Instance-Conditioned GAN

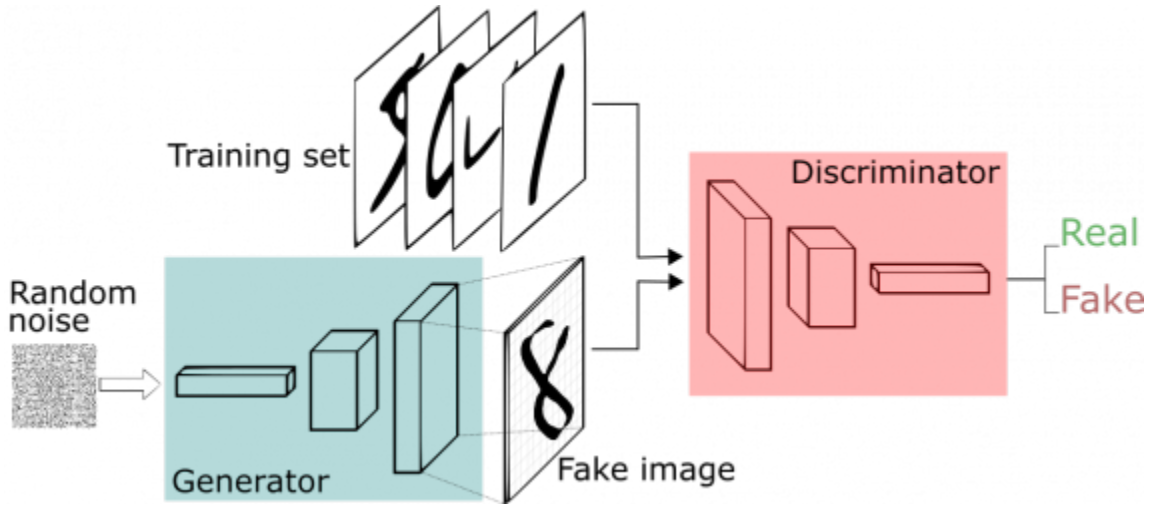
Чернышёв Александр, 617 группа

Напоминание: GAN

GAN (Generative adversarial network) — генеративно-состязательная сеть.

- Впервые GAN описан в работе Ian’a Goodfellow и других исследователей в 2014 году.
- Генератор генерирует образцы.
- Дискриминатор пытается отличить сгенерированные (fake) образцы от подлинных (real).

Обычно GAN не может полностью покрыть выборку.



