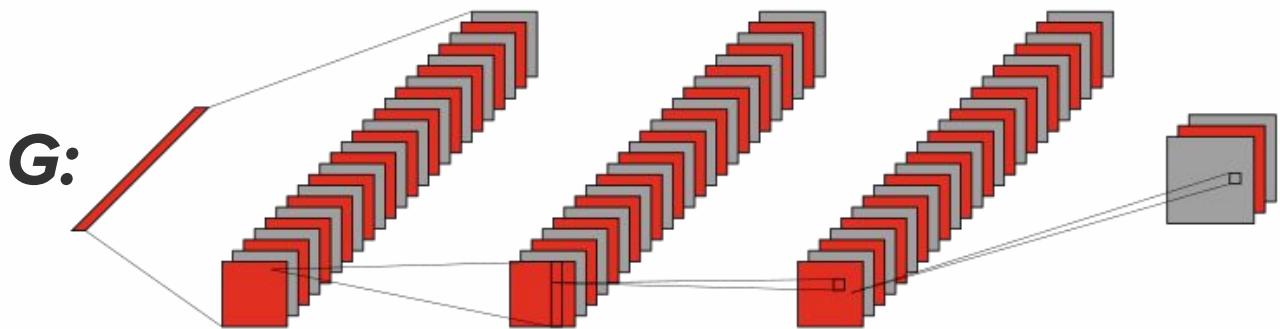
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

- ❖ Обучение проходит в два шага
- ❖ Обучение дискриминатора:
  - сэмплируем настоящие картинки, считаем ошибку
  - генерируем сэмплы, считаем ошибку
  - обновляем веса дискриминатора
- ❖ Обучение генератора:
  - генерируем сэмплы
  - считаем ошибку дискриминатора
  - прорасываем градиенты в генератор
  - обновляем веса генератора

# Обучение генератора: minimax game

---

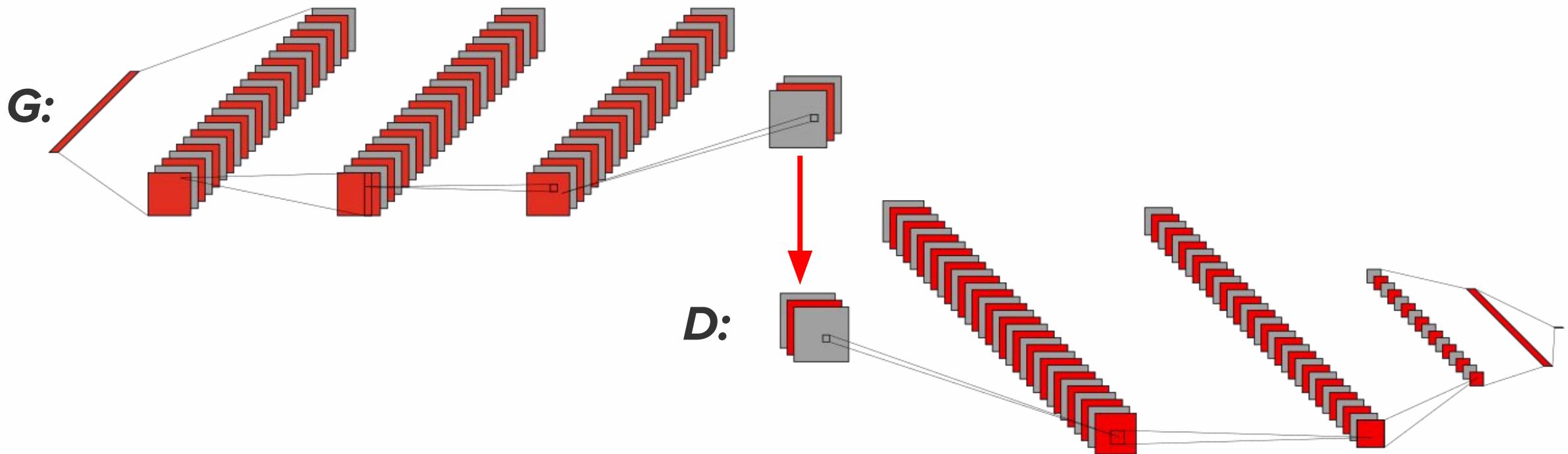
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$



# Обучение генератора: minimax game

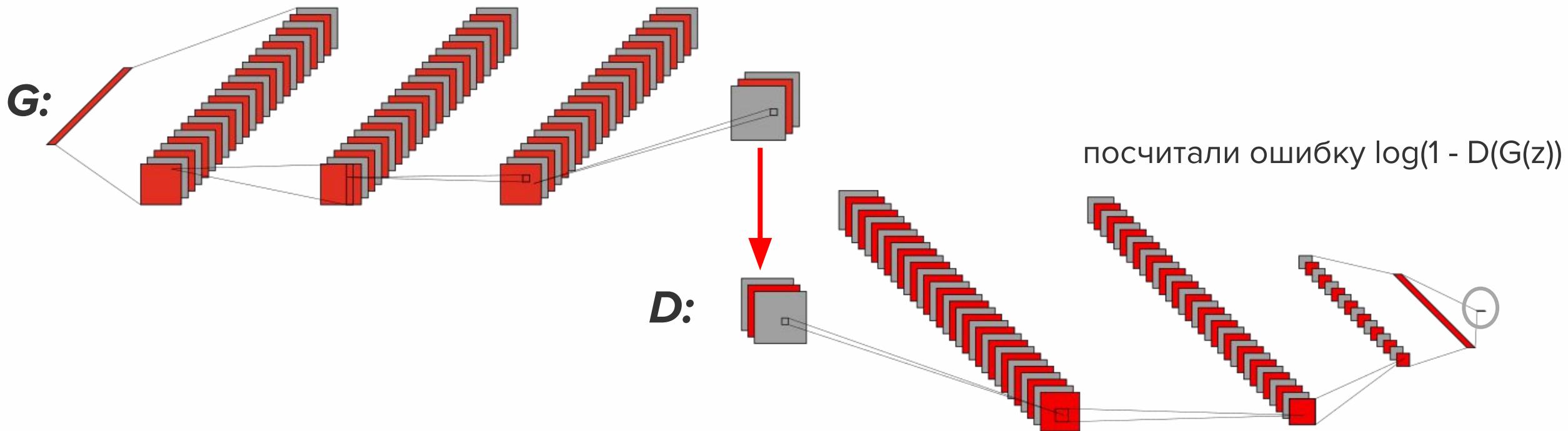
---

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



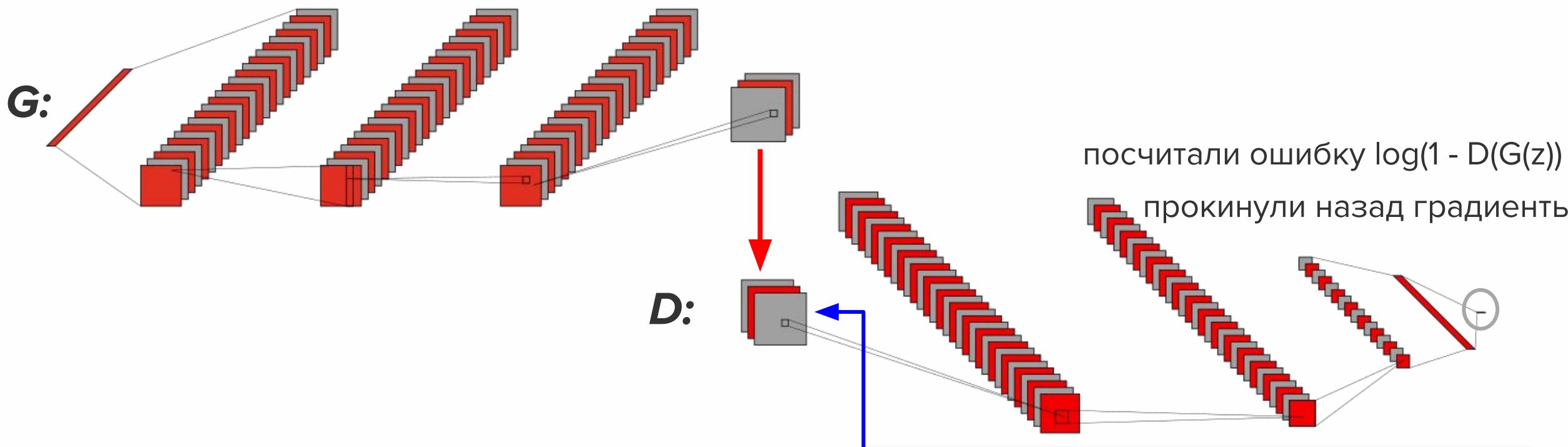
# Обучение генератора: minimax game

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



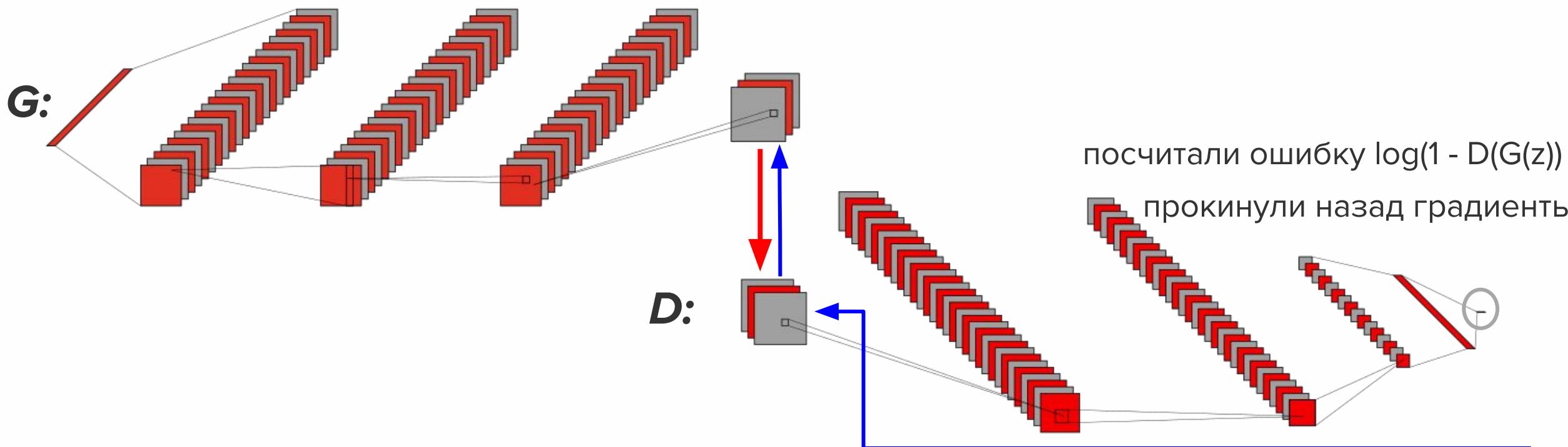
# Обучение генератора: minimax game

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



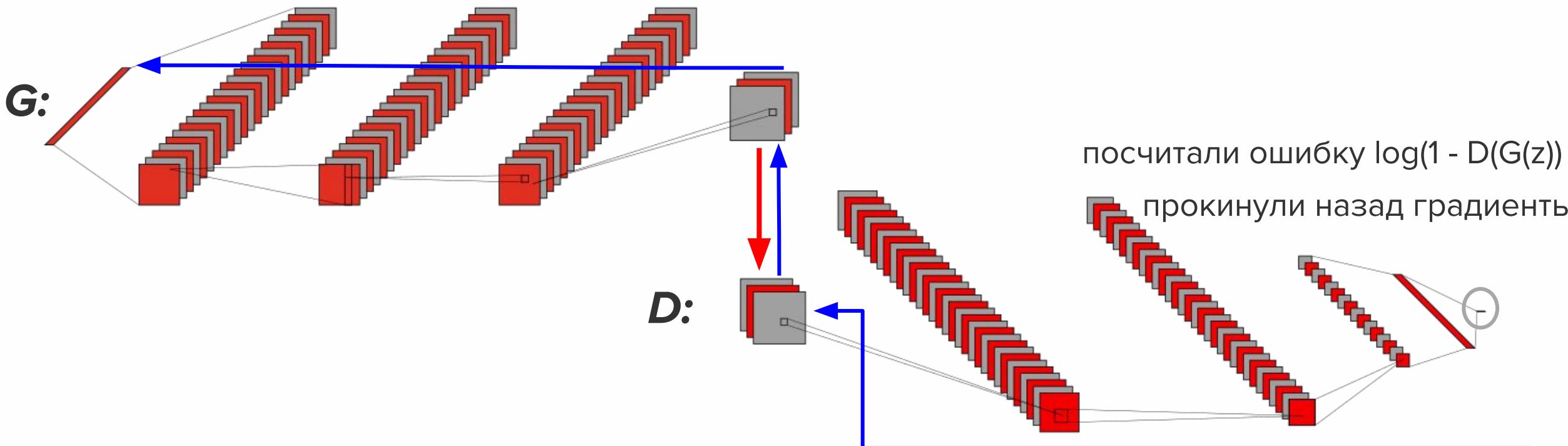
# Обучение генератора: minimax game

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



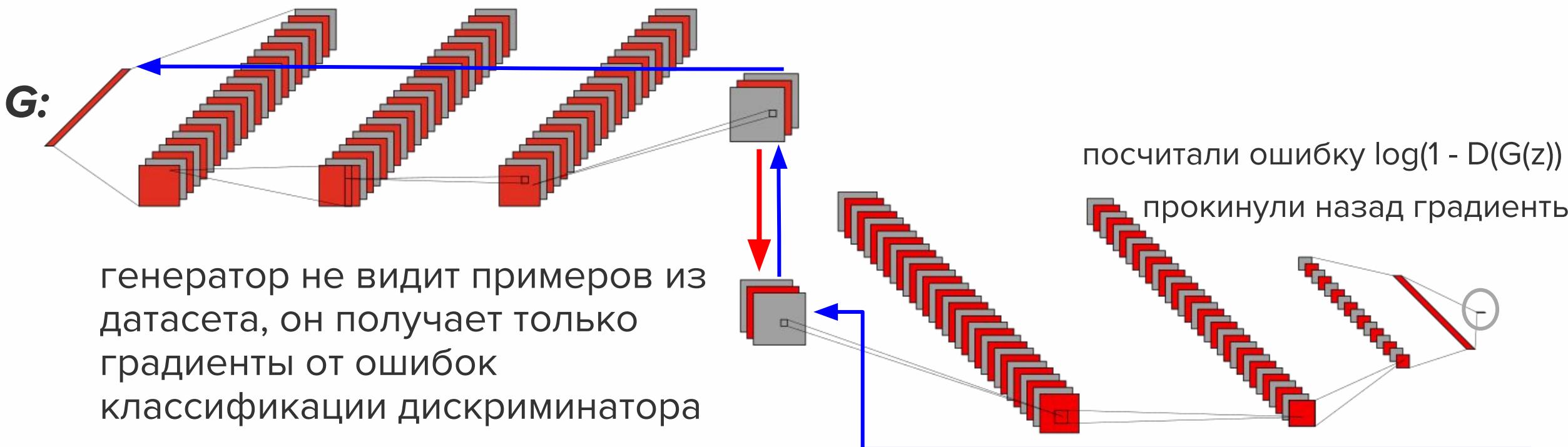
# Обучение генератора: minimax game

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



# Обучение генератора: minimax game

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



# Обучение: minimax game. Затухание

- ❖ Генератор G пытается сделать  $D(G(z)) = 1$
- ❖ Дискриминатор D пытается сделать  $D(G(z)) = 0$

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

- ❖ Учим дискриминатор правильно отличать примеры из датасета и из  $G(z)$
- ❖ Дискриминатор максимизирует выражение  $V(D, G)$
- ❖ Учим генератор обманывать дискриминатор. Генератор минимизирует  $\log(1 - D(G(z)))$

# Обучение: minimax game.

- ❖ Генератор G пытается сделать ❖ Дискриминатор D пытается сделать  
 $D(G(z)) = 1$   $D(G(z)) = 0$

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

- ❖ Учим дискриминатор правильно отличать примеры из датасета и из  $G(z)$
- ❖ Дискриминатор максимизирует выражение  $V(D, G)$
- ❖ Учим генератор обманывать дискриминатор. Генератор минимизирует  $\log(1 - D(G(z)))$
- ❖ **Есть проблема.**

# Обучение: minimax game. Затухание градиентов

- ❖ Генератор G пытается сделать  $D(G(z)) = 1$
- ❖ Дискриминатор D пытается сделать  $D(G(z)) = 0$

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

- ❖ Учим дискриминатор правильно отличать примеры из датасета и из  $G(z)$
- ❖ Дискриминатор максимизирует выражение  $V(D, G)$
- ❖ Учим генератор обманывать дискриминатор. Генератор минимизирует  $\log(1 - D(G(z)))$
- ❖ **Есть проблема:** если  $D$  уже хорошо обучился, а  $G$  всё еще генерирует плохие картинки, слагаемое  $\log(1 - D(G(z)))$  начинает давать очень маленькие градиенты.



# Обучение: minimax game. Решение проблемы

---

**Было:**

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



# Обучение: minimax game. Решение проблемы

---

**Было:**

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

**Стало:**

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

**G:**  $\max_G \mathbb{E}_{z \sim p_z(z)} \log(D(G(z)))$

# Обучение: minimax game. Решение проблемы

**Было:**

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

**Стало:**

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

$$G: \max_G \mathbb{E}_{z \sim p_z(z)} \log(D(G(z)))$$



# Обучение: minimax game. Оптимальный D

---

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



# Обучение: minimax game. Оптимальный D

---

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

$$\frac{\partial V(D, G)}{\partial D(x)} =$$

# Обучение: minimax game. Оптимальный D

---

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

$$\frac{\partial V(D, G)}{\partial D(x)} = \frac{\partial [p_{\text{data}}(x) \log D(x) + p_{\text{gen}}(x) \log(1 - D(x))]}{\partial D(x)} =$$

# Обучение: minimax game. Оптимальный D

---

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

$$\frac{\partial V(D, G)}{\partial D(x)} = \frac{\partial [p_{\text{data}}(x) \log D(x) + p_{\text{gen}}(x) \log(1 - D(x))]}{\partial D(x)} =$$

$$\frac{p_{\text{data}}(x)}{D(x)} - \frac{p_{\text{gen}}(x)}{1 - D(x)} = 0$$

# Обучение: minimax game. Оптимальный D

---

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

$$\frac{\partial V(D, G)}{\partial D(x)} = \frac{\partial [p_{\text{data}}(x) \log D(x) + p_{\text{gen}}(x) \log(1 - D(x))]}{\partial D(x)} =$$

$$\frac{p_{\text{data}}(x)}{D(x)} - \frac{p_{\text{gen}}(x)}{1 - D(x)} = 0$$

$$p_{\text{data}}(x)(1 - D(x)) = p_{\text{gen}}(x)D(x)$$

# Обучение: minimax game. Оптимальный D

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

$$\frac{\partial V(D, G)}{\partial D(x)} = \frac{\partial [p_{\text{data}}(x) \log D(x) + p_{\text{gen}}(x) \log(1 - D(x))]}{\partial D(x)} =$$

$$\frac{p_{\text{data}}(x)}{D(x)} - \frac{p_{\text{gen}}(x)}{1 - D(x)} = 0$$

$$p_{\text{data}}(x)(1 - D(x)) = p_{\text{gen}}(x)D(x)$$

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)}$$

# Обучение: minimax game. Оптимальный D

---

- ❖ Если дискриминатор идеально емкий (может описать любую функцию), он в каждой точке пространства будет предсказывать отношение истинного вероятностного распределения к сумме истинного и генерируемого распределений.

$$D(x) = \frac{p_{data}(x)}{p_{data}(x) + p_{gen}(x)}$$

# Обучение: minimax game. Оптимальный D

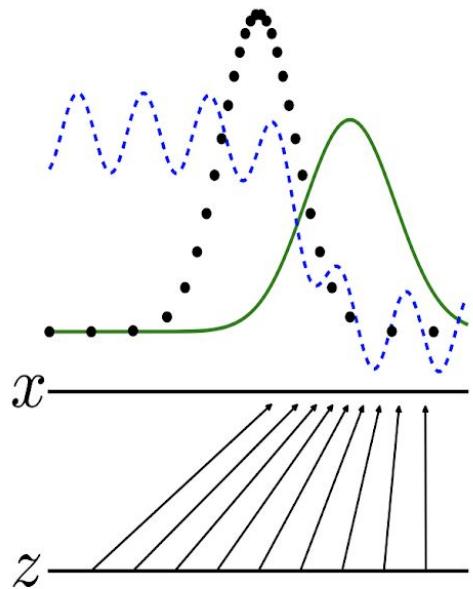
---

- ❖ Если дискриминатор идеально емкий (может описать любую функцию), он в каждой точке пространства будет предсказывать отношение истинного вероятностного распределения к сумме истинного и генерируемого распределений.

$$D(x) = \frac{p_{data}(x)}{p_{data}(x) + p_{gen}(x)} = \frac{1}{2}$$

это происходит, когда  $p_{data} = p_{gen}$

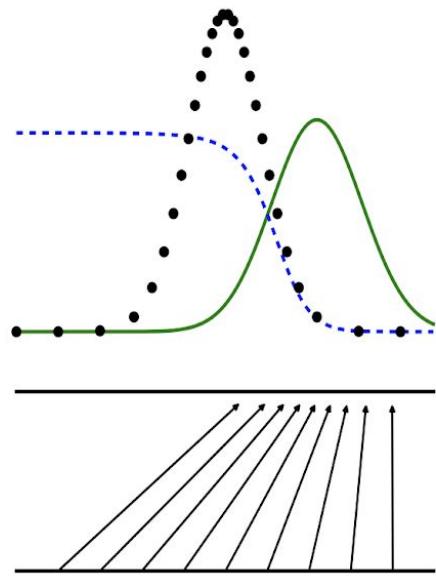
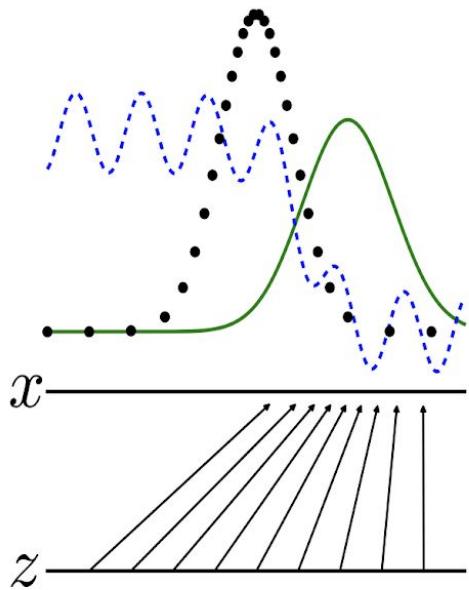
# Обучение: minimax game.



(a)

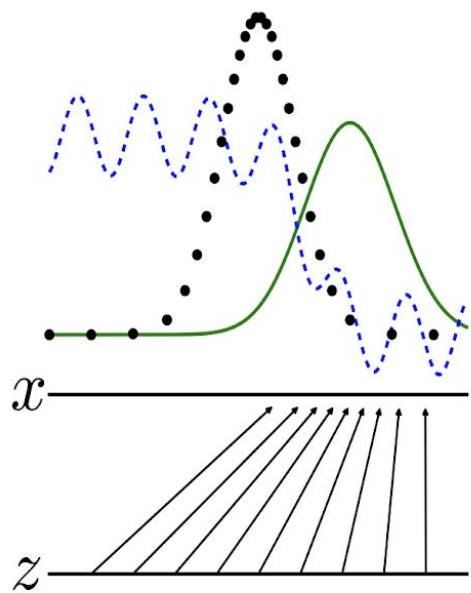
- · · · · распределение данных (целевое распределение)
- — — распределение ответов дискриминатора (дискриминативное распределение)
- распределение ответов генератора (генерируемое распределение)

# Обучение: minimax game.

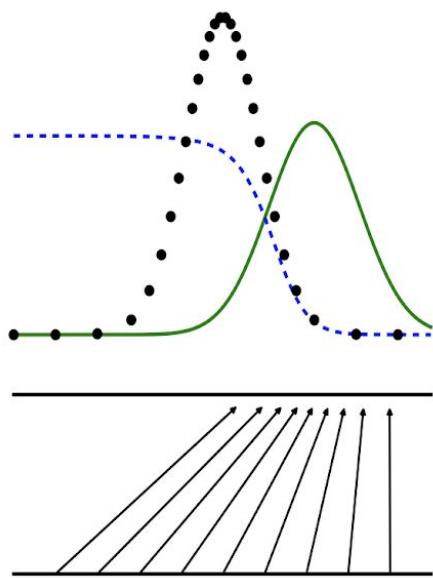


- · · · · распределение данных (целевое распределение)
- — — распределение ответов дискриминатора (дискриминативное распределение)
- распределение ответов генератора (генерируемое распределение)

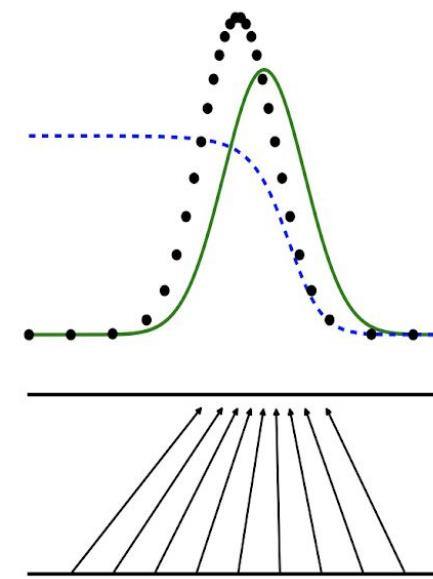
# Обучение: minimax game.



