Self-Supervised Representation Learning for Images

Contrastive learning

Иван Сафонов
БПМИ 182

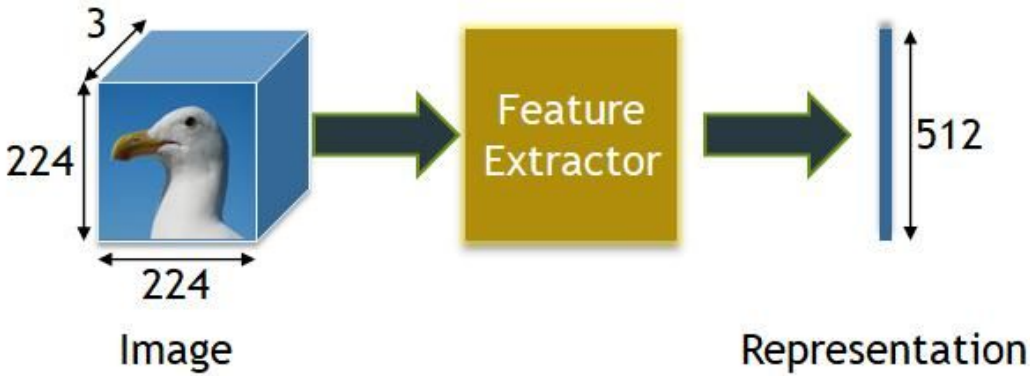
23.03.2021

План

- Введение
- SimCLR
- MoCo
- MoCo v2
- Результаты сравнения, эффективности

Введение

- хотим обучить encoder f(Image), который будет вычислять хорошее векторное представление для картинки
- данные: много неразмеченных картинок



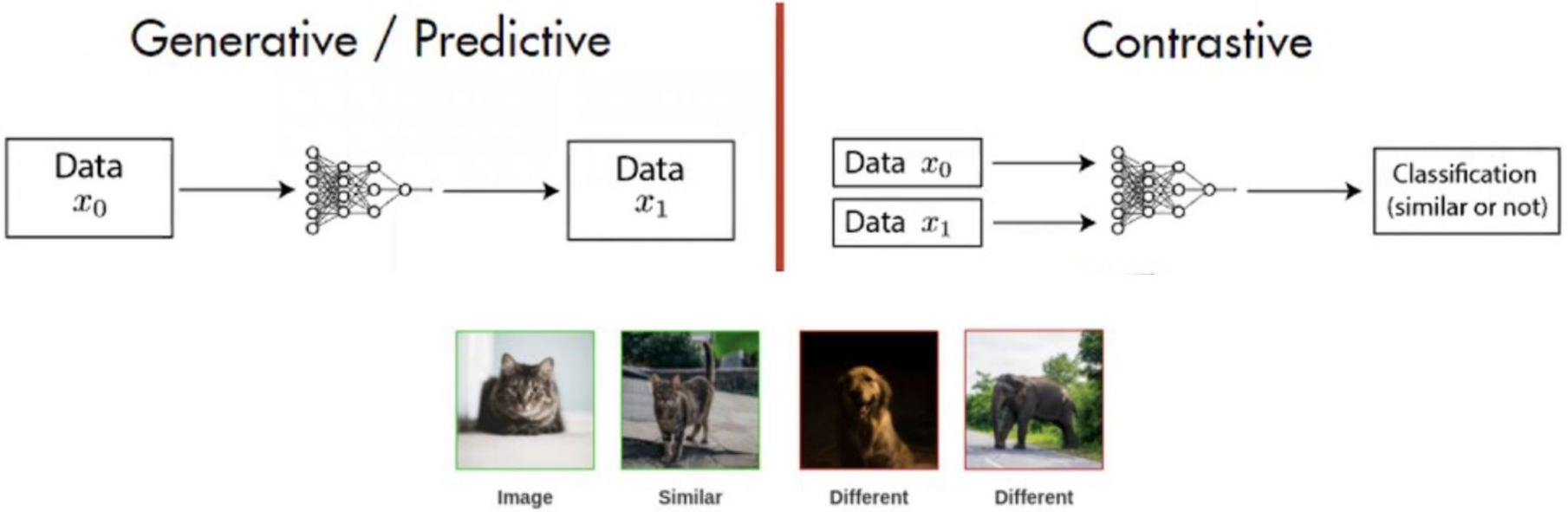
Зачем?