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

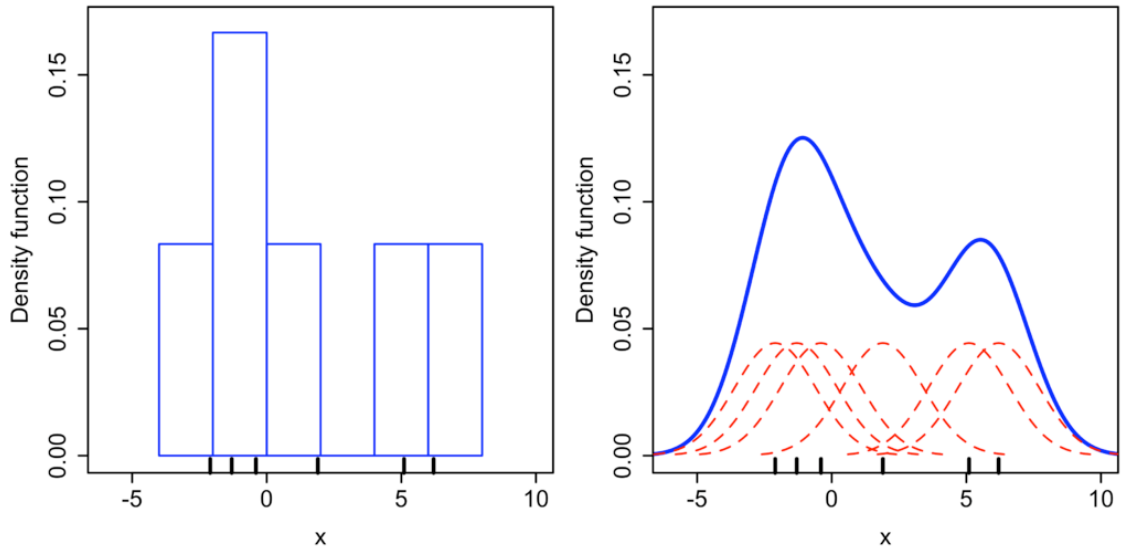
GAN: примеры из оригинальной статьи

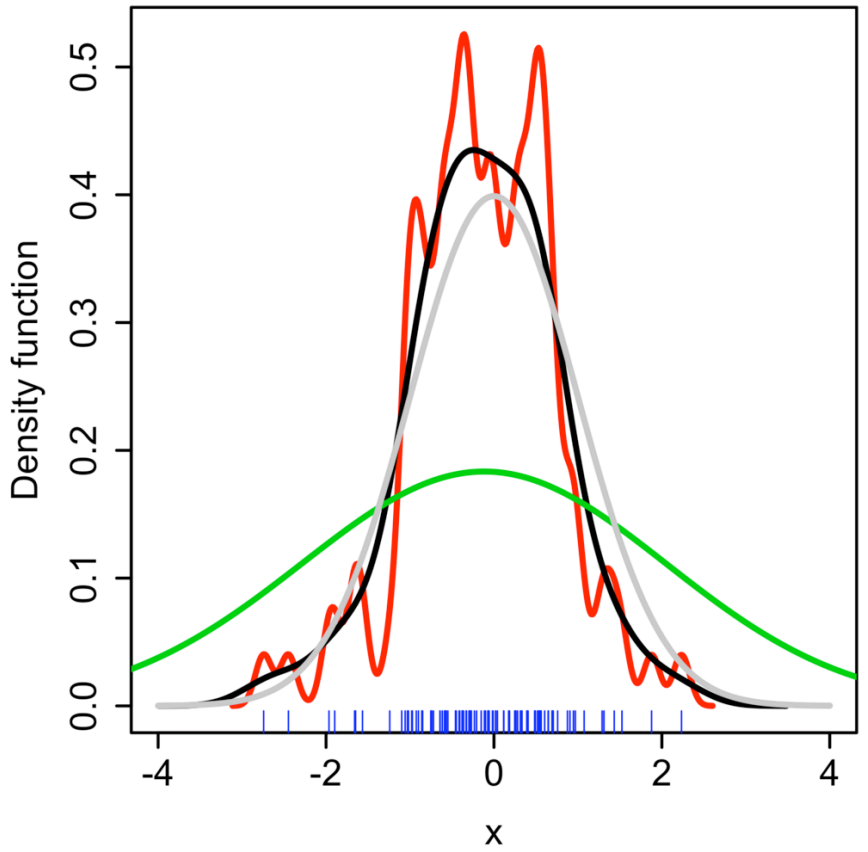
В правой колонке находится самый близкий к сгенерированному объект из обучающей выборки.

Напоминание: KDE

- KDE (Kernel Density Estimation, ядерная оценка плотности) — способ оценки плотности случайной величины.
- $K(x)$ — ядро, ядерная функция.
- h — ширина окна (сглаживающий параметр).

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

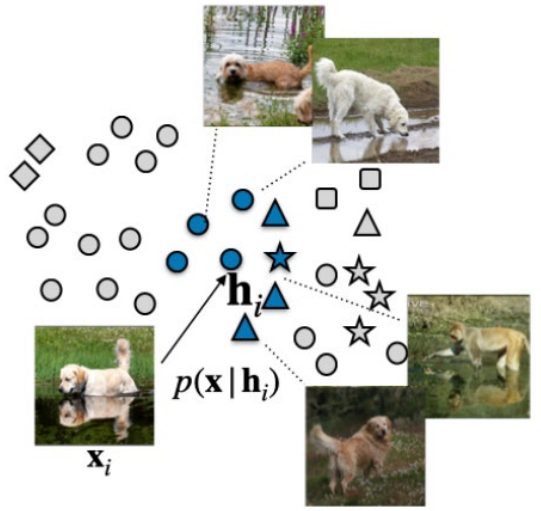




100 точек из стандартного нормального распределения.
Серая кривая: истинная плотность.
Красная кривая: KDE с h=0,05.
Чёрная кривая: KDE с h=0,337.
Зелёная кривая: KDE с h=2.