(a)



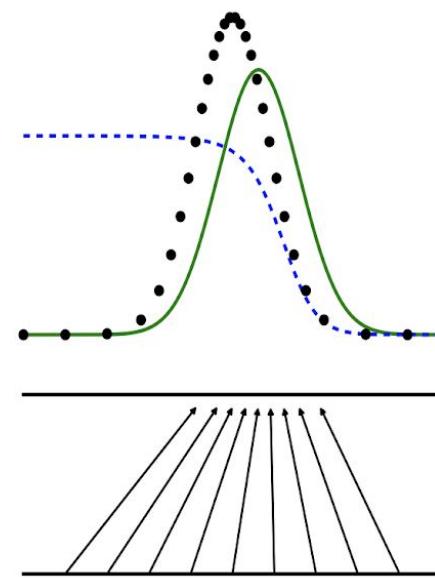
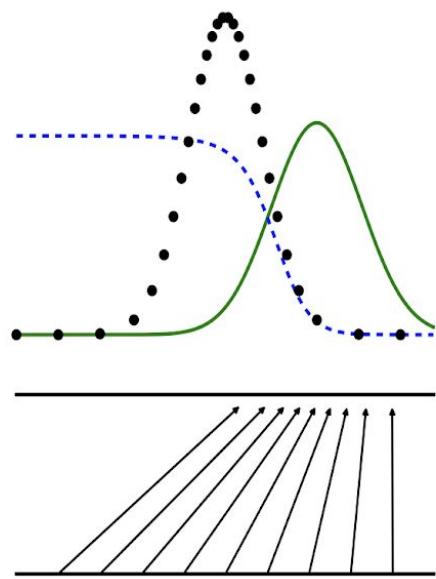
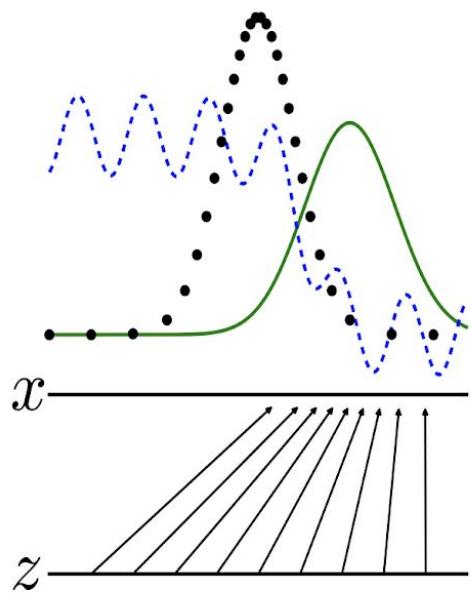
(b)



(c)

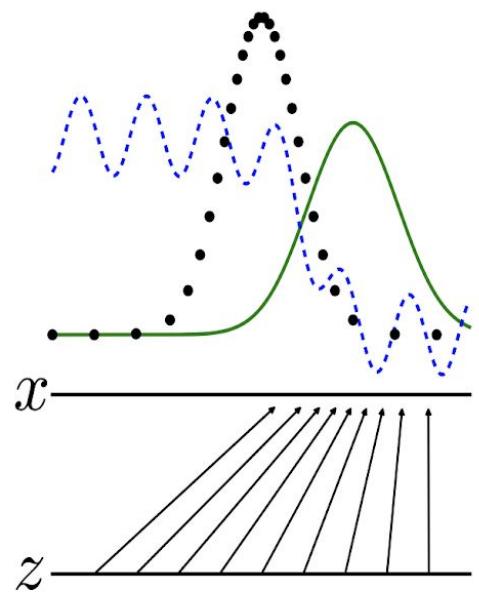
- · · · · распределение данных (целевое распределение)
- — — распределение ответов дискриминатора (дискриминативное распределение)
- распределение ответов генератора (генерируемое распределение)

# Обучение: minimax game.

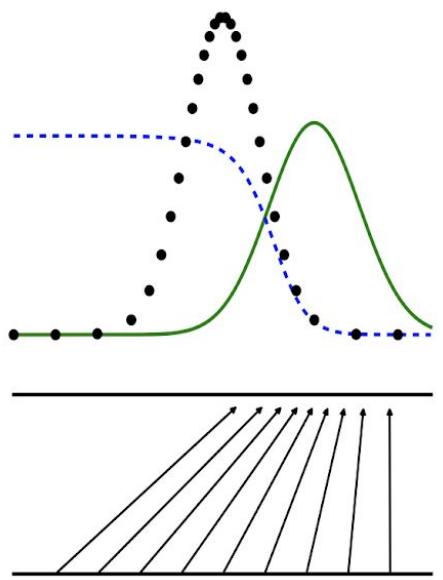


- · · · · распределение данных (целевое распределение)
- — — распределение ответов дискриминатора (дискриминативное распределение)
- распределение ответов генератора (генерируемое распределение)

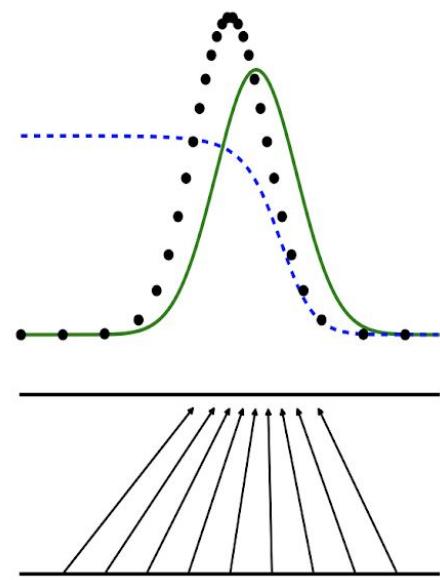
# Обучение: minimax game.



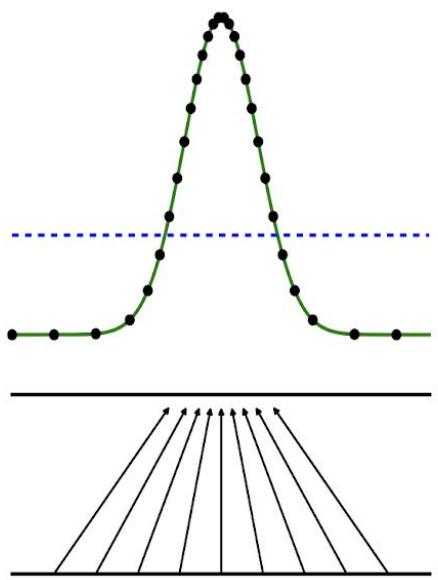
(a)



(b)



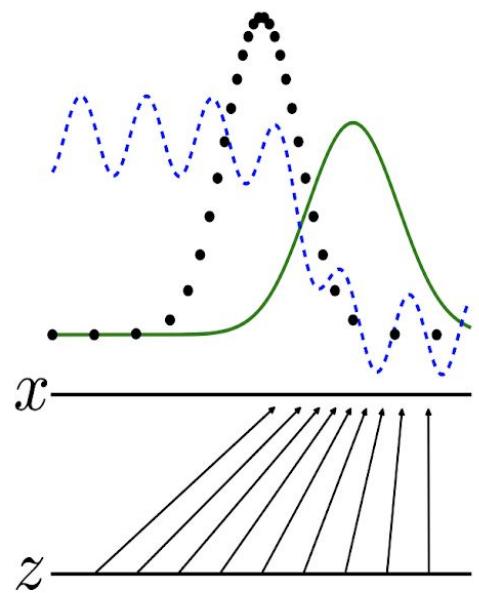
(c)



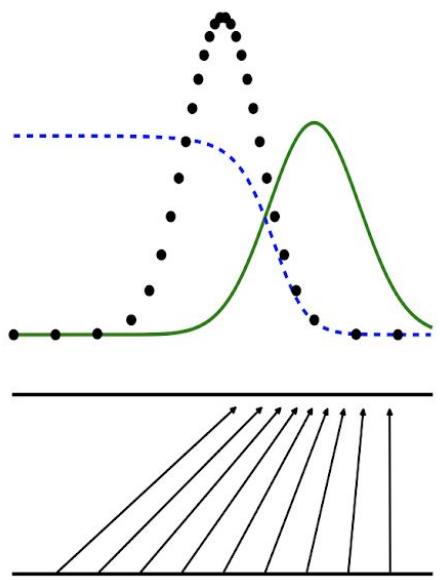
(d)

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$

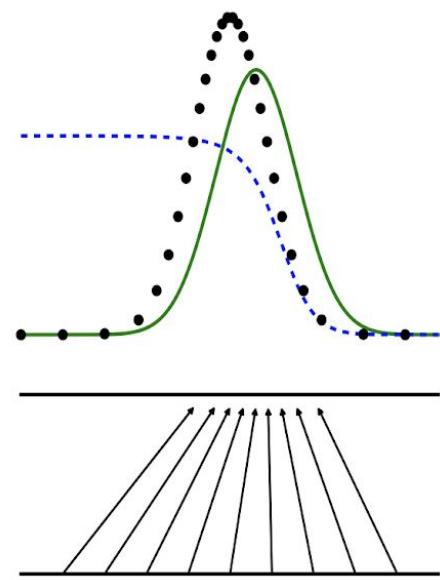
# Обучение: minimax game.



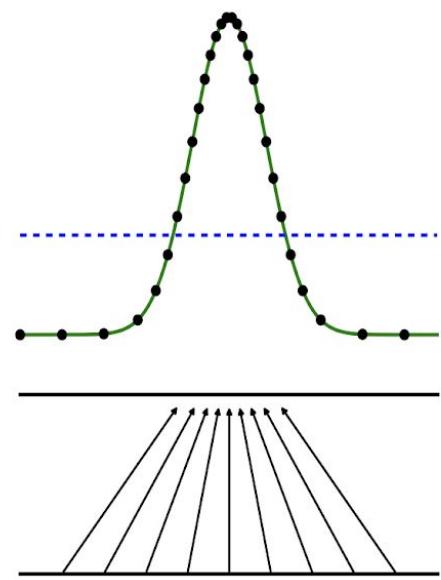
(a)



(b)



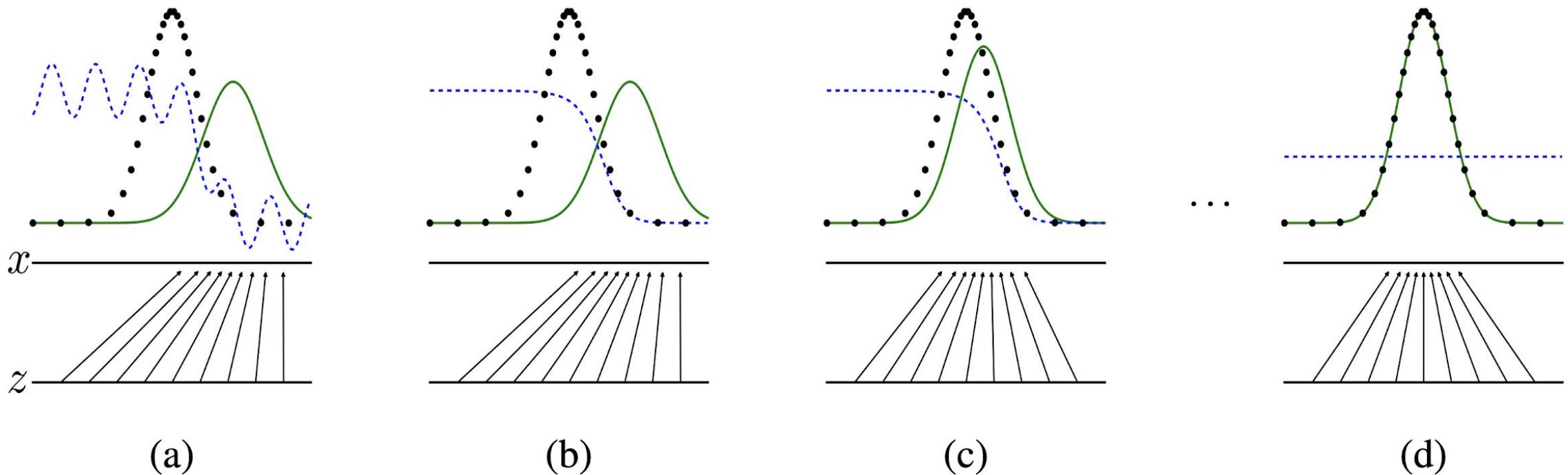
(c)



(d)

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)} = \frac{1}{2}$$

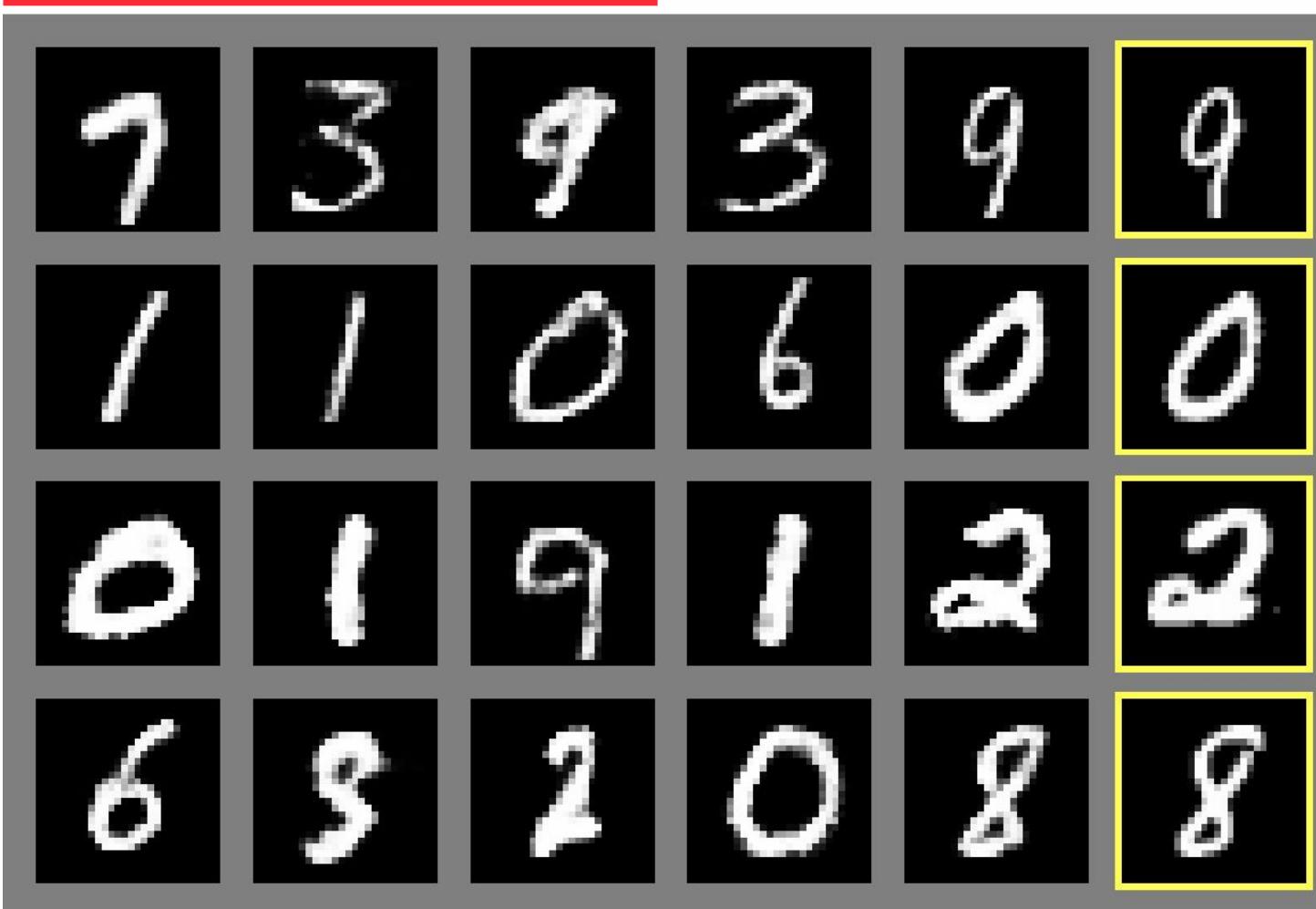
# Обучение: minimax game.



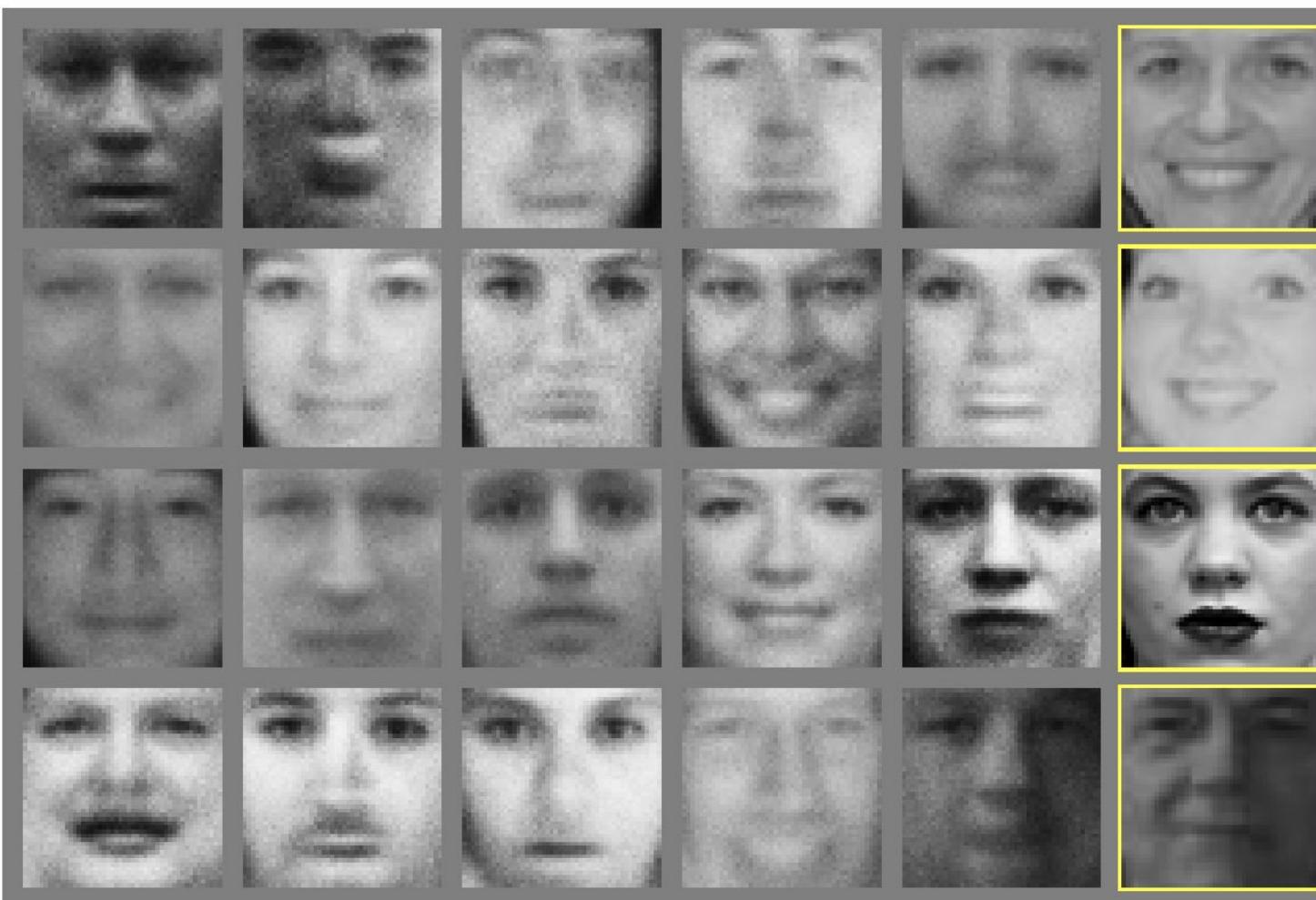
$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)} = \frac{1}{2}$$

**Оценка этого отношения с помощью обучения с учителем - главный механизм аппроксимации, который помогает генеративным сетям работать**

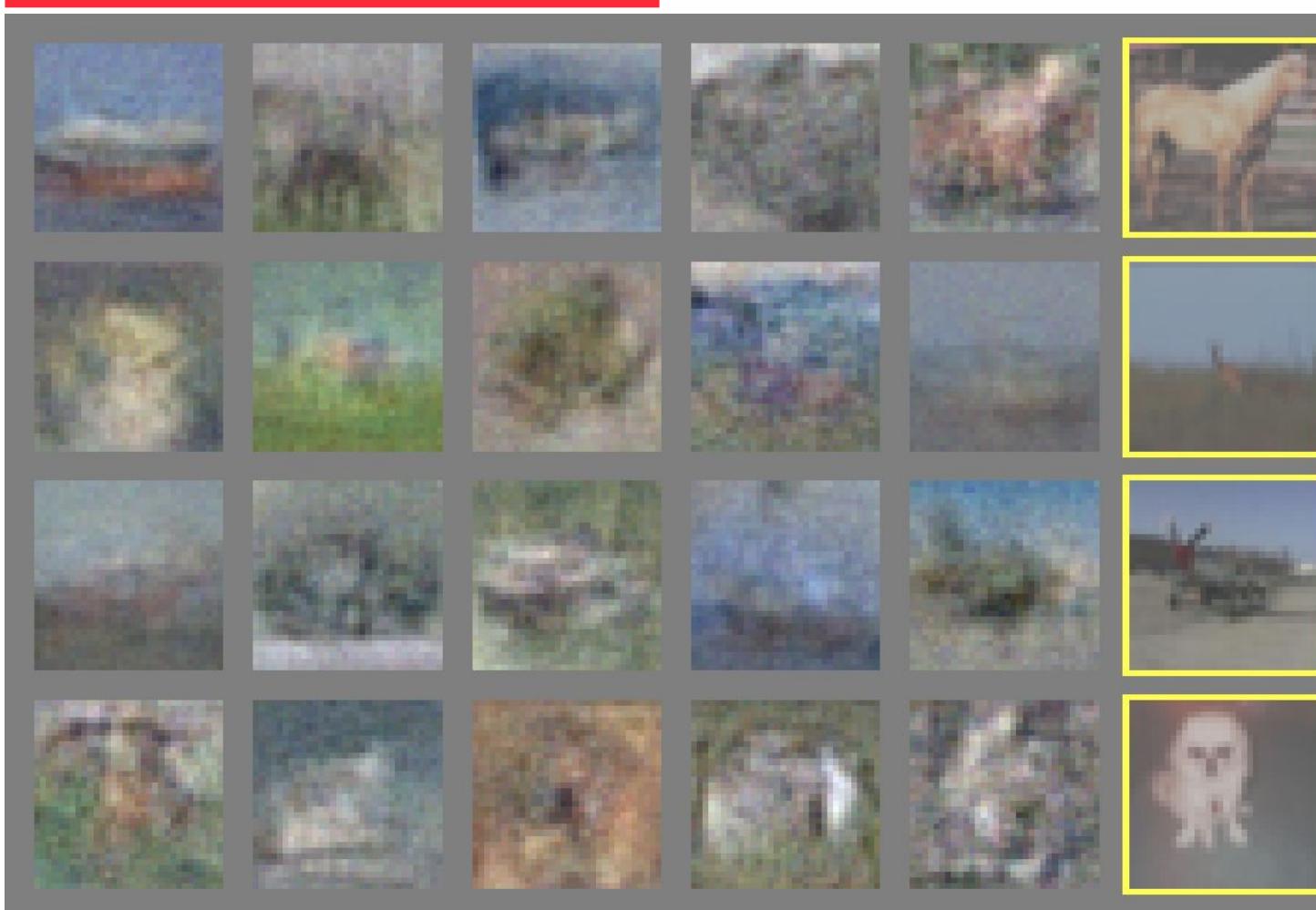
# Оригинальная статья. Примеры работы



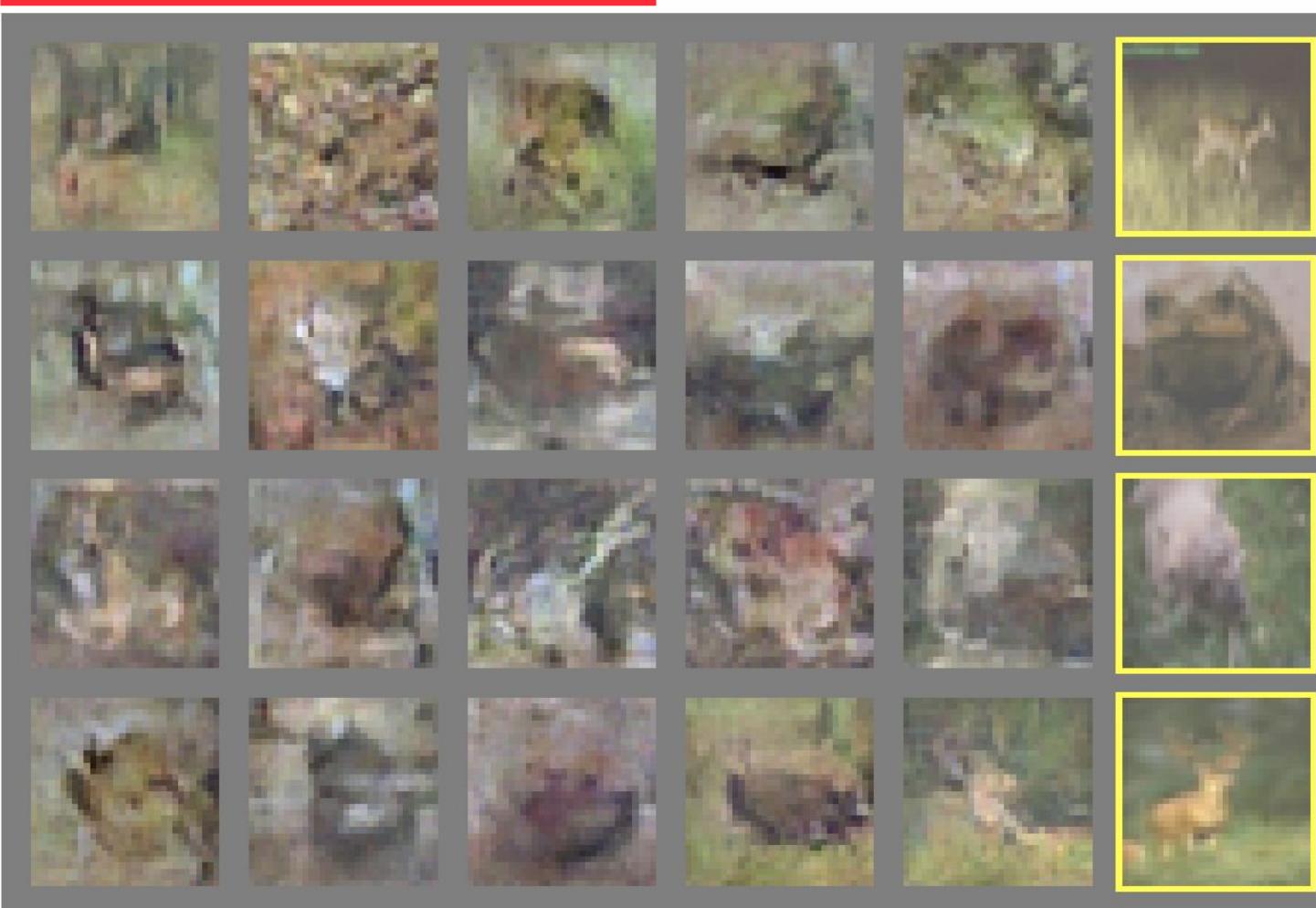
# Оригинальная статья. Примеры работы



# Оригинальная статья. Примеры работы



# Оригинальная статья. Примеры работы



# DCGAN. Усовершенствование оригинальной статьи

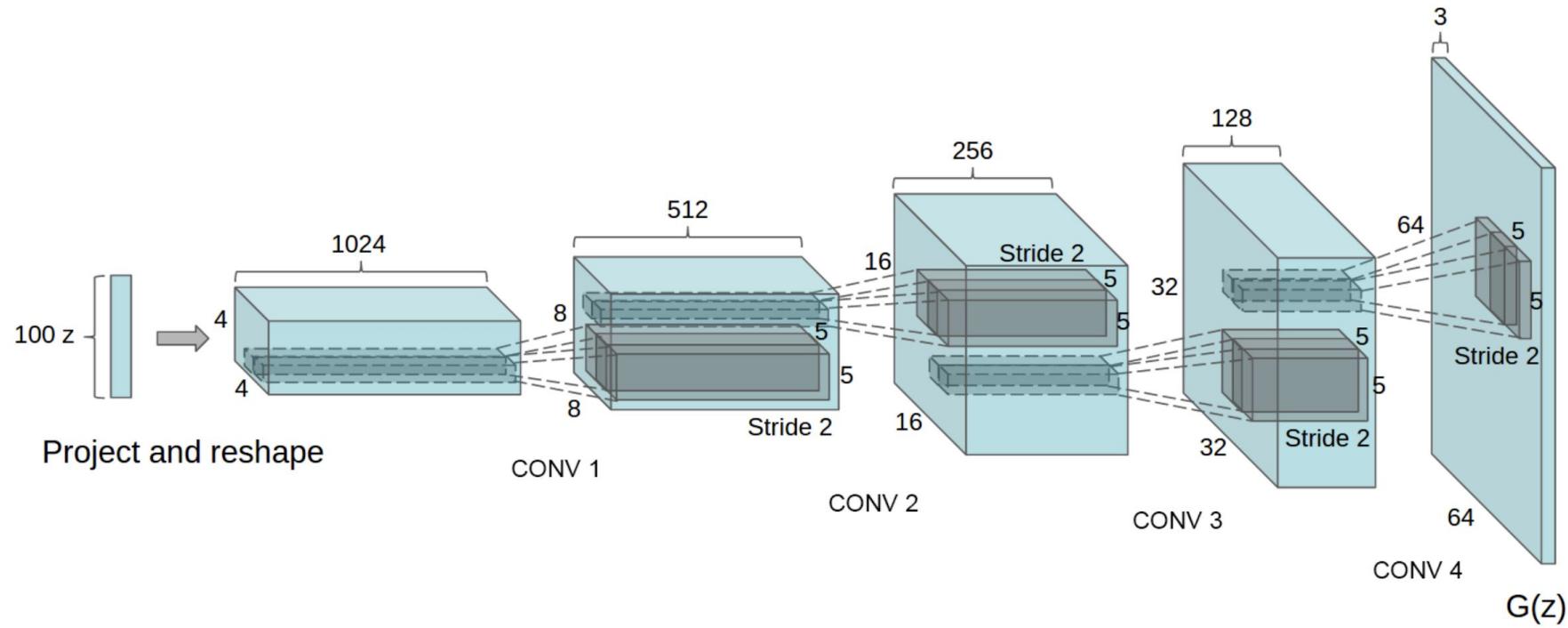


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution  $Z$  is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions)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