- мало supervised данных, но много unsupervised
- можно использовать в supervised задаче обученный на unsupervised данных encoder f, например сделав fine-tuning
- можно получать признаки, которые будут полезны для множества задач

Способ оценивания алгоритма

- обучим алгоритм на ImageNet как на unsupervised данных
- построим линейную модель на ImageNet на представлениях, полученных с помощью encoder-а f
- accuracy этой модели будет говорить об эффективности представлений

Contrastive Learning



SimCLR (Simple Framework for Contrastive Learning of Visual Representations)

Идея:
Будем обучать представления, максимизируя схожесть представлений между двумя случайными аугментациями одной картинки.

SimCLR: составные части метода

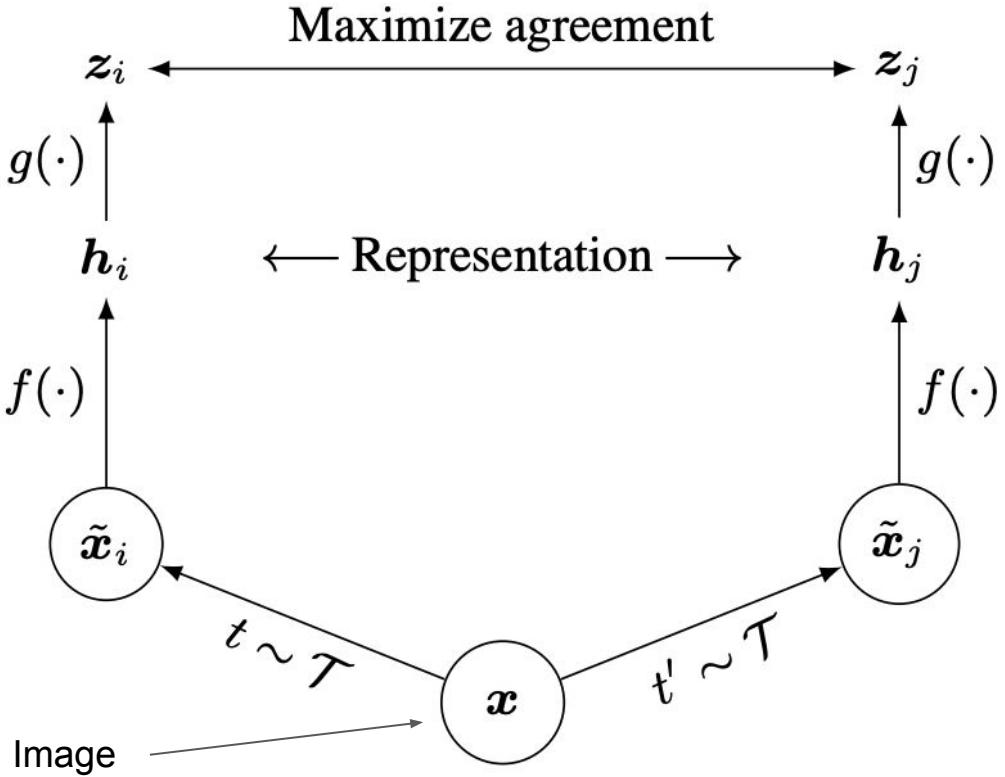
1. Распределение $\mathcal{T}$ на аугментациях картинки
2. Нейронная сеть encoder f: Image $\rightarrow \mathbb{R}^n$
3. Нейронная сеть g: $\mathbb{R}^n \rightarrow \mathbb{R}^m$ (сеть проекции)
4. Функция схожести двух векторов $\boldsymbol{sim} : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$

Авторы статьи использовали:

