Anycost GANs for Interactive Image Synthesis and Editing

Докладчик: Петр Молодык

Практик: Вадим Павлов

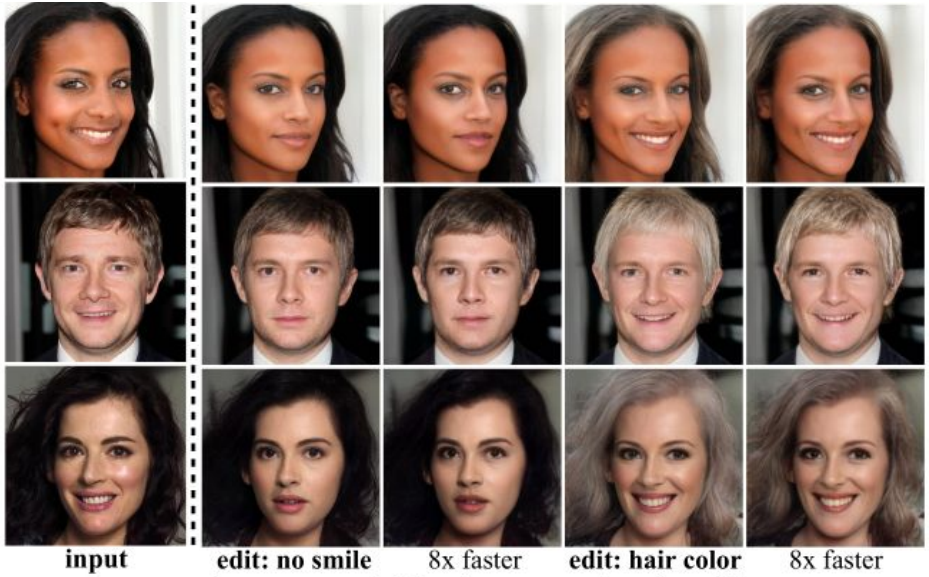
Рецензент: Ольга Агапова

Хакер: Артем Алекберов

Постановка задачи

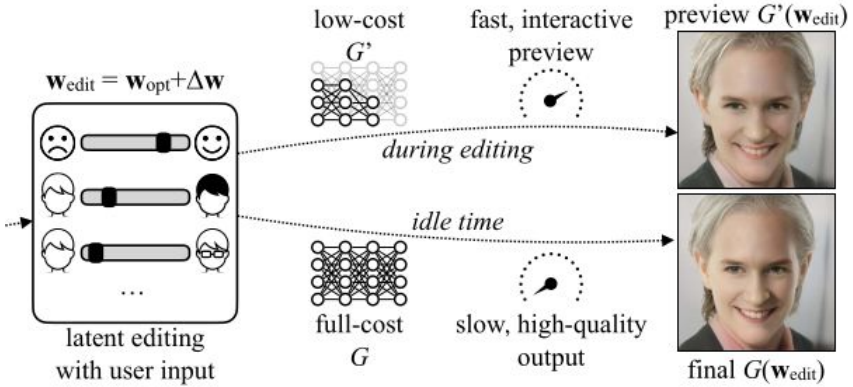
Создать GAN, который может генерировать изображения более низкого разрешения в разы быстрее чем полноразмерные, причем чтобы результаты “быстрого” режима были визуально похожи на полные.

Это нужно, например, в инструментах редактирования фото и видео для комфорта пользователей (полный GAN работает несколько секунд)