# DCGAN. Результаты работы на датасете LSUN



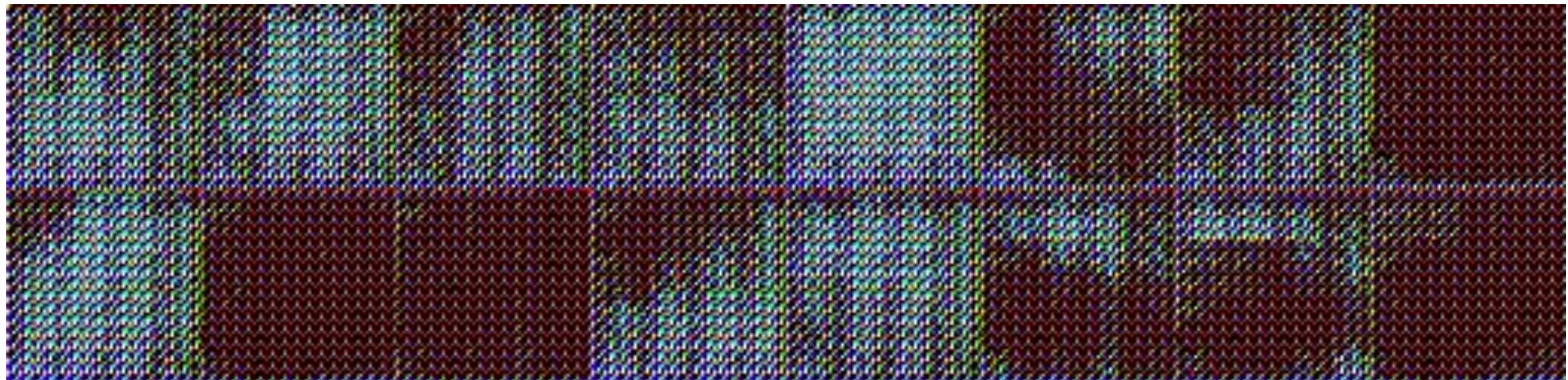
# DCGAN. Результаты работы на датасете LSUN



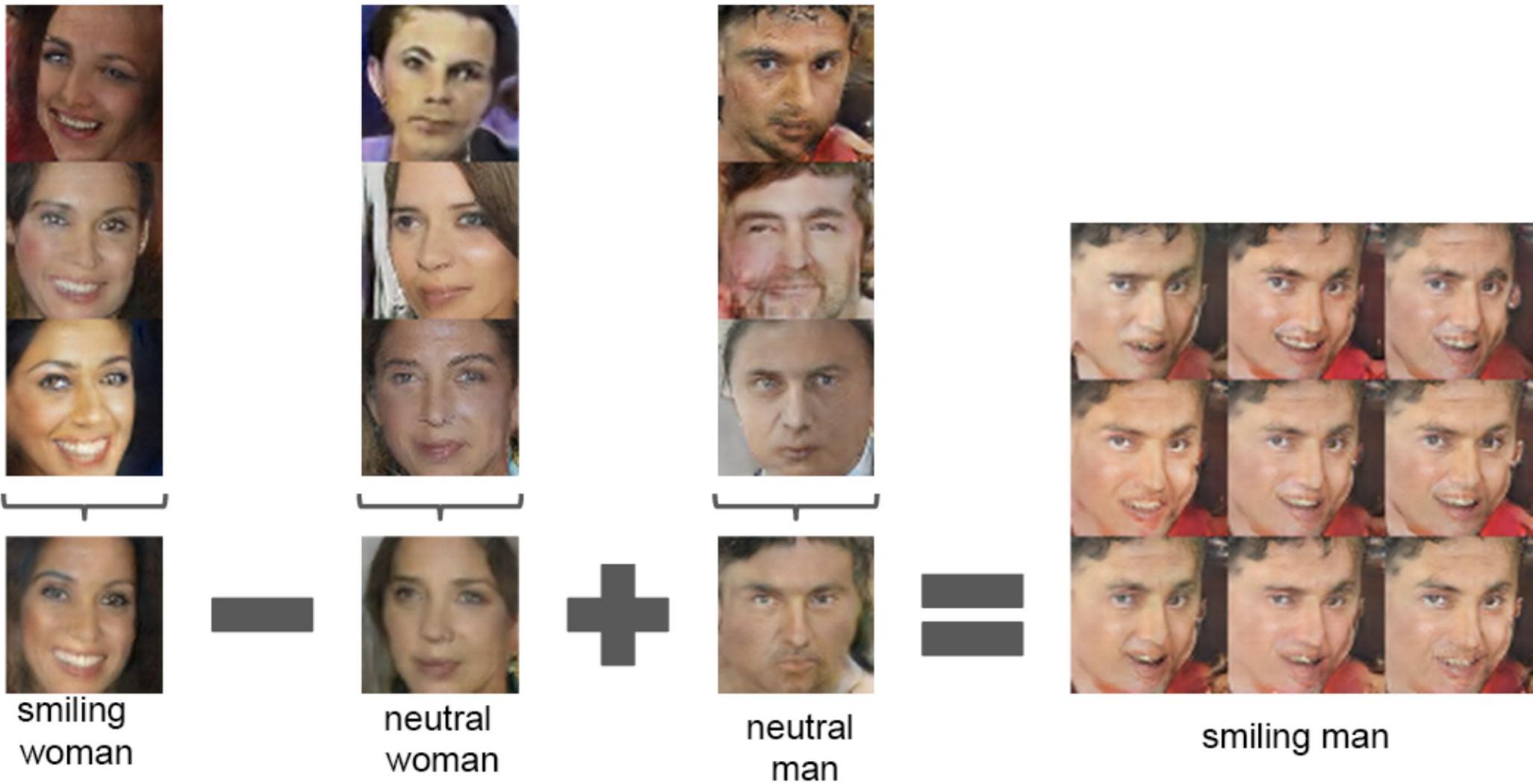


# DCGAN. Результаты работы без BatchNorm

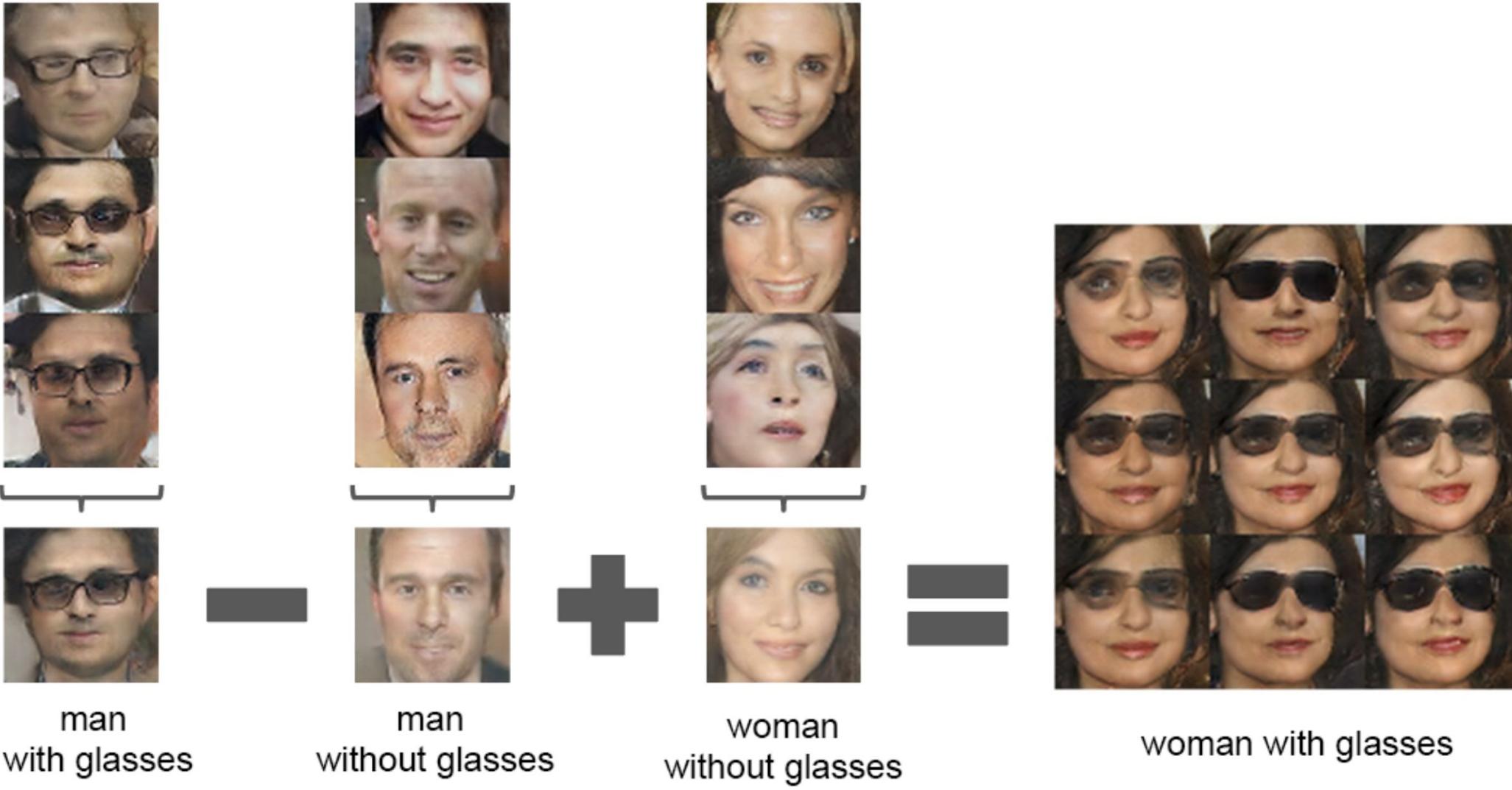
---



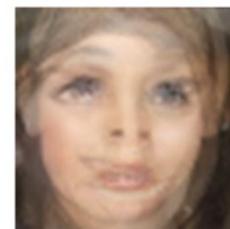
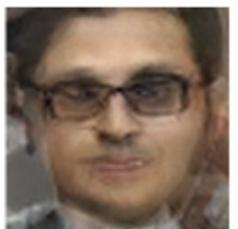
# DCGAN. Векторная арифметика



# DCGAN. Векторная арифметика

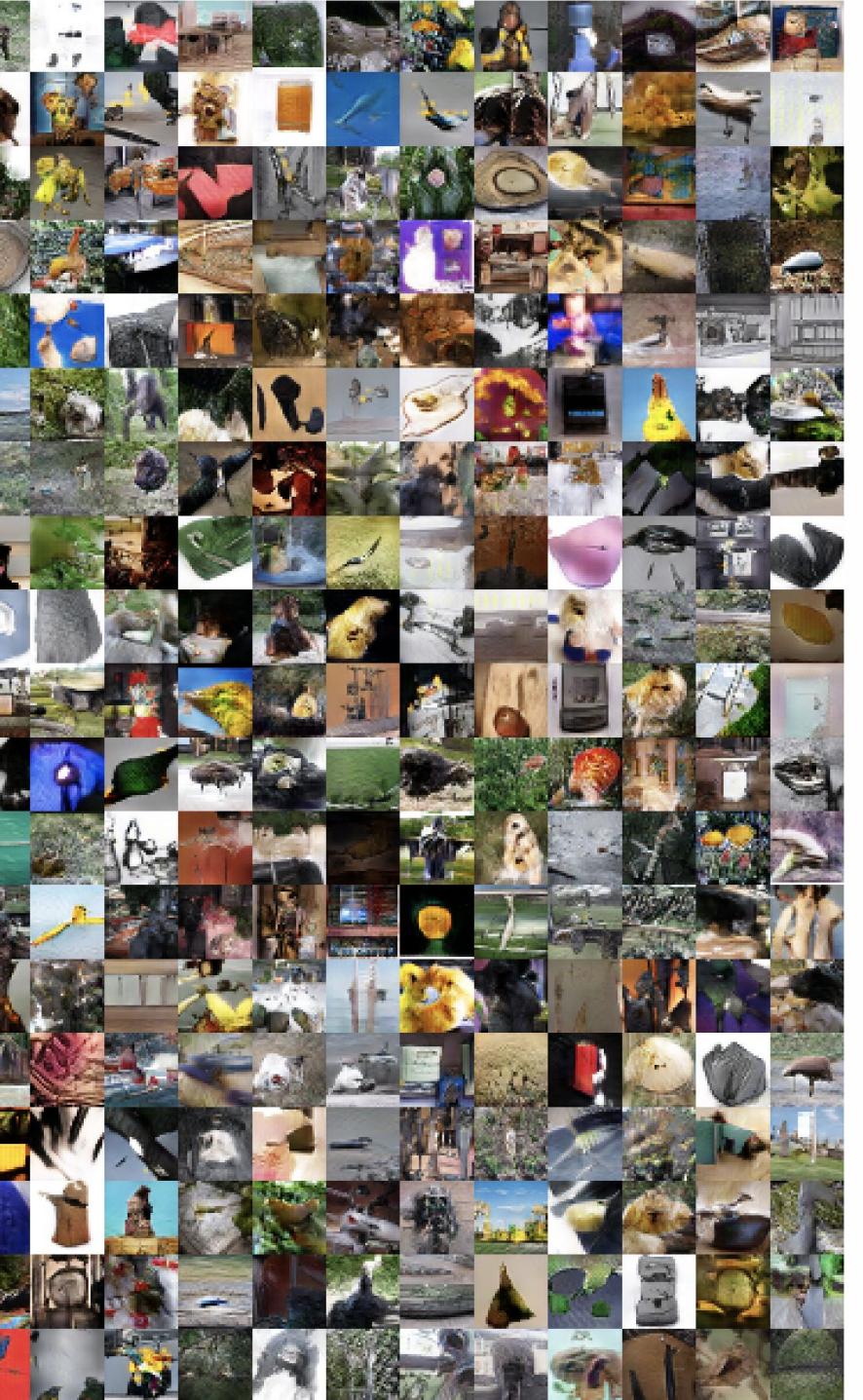


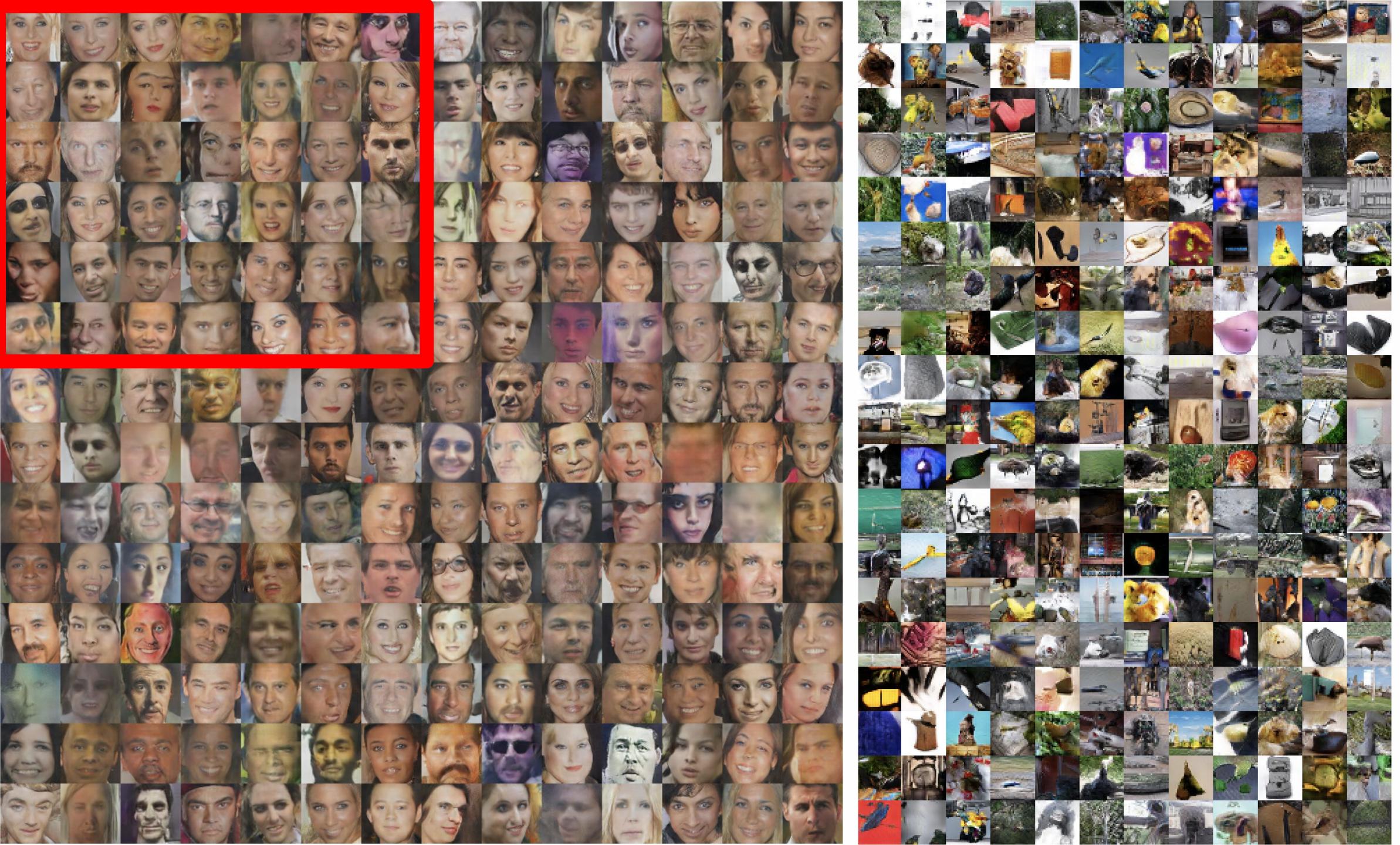
# DCGAN. Пиксельная арифметика

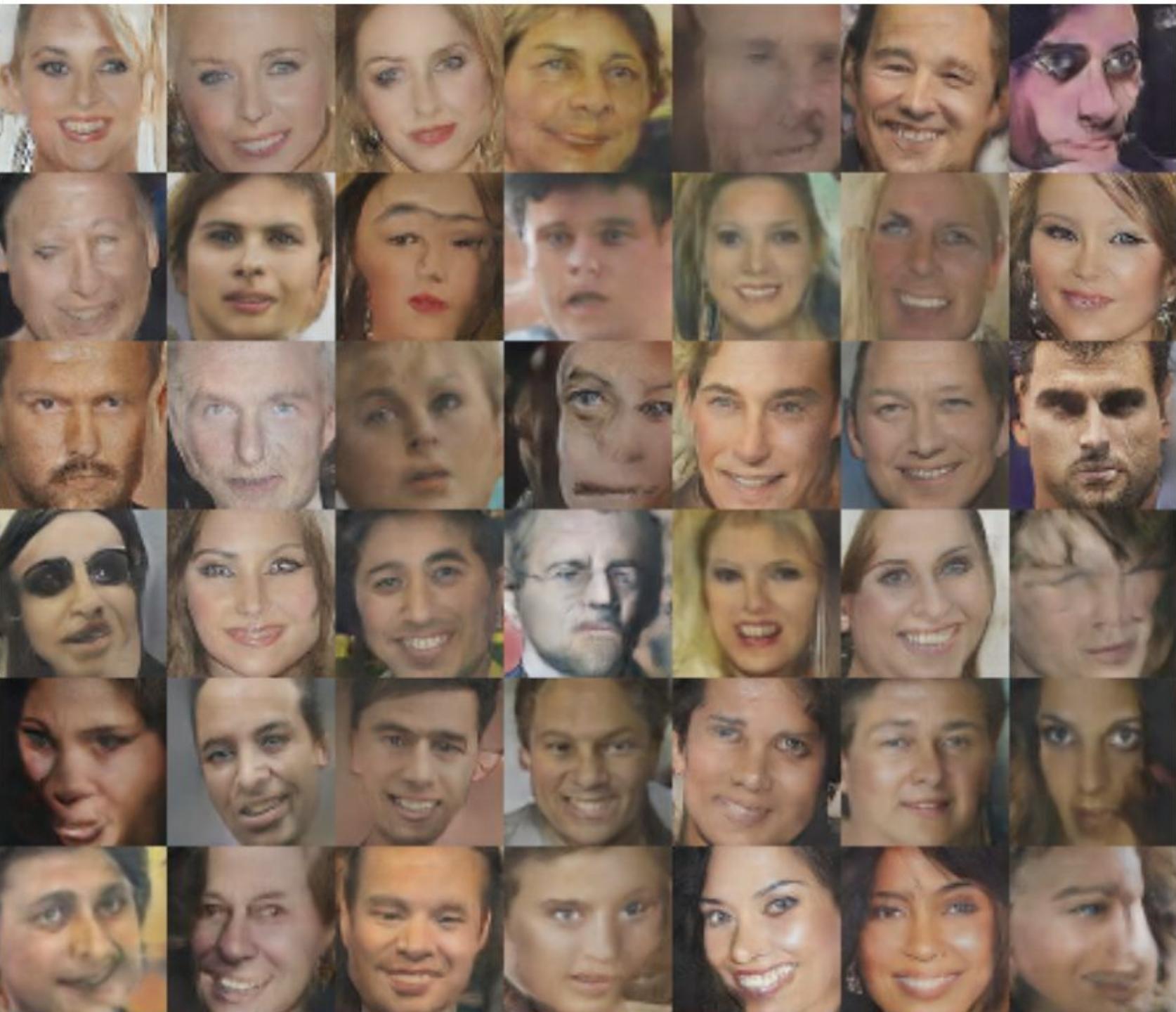


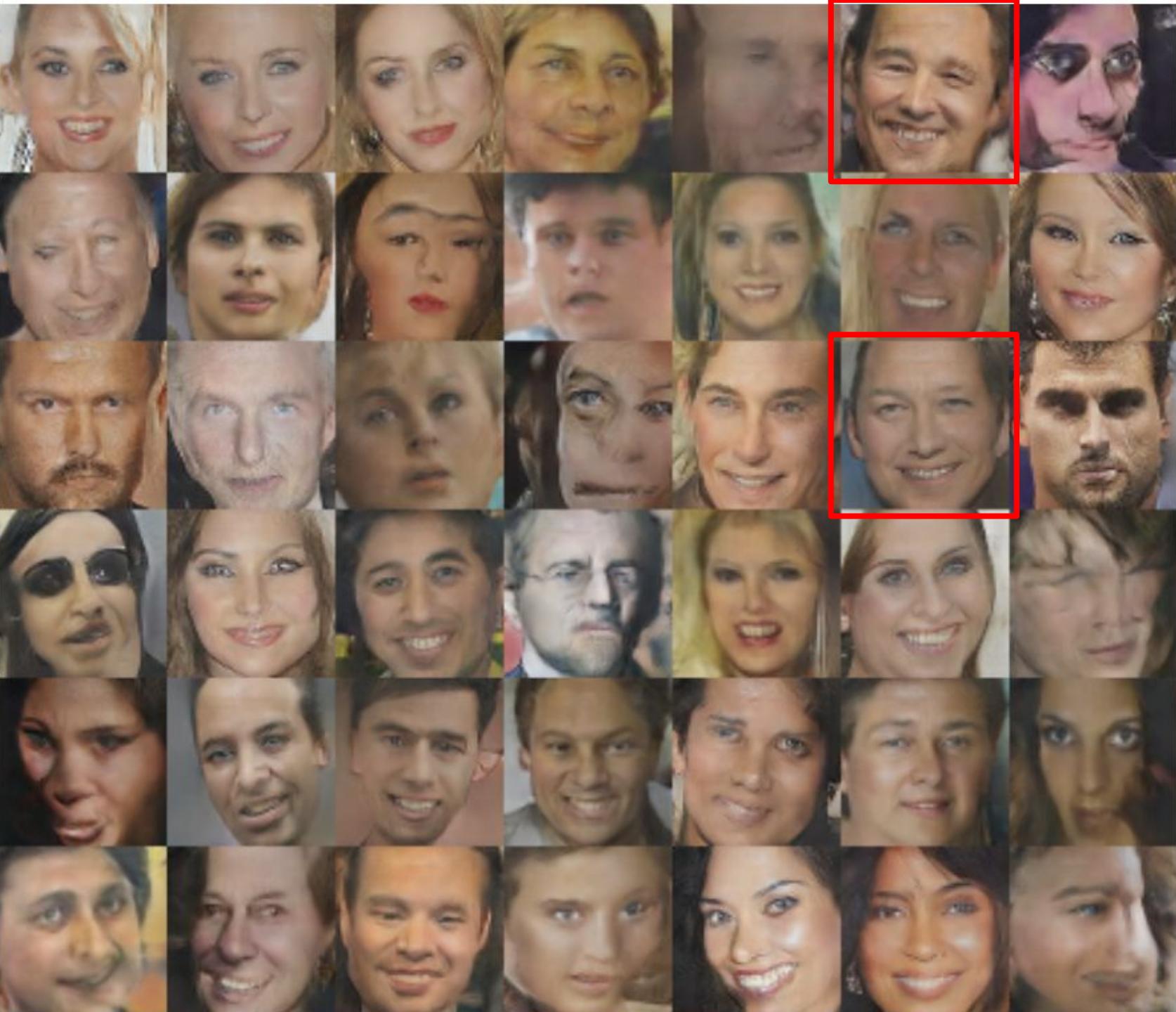
Results of doing the same  
arithmetic in pixel space

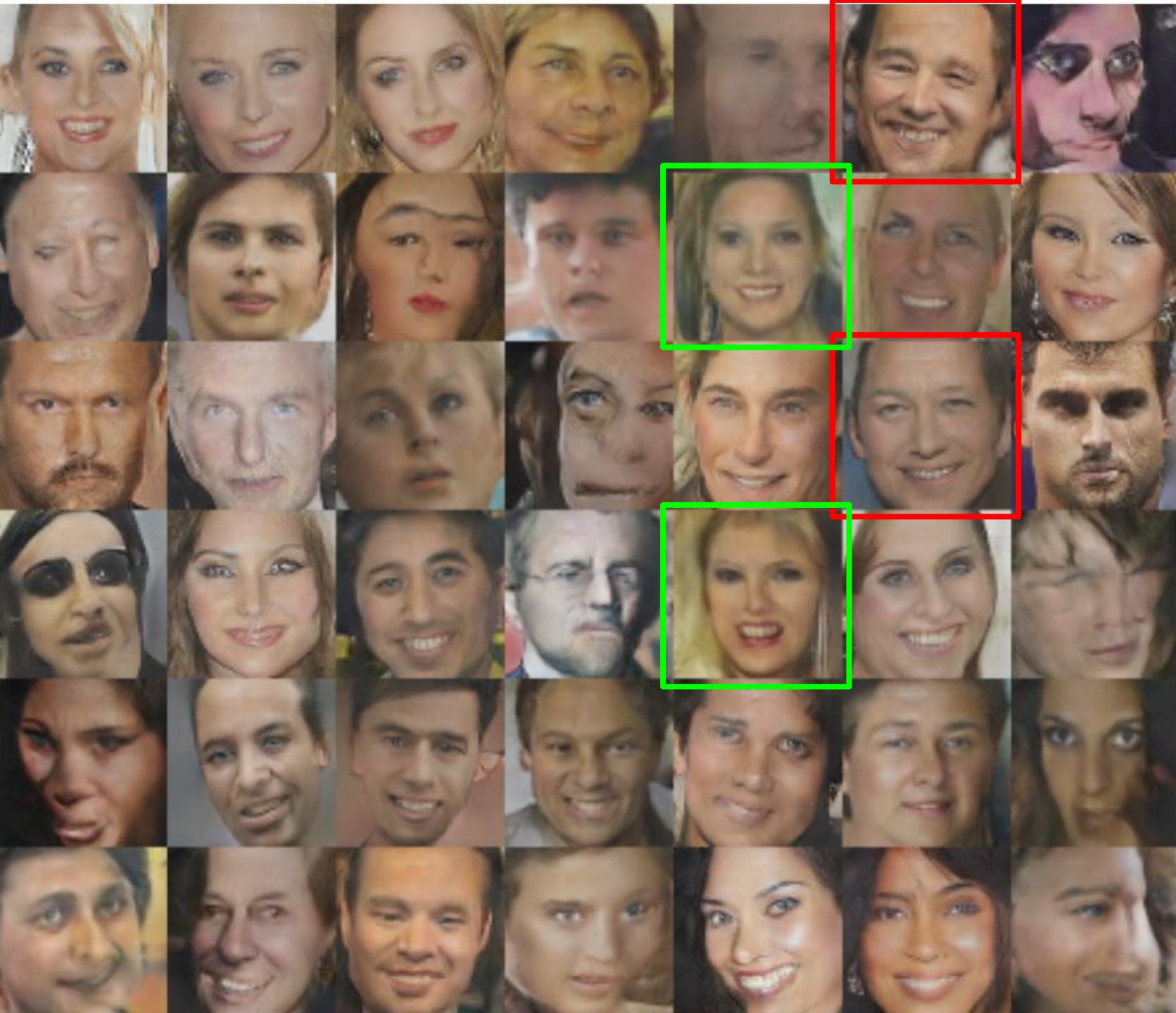














DCGAN

Published as a conference paper at ICLR 2018

---

# DO GANs LEARN THE DISTRIBUTION? SOME THEORY AND EMPIRICS

**Sanjeev Arora**

Department of Computer Science  
Princeton University  
Princeton, NJ 08544, USA  
[arora@cs.princeton.edu](mailto:arora@cs.princeton.edu)

**Andrej Risteski**

Applied Mathematics Department and IDSS  
Massachusetts Institute of Technology  
Cambridge, MA 02139, USA  
[risteski@mit.edu](mailto:risteski@mit.edu)

**Yi Zhang**

Department of Computer Science  
Princeton University  
Princeton, NJ 08544, USA  
[y.zhang@cs.princeton.edu](mailto:y.zhang@cs.princeton.edu)

<https://arxiv.org/abs/1706.08224>



# Проблема идеального генератора

---

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

$$G: \max_G \mathbb{E}_{z \sim p_z(z)} \log(D(G(z)))$$



# Проблема идеального генератора

---

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

**G:**  $\max_G \mathbb{E}_{z \sim p_z(z)} \log(D(G(z)))$

Если дискриминатор зафиксирован (не учится), то что нужно делать генератору, чтобы всегда обманывать дискриминатор?

$$G(z) = \operatorname{argmax}_x D(x)$$



# Проблема идеального генератора

---

- ❖ Если дискриминатор остановлен и не учится, оптимальный генератор будет для любого входа возвращать такую картинку, для которой дискриминатор возвращает максимальную вероятность.

$$G(z) = \operatorname{argmax}_x D(x)$$

# Проблема идеального генератора

---

- ❖ Если дискриминатор остановлен и не учится, оптимальный генератор будет для любого входа возвращать такую картинку, для которой дискриминатор возвращает максимальную вероятность.