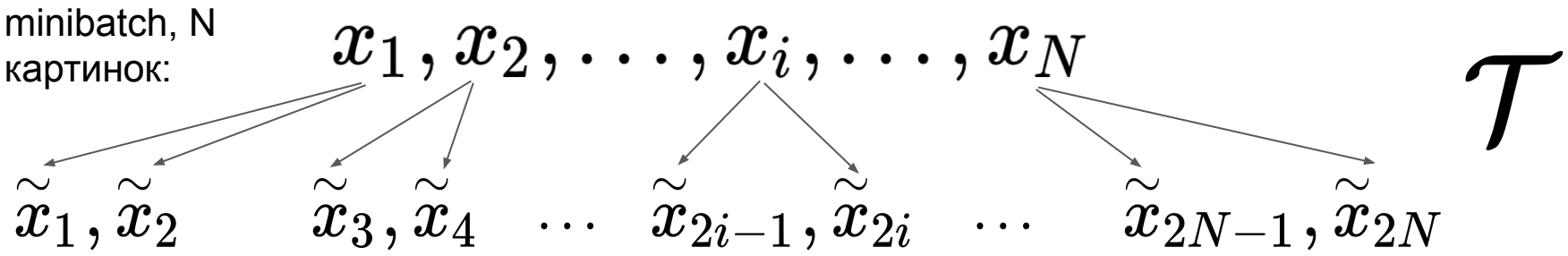
1. $\mathcal{T}$ это композиция случайного обрезания, цветового искажения и случайного Гауссовского размытия
2. f: ResNet-50
3. g: MLP с одним скрытым слоем (нелинейность - ReLU)
4. Функция схожести двух векторов: $sim(u, v) = (u^T v)/(||u||_2 ||v||_2)$ cosine similarity

SimCLR: схема работы

$$\widetilde{x_i} = t(x)$$
$$h_i = f(\widetilde{x_i})$$
$$z_i = g(h_i)$$
$$\widetilde{x_j} = t'(x)$$
$$h_j = f(\widetilde{x_j})$$
$$z_j = g(h_j)$$
$$\text{agreement} = sim(z_i, z_j)$$



SimCLR: мини-supervised задача



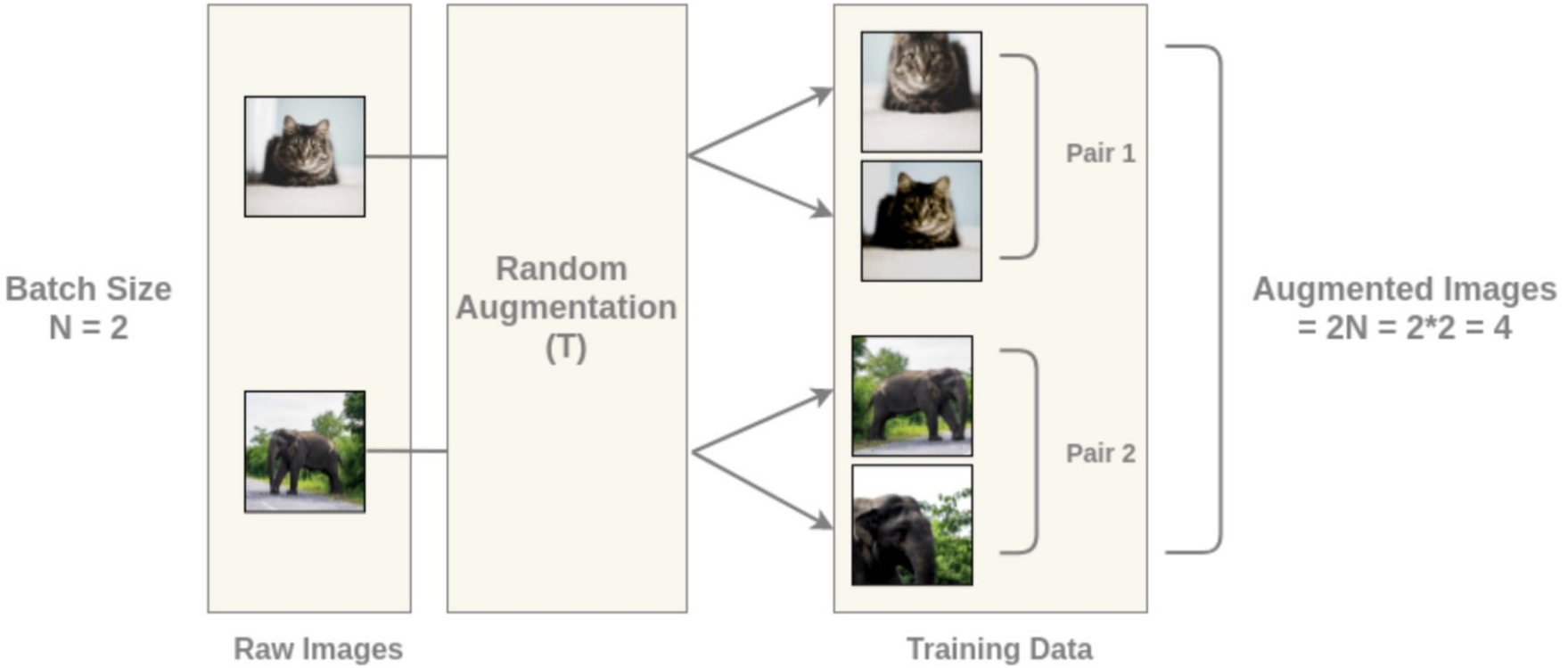
Задача: для $\widetilde{x}_i$ определить верно ли, что $\widetilde{x}_j$ с ним в паре.