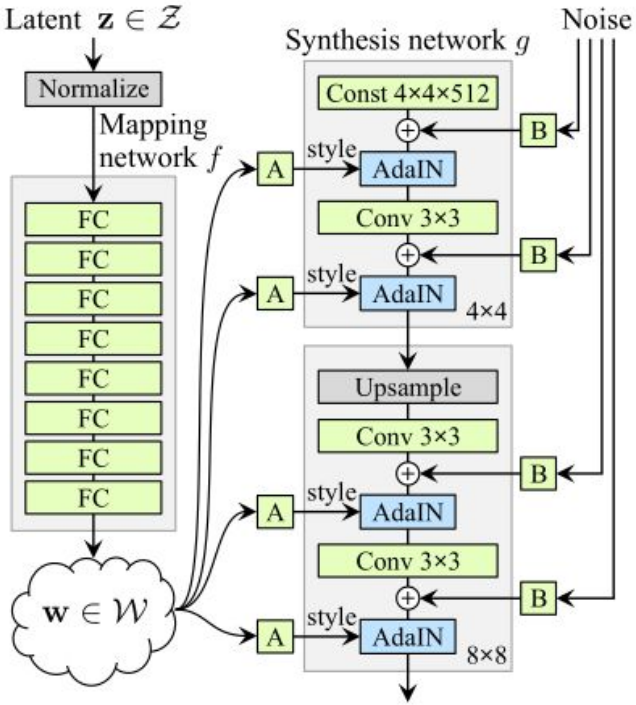
Идея AnycostGAN

Построим GAN, в котором можно будет брать подмножество весов, выбрасывая некоторые слои и каналы, и при этом все равно получать адекватный результат. Тогда выбирая количество слоев и каналов можно получать несколько градаций, обменивая скорость работы на качество.

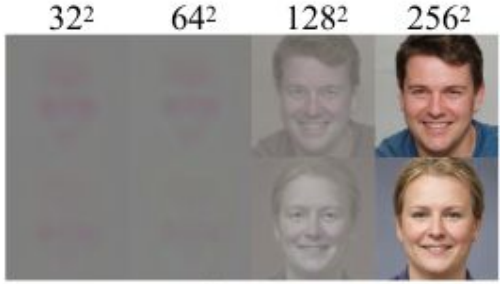


Вспомним StyleGAN

AnycostGAN использует схему генератора из StyleGAN, в которой шум сначала переводится набором полносвязных слоев в пространство большей размерности (в данном случае 18x512). В пространстве такой размерности можно проводить изменения, которые будут давать нужный эффект на итоговом изображении. В остальной части генератора мы будем использовать только часть весов и “останавливаться” на нужном разрешении.



Anycost - генератор



(a) vanilla StyleGAN2

В обычном StyleGAN уже есть слои, которые возвращают объекты меньшего разрешения, но сами по себе при стандартном обучении они не сохраняют изображение.

Авторы предлагают новую технику обучения - multi-resolution training. Первые k слоев генератора рассматриваются как отдельный генератор x_k, который должен возвращать изображения меньшего разрешения.

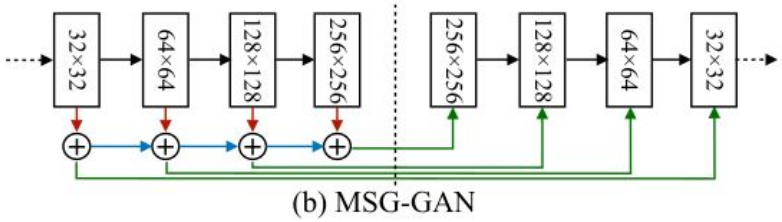
$$\tilde{\mathbf{x}} = G(\mathbf{w}) = g^K \circ g^{K-1} \circ ... \circ g^k \circ ... \circ g^2 \circ g^1(\mathbf{w})$$

$$\tilde{\mathbf{x}}^k = G^k(\mathbf{w}) = g^k \circ g^{k-1} \circ ... \circ g^2 \circ g^1(\mathbf{w}), k \leq K,$$

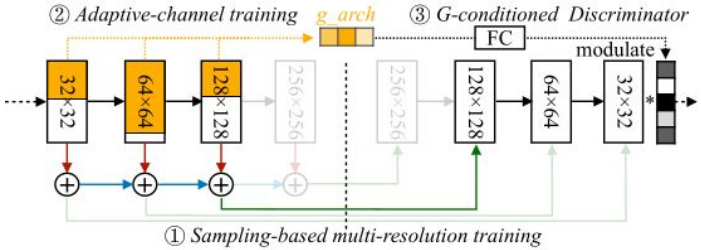
Multi-resolution training

Основа метода - семплирование разрешения на каждой итерации метода. Случайно выбирается k и используется генератор x_k, а слои более высокого разрешения не исполняются вовсе.