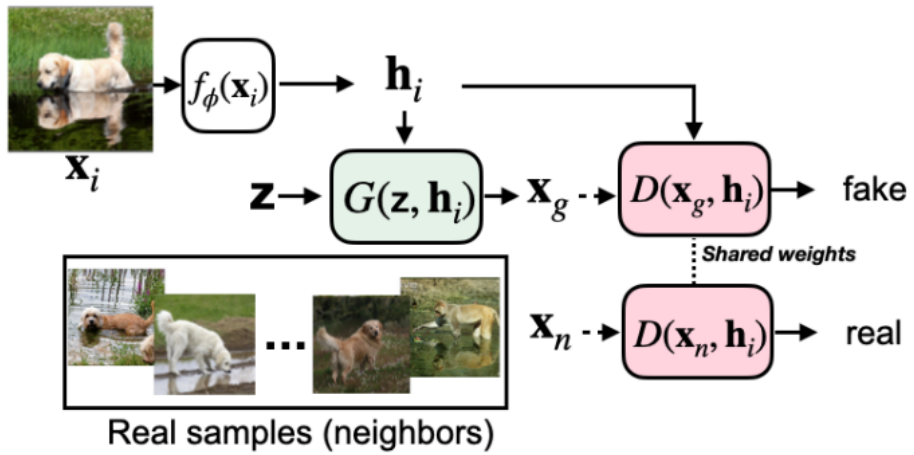
Instance-conditioned GAN

- Основная идея: моделируем распределение сложного набора данных $p(x)$ в виде взвешенной суммы распределений, обусловленных на объекты из набора данных $p(x \mid h_i)$: $p(\mathbf{x}) \approx \frac{1}{M} \sum_i p(\mathbf{x}|\mathbf{h}_i)$.
- $h_i = f_\phi(x_i)$, f_ϕ — функция для извлечения высокоуровневых признаков из объекта (глубокая CNN).
- Генератор и дискриминатор обусловим на h_i.
- Во время обучения:
  - Выбираем произвольный объект x_i из набора данных.
  - Берем множество ближайших к нему соседей (k штук): A_i.
  - Генератор и дискриминатор обуславливаем на $h_i = f_\phi(x_i)$.
  - Дискриминатору подаем на вход выход генератора с пометкой `fake` и соседей x_i из A_i с пометкой `real`.
  - Далее обучаем так же, как и обычный GAN.



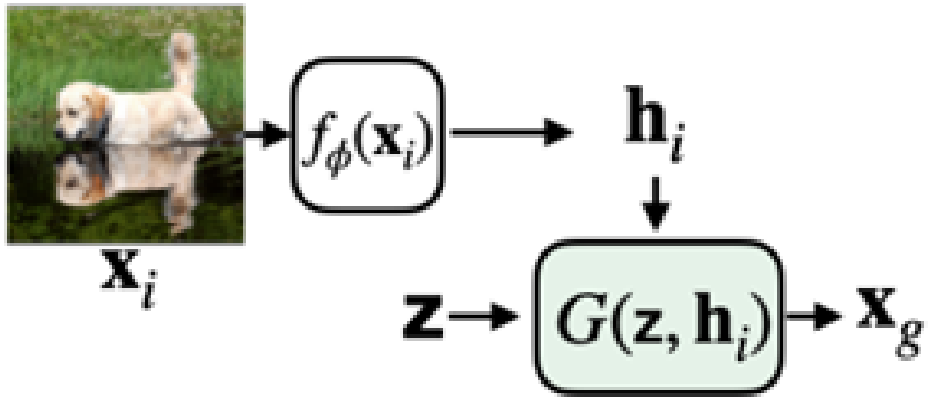
(a) Neighborhood $\mathcal{A}_i$ of instance $\mathbf{h}_i$

$$\min_G \max_D \mathbb{E}_{\mathbf{x}_i \sim p(\mathbf{x}), \mathbf{x}_n \sim \mathcal{U}(\mathcal{A}_i)} [\log D(\mathbf{x}_n, f_\phi(\mathbf{x}_i))] +$$
$$\mathbb{E}_{\mathbf{x}_i \sim p(\mathbf{x}), \mathbf{z} \sim p(\mathbf{z})} [\log(1 - D(G(\mathbf{z}, f_\phi(\mathbf{x}_i)), f_\phi(\mathbf{x}_i)))]$$



Instance-conditioned GAN

- Во время инференса:
  - Выбираем произвольный объект x_i из набора данных.
  - Генератор обуславливаем на $h_i = f_\phi(x_i)$ и (как и обычно) подаем шум на вход.