2N мини-задач классификации:

$\widetilde{x}_{2i-1}$: positive $\widetilde{x}_{2i}$, others are negative

$\widetilde{x}_{2i}$: positive $\widetilde{x}_{2i-1}$, others are negative

SimCLR: мини-supervised задача



SimCLR: loss

Если (i, j) — положительная пара, то loss function для этой пары определяется так:

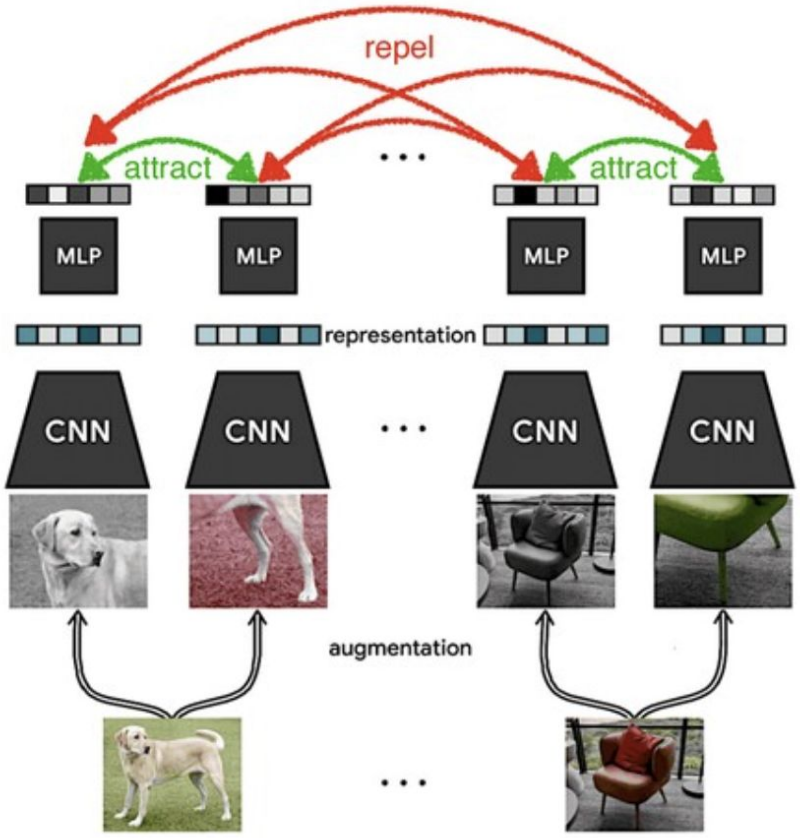
$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

τ — температурный параметр

Оптимизируем сети f и g, минимизируя суммарный loss для всех 2N мини-задач:

$$\mathcal{L} = \sum_{k=1}^{N} \left(\ell_{2k-1,2k} + \ell_{2k,2k-1} \right)$$

SimCLR: псевдокод



input: batch size N, constant τ, structure of $f, g, \mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