В MSG-GAN используется другой подход - в дискриминатор подаются сразу выходы всех промежуточных слоев, но это проигрывает Anycost-GAN на крупных датасетах.



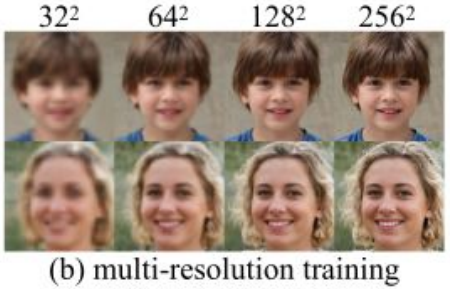
(b) MSG-GAN



(c) Anycost GAN

Подбор каналов

Такой генератор с лоссом $\mathcal{L}_{\text{multi-res}} = \mathbb{E}_{\mathbf{x},k}[\log D(\mathbf{x}^k)] + \mathbb{E}_{\mathbf{w},k}[\log(1 - D(G^k(\mathbf{w})))]$.



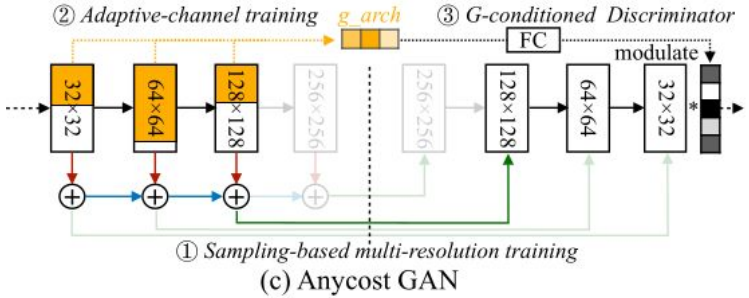
(b) multi-resolution training

уже вполне себе работает (см картинку b), но сам по себе выбор разрешения не позволяет достаточно сильно уменьшать количество вычислений. Надо еще убрать часть каналов. Для этого на каждом шаге для каждого слоя выберем число α_i - долю используемых каналов (оранжевые на схеме). При этом мы хотим брать не любые каналы, а самые важные, поэтому перед

выбором каналов мы их сортируем по модулю

чисел в ядре свертки и берем α_i наибольших.

Тогда лосс принимает вид:



(c) Anycost GAN

$$\mathcal{L}_{\text{ada-ch}} = \mathbb{E}_{\mathbf{x},k}[\log D(\mathbf{x}^k)] + \mathbb{E}_{\mathbf{w},k,\mathbb{C}}[\log(1 - D(G^k_{\mathbb{C}}(\mathbf{w})))].$$

Постоянство приближений

Теперь модель работает быстро и изображения вполне адекватные, но важно, чтобы предсказания разного разрешения были визуально как можно ближе к результату полной модели, чтобы их можно было использовать в качестве превью при редактировании. Для этого в лосс добавляется еще один компонент, который отвечает за схожесть предсказаний неполных генераторов.

В качестве расстояния авторы используют комбинацию MSE и LPIPS.

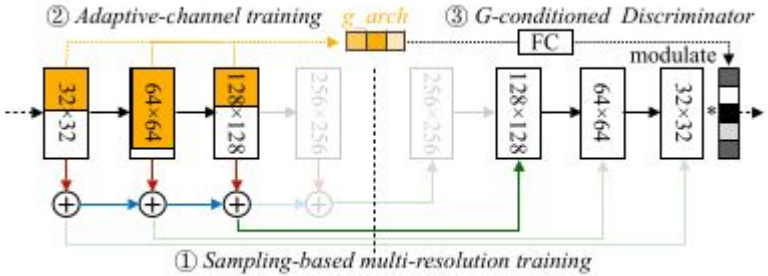
$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{ada-ch}} + \mathbb{E}_{\mathbf{w},k,\mathbb{C}}[\ell(G^k_{\mathbb{C}}(\mathbf{w}), G(\mathbf{w}))]$$