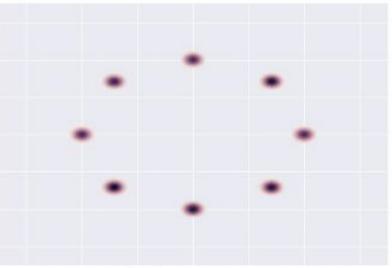
$$G(z) = \operatorname{argmax}_x D(x)$$

- ❖ Проблема в том, что мы никак не заставляет генератор вести себя по-другому, поэтому генератор может для какой-то части латентного пространства возвращать очень похожие выходы. Такая проблема называется **Mode Collapse**



# Mode collapse

---

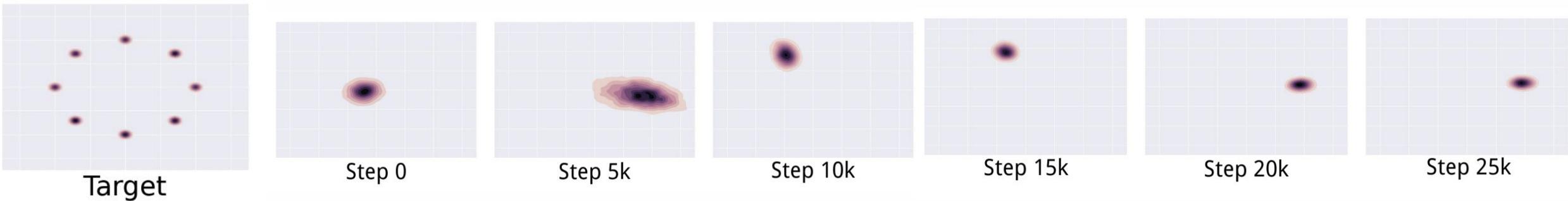


Target

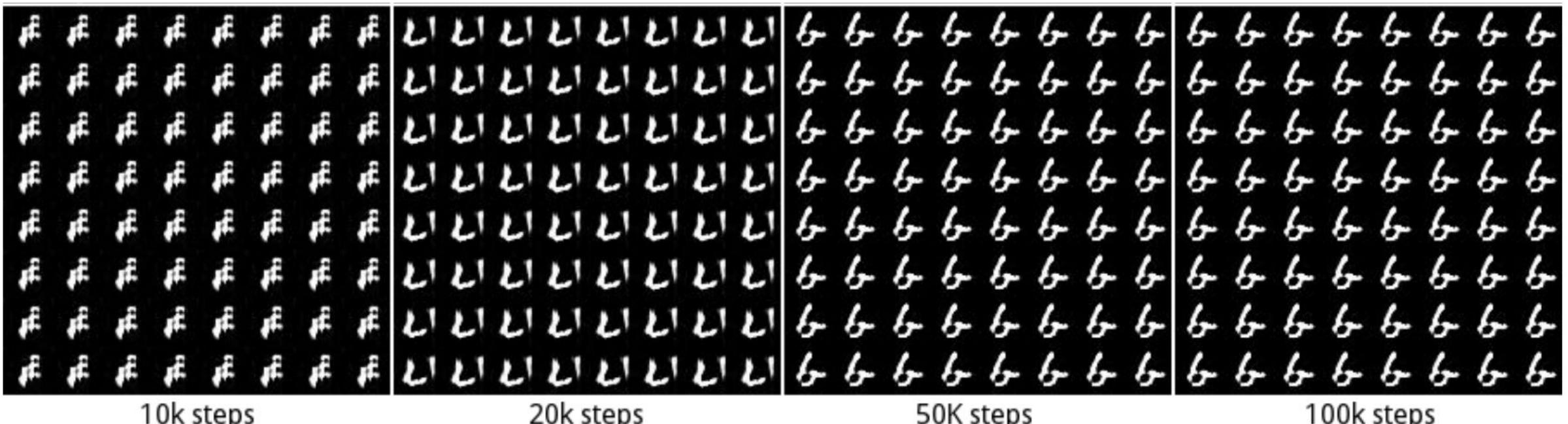
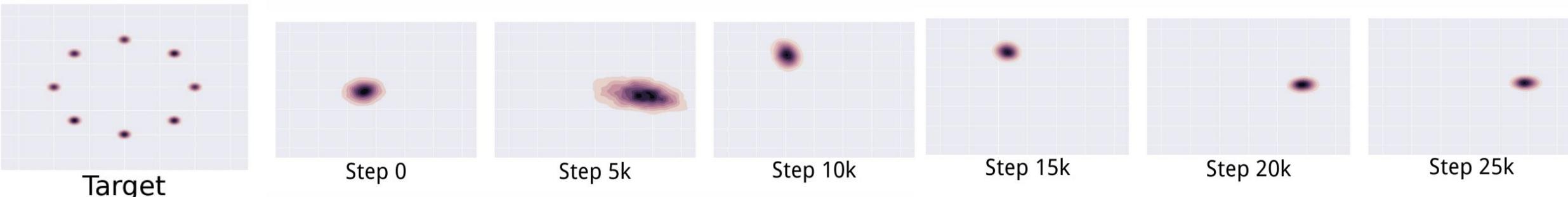


# Mode collapse

---



# Mode collapse









# Что мы уже прошли

---

- ❖ Оригинальная статья:
  - учим дискриминатор как бинарный классификатор
  - используем градиенты ошибки классификации для обучения дискриминатора и генератора
  - столкнулись с затуханием градиентов генератора
    - поменяли функционал ошибки генератора
  - узнали, что идеальный дискриминатор пытается предсказать отношение плотностей распределений
  - столкнулись с проблемой mode collapse. В 2020 она до сих пор не решена
    - причиной считается поведение генератора, который стремится для всего латентного пространства вернуть одинаковый выход



# Что будет дальше

---

- ❖ Вспомним что такое KL и JS дивергенции
- ❖ Столкнемся с проблемами сходимости у сети из оригинальной статьи
- ❖ Предложим решение проблем



# KL-дивергенция

---

- ❖ Несимметричный функционал, показывающий меру близости двух вероятностных распределений

# KL-дивергенция

- ❖ Несимметричный функционал, показывающий меру близости двух вероятностных распределений

$$D_{KL}(P \parallel Q) = \sum_{i=1}^n p_i \log \frac{p_i}{q_i} \quad \text{для дискретных распределений}$$

# KL-дивергенция

- ❖ Несимметричный функционал, показывающий меру близости двух вероятностных распределений

$$D_{KL}(P \parallel Q) = \sum_{i=1}^n p_i \log \frac{p_i}{q_i} \quad \text{для дискретных распределений}$$

$$D_{KL}(P \parallel Q) = \int_X p(x) \log \frac{p(x)}{q(x)} dx \quad \begin{aligned} &\text{для абсолютно непрерывных} \\ &\text{распределений P и Q, где} \\ &p(x) \text{ и } q(x) - \text{плотности} \end{aligned}$$

# KL-дивергенция

- ❖ Несимметричный функционал, показывающий меру близости двух вероятностных распределений

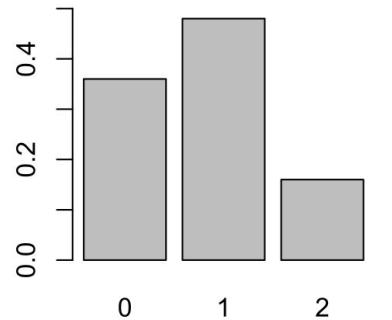
$$D_{KL}(P \parallel Q) = \sum_{i=1}^n p_i \log \frac{p_i}{q_i} \quad \text{для дискретных распределений}$$

$$D_{KL}(P \parallel Q) = \int_X p(x) \log \frac{p(x)}{q(x)} dx \quad \begin{aligned} &\text{для абсолютно непрерывных} \\ &\text{распределений } P \text{ и } Q, \text{ где} \\ &p(x) \text{ и } q(x) - \text{плотности} \end{aligned}$$

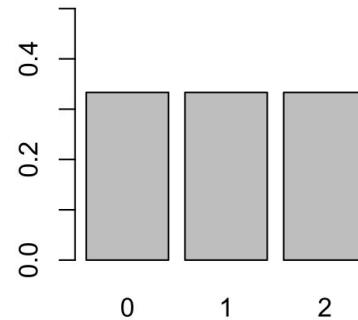
$$D_{KL}(P \parallel Q) \neq D_{KL}(Q \parallel P)$$

# KL-дивергенция. Дискретный случай

**Distribution P**  
Binomial with  $p = 0.4$ ,  $N = 2$

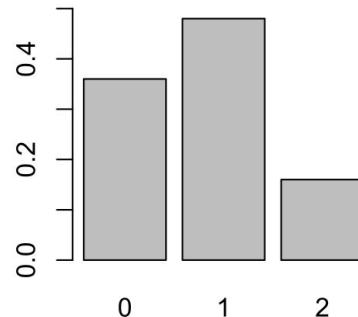


**Distribution Q**  
Uniform with  $p = 1/3$

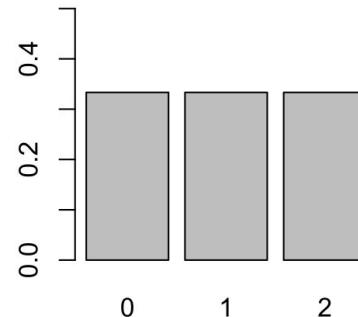


# KL-дивергенция. Дискретный случай

Distribution P  
Binomial with  $p = 0.4$ ,  $N = 2$



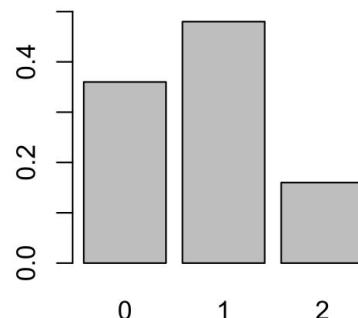
Distribution Q  
Uniform with  $p = 1/3$



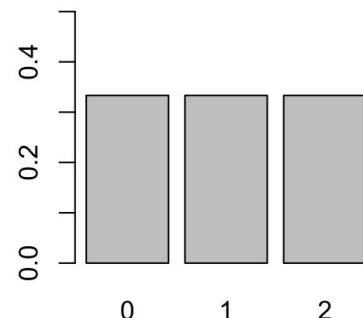
x	1	2	3
P(x)	0.36	0.48	0.16
Q(x)	0.333	0.333	0.333

# KL-дивергенция. Дискретный случай

Distribution P  
Binomial with  $p = 0.4$ ,  $N = 2$



Distribution Q  
Uniform with  $p = 1/3$



x	1	2	3
P(x)	0.36	0.48	0.16
Q(x)	0.333	0.333	0.333

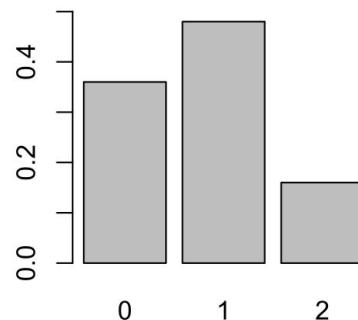
$$D_{\text{KL}}(P \parallel Q) =$$

$$D_{\text{KL}}(Q \parallel P) =$$

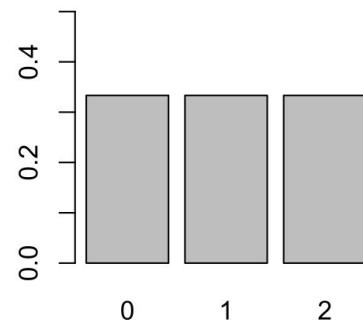
:

# KL-дивергенция. Дискретный случай

Distribution P  
Binomial with  $p = 0.4$ ,  $N = 2$



Distribution Q  
Uniform with  $p = 1/3$



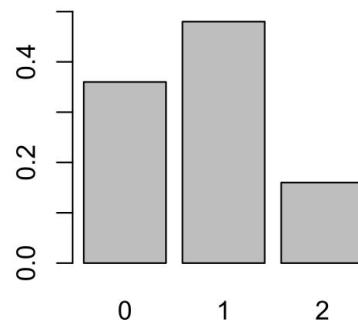
x	1	2	3
P(x)	0.36	0.48	0.16
Q(x)	0.333	0.333	0.333

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \ln \left( \frac{P(x)}{Q(x)} \right)$$

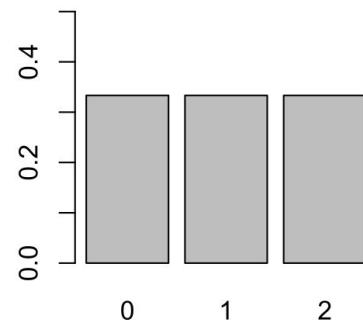
$$D_{\text{KL}}(Q \parallel P) =$$

# KL-дивергенция. Дискретный случай

Distribution P  
Binomial with  $p = 0.4$ ,  $N = 2$



Distribution Q  
Uniform with  $p = 1/3$



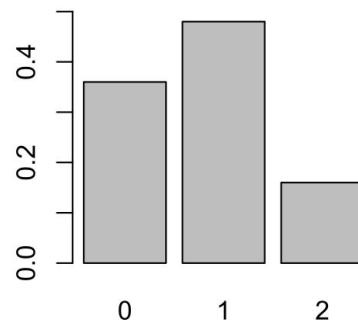
x	1	2	3
P(x)	0.36	0.48	0.16
Q(x)	0.333	0.333	0.333

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \ln \left( \frac{P(x)}{Q(x)} \right) = 0.36 \ln \left( \frac{0.36}{0.333} \right) + 0.48 \ln \left( \frac{0.48}{0.333} \right) + 0.16 \ln \left( \frac{0.16}{0.333} \right)$$

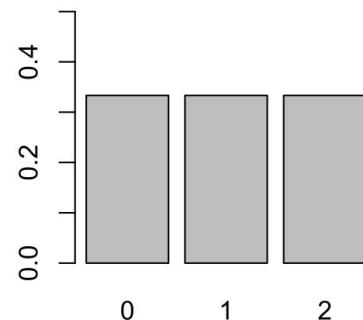
$$D_{\text{KL}}(Q \parallel P) =$$

# KL-дивергенция. Дискретный случай

Distribution P  
Binomial with  $p = 0.4$ ,  $N = 2$



Distribution Q  
Uniform with  $p = 1/3$



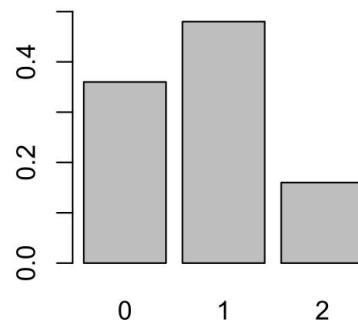
x	1	2	3
P(x)	0.36	0.48	0.16
Q(x)	0.333	0.333	0.333

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \ln \left( \frac{P(x)}{Q(x)} \right) = 0.36 \ln \left( \frac{0.36}{0.333} \right) + 0.48 \ln \left( \frac{0.48}{0.333} \right) + 0.16 \ln \left( \frac{0.16}{0.333} \right) = 0.085299$$

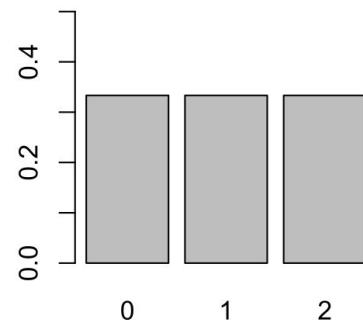
$$D_{\text{KL}}(Q \parallel P) =$$

# KL-дивергенция. Дискретный случай

Distribution P  
Binomial with  $p = 0.4$ ,  $N = 2$



Distribution Q  
Uniform with  $p = 1/3$



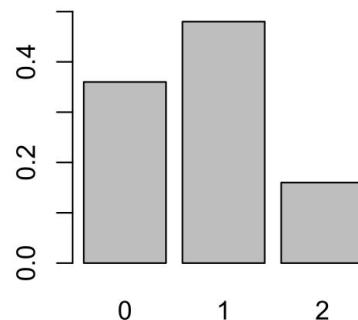
x	1	2	3
P(x)	0.36	0.48	0.16
Q(x)	0.333	0.333	0.333

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \ln \left( \frac{P(x)}{Q(x)} \right) = 0.36 \ln \left( \frac{0.36}{0.333} \right) + 0.48 \ln \left( \frac{0.48}{0.333} \right) + 0.16 \ln \left( \frac{0.16}{0.333} \right) = 0.085299$$

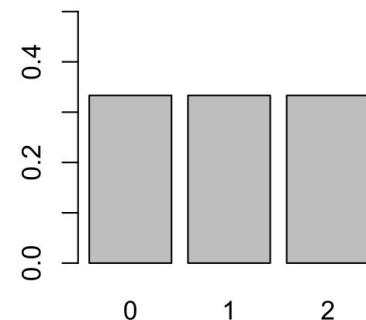
$$D_{\text{KL}}(Q \parallel P) = \sum_{x \in \mathcal{X}} Q(x) \ln \left( \frac{Q(x)}{P(x)} \right)$$

# KL-дивергенция. Дискретный случай

Distribution P  
Binomial with  $p = 0.4$ ,  $N = 2$



Distribution Q  
Uniform with  $p = 1/3$



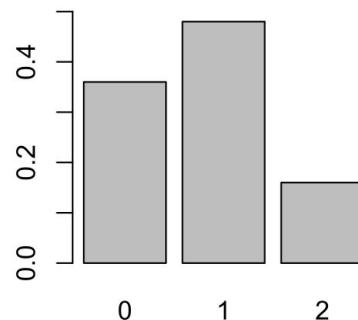
x	1	2	3
P(x)	0.36	0.48	0.16
Q(x)	0.333	0.333	0.333

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \ln \left( \frac{P(x)}{Q(x)} \right) = 0.36 \ln \left( \frac{0.36}{0.333} \right) + 0.48 \ln \left( \frac{0.48}{0.333} \right) + 0.16 \ln \left( \frac{0.16}{0.333} \right) = 0.085299$$

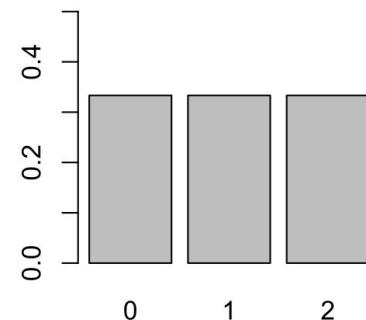
$$D_{\text{KL}}(Q \parallel P) = \sum_{x \in \mathcal{X}} Q(x) \ln \left( \frac{Q(x)}{P(x)} \right) = 0.333 \ln \left( \frac{0.333}{0.36} \right) + 0.333 \ln \left( \frac{0.333}{0.48} \right) + 0.333 \ln \left( \frac{0.333}{0.16} \right)$$

# KL-дивергенция. Дискретный случай

Distribution P  
Binomial with  $p = 0.4$ ,  $N = 2$



Distribution Q  
Uniform with  $p = 1/3$



x	1	2	3
P(x)	0.36	0.48	0.16
Q(x)	0.333	0.333	0.333

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \ln \left( \frac{P(x)}{Q(x)} \right) = 0.36 \ln \left( \frac{0.36}{0.333} \right) + 0.48 \ln \left( \frac{0.48}{0.333} \right) + 0.16 \ln \left( \frac{0.16}{0.333} \right) = 0.085299$$

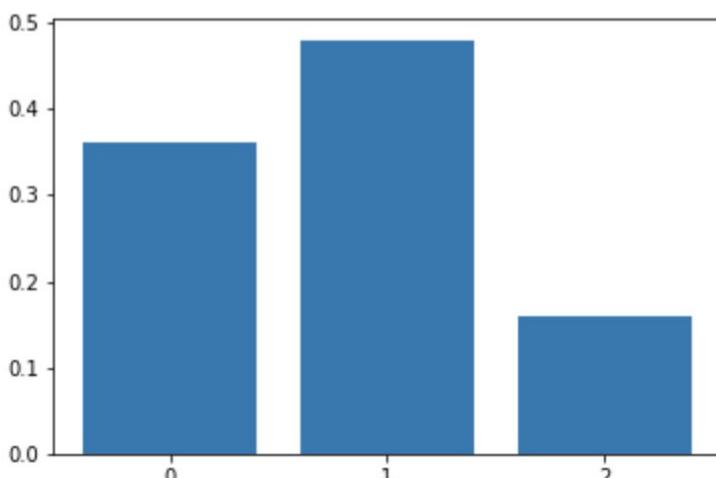
$$D_{\text{KL}}(Q \parallel P) = \sum_{x \in \mathcal{X}} Q(x) \ln \left( \frac{Q(x)}{P(x)} \right) = 0.333 \ln \left( \frac{0.333}{0.36} \right) + 0.333 \ln \left( \frac{0.333}{0.48} \right) + 0.333 \ln \left( \frac{0.333}{0.16} \right) = 0.097455$$

# KL-дивергенция. Код

## Биномиальное распределение

```
rv = binom(2, 0.4)
x = np.arange(3)
binom_pmf = rv.pmf(x)
print ("Probability mass function values: {}".format(binom_pmf))
plt.bar(x, binom_pmf, width=0.8, tick_label=x)
plt.show()
```

Probability mass function values: [0.36 0.48 0.16]

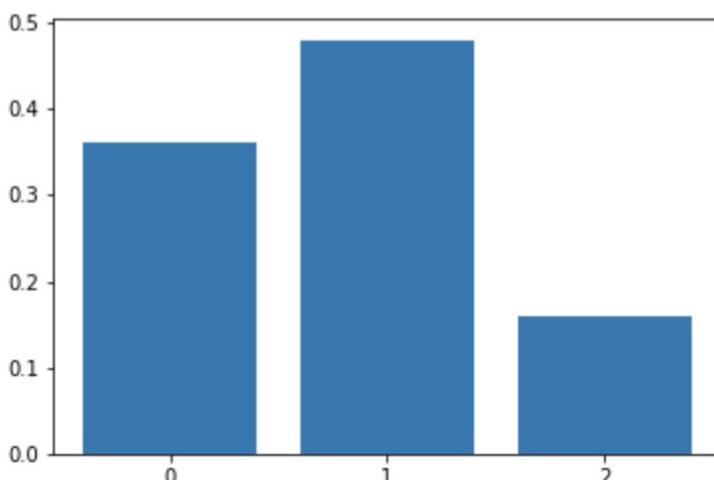


# KL-дивергенция. Код

## Биномиальное распределение

```
rv = binom(2, 0.4)
x = np.arange(3)
binom_pmf = rv.pmf(x)
print ("Probability mass function values: {}".format(binom_pmf))
plt.bar(x, binom_pmf, width=0.8, tick_label=x)
plt.show()
```

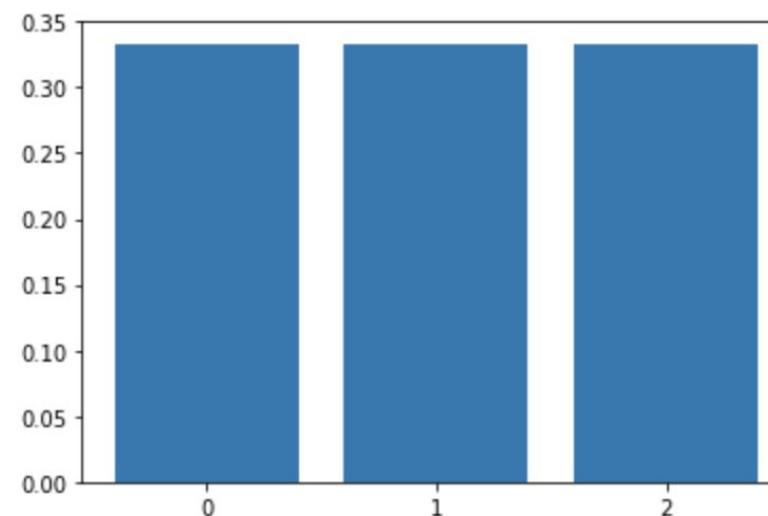
Probability mass function values: [0.36 0.48 0.16]



## Равномерное распределение

```
: uniform_pmf = 1 / len(x) * np.ones_like(x)
print ("Probability mass function values: {}".format(uniform_pmf))
plt.bar(x, uniform_pmf, width=0.8, tick_label=x)
plt.show()
```

Probability mass function values: [0.33333333 0.33333333 0.33333333]