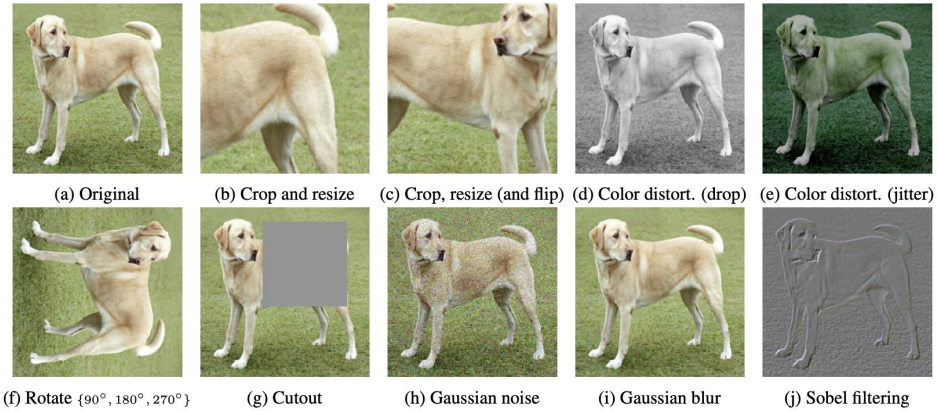
SimCLR: итог

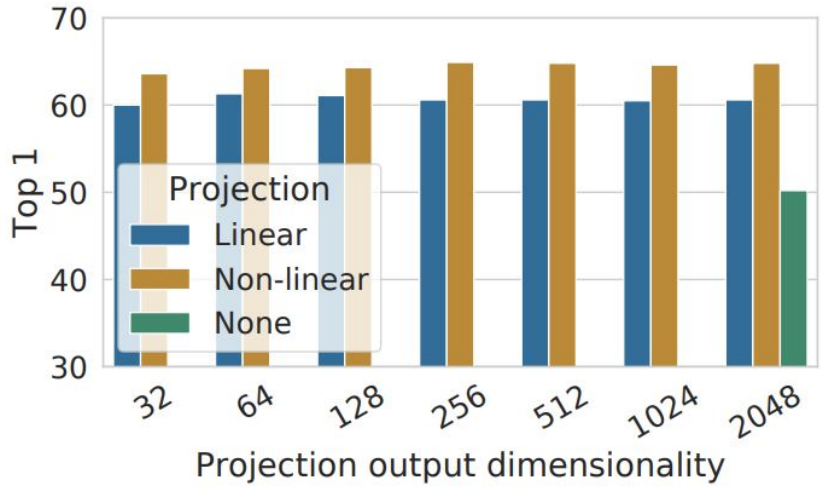
- Получаем обученный encoder f, который по картинке выдает информативное представление
- Сеть g нам больше не нужна, она была нужна для того, чтобы вычислять similarity не в исходном пространстве, а после проекции в некоторое другое пространство

SimCLR: важность деталей

Авторами статьи было исследовано и показано, что:

1. Для эффективности представлений важен выбор аугментаций
2. Использование проекции представлений для сравнения схожести дает лучший результат, чем просто сравнение схожести представлений
3. Больший размер mini-batch N дает результат лучше
4. Использование большего количества эпох дает результат лучше





MoCo (Momentum Contrast)

Другой метод Self-Supervised Representation Learning, основанный на той же идее contrastive learning.