(c) w/o consistency loss


(d) w/ consistency loss

Дискриминатор

Обычный дискриминатор не справляется с тем, чтобы адекватно обучаться в ситуации, когда постоянно меняется количество каналов генератора, поэтому при обучении на каждом шаге после выбора каналов в генераторе выбранный набор кодируется и полученный вектор пропускается через полносвязный слой g_arch, который потом поэлементно домножается на каналы последнего слоя дискриминатора. Таким образом дискриминатор получает информацию о выбранных в генераторе слоях.



(c) Anycost GAN

Редактирование изображений

Для редактирования надо сначала спроектировать изображение в пространство W, а потом меняя проекцию менять выход генератора.

Сначала обучим модель-проектор: $E^* = \arg\min_E \mathbb{E}_{\mathbf{x}} \ell(G(E(\mathbf{x})), \mathbf{x})$

Потом для конкретной картинки улучшим проекцию: $\mathbf{w}^* = \arg\min_{\mathbf{w}} \ell(G(\mathbf{w}), \mathbf{x})$

На практике распределение проектора отличается от распределения изначального шума, поэтому надо отдельно проследить за постоянством.

$$E^* = \arg\min_E \mathbb{E}_{\mathbf{x}}[\ell(G(E(\mathbf{x})), \mathbf{x}) + \alpha \mathbb{E}_{k,\mathbb{C}} \ell(G^k_{\mathbb{C}}(E(\mathbf{x})), \mathbf{x})]$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}}[\ell(G(\mathbf{w}), \mathbf{x}) + \alpha \mathbb{E}_{k,\mathbb{C}} \ell(G^k_{\mathbb{C}}(\mathbf{w}, \mathbf{x})]$$

Результаты

Авторы взяли генератор на базе StyleGAN2. Тестировали на FFHQ(лица) и на LSUN(машины).

Генерировали объекты 4 разрешений (1024, 512, 256, 128 для FFHQ).

Выбирали долю каналов (α) из $[0.25, 0.5, 0.75, 1]$

Сравнивали **FID** с MSG-GAN и с отдельными ганами по разрешениям.

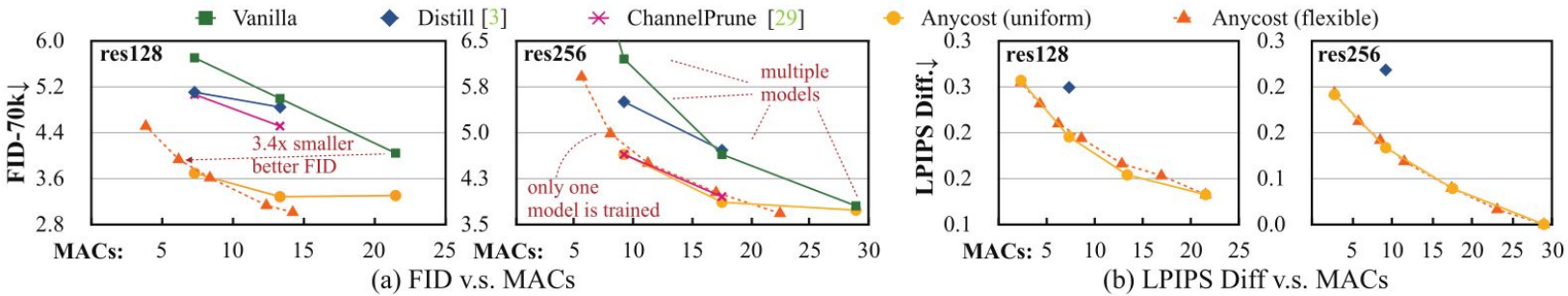
Resolution	1024	512	256	128	64	32
Single-resolution	3.25	4.17	3.76	4.04	3.32	2.41
MSG-GAN [41]	-	-	4.79	6.34	2.7	3.04
Ours (low res)	-	-	3.49	**3.26**	**2.52**	**2.18**
Ours (high res)	**2.99**	**3.08**	**3.35**	3.98	-	-

