

Компьютерное зрение

Лекция 9. Representation learning

27.06.2020
Руслан Рахимов

Introduction

В машинном обучении, необходимо:

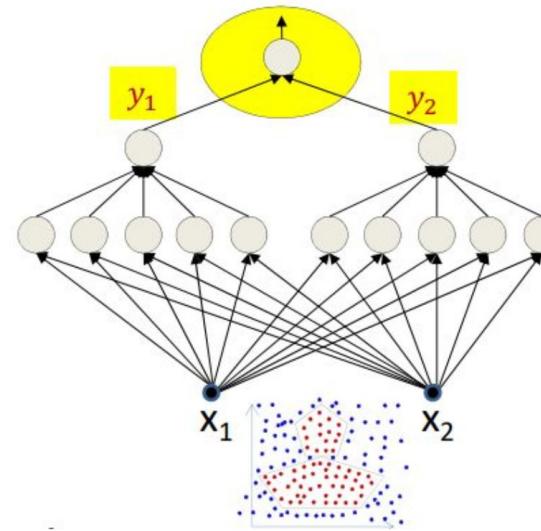
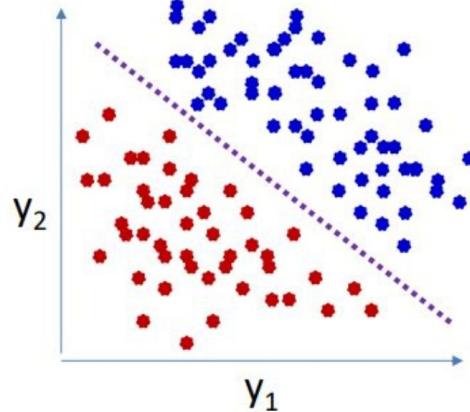
- иметь хорошие признаки (features)
- модель поверх признаков (классификатор, регрессор)

Сегодня поговорим про признаки

Если мы создаем признаки вручную, это называется **feature engineering** (очень важно в SVM, decision trees, NLP алгоритмах и т.д.)

Если мы делаем это автоматически, это называется **feature learning** или **representation learning**

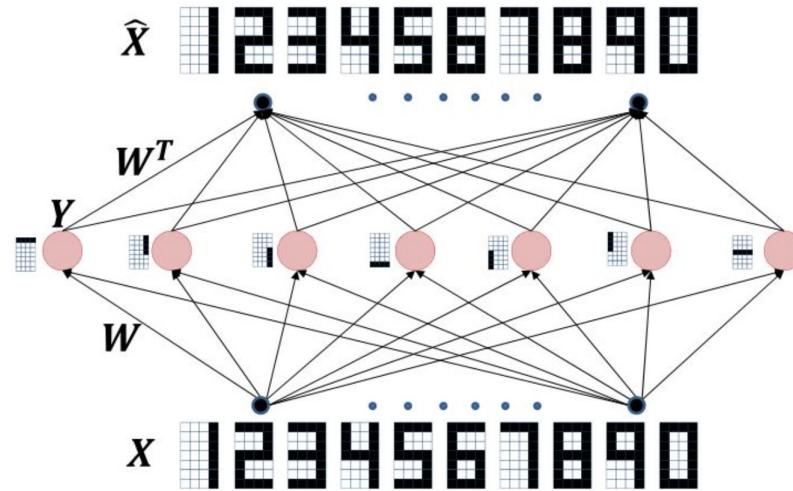
Пример



Deep Learning объединяет representation learning и классификацию в одну задачу

Пример

Нейронные сети могут выучить представление данных в unsupervised режиме. Например Autoencoders (Автоенкодеры)



Модель на картинке по сути non-linear PCA

Representation learning как способ думать

- Почему CNN хороши для картинок?
 - Признаки которые вы ищете на картинках обычно translation-invariant
- Почему RNN хороши для последовательностей?
 - Так как зависимости в последовательностях могут быть долговременными
- Почему attention полезен в seq2seq задачах?
 - Потому что слово на выходе зависит от определенных слов на входе

Вместо того, чтобы использовать случайный набор инструментов, моделей для своей задачи, подумайте, какие зависимости в данных вы хотите искать