MoCo: составные части метода

1. Распределение $\mathcal{T}$ на аугментациях картинки
2. Нейронная сеть encoder f_q : Image $\rightarrow \mathbb{R}^n$
3. Нейронная сеть encoder f_k : Image $\rightarrow \mathbb{R}^n$

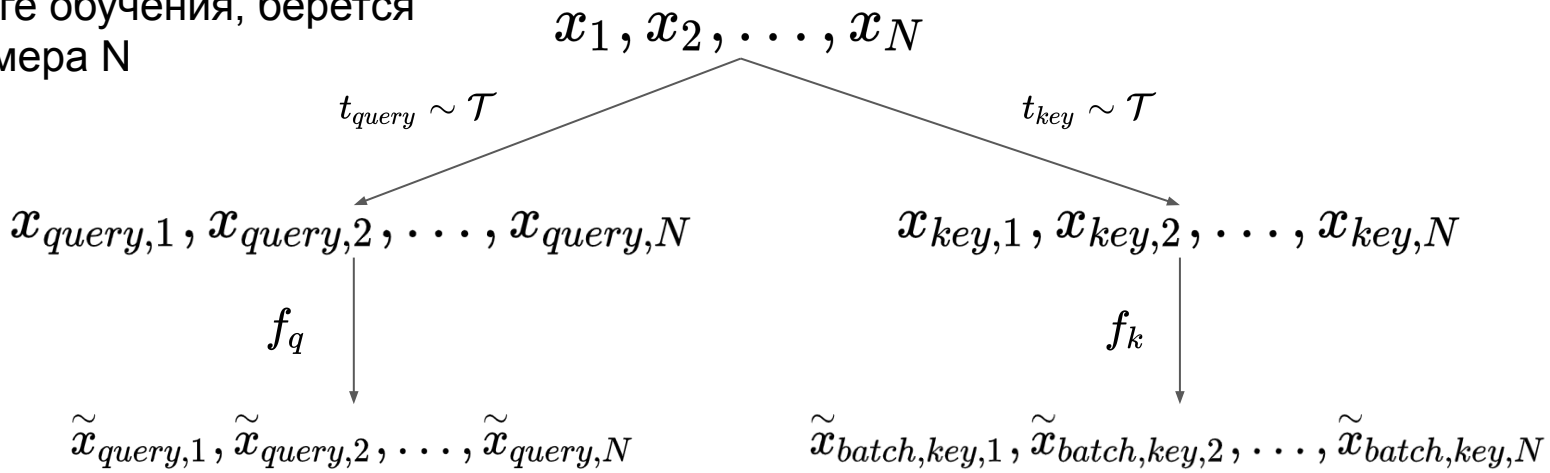
- f_q и f_k аналогично берутся ResNet-50
- будет обучаться два разных encoder-а; тот, который в итоге будет результатом это f_q

MoCo: алгоритм

Будет поддерживаться очередь размера K, состоящая из представлений

$$queue := \{\widetilde{x}_{key,1}, \widetilde{x}_{key,2}, \ldots, \widetilde{x}_{key,K}\}$$

На каждом шаге обучения, берется mini-batch размера N



MoCo: алгоритм

Добавим представления $\widetilde{x}_{batch,key,1}, \widetilde{x}_{batch,key,2}, \ldots, \widetilde{x}_{batch,key,N}$ в очередь.

Для каждого $\widetilde{x}_{query,i}$ получается своя мини-задача классификации:

Среди K+N представлений из очереди найти представление, соответствующее той же картинке.

MoCo: loss function

loss function для одной такой мини-задачи классификации будет:

$$\mathcal{L}_{q,k^+,\{k^-\}} = -\log \frac{\exp(q \cdot k^+/\tau)}{\exp(q \cdot k^+/\tau) + \sum_{k^-} \exp(q \cdot k^-/\tau)}$$

InfoNCE loss

(q — представление запрос, k^+ — единственное положительное представление ключ из очереди, $\{k^-\}$ — отрицательные представления ключи из очереди)