# KL-дивергенция. Код

---

## KL-дивергенция своими руками

```
|: # https://towardsdatascience.com/kl-divergence-python-example-b87069e4b810
def kl_divergence(p, q):
    return np.sum(np.where(p != 0, p * np.log(p / q), 0))
```

$$D_{\text{KL}}(P \parallel Q)$$

```
|: print (%.7f % kl_divergence(binom_pmf, uniform_pmf))
```

0.0852996

$$D_{\text{KL}}(Q \parallel P)$$

```
|: print (%.6f % kl_divergence(uniform_pmf, binom_pmf))
```

0.097455

# KL-дивергенция. Код

---

## KL-дивергенция из scipy

```
: # https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.kl_div.html
from scipy.special import kl_div

: print ("% .7f" % np.sum(kl_div(binom_pmf, uniform_pmf)))
0.0852996

: print ("% .6f" % np.sum(kl_div(uniform_pmf, binom_pmf)))
0.097455
```



## JS-дивергенция

---

$$\text{JSD}(P \parallel Q) = \frac{1}{2}D_{KL}(P \parallel M) + \frac{1}{2}D_{KL}(Q \parallel M)$$



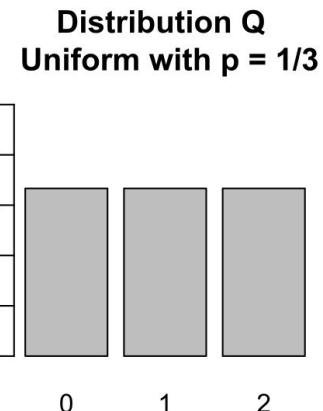
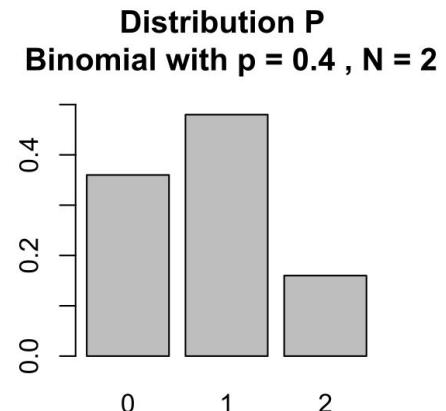
## JS-дивергенция

---

$$\text{JSD}(P \parallel Q) = \frac{1}{2}D_{KL}(P \parallel M) + \frac{1}{2}D_{KL}(Q \parallel M)$$

$$M = \frac{1}{2}(P + Q)$$

# JS-дивергенция. Дискретный случай



x	1	2	3
P(x)	0.36	0.48	0.16
Q(x)	0.333	0.333	0.333
M(x)	0.347	0.407	0.247

$$D_{KL}(P \parallel M) = \sum_{x \in \mathcal{X}} P(x) \ln \left( \frac{P(x)}{M(x)} \right) = 0.36 \ln \left( \frac{0.36}{0.347} \right) + 0.48 \ln \left( \frac{0.48}{0.407} \right) + 0.16 \ln \left( \frac{0.16}{0.247} \right) = 0.02295$$

$$D_{KL}(Q \parallel M) = \sum_{x \in \mathcal{X}} Q(x) \ln \left( \frac{Q(x)}{M(x)} \right) = 0.333 \ln \left( \frac{0.333}{0.347} \right) + 0.333 \ln \left( \frac{0.333}{0.407} \right) + 0.333 \ln \left( \frac{0.333}{0.247} \right) = 0.01895$$

$$JSD(P \parallel Q) = \frac{1}{2} D_{KL}(P \parallel M) + \frac{1}{2} D_{KL}(Q \parallel M) = 0.02095$$

# JS-дивергенция. Дискретный случай

---

## JS-дивергенция своими руками

```
: def js_divergence(p, q):
    m = (p + q) / 2
    return (kl_divergence(p, m) + kl_divergence(q, m)) / 2
```

```
: print ("%.7f" % js_divergence(binom_pmf, uniform_pmf))
```

0.0224599

```
: print ("%.7f" % js_divergence(uniform_pmf, binom_pmf))
```

0.0224599

# JS-дивергенция. Дискретный случай

## JS-дивергенция

```
# https://scipy.github.io/devdocs/generated/scipy.spatial.distance.jensenshannon.html
from scipy.spatial.distance import jensenshannon
```

Compute the Jensen-Shannon distance (metric) between two 1-D probability arrays. This is the square root of the Jensen-Shannon divergence. The Jensen-Shannon distance between two probability vectors `p` and `q` is defined as,  $\sqrt{\frac{D(p||m)+D(q||m)}{2}}$ , where `m` is the pointwise mean of `p` and `q` and `D` is the Kullback-Leibler divergence.

```
print ("% .7f" % jensenshannon(binom_pmf, uniform_pmf) ** 2)
```

```
0.0224599
```

```
print ("% .7f" % jensenshannon(uniform_pmf, binom_pmf) ** 2)
```

```
0.0224599
```



# Оригинальная статья. Функционал потерь

---

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

# Оригинальная статья. Функция потерь

---

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

В оптимальной точке:  $p_{\text{data}} = p_{\text{gen}}$ ,  $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)} = \frac{1}{2}$

# Оригинальная статья. Функция потерь

---

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

В оптимальной точке:  $p_{\text{data}} = p_{\text{gen}}$ ,  $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)} = \frac{1}{2}$

$$\begin{aligned} \min_G V(D^*, G) &= \int_x \left( p_r(x) \log D^*(x) + p_g(x) \log(1 - D^*(x)) \right) dx \\ &= \int_x \left( p_r(x) \log \frac{p_r(x)}{p_r(x) + p_g(x)} + p_g(x) \log \frac{p_g(x)}{p_r(x) + p_g(x)} \right) dx \end{aligned}$$

# Оригинальная статья. Функция потерь

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

В оптимальной точке:  $p_{\text{data}} = p_{\text{gen}}$ ,  $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)} = \frac{1}{2}$

$$\begin{aligned} \min_G V(D^*, G) &= \int_x \left( p_r(x) \log D^*(x) + p_g(x) \log(1 - D^*(x)) \right) dx \\ &= \int_x \left( p_r(x) \log \frac{p_r(x)}{p_r(x) + p_g(x)} + p_g(x) \log \frac{p_g(x)}{p_r(x) + p_g(x)} \right) dx \end{aligned}$$

$$\begin{aligned} D_{JS}(p_r \| p_g) &= \frac{1}{2} D_{KL}(p_r || \frac{p_r + p_g}{2}) + \frac{1}{2} D_{KL}(p_g || \frac{p_r + p_g}{2}) \\ &= \frac{1}{2} \left( \int_x p_r(x) \log \frac{2p_r(x)}{p_r(x) + p_g(x)} dx \right) + \frac{1}{2} \left( \int_x p_g(x) \log \frac{2p_g(x)}{p_r(x) + p_g(x)} dx \right) \\ &= \frac{1}{2} \left( \log 2 + \int_x p_r(x) \log \frac{p_r(x)}{p_r(x) + p_g(x)} dx \right) + \\ &\quad \frac{1}{2} \left( \log 2 + \int_x p_g(x) \log \frac{p_g(x)}{p_r(x) + p_g(x)} dx \right) \\ &= \frac{1}{2} \left( \log 4 + \min_G V(D^*, G) \right) \end{aligned}$$

# Оригинальная статья. Функция потерь

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

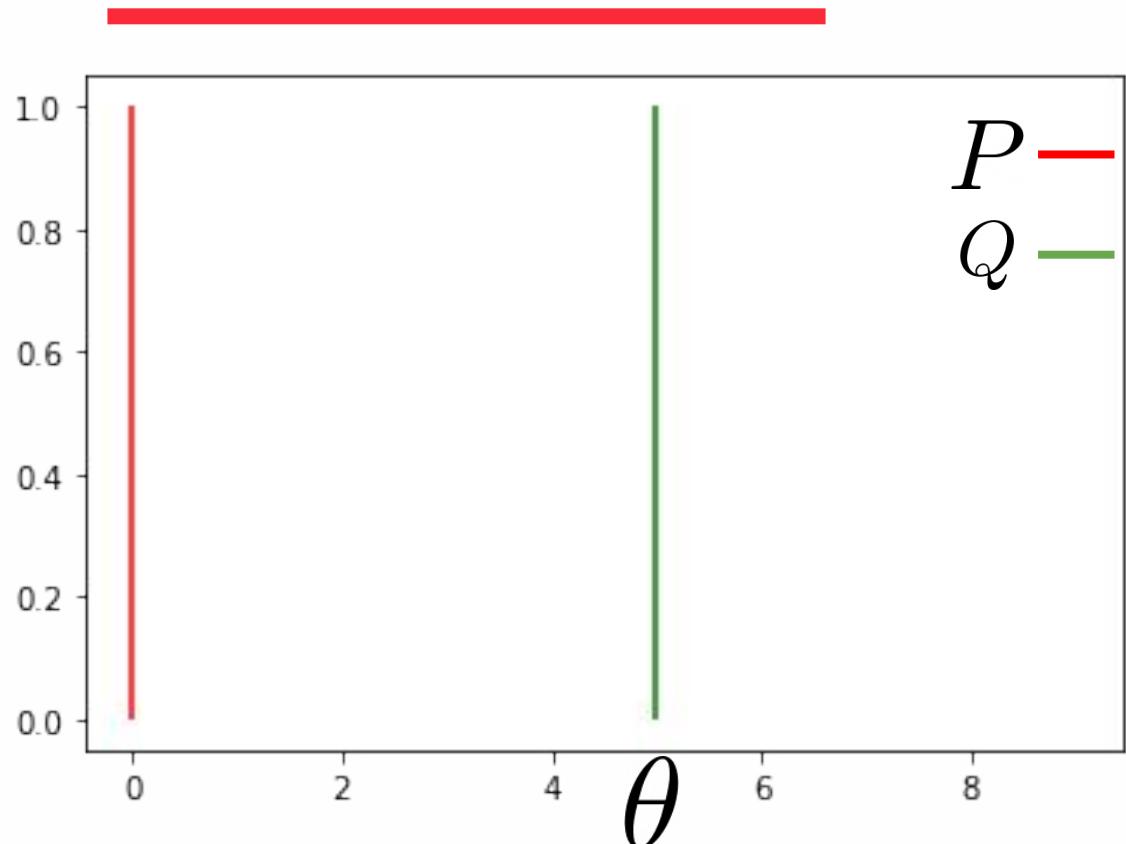
В оптимальной точке:  $p_{\text{data}} = p_{\text{gen}}$ ,  $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)} = \frac{1}{2}$

$$\begin{aligned} \min_G V(D^*, G) &= \int_x \left( p_r(x) \log D^*(x) + p_g(x) \log(1 - D^*(x)) \right) dx \\ &= \int_x \left( p_r(x) \log \frac{p_r(x)}{p_r(x) + p_g(x)} + p_g(x) \log \frac{p_g(x)}{p_r(x) + p_g(x)} \right) dx \end{aligned}$$

$$\begin{aligned} D_{JS}(p_r \| p_g) &= \frac{1}{2} D_{KL}(p_r \| \frac{p_r + p_g}{2}) + \frac{1}{2} D_{KL}(p_g \| \frac{p_r + p_g}{2}) \\ &= \frac{1}{2} \left( \int_x p_r(x) \log \frac{2p_r(x)}{p_r(x) + p_g(x)} dx \right) + \frac{1}{2} \left( \int_x p_g(x) \log \frac{2p_g(x)}{p_r(x) + p_g(x)} dx \right) \\ &= \frac{1}{2} \left( \log 2 + \int_x p_r(x) \log \frac{p_r(x)}{p_r(x) + p_g(x)} dx \right) + \\ &\quad \frac{1}{2} \left( \log 2 + \int_x p_g(x) \log \frac{p_g(x)}{p_r(x) + p_g(x)} dx \right) \\ &= \frac{1}{2} \left( \log 4 + \min_G V(D^*, G) \right) \end{aligned}$$

$$L(G, D) = -2 \log(2) + 2D_{JS}(p_r \| p_g)$$

# Проблемы KL и JS дивергенций. Пример



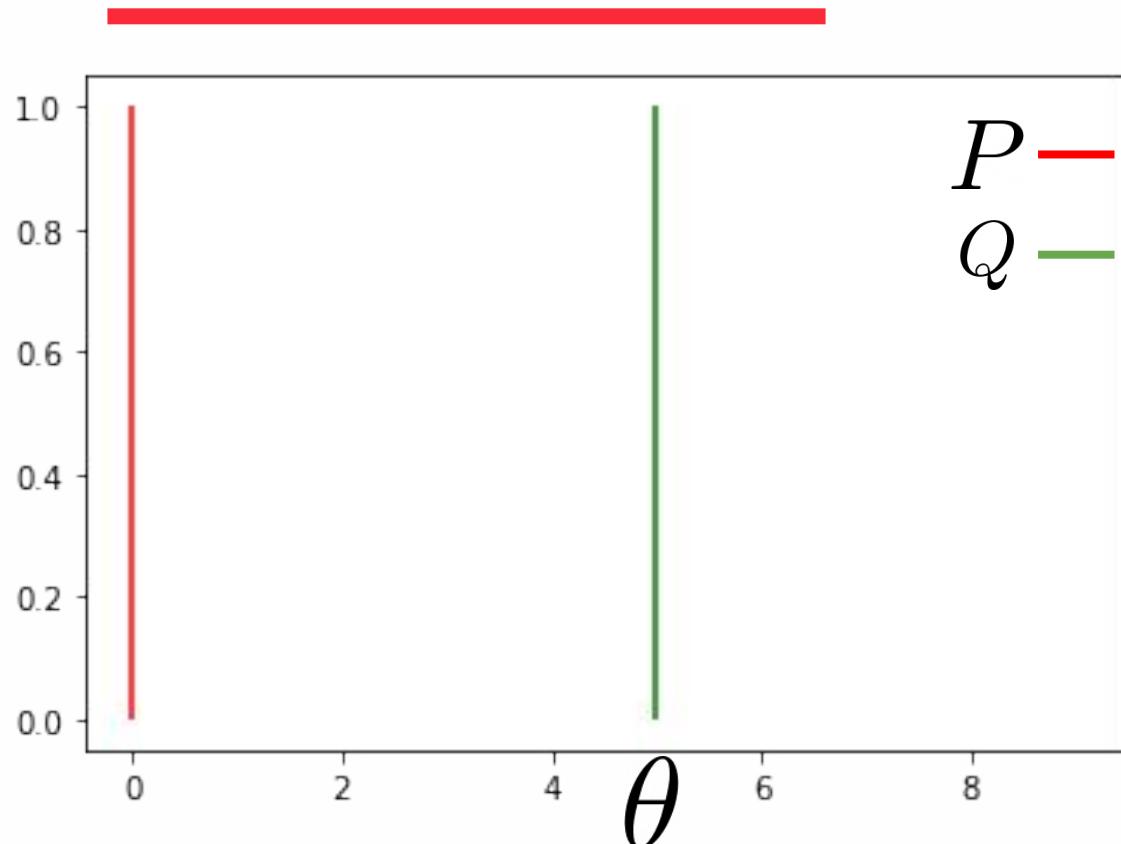
$$D_{\text{KL}}(P \parallel Q) =$$

$$\text{JSD}(P \parallel Q) =$$

$\forall(x, y) \in P, x = 0$  and  $y \sim U(0, 1)$

$\forall(x, y) \in Q, x = \theta, 0 \leq \theta \leq \infty$  and  $y \sim U(0, 1)$

# Проблемы KL и JS дивергенций. Пример



$$D_{KL}(P \parallel Q) =$$

$$\text{JSD}(P \parallel Q) =$$

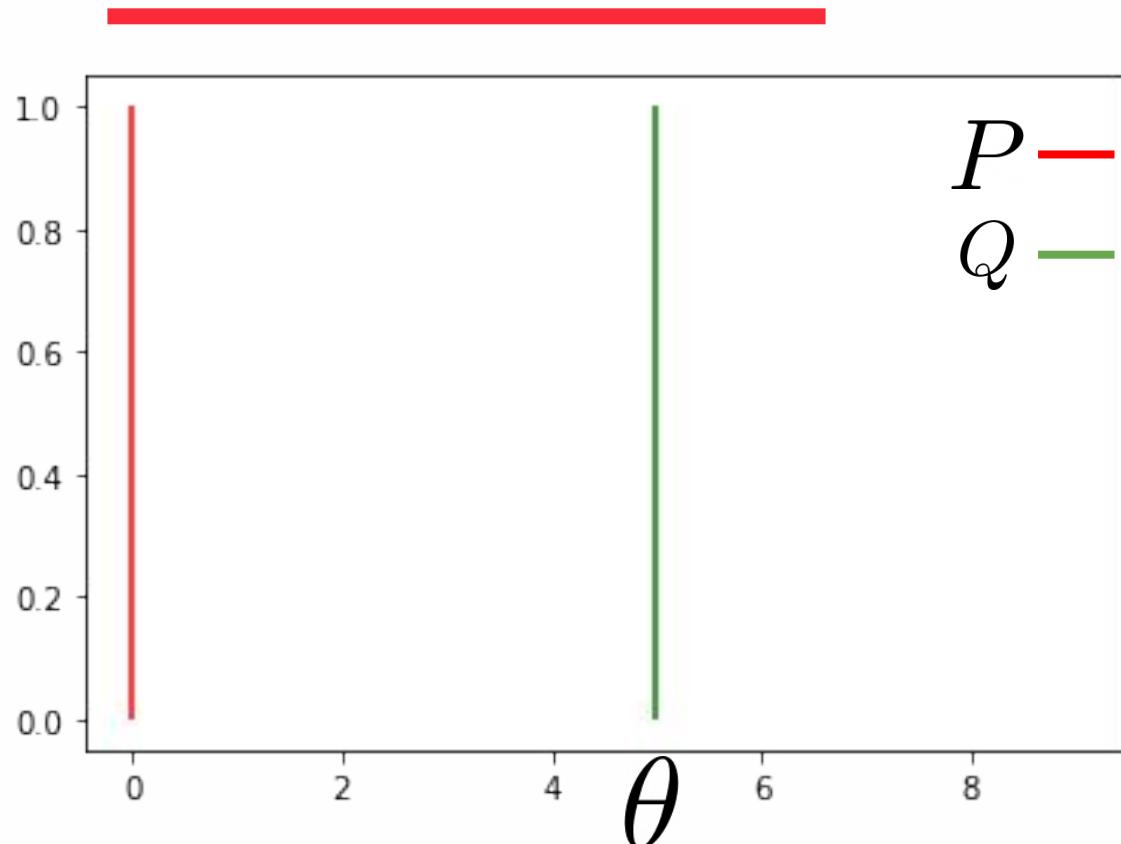
$\forall(x, y) \in P, x = 0 \text{ and } y \sim U(0, 1)$

$\forall(x, y) \in Q, x = \theta, 0 \leq \theta \leq \infty \text{ and } y \sim U(0, 1)$

$$D_{KL}(P \parallel Q) = \sum_{x=0, y \sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

$$D_{KL}(Q \parallel P) = \sum_{x=\theta, y \sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

# Проблемы KL и JS дивергенций. Пример



$$D_{KL}(P \parallel Q) =$$

$$\text{JSD}(P \parallel Q) =$$

$\forall(x, y) \in P, x = 0 \text{ and } y \sim U(0, 1)$

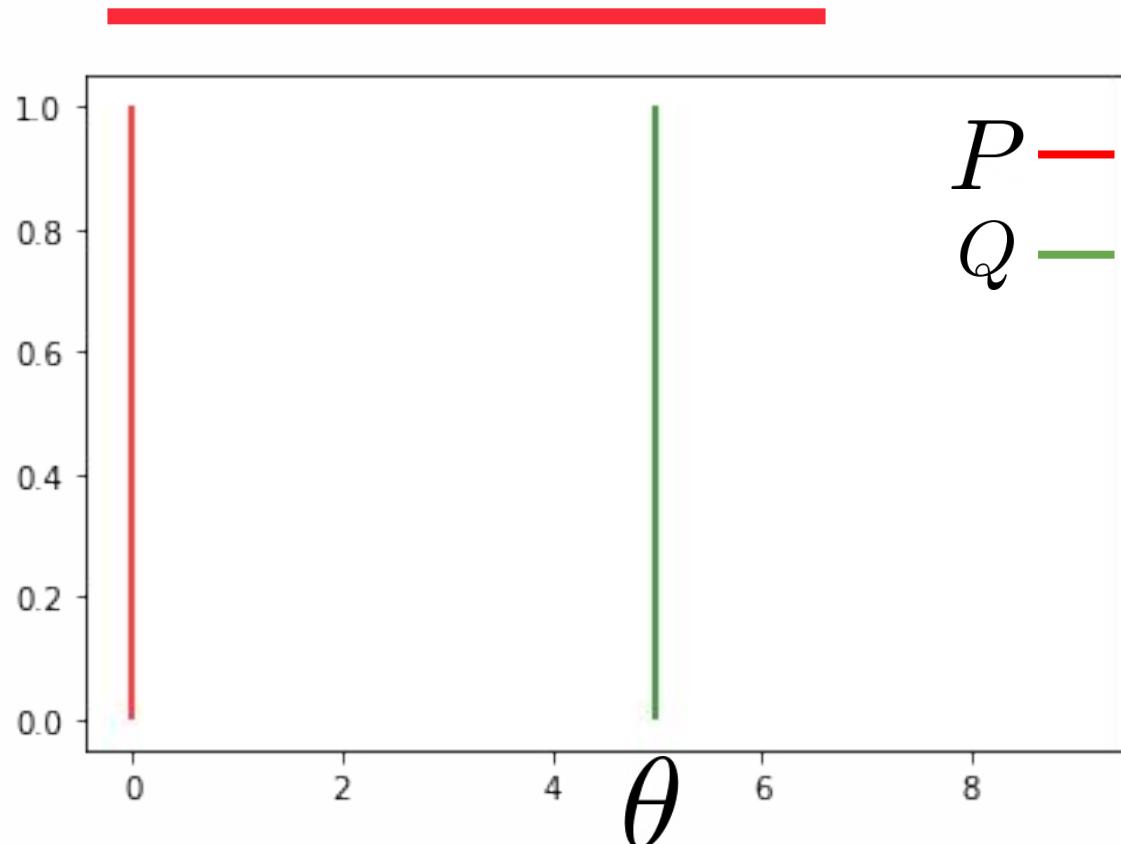
$\forall(x, y) \in Q, x = \theta, 0 \leq \theta \leq \infty \text{ and } y \sim U(0, 1)$

$$D_{KL}(P \parallel Q) = \sum_{x=0, y \sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

$$D_{KL}(Q \parallel P) = \sum_{x=\theta, y \sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

$$D_{JS}(P, Q) = \frac{1}{2} \left( \sum_{x=0, y \sim U(0,1)} 1 \cdot \log \frac{1}{1/2} + \sum_{x=0, y \sim U(0,1)} 1 \cdot \log \frac{1}{1/2} \right) = \log 2$$

# Проблемы KL и JS дивергенций. Пример



$$D_{KL}(P \parallel Q) = \infty$$

$$\text{JSD}(P \parallel Q) = \ln(2) = 0.69314$$

$\forall(x, y) \in P, x = 0 \text{ and } y \sim U(0, 1)$

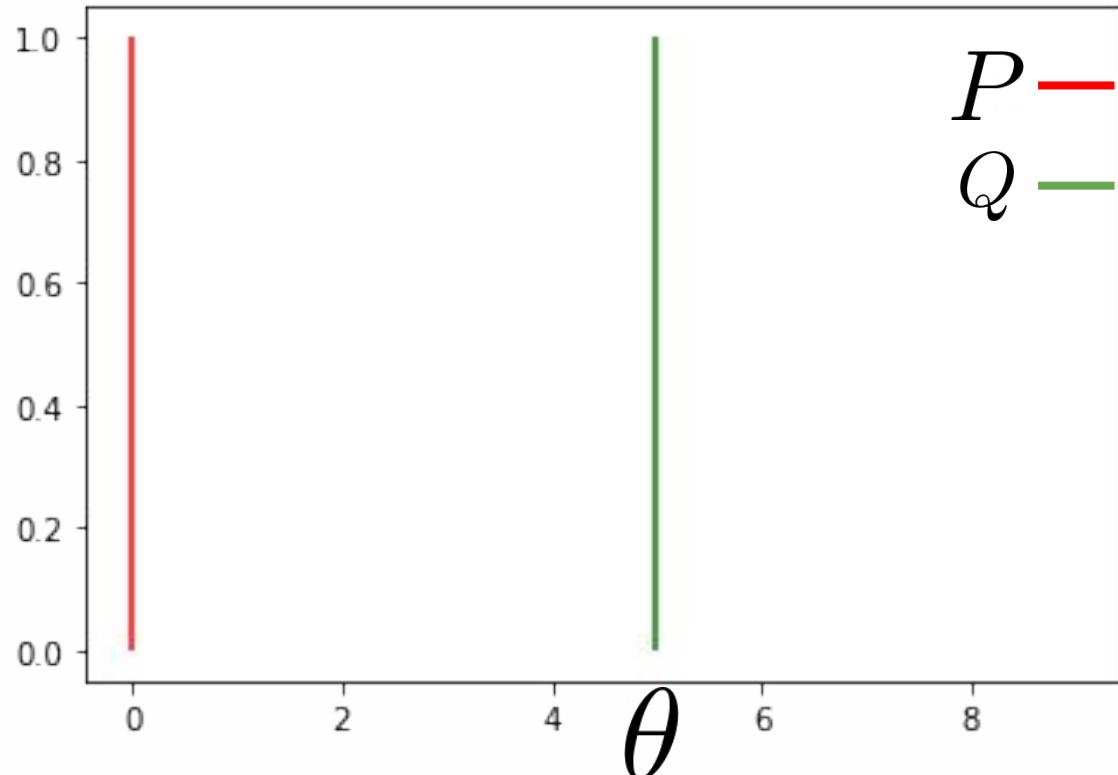
$\forall(x, y) \in Q, x = \theta, 0 \leq \theta \leq \infty \text{ and } y \sim U(0, 1)$

$$D_{KL}(P \parallel Q) = \sum_{x=0, y \sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

$$D_{KL}(Q \parallel P) = \sum_{x=\theta, y \sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

$$D_{JS}(P, Q) = \frac{1}{2} \left( \sum_{x=0, y \sim U(0,1)} 1 \cdot \log \frac{1}{1/2} + \sum_{x=\theta, y \sim U(0,1)} 1 \cdot \log \frac{1}{1/2} \right) = \log 2$$

# Проблемы KL и JS



$$D_{\text{KL}}(P \parallel Q) = \infty$$

$$\text{JSD}(P \parallel Q) = \ln(2) = 0.693$$

[https://github.com/BorisLestsov/MADE/blob/master/seminar10-gan/lecture\\_experiments.ipynb](https://github.com/BorisLestsov/MADE/blob/master/seminar10-gan/lecture_experiments.ipynb)

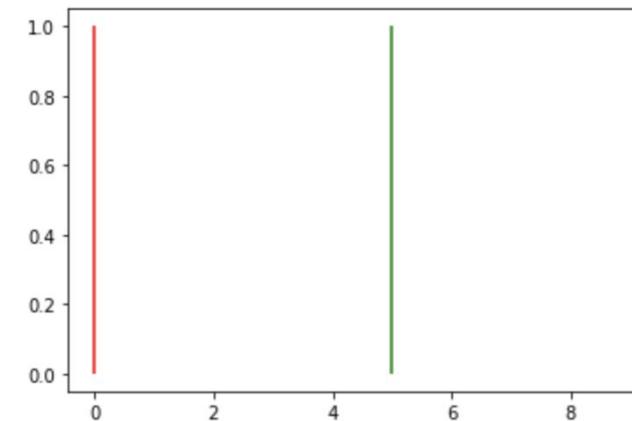
```
# https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.unit_impulse.html
from scipy.signal import unit_impulse
```

```
omega = 5
n_observations = 10
```

```
p = unit_impulse(n_observations)
q = unit_impulse(n_observations, idx=omega)
print ('p: {}'.format(p))
print ('q: {}'.format(q))
```

```
p: [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
q: [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