Н: для sentiment analysis CNN подходят лучше чем RNN

Как находить representation

- end-to-end (обычный подход)
- unsupervised (autoencoders)
- на альтернативной задаче
- использовать предобученную модель (н: word embeddings)

Если мы используем готовые представления данных, нас интересуют:

- возможность файнтюнить представления
- фиксировать представления и добавлять больше слоев в сетку

Например

- Transfer learning

Учим модель на простой задаче, где есть много данных, переносим ее на нашу задачу где данных мало, меняя последний слой сети

Например

- Semi-supervised learning

Мы хотим научить переводчик Английский-Русский. Одно из решений:

- Учим encoder-decoder на английских текстах, берем encoder
- Учим encoder-decoder на русских текстах, берем decoder
- Учим attention на текстах английский-русский

Unimodal representations

Какие есть уже хорошие предобученные представления данных?

Зависит от модальности: картинки, звук, аудио, видео, ...

- **Images:** берем модель предобученную например на ImageNet (domain-specific! т.е. для медицинских данных ImageNet явно не подойдет)
- **Text:** Word2Vec, BERT, и т.д.
- **Audio:** VGGish
- **Video:** кодируем отдельно звуковой и визуальный ряд
 - в случае визуального ряда, вытаскиваем кадры на фиксированном интервале
 - получаем представления каждого кадра
 - Далее представления можно усреднять либо оставить как последовательность

Multimodal representations

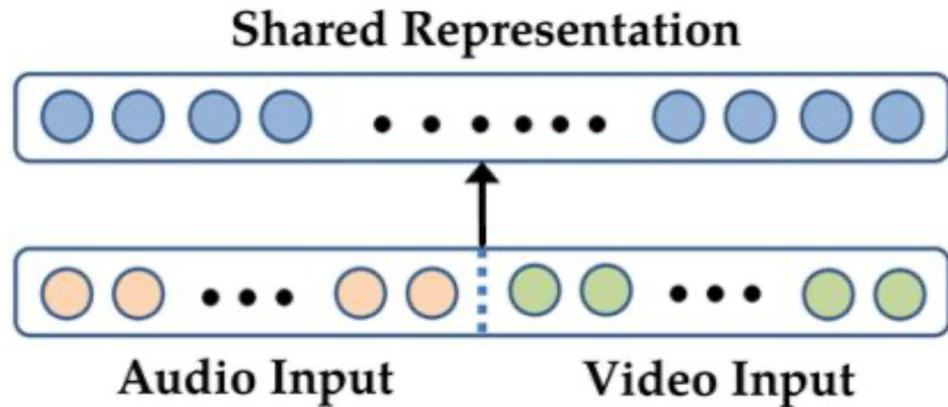
Допустим мы строим вопросно-ответную систему по изображению

Можно легко получить представления каждой модальности в отдельности, картинки и звука

Как теперь будем объединять?

Shallow representations

Просто конкатенируем!



Плюсы: Очень просто

Минусы: Не улавливаем большую часть взаимодействия между модальностями -> weak representation

Bilinear Pooling

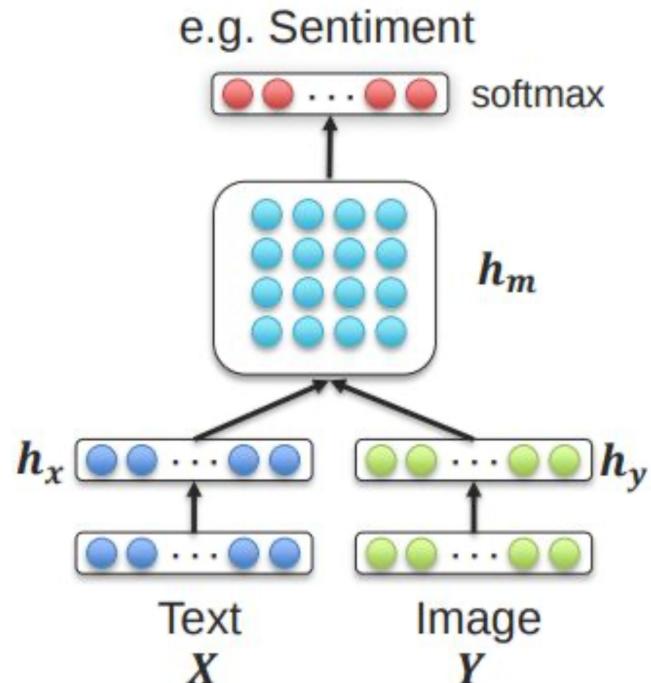
Считаем outer product двух векторов и получаем большой мультимодальный вектор на выходе

Плюсы:

- Улавливаем почти каждый паттерн
- Отлично для “дополняющих” друг друга модальностей, т.е. которые содержат разную информацию

Минусы:

- Можем получить много лишней информации (redundancy)



Autoencoders

Единое представление для нескольких модальностей!