Instance-conditioned GAN

- Можно генератор и дискриминатор обусловить еще и на метку класса объекта.
- Во время обучения:
  - Выбираем произвольный объект x_i из набора данных.
  - Берем множество ближайших к нему соседей (k штук, не обязательно из того же класса): A_i.
  - Для каждого объекта (x_j, y_j) из A_i генератор и дискриминатор обуславливаем на $h_i = f_\phi(x_i)$ и на y_j.
  - Дискриминатору подаем на вход выход генератора с пометкой `fake` и соседей x_i из A_i с пометкой `real`.
  - Далее обучаем так же, как и обычный GAN.
- Во время инференса дан класс y:
  - Выбираем произвольный объект x_i из набора данных.
  - Генератор обуславливаем на $h_i = f_\phi(x_i)$ и на y, и (как и обычно) подаем шум на вход.

Эксперименты

- Наборы данных:
  - ImageNet;
  - COCO-Stuff;
  - ImageNet-LT;
  - Для transfer learning: Cityscapes, MetFaces, PACS и Sketches.
- Метрики качества:
  - Fréchet Inception Distance (FID);
  - Inception Score (IS);
  - LPIPS.
- Нейронные сети:
  - f_ϕ — ResNet50;
  - GAN — BigGan или StyleGAN2.

Результаты

- Неразмеченная выборка:

ImageNet

Method	Res.	↓FID	↑IS
Self-sup. GAN [41]	64	19.2*	16.5*
Uncond. BigGAN†	64	16.9* ± 0.0	14.6* ± 0.1
IC-GAN	64	10.4* ± 0.1	21.9* ± 0.1
IC-GAN + DA (**d**,**i**)	64	**9.2*** ± 0.0	**23.5*** ± 0.1
MGAN [23]	128	58.9	13.2
PacGAN2 [32]	128	57.5	13.5
Logo-GAN-AE [45]	128	50.9	14.4
Self-cond. GAN [33]	128	41.7	14.9
Uncond. BigGAN [36]	128	25.3	20.4
SS-cluster GAN [36]	128	22.0	23.5
PGMGAN [2]	128	21.7	23.3
IC-GAN	128	13.2 ± 0.0	45.5 ± 0.2
IC-GAN + DA (**d**,**i**)	128	**11.7** ± 0.0	**48.7** ± 0.1
ADM [12]	256	26.2	39.7
IC-GAN ($ch \times 64$)	256	17.0 ± 0.2	53.0 ± 0.4
IC-GAN ($ch \times 64$) + DA (**d**,**i**)	256	17.4 ± 0.1	53.5 ± 0.5
IC-GAN ($ch \times 96$) + DA (**d**)	256	**15.6** ± 0.1	**59.0** ± 0.4

COCO-Stuff (с transfer learning с ImageNet результаты лучше!)

		↓FID				↑ LPIPS
128×128	# prms.	train	eval	eval seen	eval unseen	eval
LostGANv2 [48]	41 M	12.8 ± 0.1	40.7 ± 0.3	80.0 ± 0.4	55.2 ± 0.5	0.45 ± 0.1
OC-GAN [49]	170 M	—	45.1 ± 0.3	85.8 ± 0.5	60.1 ± 0.2	0.13 ± 0.1
Unconditional (BigGAN)	18 M	17.9 ± 0.1	46.9 ± 0.5	103.8 ± 0.8	60.9 ± 0.7	0.68 ± 0.1
IC-GAN (BigGAN)	22 M	16.8 ± 0.1	44.9 ± 0.5	81.5 ± 1.3	60.5 ± 0.5	0.67 ± 0.1
IC-GAN (BigGAN, transf.)	77 M	**8.5** ± 0.0	**35.6** ± 0.2	77.0 ± 1.0	**48.9** ± 0.2	**0.69** ± 0.1
Unconditional (StyleGAN2)	23 M	8.8 ± 0.1	37.8 ± 0.2	92.1 ± 1.0	53.2 ± 0.5	0.68 ± 0.1
IC-GAN (StyleGAN2)	24 M	8.9 ± 0.0	36.2 ± 0.2	**74.3** ± 0.8	50.8 ± 0.3	0.67 ± 0.1
256×256						
LostGANv2 [48]	46 M	18.0 ± 0.1	47.6 ± 0.4	88.5 ± 0.4	62.0 ± 0.6	0.56 ± 0.1
OC-GAN [49]	190 M	—	57.0 ± 0.1	98.7 ± 1.2	71.4 ± 0.5	0.21 ± 0.1
Unconditional (BigGAN)	21 M	51.0 ± 0.1	81.6 ± 0.5	135.1 ± 1.6	95.8 ± 1.1	**0.77** ± 0.1
IC-GAN (BigGAN)	26 M	24.6 ± 0.1	53.1 ± 0.4	88.5 ± 1.8	69.1 ± 0.6	0.73 ± 0.1
IC-GAN (BigGAN, transf.)	90 M	13.9 ± 0.1	**40.9** ± 0.3	79.4 ± 1.2	**55.6** ± 0.6	0.76 ± 0.1
Unconditional (StyleGAN2)	23 M	**7.1** ± 0.0	44.6 ± 0.4	98.1 ± 1.7	59.9 ± 0.5	0.76 ± 0.1
IC-GAN (StyleGAN2)	25 M	9.6 ± 0.0	41.4 ± 0.2	**76.7** ± 0.6	57.5 ± 0.5	0.74 ± 0.1