Сравнение с другими методами сжатия GAN

Vanilla - отдельные ганы по размерностям

ChannelPrune - техника сжатия сверточных слоев

uniform/flexible - способ выбора доли слоев (для каждого слоя или одна на всех)



Практик-исследователь

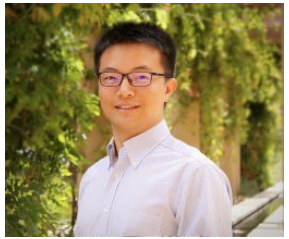
Авторы статьи

Ji Lin - Ph.D. student, MIT.

- Много статей про сжатие нейронных сетей
  - Defensive quantization: When efficiency meets robustness
  - Mcunet: Tiny deep learning on iot devices, Runtime neural pruning
- У него уже была статья, где применялась дистилляция к Conditional GAN
  - Gan compression: Efficient architectures for interactive conditional gans.

Jun-Yan Zhu - Assistant Professor, Carnegie Mellon University

- Много статей про generative adversarial networks
- Автор CycleGAN
- Соавтор статьи про дистилляцию Conditional GAN



Song Han - Assistant Professor, MIT EECS

- Много статей про сжатие нейронных сетей
  - Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding - 6к цитирований

История статьи

1. 1 февраля 2019 года выпущена статья “Compressing GANs using Knowledge Distillation” - используется обычная дистилляция для сжатия DCGAN
2. 19 марта 2020 года Ji Lin и Jun-Yan Zhu выпускают статью “GAN Compression: Efficient Architectures for Interactive Conditional GANs” - используется поканальная дистилляция (при этом дискриминатор не меняют). Принята на CVPR 2020
3. 4 марта 2021 года выпущена статья “Anycost GANs for Interactive Image Synthesis and Editing”. Принята на CVPR 2021
4. Конференция проходила онлайн в формате постеров. Никаких наград статья не получила.

> MAIN CONFERENCE
>
> All papers will be presented in the same manner. Each paper will have a five minute pre-recorded video and a PDF of the poster. An asynchronous text chat will be available for each paper. Attendees can view the papers and videos on demand at any time. Authors will also have individual Q&A sessions at the posted times below.
>
> All posted times are EDT but the chart linked below has all time zones’ conversions. When the virtual site is up, you will be able to select which sessions you are interested in and it will populate your own schedule.

5. На данный момент статья имеет 2 цитирования
   a. В первой перечисляют при обзоре области
   b. Во второй указывают как пример эффективного по времени работы GAN

Что дальше?

- Попробовать применить новые техники обратно к conditional GAN или другим GANам
- Сделать более быстрым обучение GANов
- Имплементировать это решение в фоторедакторы, как изначально и хотели авторы

Теперь немного кода

Две идеи статьи для пробы

1. В процессе обучения вырезать часть параметров из самой нейросети в зависимости от важности и обучать сразу и полную модель, и случайную подмодель с самыми важными на текущий момент весами

   А что будет, если попробовать не свертки, а линейные слои?

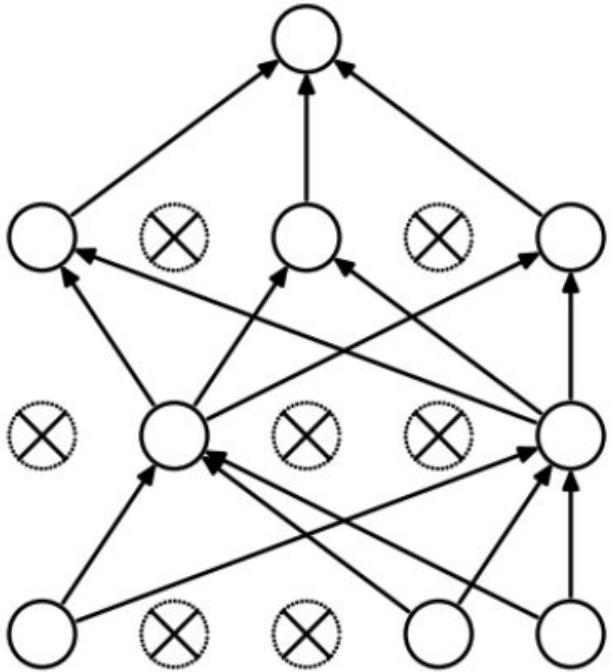
2. Обучать результат в разных разрешениях на разных уровнях нейросети