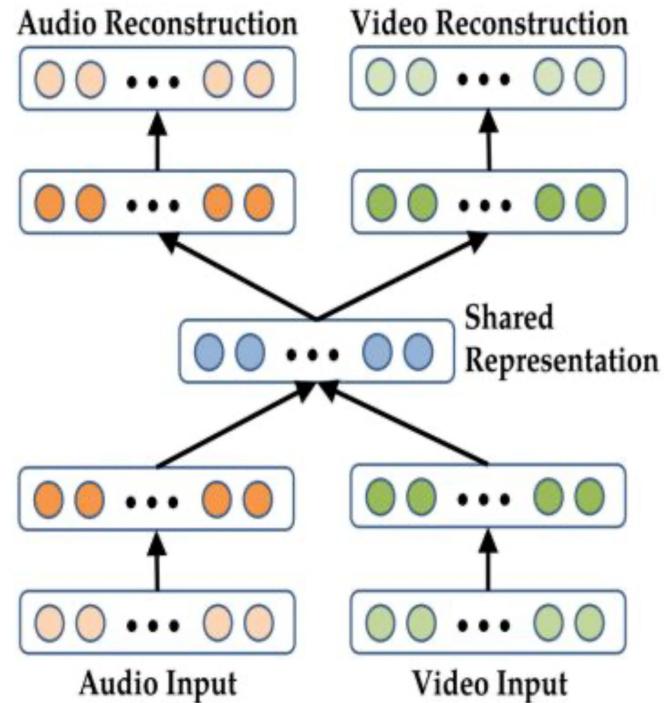
Учим в unsupervised режиме. Когда применяем представлений для другой задачи оставляем только Encoder.

Плюсы:

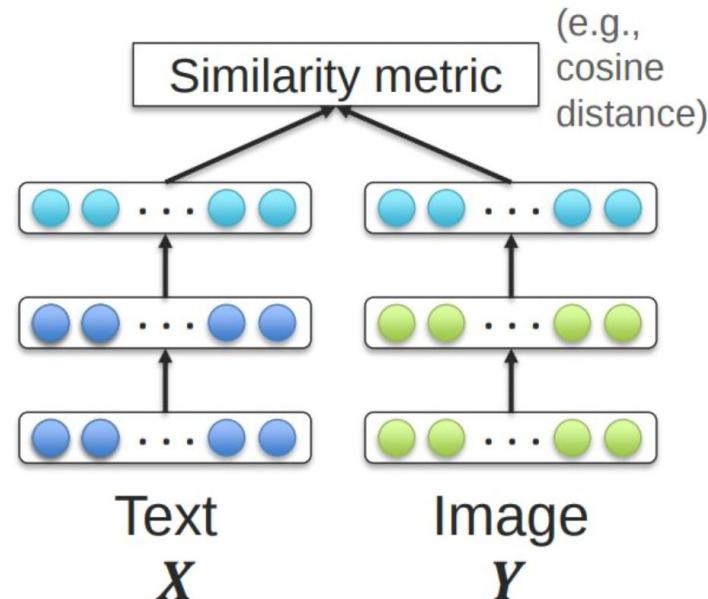
- robust representation
- если тренируем правильно, можно восстанавливать недостающие модальности

Минусы:

Требует отдельной процедуры обучения, т.к. есть decoder и зачастую не state-of-the-art по сравнению с pooled или coordinated representation