```
plt.vlines(np.arange(len(p)), 0, p, colors='r')
plt.vlines(np.arange(len(p)), 0, q, colors='g')
plt.show()
```



$$D_{\text{KL}}(P \parallel Q)$$

```
np.sum(kl_div(p, q))
```

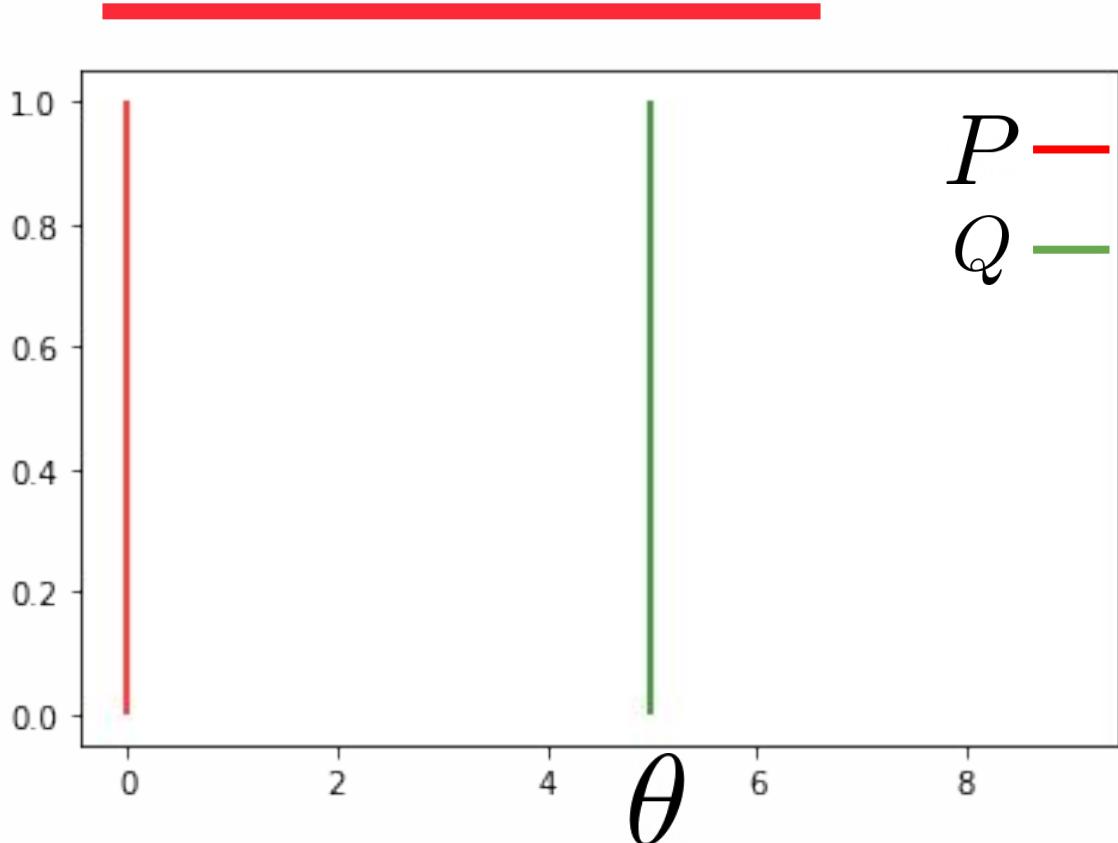
```
inf
```

$$\text{JSD}(P \parallel Q)$$