Результаты

- Размеченная выборка (размечены классы):

ImageNet

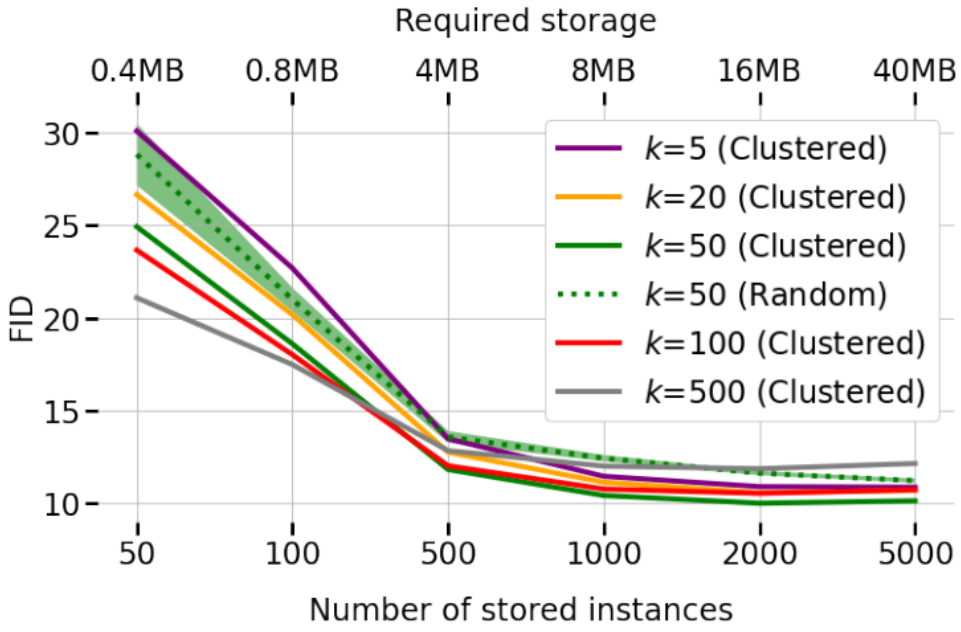
	Res.	↓FID	↑IS
BigGAN* [5]	64	12.3 ± 0.0	27.0 ± 0.2
BigGAN* [5] +DA (**d**)	64	10.2 ± 0.1	30.1 ± 0.1
IC-GAN	64	8.5 ± 0.0	39.7 ± 0.2
IC-GAN + DA(**d**, **i**)	64	**6.7** ± 0.0	**45.9** ± 0.3
BigGAN* [5]	128	9.4 ± 0.0	98.7 ± 1.1
BigGAN* [5]. DA(**d**)	128	**8.0** ± 0.0	107.2 ± 0.9
IC-GAN	128	10.6 ± 0.1	100.1 ± 0.5
IC-GAN + DA(**d**, **i**)	128	9.5 ± 0.1	**108.6** ± 0.7
BigGAN* [5] ($ch \times 64$)	256	8.0 ± 0.1	139.1 ± 0.3
BigGAN* [5] ($ch \times 64$) + DA(**d**)	256	8.3 ± 0.1	125.0 ± 1.1
IC-GAN ($ch \times 64$)	256	8.3 ± 0.1	143.7 ± 1.1
IC-GAN ($ch \times 64$) + DA(**d**, **i**)	256	**7.5** ± 0.0	152.6 ± 1.1
BigGAN† [5] ($ch \times 96$)	256	8.1	144.2
IC-GAN ($ch \times 96$) + DA(**d**)	256	8.2 ± 0.1	**173.8** ± 0.9