Coordinated representation

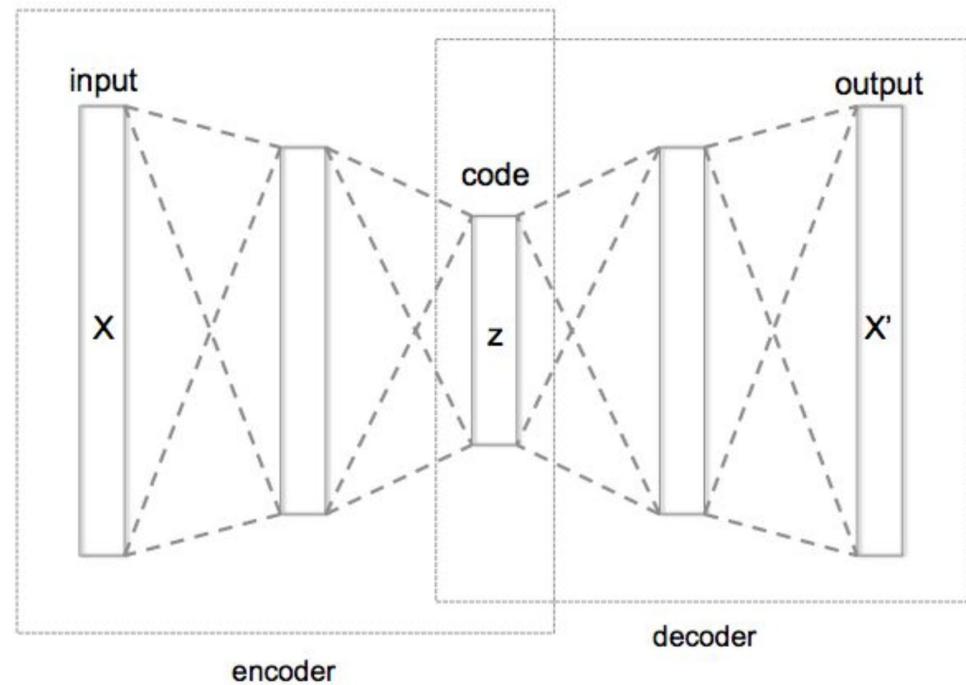


Подробнее об autoencoders

Autoencoder

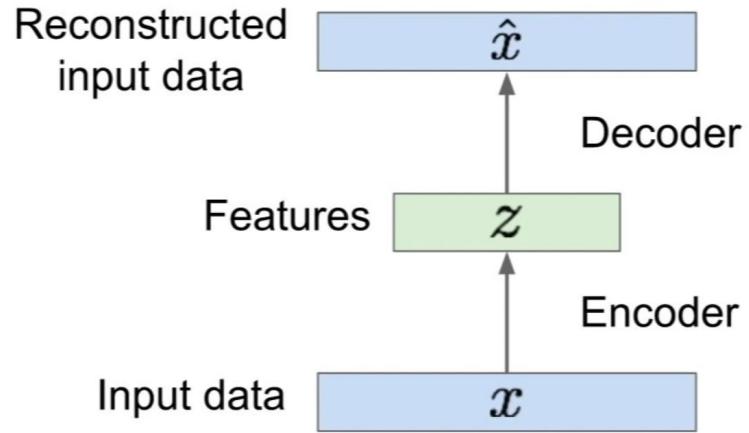
Выполняет две задачи:

- Representation learning
- Dimensionality reduction



Autoencoder

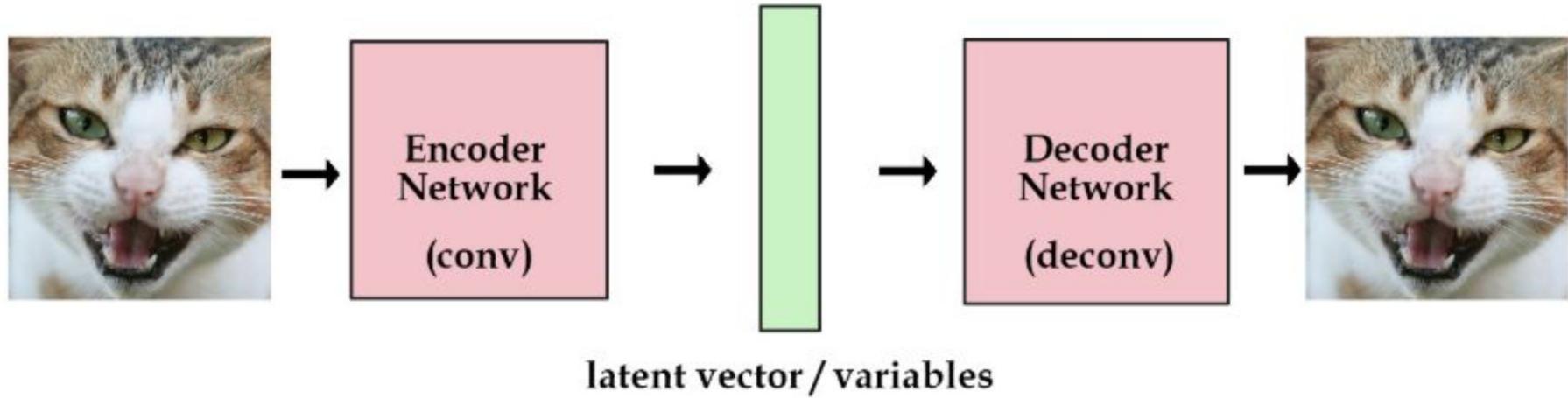
- входные данные имеют большую размерность
- encoder уменьшает размерность
- decoder восстанавливает оригинальные данные
- возможные архитектуры включают:
 - Linear layers + нелинейность
 - Conv, Deconv
 - LSTM, RNN, GRU, etc.
- L2 loss, VGG loss
- обычно очень полезны когда пытаемся извлечь важные признаки



**Скрытое представление проходит через bottleneck
Узнали?**

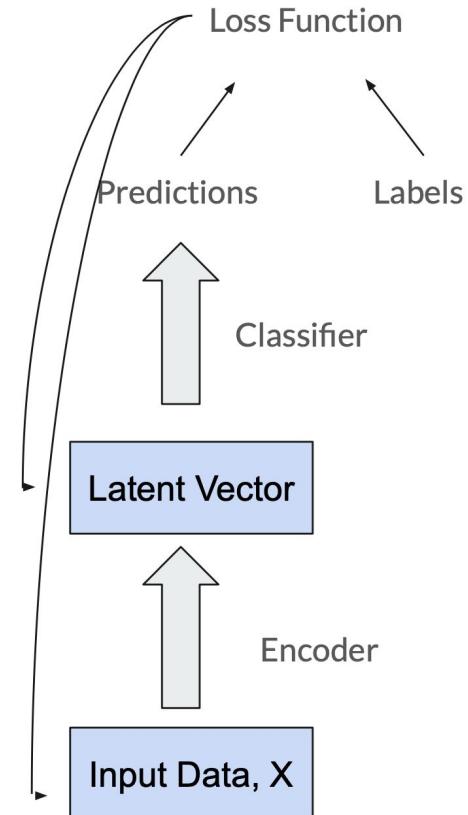


Autoencoder



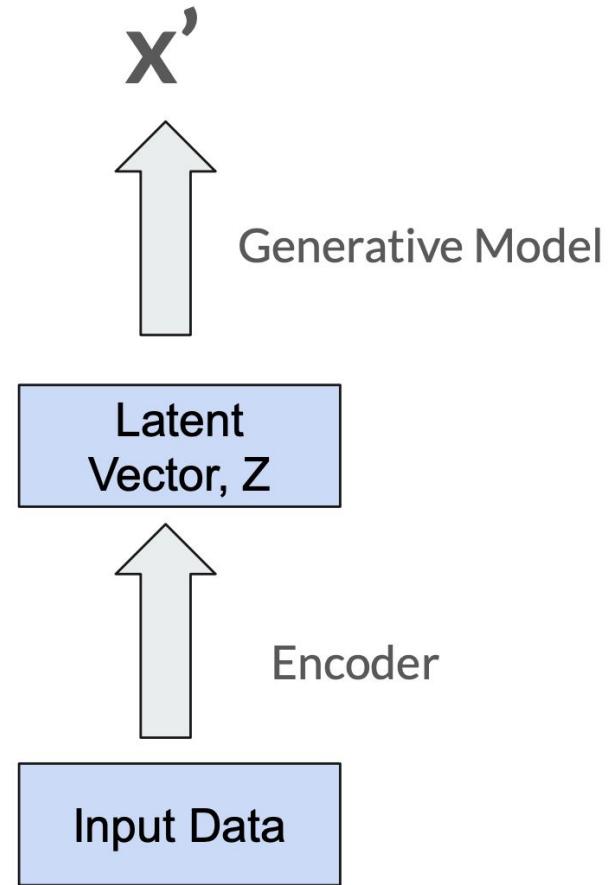
Autoencoder

- можно применять в supervised learning задачах
- оставляем encoder для извлечения признаков
- убираем decoder
- объединяя с классификатором, файнтюним
- имеет смысл когда имеется большое количество неразмеченных данных и мало размеченных