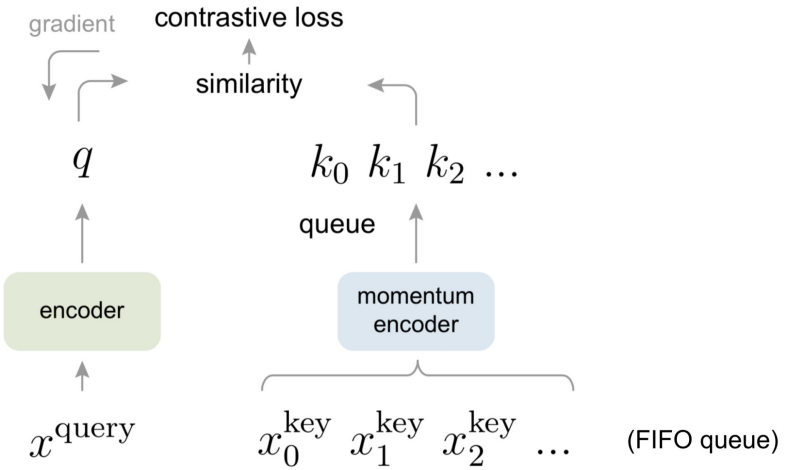
Суммарный loss L_{step} на этом шаге — это сумма InfoNCE по всем $q = \widetilde{x}_{query,i}$

MoCo: алгоритм

Удаляем первые N представлений из очереди (таким образом ее размер снова K).

Делаем один backpropagation шаг для f_q, оптимизирующий L_{step}.

Параметры f_k меняются с помощью momentum через параметры f_q: $\theta_k \leftarrow m\theta_k + (1-m)\theta_q$



MoCo: псевдокод

```
# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature

f_k.params = f_q.params # initialize
for x in loader: # load a minibatch x with N samples
   x_q = aug(x) # a randomly augmented version
   x_k = aug(x) # another randomly augmented version

   q = f_q.forward(x_q) # queries: NxC
   k = f_k.forward(x_k) # keys: NxC
   k = k.detach() # no gradient to keys

   # positive logits: Nx1
   l_pos = bmm(q.view(N,1,C), k.view(N,C,1))

   # negative logits: NxK
   l_neg = mm(q.view(N,C), queue.view(C,K))

   # logits: Nx(1+K)
   logits = cat([l_pos, l_neg], dim=1)

   # contrastive loss, Eqn.(1)
   labels = zeros(N) # positives are the 0-th
   loss = CrossEntropyLoss(logits/t, labels)

   # SGD update: query network
   loss.backward()
   update(f_q.params)

   # momentum update: key network
   f_k.params = m*f_k.params+(1-m)*f_q.params

   # update dictionary
   enqueue(queue, k) # enqueue the current minibatch
   dequeue(queue) # dequeue the earliest minibatch
```

`bmm`: batch matrix multiplication; `mm`: matrix multiplication; `cat`: concatenation.

MoCo: детали

- Невозможно обучать f_k с помощью backpropagation, потому что представления в очереди получены на разных шагах (т.е. с разными encoder-ами)
- Использование одинаковых encoder-ов f_k и f_q, вместо обновления f_k с помощью momentum приводит к тому, что нет сходимости.

momentum m	0	0.9	0.99	0.999	0.9999
accuracy (%)	*fail*	55.2	57.8	59.0	58.9

MoCo vs SimCLR

Преимущества SimCLR:

- более грамотное сравнение similarity (проекция на другое пространство)
- подбор эффективных аугментаций

Преимущества MoCo:

- За размер мини-задачи классификации отвечает размер очереди K, а размер batch N не влияет на эффективность представлений ⇒ обучение не требует огромных вычислительных мощностей

MoCo v2

Авторы статьи добавили в MoCo две идеи из SimCLR:

- считать similarity после проекции на другое пространство (с помощью обучаемой MLP)
- больше аугментаций (добавлено Гауссовское размытие, увеличено искажение цветов)

В результате получился state-of-the-art алгоритм (превзошедший MoCo и SimCLR)