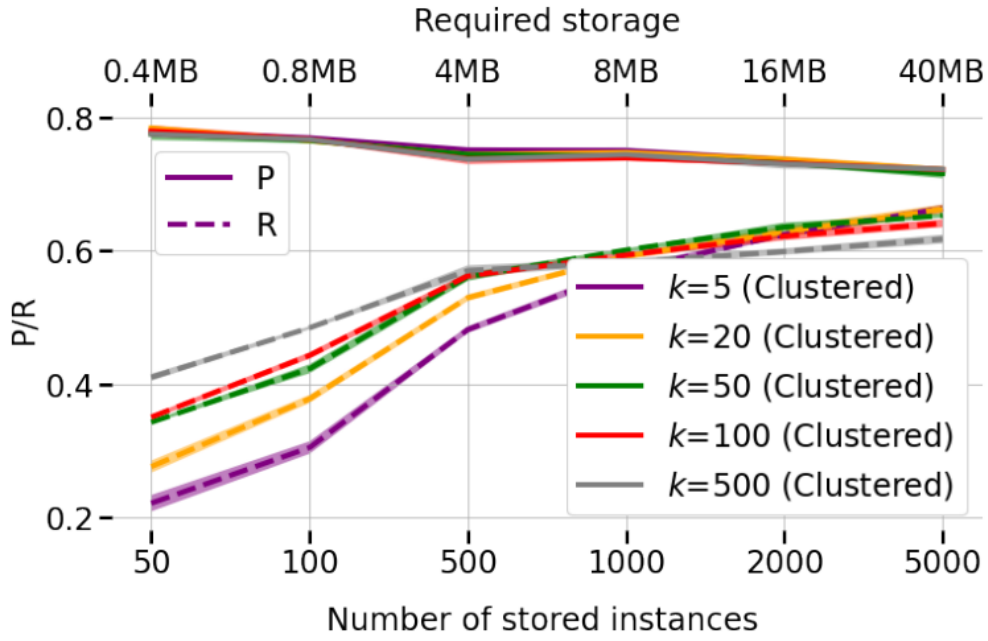
ImageNet-LT (бывает мало объектов при обучении)

	Res.	↓train FID	↑train IS	↓val FID	many/med/few ↓val FID	↑val IS
BigGAN* [5]	64	27.6 ± 0.1	18.1 ± 0.2	28.1 ± 0.1	28.8 / 32.8 / 48.4 ± 0.2	16.0 ± 0.1
IC-GAN	64	**23.2** ± 0.1	**19.5** ± 0.1	**23.4** ± 0.1	**23.8 / 28.0 / 42.7** ± 0.1	**17.6** ± 0.1
BigGAN* [5]	128	31.4 ± 0.1	30.6 ± 0.1	35.4 ± 0.1	34.0 / 43.5 / 64.4 ± 0.2	24.9 ± 0.2
IC-GAN	128	**23.4** ± 0.1	**39.6** ± 0.2	**24.9** ± 0.1	**24.3 / 31.4 / 53.6** ± 0.3	**32.5** ± 0.1
BigGAN* [5]	256	27.8 ± 0.0	58.2 ± 0.2	31.4 ± 0.1	28.1 / 40.9 / 67.6 ± 0.3	44.7 ± 0.2
IC-GAN	256	**21.7** ± 0.1	**66.5** ± 0.3	**23.4** ± 0.1	**20.6 / 32.4 / 60.0** ± 0.2	**51.7** ± 0.1

Подбор количества соседей и числа хранимых образцов

- k — количество соседей (аналог ширины окна в KDE).
- Кластеризуем все обучающие объекты с помощью k-means, сохраняем в модели наиближайшие к центроидам кластеров объекты и их соседей.
- Precision — про качество изображений, recall — про разнообразие.
- 1000 хранимых изображений достаточно, $k = 50$ — лучший выбор.