```
jensenshannon(p, q) ** 2 # amo ln(2)
```

0.6931471805599452

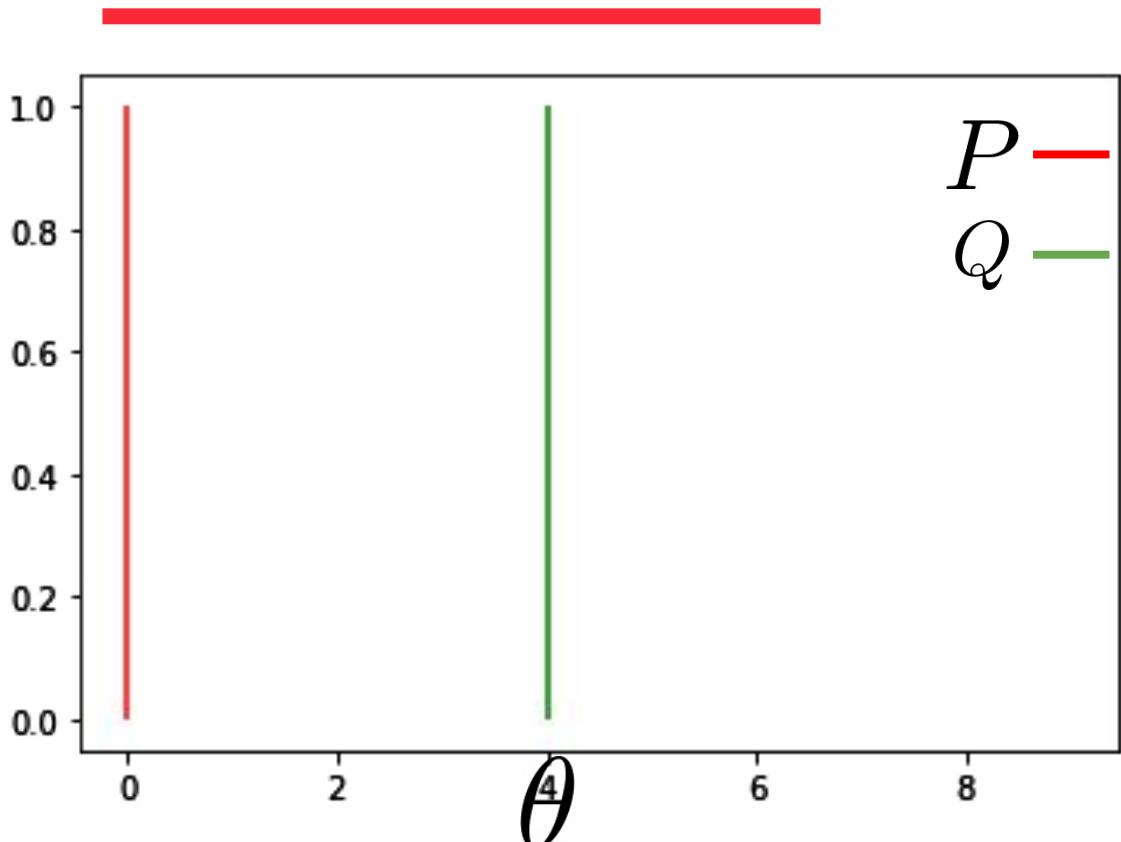
# Проблемы KL и JS дивергенций. Пример



$$D_{\text{KL}}(P \parallel Q) = \infty$$

$$\text{JSD}(P \parallel Q) = \ln(2) = 0.69314$$

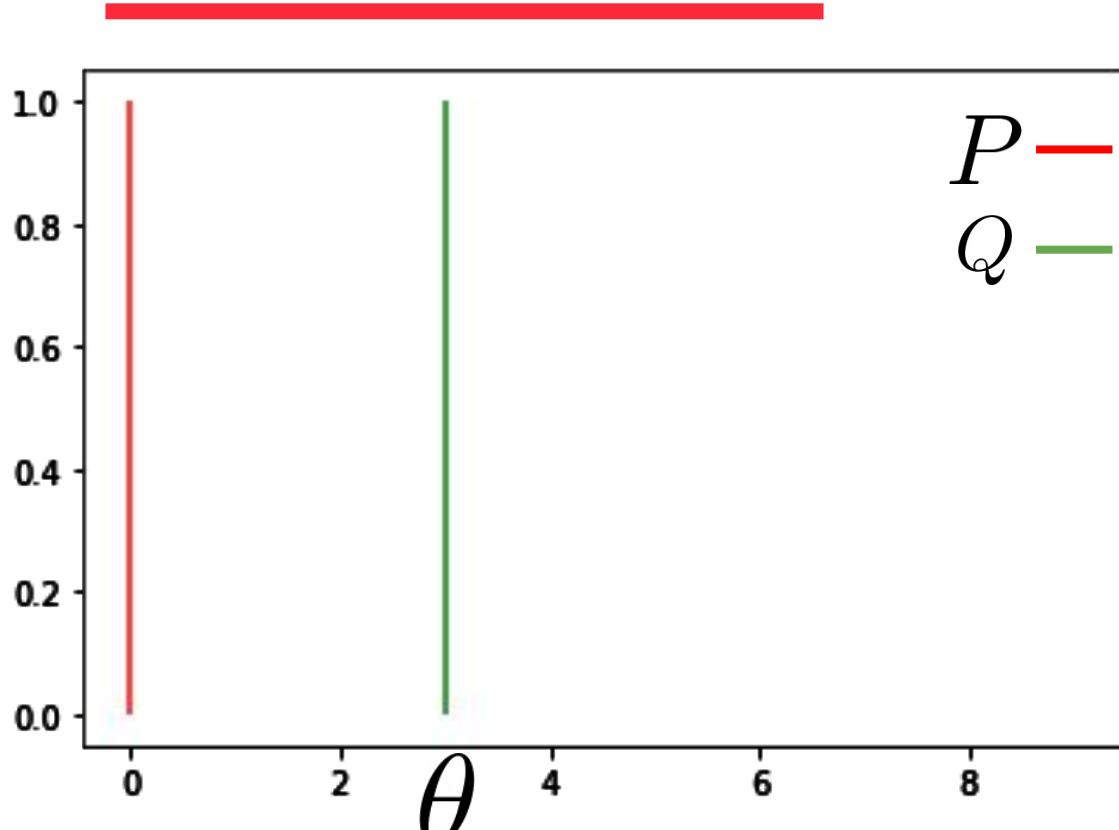
# Проблемы KL и JS дивергенций. Пример



$$D_{\text{KL}}(P \parallel Q) = \infty$$

$$\text{JSD}(P \parallel Q) = \ln(2) = 0.69314$$

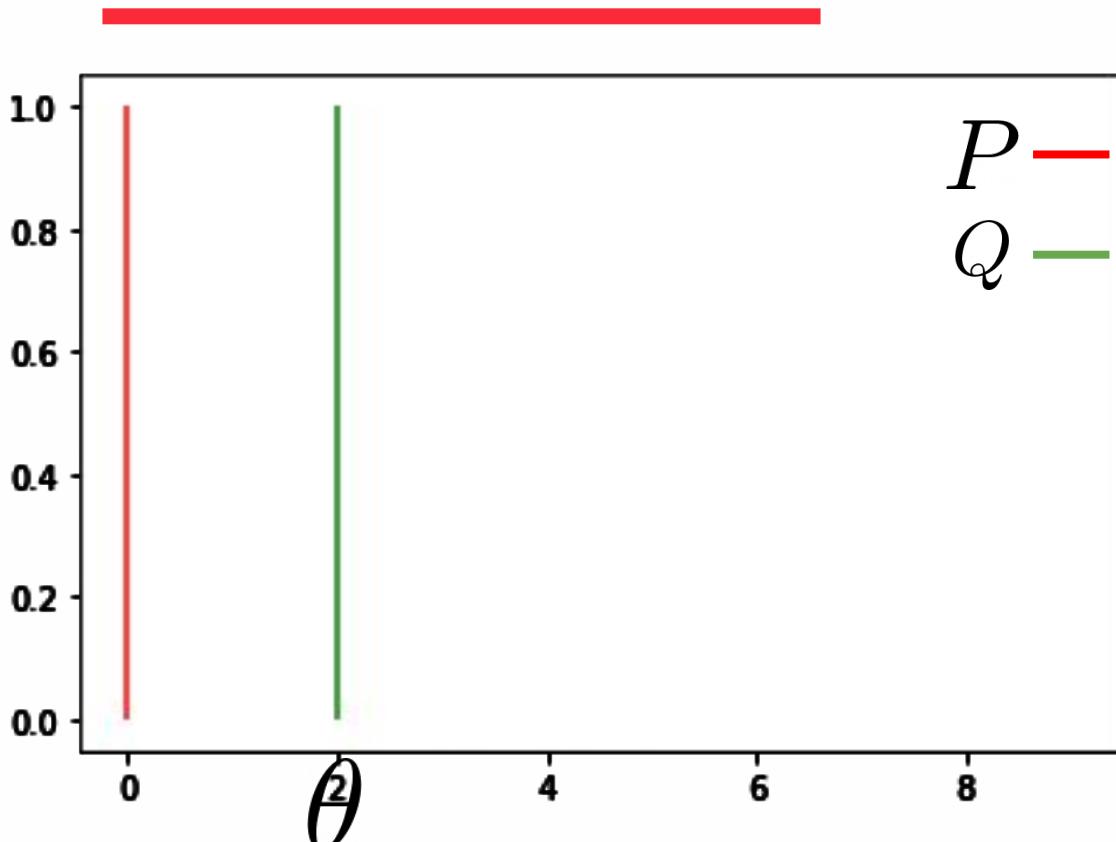
# Проблемы KL и JS дивергенций. Пример



$$D_{\text{KL}}(P \parallel Q) = \infty$$

$$\text{JSD}(P \parallel Q) = \ln(2) = 0.69314$$

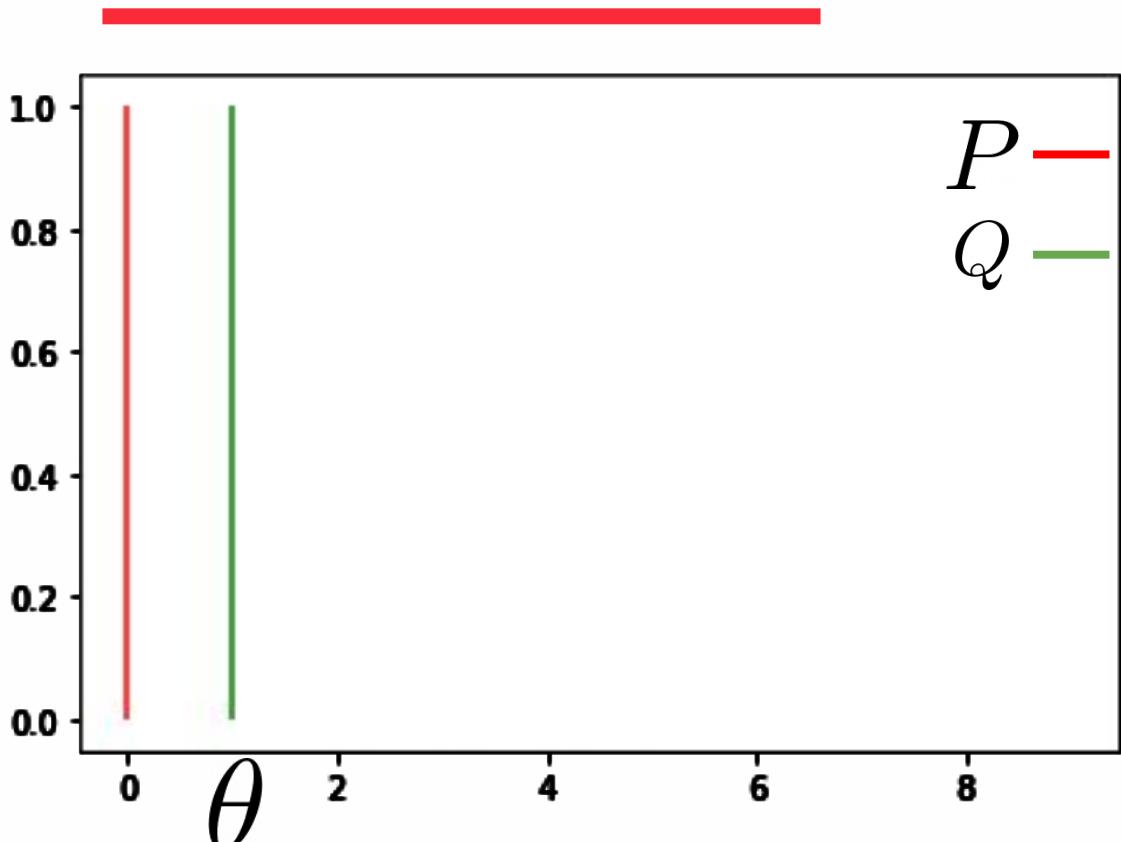
# Проблемы KL и JS дивергенций. Пример



$$D_{\text{KL}}(P \parallel Q) = \infty$$

$$\text{JSD}(P \parallel Q) = \ln(2) = 0.69314$$

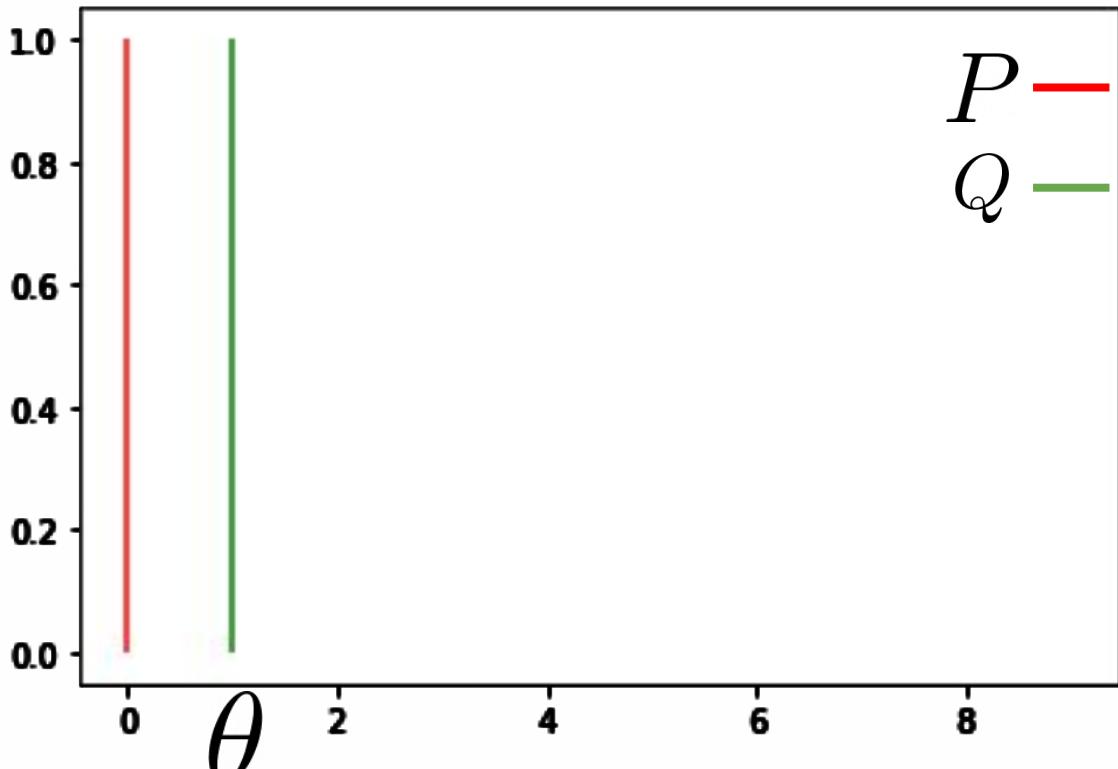
# Проблемы KL и JS дивергенций. Пример



$$D_{\text{KL}}(P \parallel Q) = \infty$$

$$\text{JSD}(P \parallel Q) = \ln(2) = 0.69314$$

# Проблемы KL и JS дивергенций. Пример

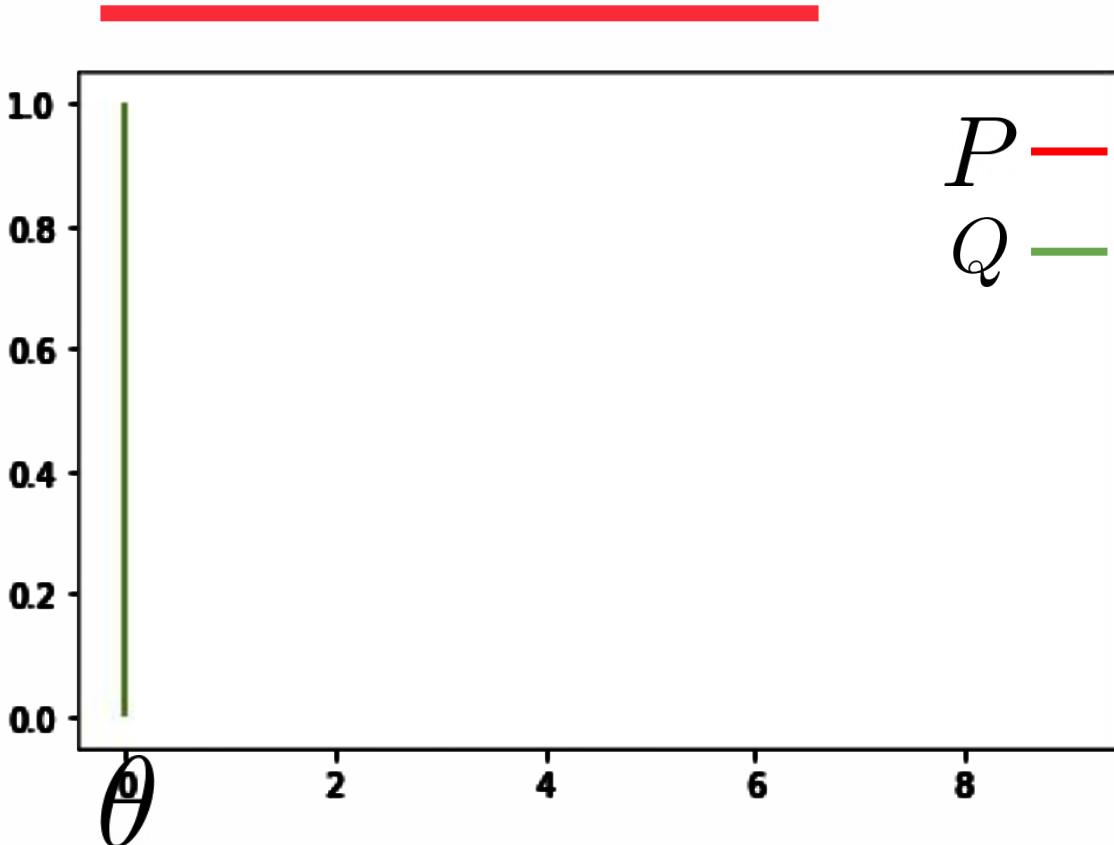


$$D_{\text{KL}}(P \parallel Q) = \infty$$

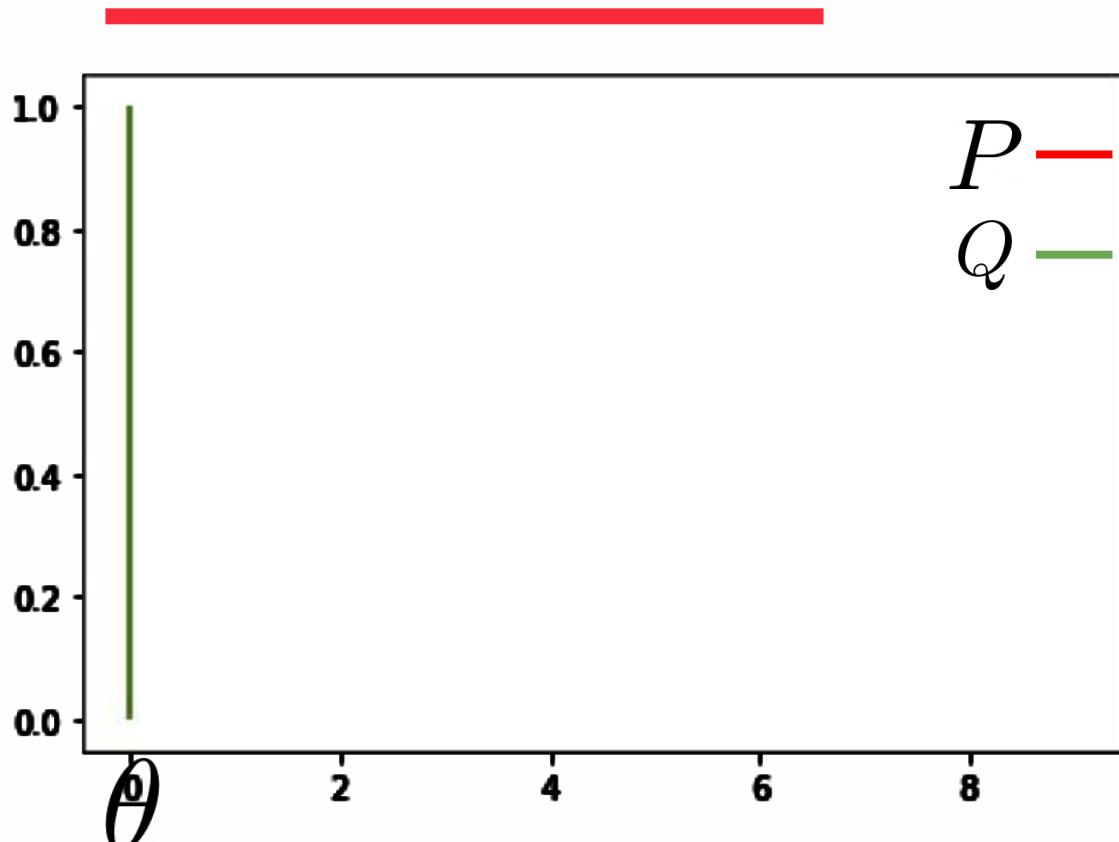
$$\text{JSD}(P \parallel Q) = \ln(2) = 0.69314$$



# Проблемы KL и JS дивергенций. Пример



# Проблемы KL и JS дивергенций. Пример



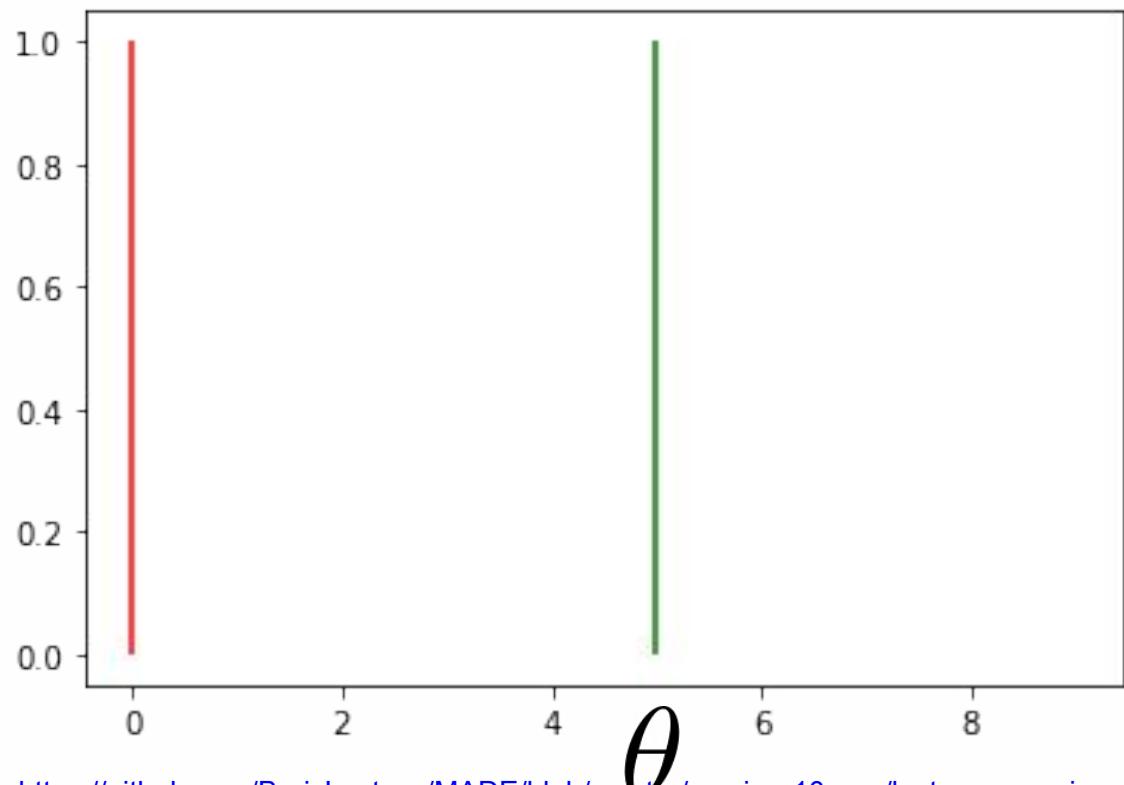
$$D_{\text{KL}}(P \parallel Q) = 0$$

$$\text{JSD}(P \parallel Q) = 0$$

# Проблемы KL и JS дивергенций. Пример

---

$$L(G, D) = -2 \log(2) + 2D_{JS}(p_r \parallel p_g)$$

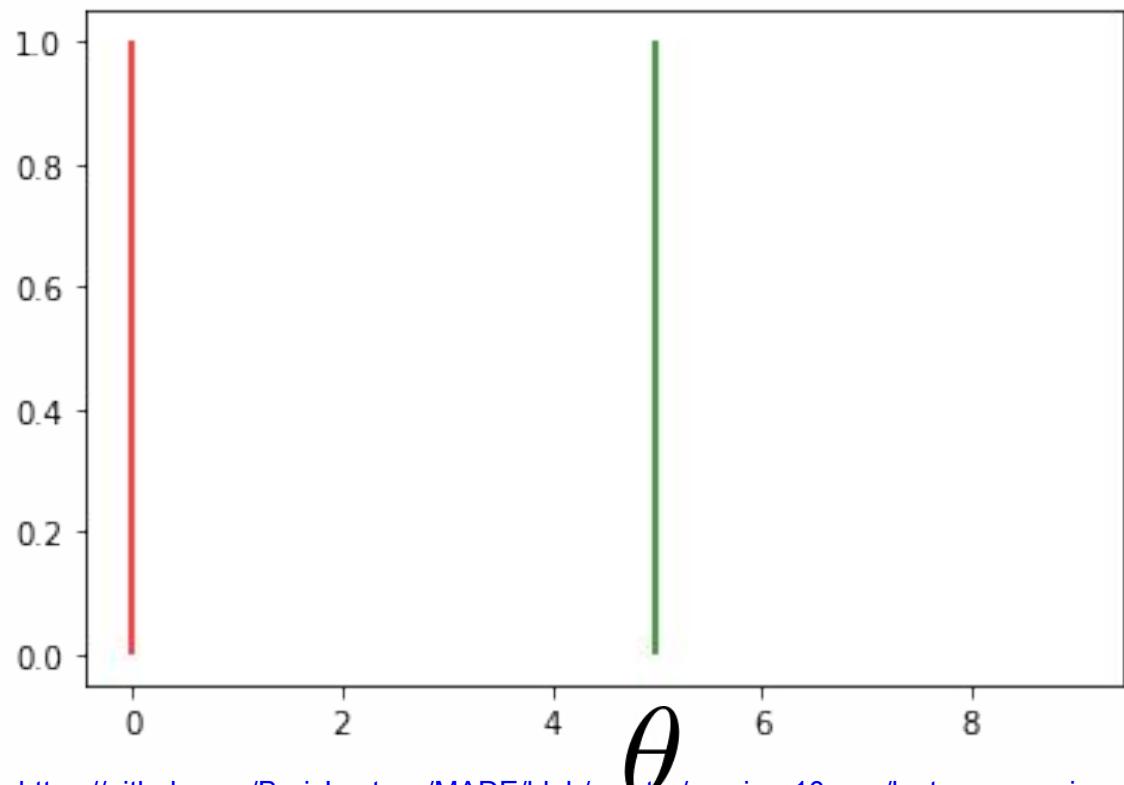


$$D_{\text{KL}}(P \parallel Q) = \begin{cases} \infty & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$
$$JSD(P \parallel Q) = \begin{cases} \ln(2) & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

# Проблемы KL и JS дивергенций. Пример

---

$$L(G, D) = -2 \log(2) + 2D_{JS}(p_r \parallel p_g)$$



$$D_{\text{KL}}(P \parallel Q) = \begin{cases} \infty & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

$$JSD(P \parallel Q) = \begin{cases} \ln(2) & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

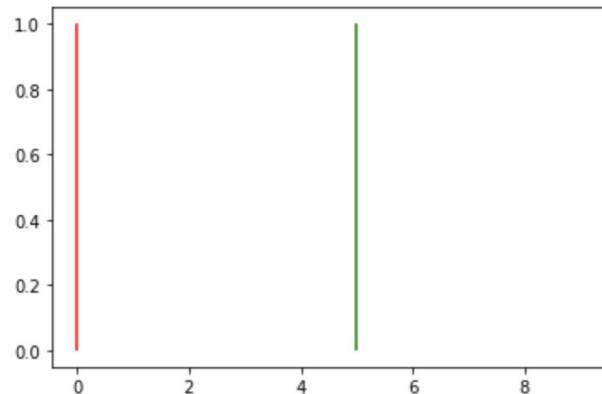
$$W_1(P \parallel Q) = \theta$$

# Проблемы KL и JS дивергенций. Пример

KL div: inf,

JS div: 0.6931471805599452,

EMD: 5.0

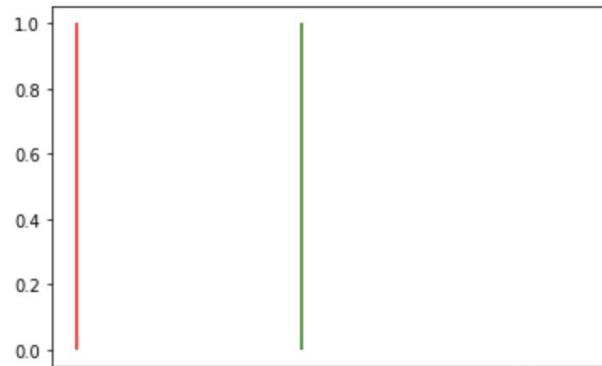


=====

KL div: inf,

JS div: 0.6931471805599452,

EMD: 4.0

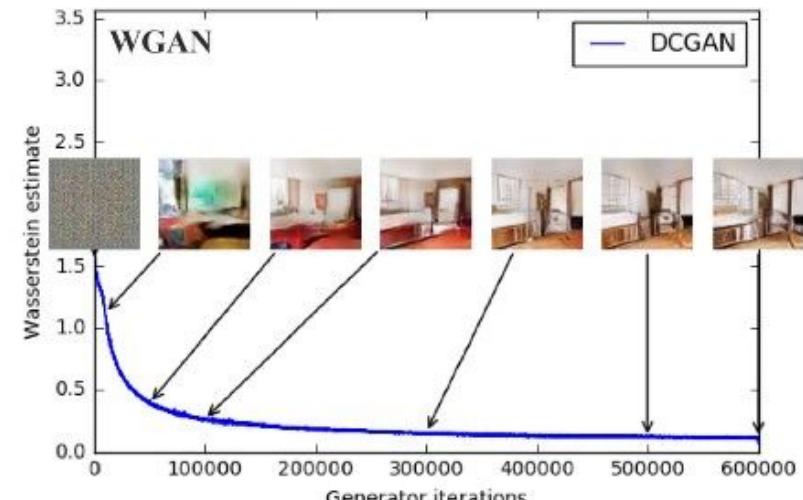
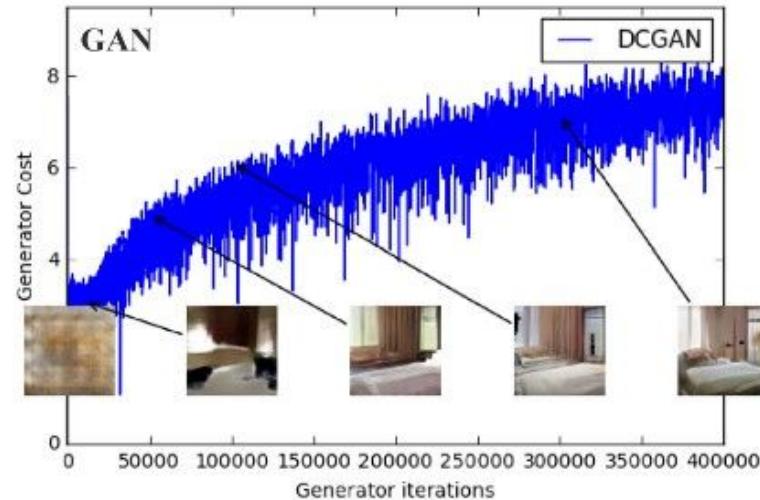
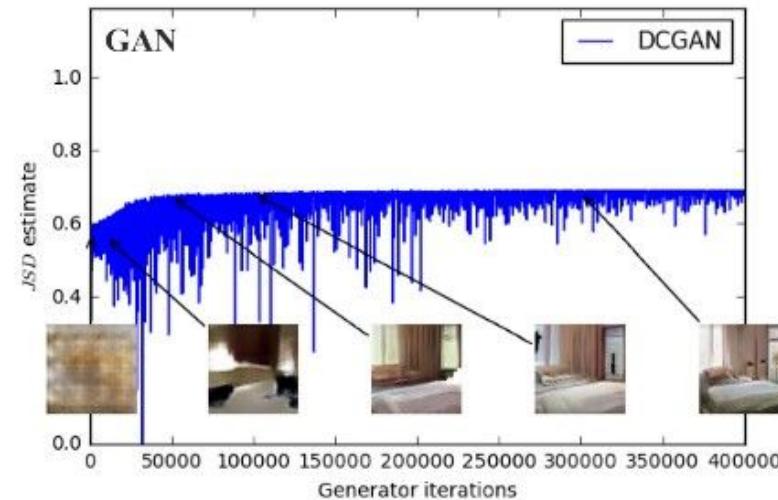


$$D_{\text{KL}}(P \parallel Q) = \begin{cases} \infty & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

$$JSD(P \parallel Q) = \begin{cases} \ln(2) & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

$$W_1(P \parallel Q) = \theta$$

# Wasserstein GAN. Качество зависит от лосса





# Что мы уже прошли. Часть 2

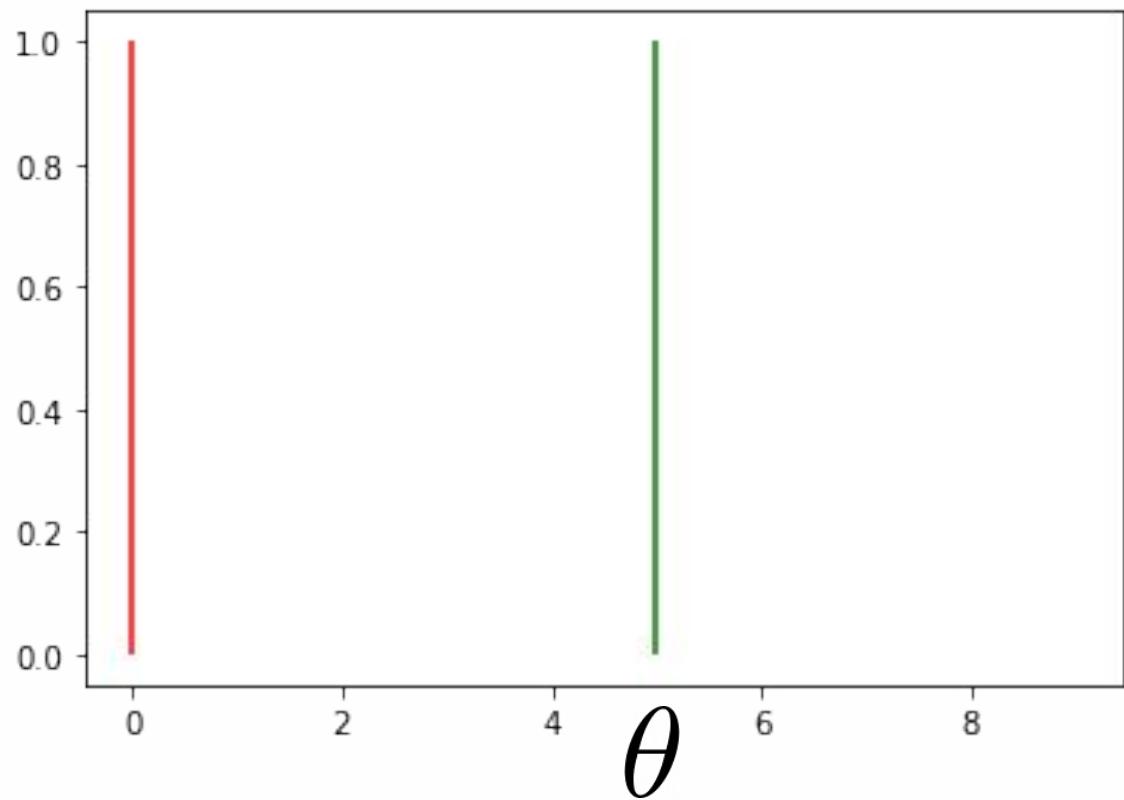
---

- ❖ Оригинальная статья:
  - учим дискриминатор как бинарный классификатор
  - используем функционал ошибки классификации
    - увидели, что он может быть выведен как сумма JS-дивергенции и константы
    - увидели, что это приводит к проблемам со сходимостью и интерпретируемостью лосса

# Проблемы KL и JS дивергенций. Пример

---

$$L(G, D) = -2 \log(2) + 2D_{JS}(p_r \parallel p_g)$$



$$D_{\text{KL}}(P \parallel Q) = \begin{cases} \infty & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

$$JSD(P \parallel Q) = \begin{cases} \ln(2) & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

$$W_1(P \parallel Q) = \theta$$



# EMD: Earth mover's distance. План

---

- ❖ Определение
- ❖ Дискретный пример
- ❖ Лосс с дистанцией Вассерштейна



# EMD: Earth mover's distance

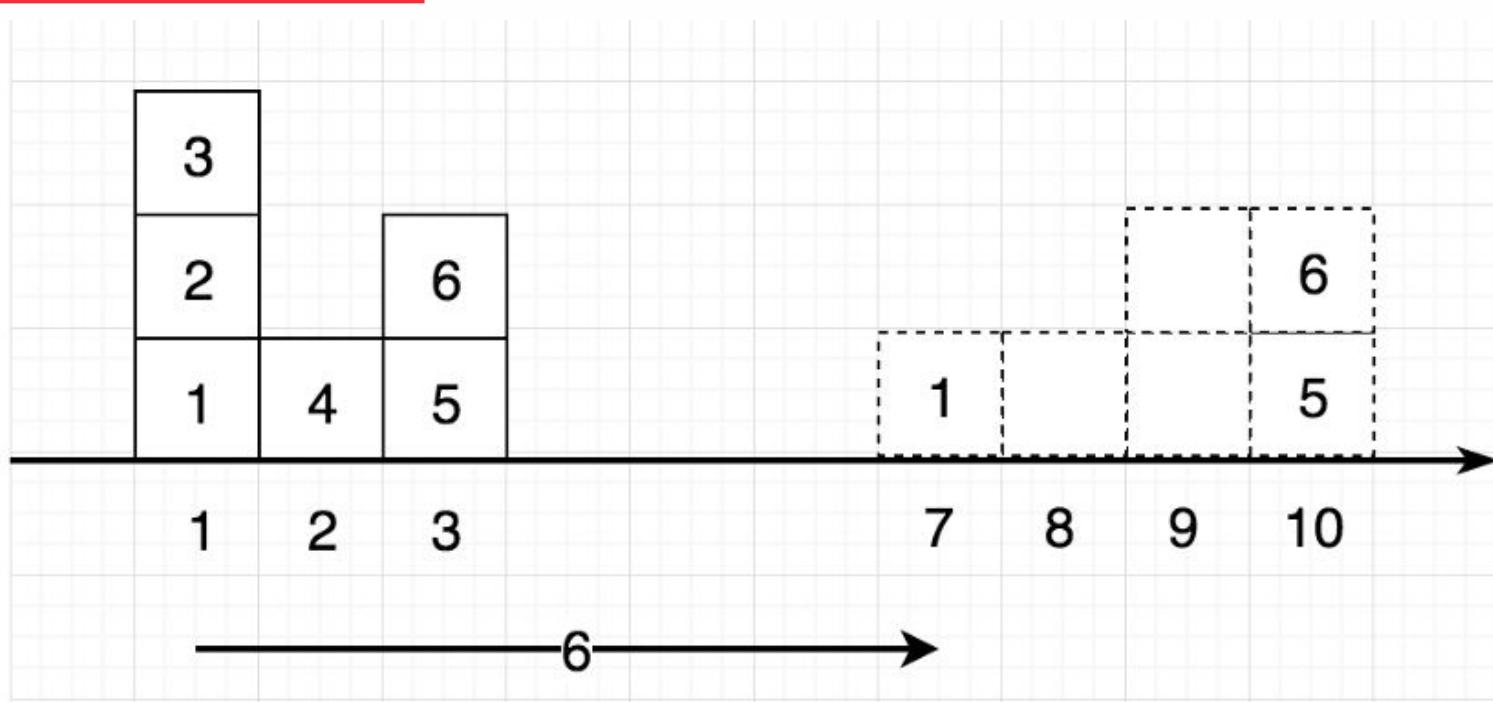
---

- ❖ Метрика Вассерштейна (Wasserstein-1 distance, Earth Mover's Distance) - метрика, вычисляемая по формуле:

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

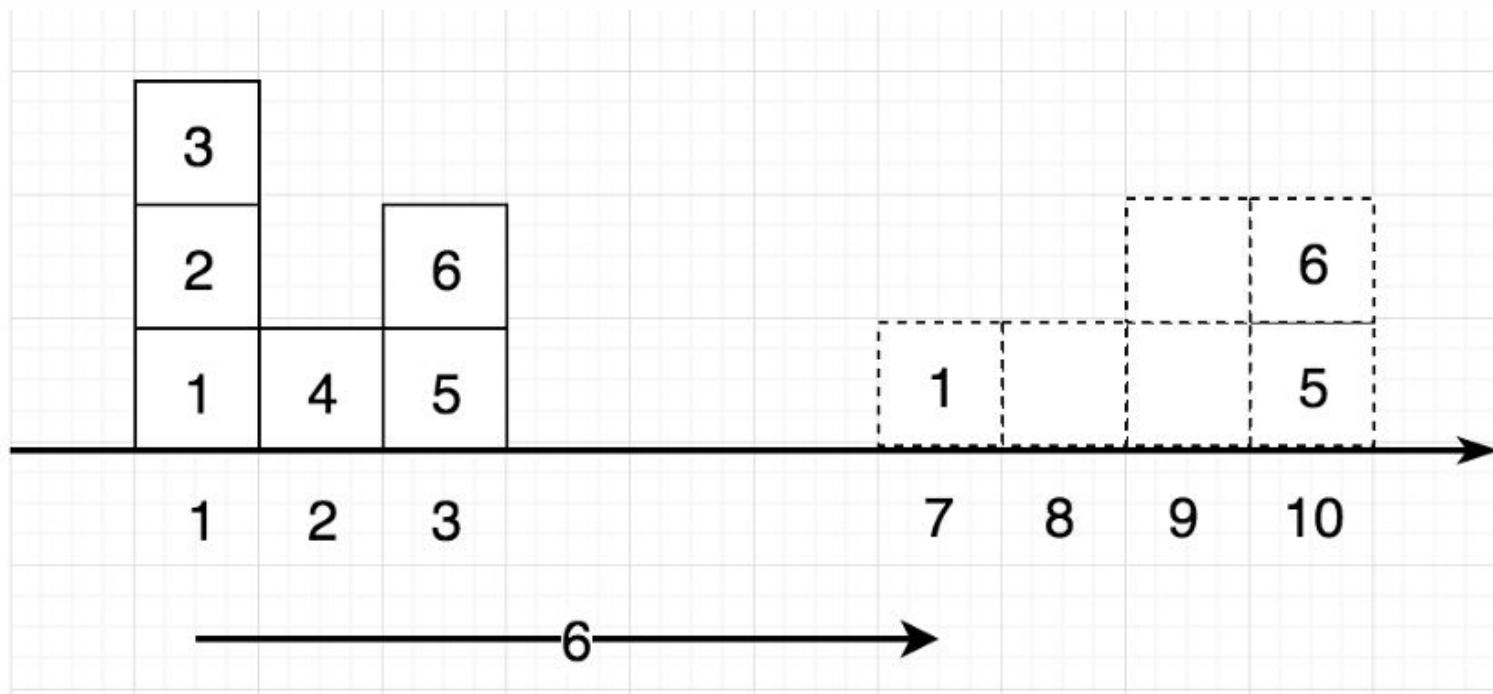
Также часто формулируется как минимальное количество энергии, которое нужно потратить, чтобы придать одной куче земли форму второй кучи земли

# EMD: Earth mover's distance. Пример



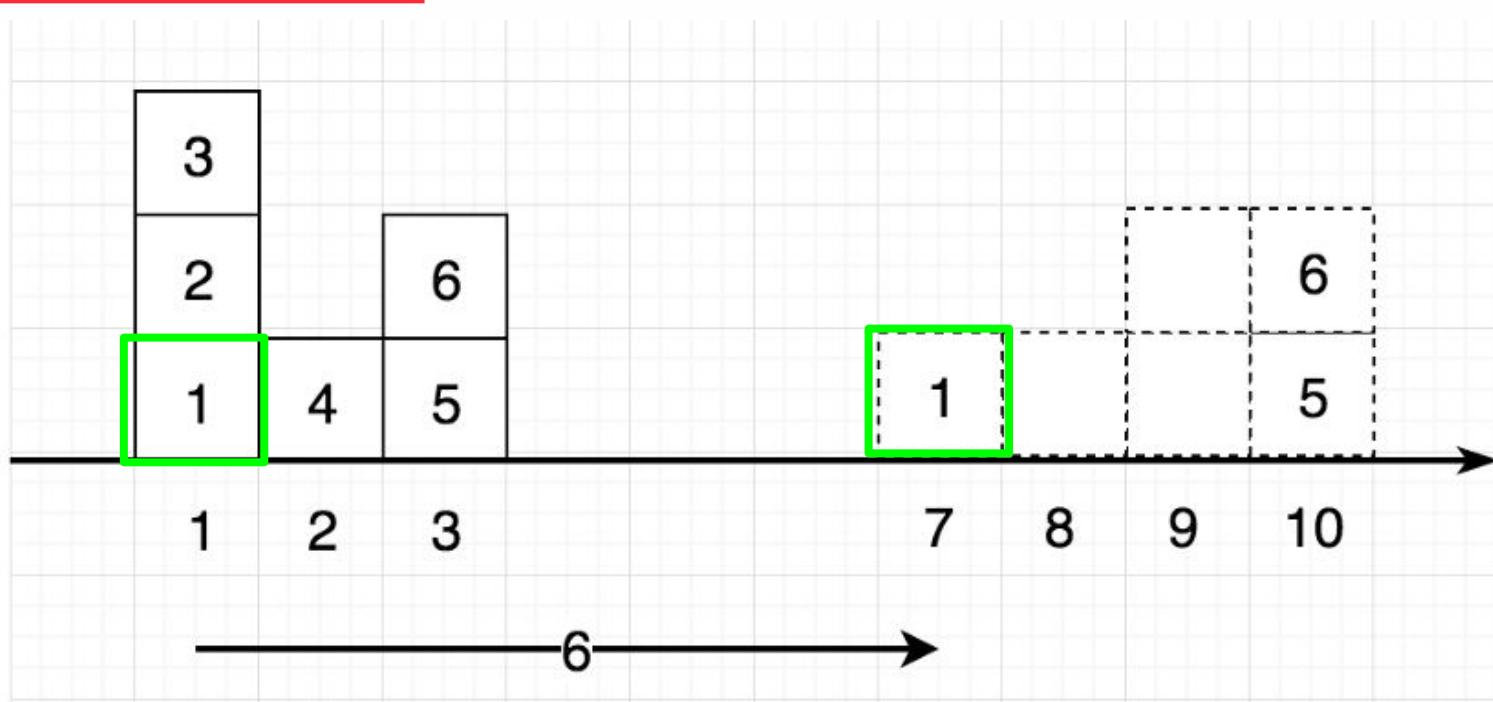
- ❖ Хотим поменять форму кучи из шести ящиков оптимальным способом

# EMD: Earth mover's distance. Пример



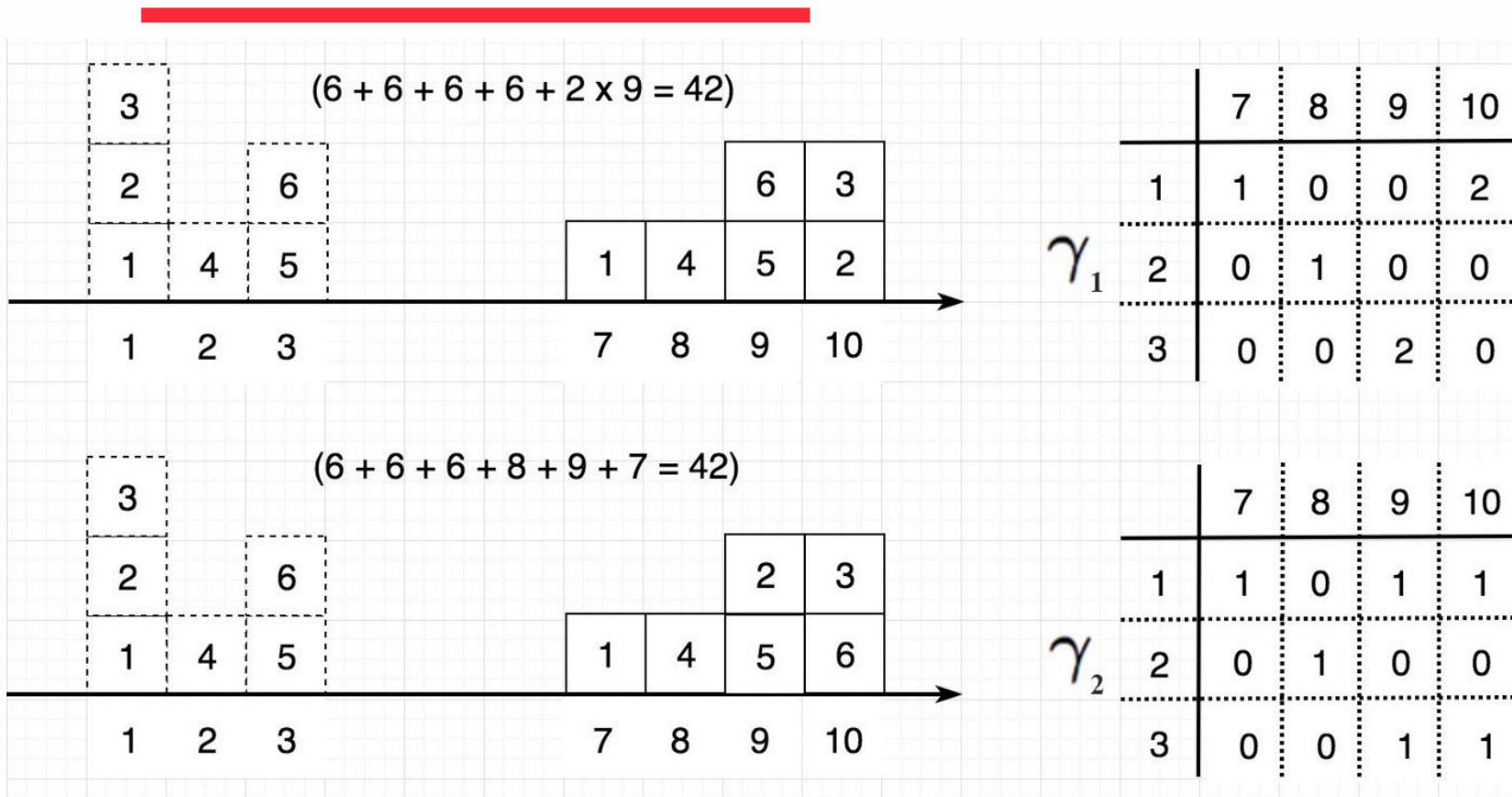
- ❖ Хотим поменять форму кучи из шести ящиков оптимальным способом
- ❖ Переносим бокс 1 из координаты 1 в координату 7. Стоимость  $7 - 1 = 6$

# EMD: Earth mover's distance. Пример



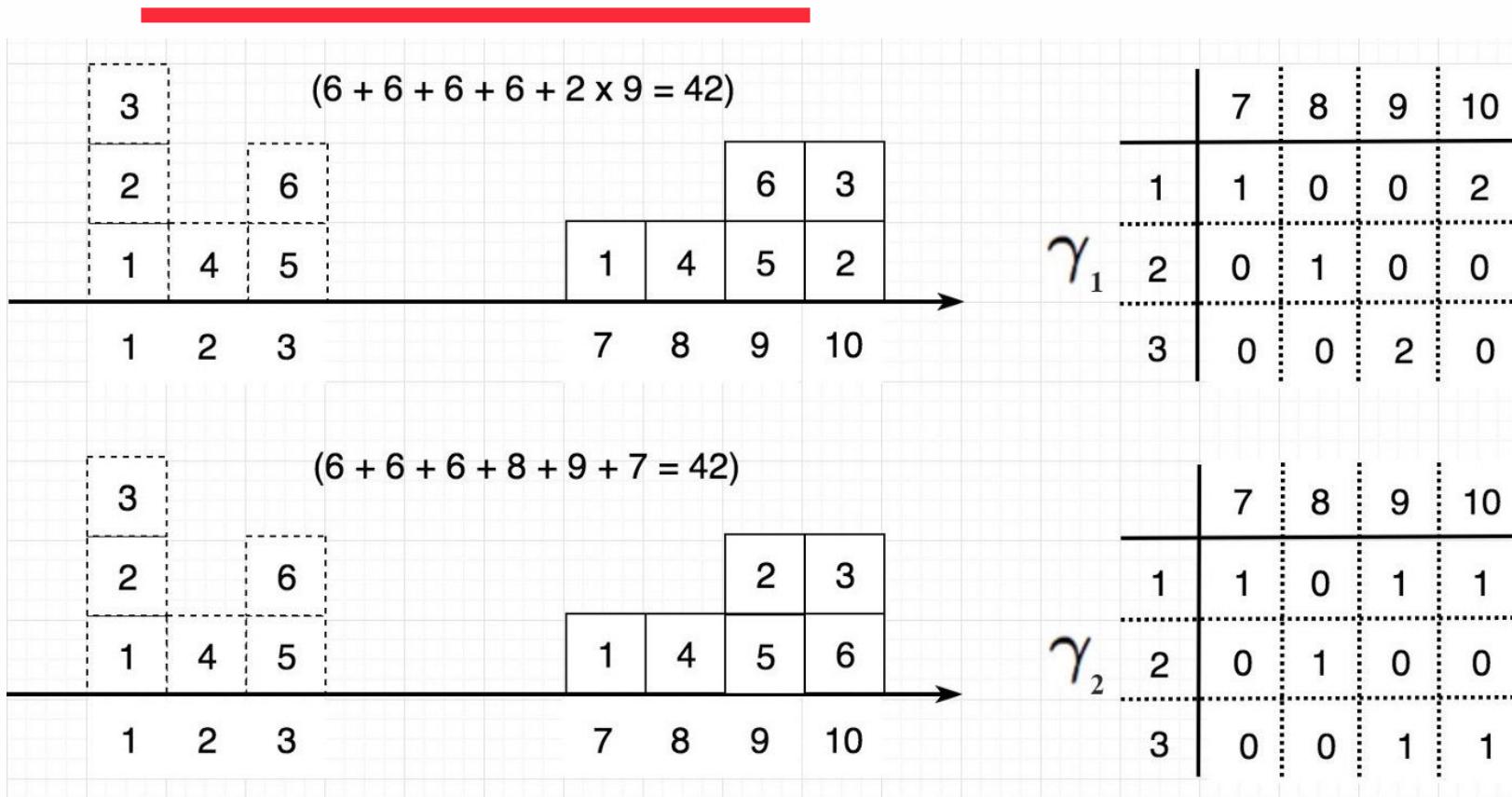
- ❖ Хотим поменять форму кучи из шести ящиков оптимальным способом
- ❖ Переносим бокс 1 из координаты 1 в координату 7. Стоимость  $7 - 1 = 6$

# EMD: Earth mover's distance. Пример



- ❖ Стратегий переноса может быть несколько.
- ❖ Обозначим  $\gamma_i$   $i$ -ую стратегию переноса

# EMD: Earth mover's distance. Пример



- ❖ Стратегий переноса может быть несколько.
- ❖ Обозначим  $\gamma_i$   $i$ -ую стратегию переноса
- ❖ Дистанция Вассерштейна - это стоимость самой дешевой стратегии переноса



# EMD: Earth mover's distance.

---

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

# EMD: Earth mover's distance.

probability distribution of  $x$  &  $y$   
- before and after the move

a.k.a. min

expected moving cost of the plan  $\gamma$

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

find one that has the lowest cost  
all possible moving plans  
distance between  $x$  &  $y$

# EMD: Earth mover's distance.

probability distribution of  $x$  &  $y$   
- before and after the move

a.k.a. min

expected moving cost of the plan  $\gamma$

find one that has the lowest cost

all possible moving plans

distance between  $x$  &  $y$

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

	7	8	9	10
1	1	0	0	2
2	0	1	0	0
3	0	0	2	0

# EMD: Earth mover's distance.

probability distribution of  $x$  &  $y$   
- before and after the move

a.k.a. min

expected moving cost of the plan  $\gamma$

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

find one that has the lowest cost

all possible moving plans

distance between  $x$  &  $y$

	7	8	9	10
1	1	0	0	2
2	0	1	0	0
3	0	0	2	0

	7	8	9	10
1	$\frac{1}{6}$	0	0	$\frac{2}{6}$
2	0	$\frac{1}{6}$	0	0
3	0	0	$\frac{2}{6}$	0

# EMD: Earth mover's distance.

probability distribution of  $x$  &  $y$   
- before and after the move

a.k.a. min

expected moving cost of the plan  $\gamma$

find one that has the lowest cost

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

all possible moving plans

distance between  $x$  &  $y$

	7	8	9	10
1	1	0	0	2
2	0	1	0	0
3	0	0	2	0

	7	8	9	10
1	$\frac{1}{6}$	0	0	$\frac{2}{6}$
2	0	$\frac{1}{6}$	0	0
3	0	0	$\frac{2}{6}$	0

$$\mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|]$$

$$= p(1, 7) \times 6 + p(1, 10) \times 9 + \dots$$

$$= 1/6 \times 6 + 2/6 \times 9 + \dots$$



# EMD: Earth mover's distance. Материалы

---

- ❖ <https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html#kullbackleibler-and-jensenshannon-divergence>
- ❖ <https://vincentherrmann.github.io/blog/wasserstein/>
- ❖ [https://medium.com/@jonathan\\_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490](https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490)
- ❖ [https://medium.com/@jonathan\\_hui/gan-spectral-normalization-893b6a4e8f53](https://medium.com/@jonathan_hui/gan-spectral-normalization-893b6a4e8f53)



# EMD: Earth mover's distance.

---

- ❖ Проблема: EMD сложно считать



# EMD: Earth mover's distance.

---