Autoencoder: минусы

- Что если мы захотим генерировать новые данные?
- Сгенерируем сами z и подадим на вход декодеру
- z содержит хорошие признаки
 - В случае лиц, z может содержать форму носа, ширину бровей и т.д.

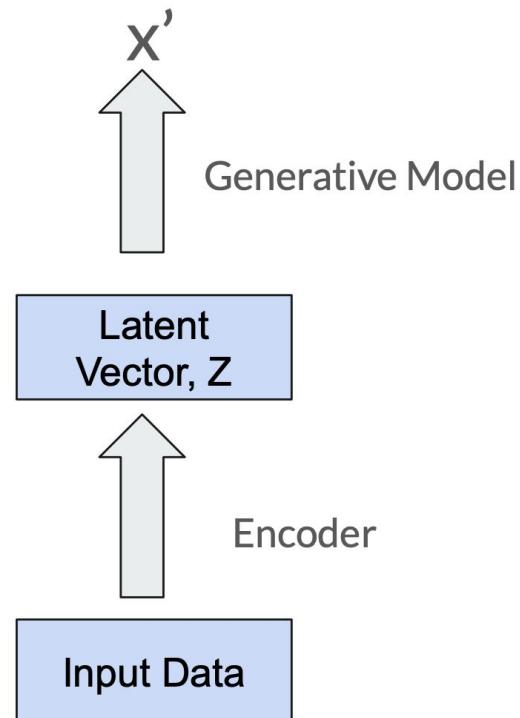


Генеративная модель на основе АЕ

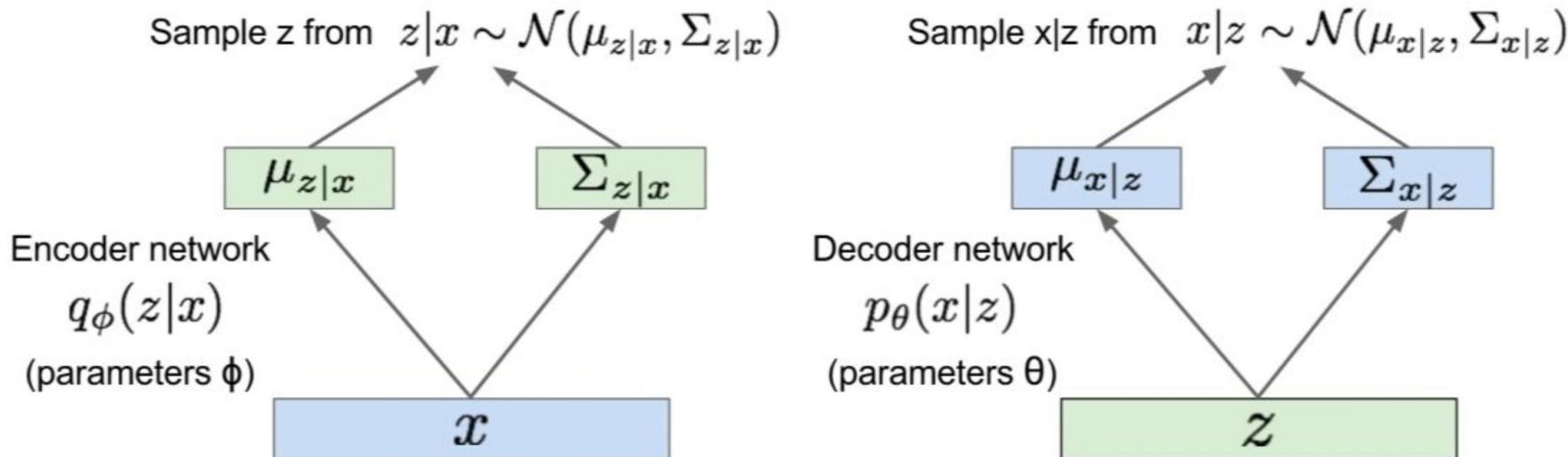
- Наложим априорное распределение на z - prior $p(z)$
- Сеть заставим учить условное распределение $p(x'|z)$
- Учим сеть максимизировать likelihood данных из трейна

$$p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$$

- **Проблема:** как считать $p(x|z)$ для любого z ?