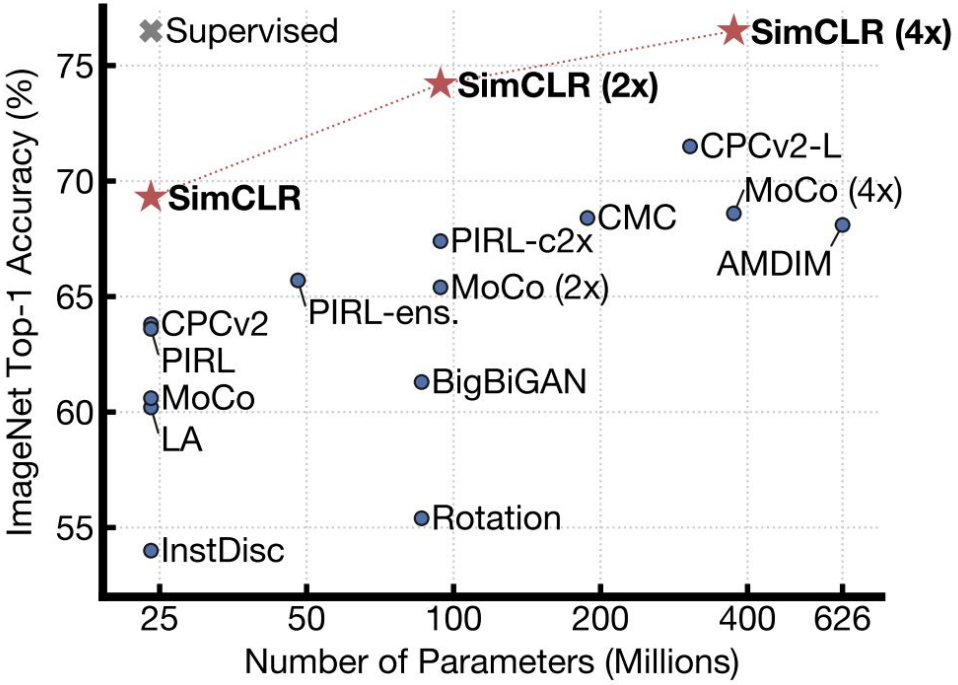
Transfer learning results (SimCLR)

Сравниваются:

- Представления, полученные из SimCLR
- Представления, полученные из supervised обучения на другую задачу

	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
Linear evaluation:												
SimCLR (ours)	**76.9**	**95.3**	80.2	48.4	**65.9**	60.0	61.2	**84.2**	**78.9**	89.2	**93.9**	**95.0**
Supervised	75.2	**95.7**	**81.2**	**56.4**	64.9	**68.8**	**63.8**	83.8	**78.7**	**92.3**	**94.1**	94.2
Fine-tuned:												
SimCLR (ours)	**89.4**	**98.6**	**89.0**	**78.2**	**68.1**	**92.1**	**87.0**	**86.6**	**77.8**	92.1	**94.1**	97.6
Supervised	88.7	98.3	**88.7**	**77.8**	67.0	91.4	**88.0**	86.5	**78.8**	**93.2**	**94.2**	**98.0**
Random init	88.3	96.0	81.9	**77.0**	53.7	91.3	84.8	69.4	64.1	82.7	72.5	92.5

Результаты сравнения [MoCo, SimCLR]



Результаты сравнения [MoCo, SimCLR, MoCo v2]

case	unsup. pre-train MLP	aug+	cos	epochs	batch	ImageNet acc.
MoCo v1 [6]				200	256	60.6
SimCLR [2]	✓	✓	✓	200	256	61.9
SimCLR [2]	✓	✓	✓	200	8192	66.6
MoCo v2	✓	✓	✓	200	256	**67.5**
*results of **longer** unsupervised training follow:*						
SimCLR [2]	✓	✓	✓	1000	4096	69.3
MoCo v2	✓	✓	✓	800	256	**71.1**

Итог

- contrastive learning методы для решения задачи Self-Supervised Representation Learning для картинок эффективны
- представления, полученные через Self-Supervised Representation Learning достигают по эффективности представления, полученные через Supervised обучение

Источники

- SimCLR: https://arxiv.org/pdf/2002.05709.pdf
- Визуализация SimCLR: https://amitness.com/2020/03/illustrated-simclr/
- Презентация про SimCLR: https://speakerdeck.com/scatterlab/simclr-a-simple-framework-for-contrastive-learning-of-visual-representations
- MoCo: https://arxiv.org/pdf/1911.05722.pdf
- MoCo v2: https://arxiv.org/pdf/2003.04297.pdf
- видео про MoCo: https://www.youtube.com/watch?v=2Undxq7jlsA
- Обзор Self-Supervised Representation Learning методов: https://lilianweng.github.io/lil-log/2019/11/10/self-supervised-learning.html