- ❖ Проблема: EMD сложно считать
- ❖ Решение: можно переформулировать выражение EMD



# EMD: Earth mover's distance.

---

- ❖ Проблема: EMD сложно считать
- ❖ Решение: можно переформулировать выражение EMD

**Было:**

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

# EMD: Earth mover's distance.

---

- ❖ Проблема: EMD сложно считать
- ❖ Решение: можно переформулировать выражение EMD

**Было:**

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

**Стало:**

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r} [f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta} [f(x)]$$

Точная верхняя грань по всем функциям  $f : \mathcal{X} \rightarrow \mathbb{R}$ , непрерывных по Лишнику с константой меньшей или равной единице (Используется дуализм Канторовича-Рубинштейна)  
<https://arxiv.org/pdf/1701.07875.pdf>



# Идея: перейти от JS-дивергенции к W-1

**Было:**

---

$$L(G, D) = -2 \log(2) + 2D_{JS}(p_r \parallel p_g)$$



# Идея: перейти от JS-дивергенции к W-1

---

**Было:**  $L(G, D) = -2 \log(2) + 2D_{JS}(p_r \parallel p_g)$

**Стало:**  $W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$



# Идея: перейти от JS-дивергенции к W-1

---

**Было:**  $L(G, D) = -2 \log(2) + 2D_{JS}(p_r \parallel p_g)$

**Стало:**  $W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$

<b>Discriminator/Critic</b>		<b>Generator</b>
<b>GAN</b>	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[ \log D\left(\mathbf{x}^{(i)}\right) + \log \left(1 - D\left(G\left(\mathbf{z}^{(i)}\right)\right)\right) \right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log \left(D\left(G\left(\mathbf{z}^{(i)}\right)\right)\right)$

# Идея: перейти от JS-дивергенции к W-1

**Было:**  $L(G, D) = -2 \log(2) + 2D_{JS}(p_r \parallel p_g)$

**Стало:**  $W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$

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

<https://arxiv.org/abs/1701.07875>

[https://medium.com/@jonathan\\_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490](https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490)

<https://www.alexirpan.com/2017/02/22/wasserstein-gan.html>

# Wasserstein GAN

---

- ❖ Решаемая задача:

**WGAN**

$$\nabla_w \frac{1}{m} \sum_{i=1}^m [f(x^{(i)}) - f(G(z^{(i)}))]$$

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^m f(G(z^{(i)}))$$

- ❖ Плюсы:

- быстрее и стабильнее сходимость
- меньше проблем с mode collapse

- ❖ Минусы:

- Дискриминатор должен быть 1-Липшиц непрерывной функцией  
(этого сложно достичь)

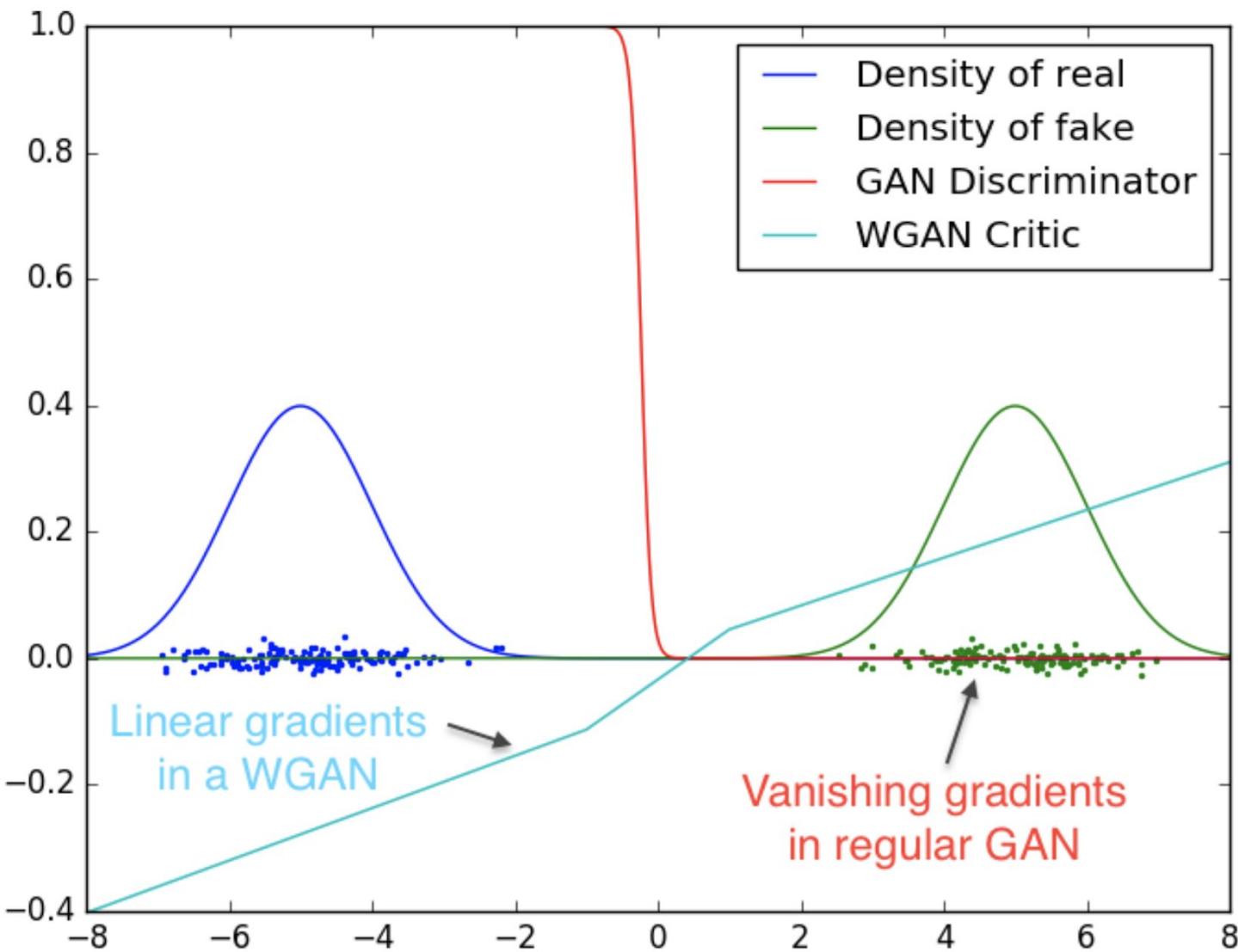


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians.

<https://arxiv.c> 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.



# Wasserstein GAN. Что почитать

---

- ❖ Оригинальная статья: <https://arxiv.org/abs/1701.07875>
- ❖ Обзор: <https://www.alexirpan.com/2017/02/22/wasserstein-gan.html>
- ❖ Разбор и объяснение деталей о Липшиц-непрерывности:  
<https://vincentherrmann.github.io/blog/wasserstein/>

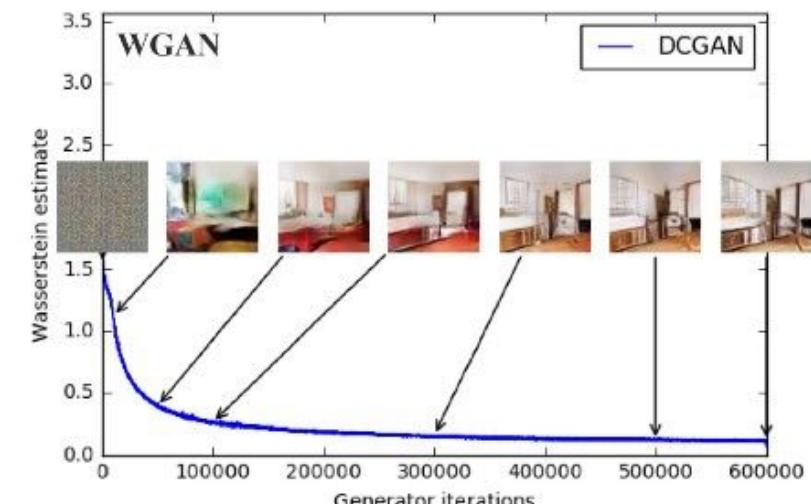
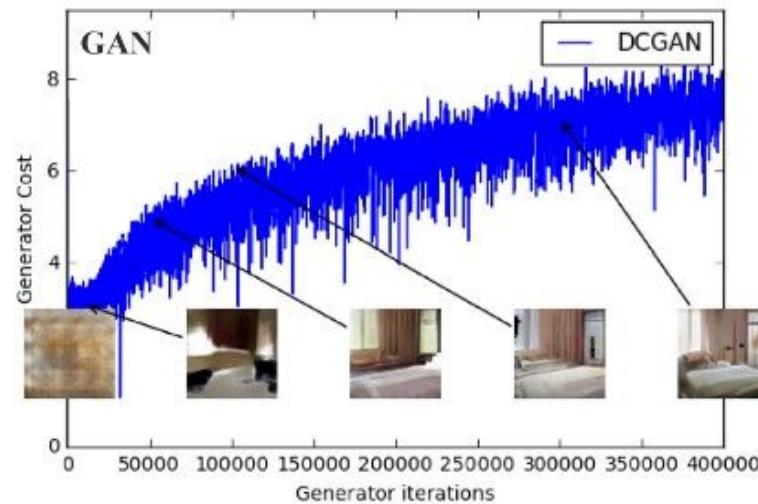
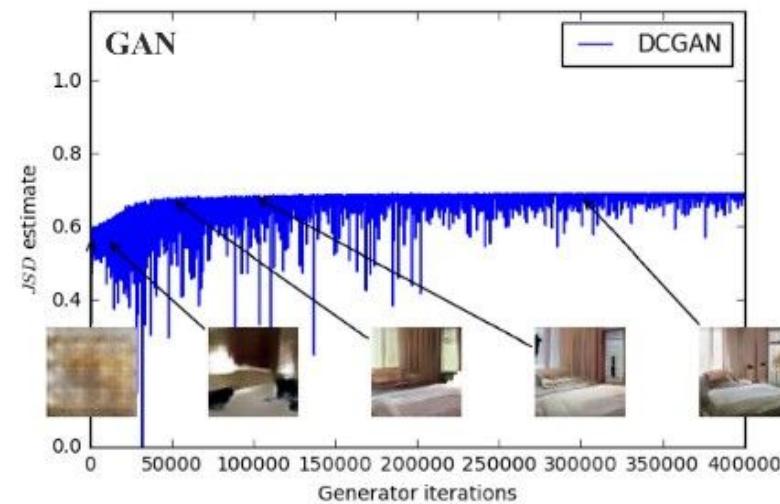


# Способы достижения Липшиц-непрерывности

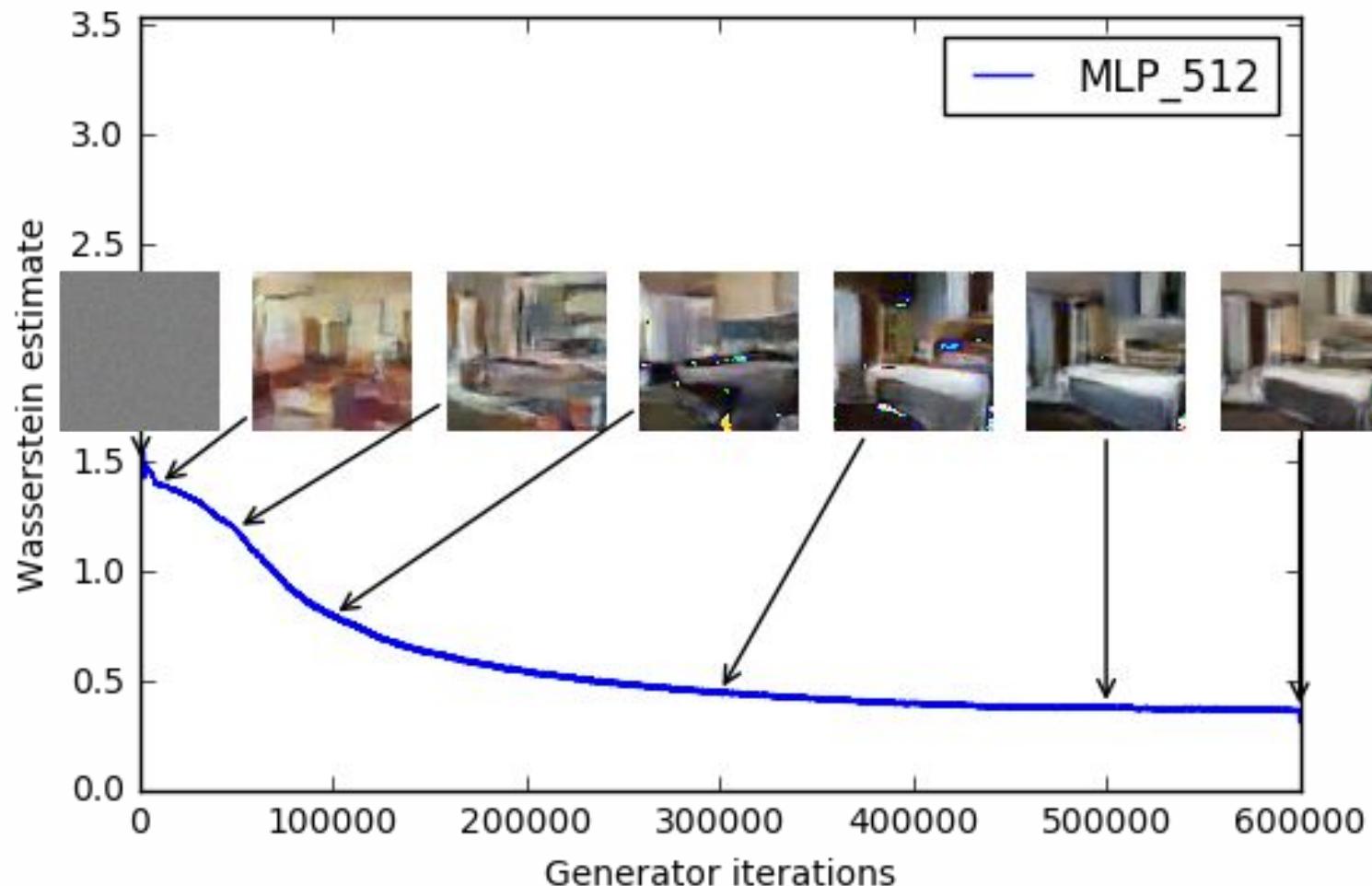
---

- ❖ Weight clipping - на каждой итерации обрезать веса до отрезка  $[-c; c]$ 
$$w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$$
$$w \leftarrow \text{clip}(w, -c, c)$$
- ❖ Gradient Penalty (WGAN-GP) - на каждой итерации делим градиент на его норму, чтобы норма итогового градиента не превышала 1
- ❖ Спектральная нормализация - делим каждый элемент матрицы весов на её спектральную норму
  - Спектральная норма - максимальное сингулярное число матрицы весов.

# Wasserstein GAN. Качество зависит от лосса



# Wasserstein GAN. Качество зависит от лосса



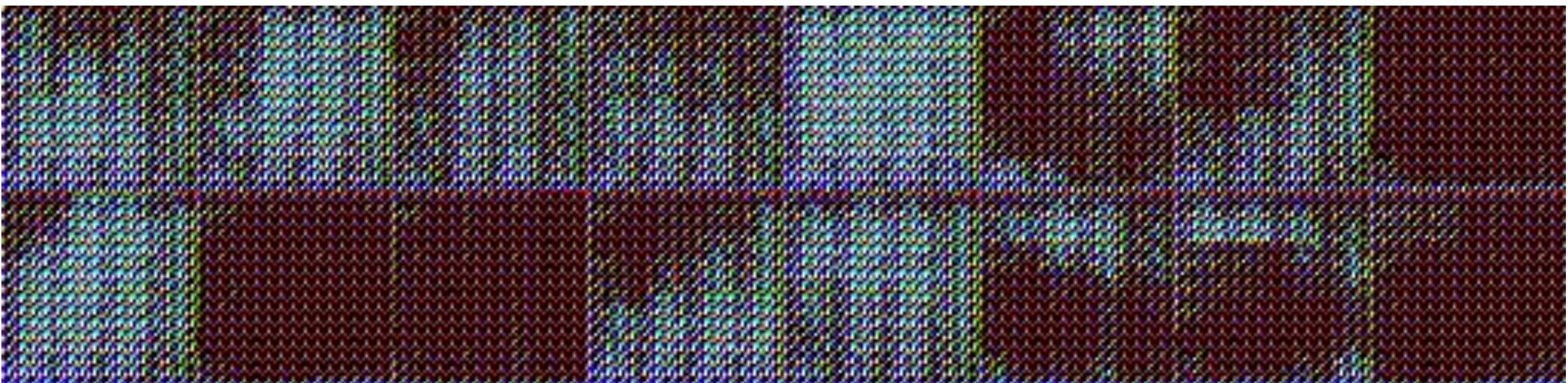
# Wasserstein GAN. Сравнение с DCGAN



Верхняя картинка: WGAN с архитектурой DCGAN. Нижняя - DCGAN

<https://arxiv.org/abs/1701.07875>

# Wasserstein GAN без BatchNorm



Верхняя картинка: WGAN с архитектурой DCGAN без BN. Нижняя - DCGAN без BN

<https://arxiv.org/abs/1701.07875>

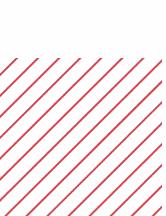


# Wasserstein GAN

---

## ❖ Статья:

- поднимает проблему использования JS-дивергентии в лоссе
  - предлагает замену: использование дистанции Вассерштейна
- Считать дистанцию Вассерштейна в лоб сложно
  - авторы переформулировывают проблему
  - получили более стабильную сходимость
- Теперь можем на каждой итерации обучения учить дискриминатор до сходимости с меньшей боязнью свалиться в mode collapse
  - значение лосса теперь коррелирует с качеством генерации
- Дискриминатор теперь называется Критиком:
  - Критик возвращает не вероятность, а действительное число
  - Отличается от дискриминатора отсутствием сигмоиды в конце



# Что мы уже прошли. Часть 1

---

- ❖ Оригинальная статья:
  - учим дискриминатор как бинарный классификатор
  - используем градиенты ошибки классификации для обучения дискриминатора и генератора
  - столкнулись с затуханием градиентов генератора
    - поменяли функционал потерь генератора
  - узнали, что идеальный дискриминатор пытается предсказать отношение плотностей распределений
  - столкнулись с проблемой mode collapse. В 2020 она до сих пор не решена
    - причиной считается поведение генератора, который стремится для всего латентного пространства вернуть одинаковый выход



# Что мы уже прошли. Часть 2

---

- ❖ Оригинальная статья:
  - учим дискриминатор как бинарный классификатор
  - используем функционал ошибки классификации
    - увидели, что он может быть выведен как сумма JS-дивергенции и константы
    - увидели, что это приводит к проблемам со сходимостью и интерпретируемостью лосса



# Что мы уже прошли. Часть 3

---

- ❖ Wasserstein GAN:
  - учим дискриминатор как бинарный классификатор
  - используем новый функционал ошибки классификации
    - дисперсия градиентов стала меньше
    - значение лосса стало интерпретируемо



# На следующей лекции:

---

- ❖ Современные подходы к генерации изображений
  - Progressive growing of GANs
  - BigGAN
  - AdaIN слои
  - StyleGAN и StyleGAN2
- ❖ Conditional GANs
- ❖ Метрики для сравнения результатов генерации

# Ссылки

---

- ❖ [Generative Adversarial Nets. Goodfellow et al.](#)
- ❖ [NIPS 2016 Tutorial: Generative Adversarial Networks. I. Goodfellow](#)
- ❖ [Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. Radford et al.](#)
- ❖ [Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Ioffe et al.](#)
- ❖ [Unrolled Generative Adversarial Networks. Metz et al.](#)
- ❖ <https://youtu.be/T4UuL7U5asA>
- ❖ <https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html#kullbackleibler-and-jensenshannon-divergence>
- ❖ <https://vincentherrmann.github.io/blog/wasserstein/>
- ❖ Progressive Growing of GANs: <https://www.youtube.com/watch?v=t640zZzIRBY>
- ❖ <https://arxiv.org/abs/1710.10196>
- ❖ <https://towardsdatascience.com/explained-a-style-based-generator-architecture-for-gans-generating-and-tuning-realistic-6cb2be0f431>
- ❖ <https://arxiv.org/pdf/1706.08224.pdf>
- ❖ Полезный и красивый семинар Саши Панчина из курса ШАДа по глубинному обучению:  
[https://colab.research.google.com/github/yandexdataschool/Practical\\_DL/blob/spring20/seminar08-generative/simple\\_1d\\_gan\\_pytorch.ipynb](https://colab.research.google.com/github/yandexdataschool/Practical_DL/blob/spring20/seminar08-generative/simple_1d_gan_pytorch.ipynb)