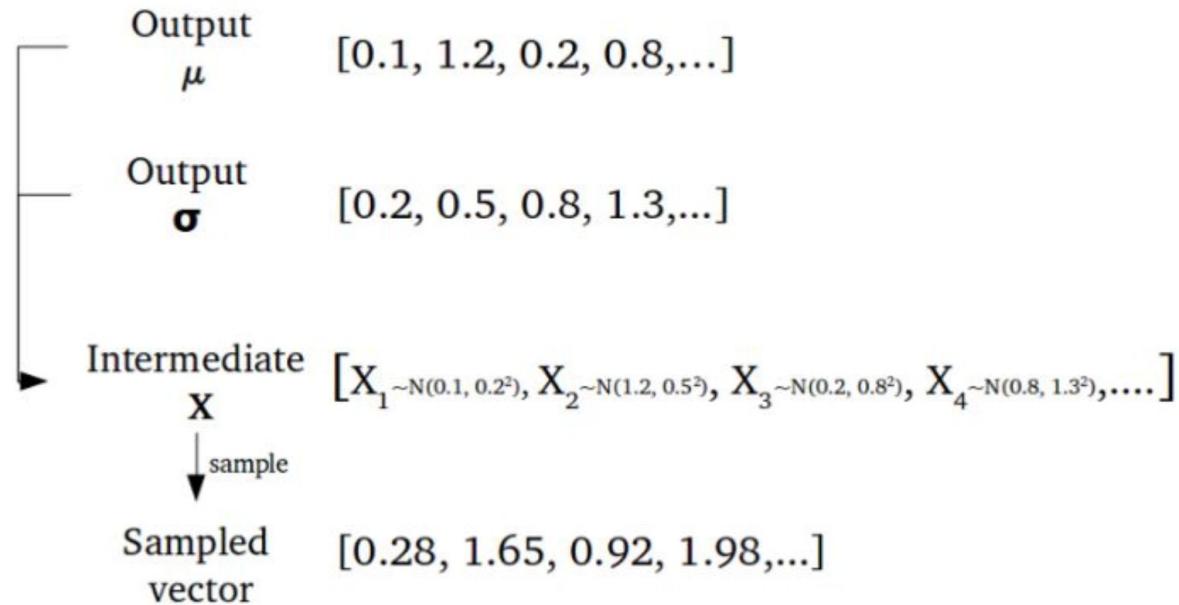


VAE



VAE

Каждый раз когда сэмплим из распределения, получаем разный x



VAE

$$\begin{aligned}
\log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z | x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z)p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z)p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] \quad (\text{Multiply by constant}) \\
&= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Logarithms}) \\
&= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z)) + D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))
\end{aligned}$$