   Какой аналог разного разрешения есть в NLP?
   Например, CLS токен в трансформерах

Адаптация для линейных слоев

1. Считаем сумму квадратов весов, исходящих из данного нейрона

2. Зануляем k нейронов с минимальной суммой



Реализация

```
def forward(self, x, scale=None, return_output=False):
    output_full = x @ self.weight.t() + self.bias
    if return_output:
        return output_full
    if scale is not None:
        mangitudes = (self.weight ** 2).sum(0)
        threshhold = torch.kthvalue(mangitudes, int(len(mangitudes) * scale)).values.item()
        strong_idxs = torch.zeros(len(mangitudes))
        strong_idxs[torch.arange(len(mangitudes))[mangitudes > threshhold]] = 1
        output = x @ (self.weight @ torch.diag(strong_idxs)).t() + self.bias
        return output, output_full
    else:
        return output_full, output_full
```

Эксперимент

- Датасет: FashionMnist

- Архитектура FC-нейросети с ReLU:
  full: 768 -> 10 -> 10
  scaled: 384 -> 5 -> 10

- Результаты:
  full: **84.5%**
  scaled: **82%**
  scaled in full: **82%** (**83%**)

Дополнительный хак:
L1-регуляризация

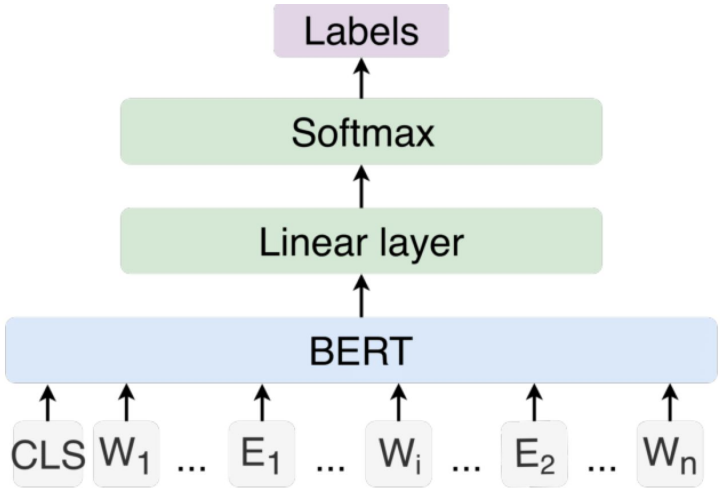
Адаптация для линейных слоев. Выводы

1. Получилось выучить две модели в одной, по аналогии тому, как предлагалось в статье
2. Качество бОльшей модели становится хуже
3. Обучение требует больше времени
4. Подбор гиперпараметров сильно важен
5. Возможно, улучшения будут более заметны на более сложных задачах и более тяжелых моделях
   Например, интересно попробовать заменить intermediate слой в трансформере, т.к часто он очень тяжелый

Адаптация MultiResolution для трансформеров

- Идея: выучить подтрансформер
  с 1го по (n / 2)-ый слой в трансформере
  с n слоями

- То есть хотим, чтобы
  $$FC(CLS_{\frac{n}{2}}) = FC(CLS_n)$$



Эксперимент

- Датасет: твиты с информацией о наличие или отсутствие побочных эффектов лекарств

- Модель: pretrained EnRuDR-BERT
  full: 12 слоев
  scaled: первые 6 слоев

- Результаты:
  full: **80%**
  scaled: **71%**
  scaled in full: **74.75%** (**77.7%**)

Спасибо за внимание