Примеры работы IC-GAN

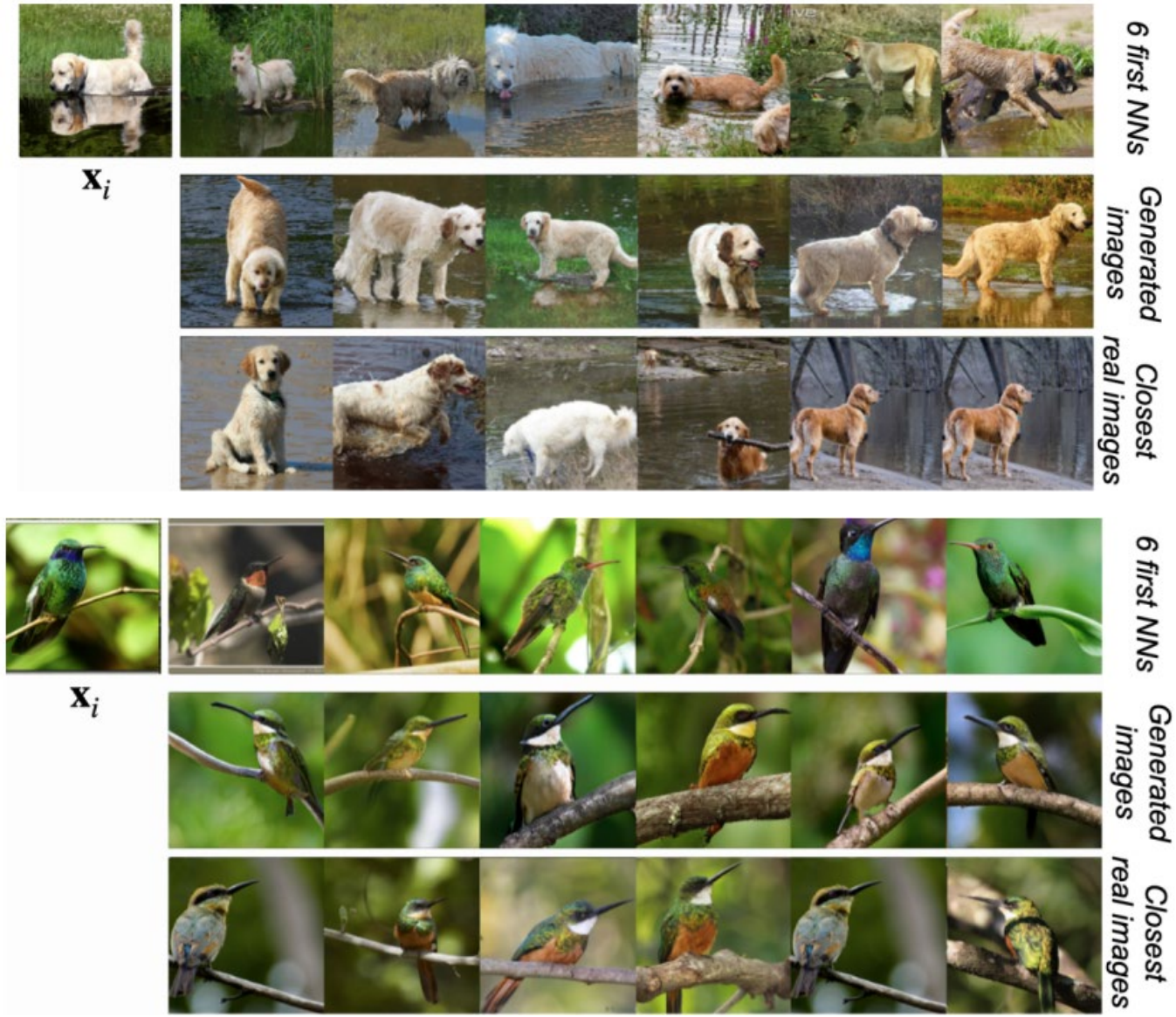
(a) IC-GAN samples

(b) Class-conditional IC-GAN samples

(c) IC-GAN transfer samples

(d) Class-conditional IC-GAN transfer samples

Примеры работы IC-GAN



Выводы

- Вклад:
  - IC-GAN без учета классов показывает лучшие результаты по сравнению с другими подходами.
  - Хорошо справляется с transfer learning, можно использовать на различных наборах данных без переобучения.
  - IC-GAN с учетом классов показывает результаты на уровне других подходов.
- Ограничения:
  - Нужно хранить часть выборки в модели (но 1000 объектов хватает).
  - Нужна предобученная сеть f_ϕ.
  - IC-GAN хорошо показывает себя на данных, на которых он не обучался, но если данные совсем не похожи на обучающую выборку, то качество сильно проседает.