Можем оценить с помощью сэмплирования

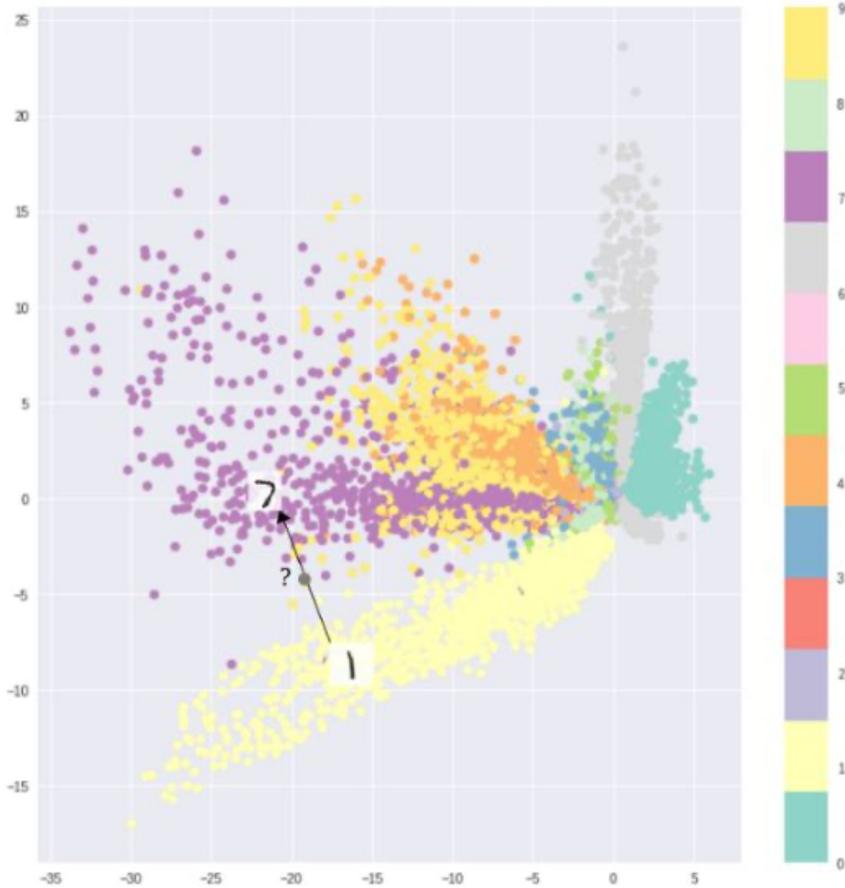
KL дивергенция между двумя гауссовскими распределениями имеет простую аналитическую форму

По определению ≥ 0

Autoencoder

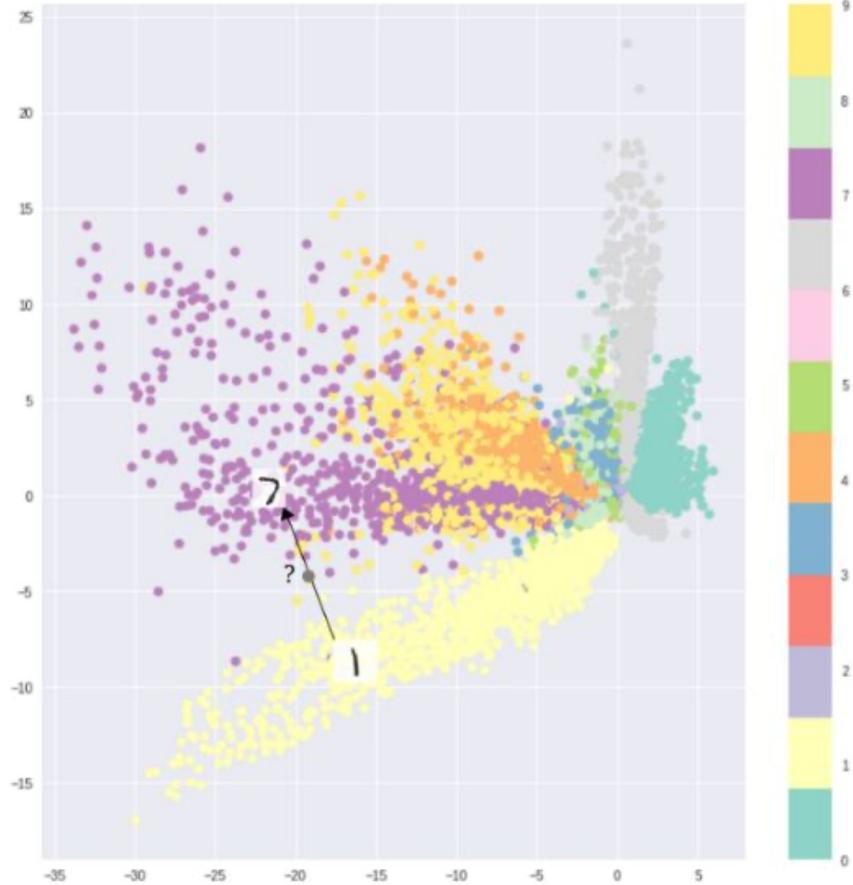
Латентное пространство (там где лежит z),
может не являться непрерывным

В этом и состоит проблема генерации новых
изображений



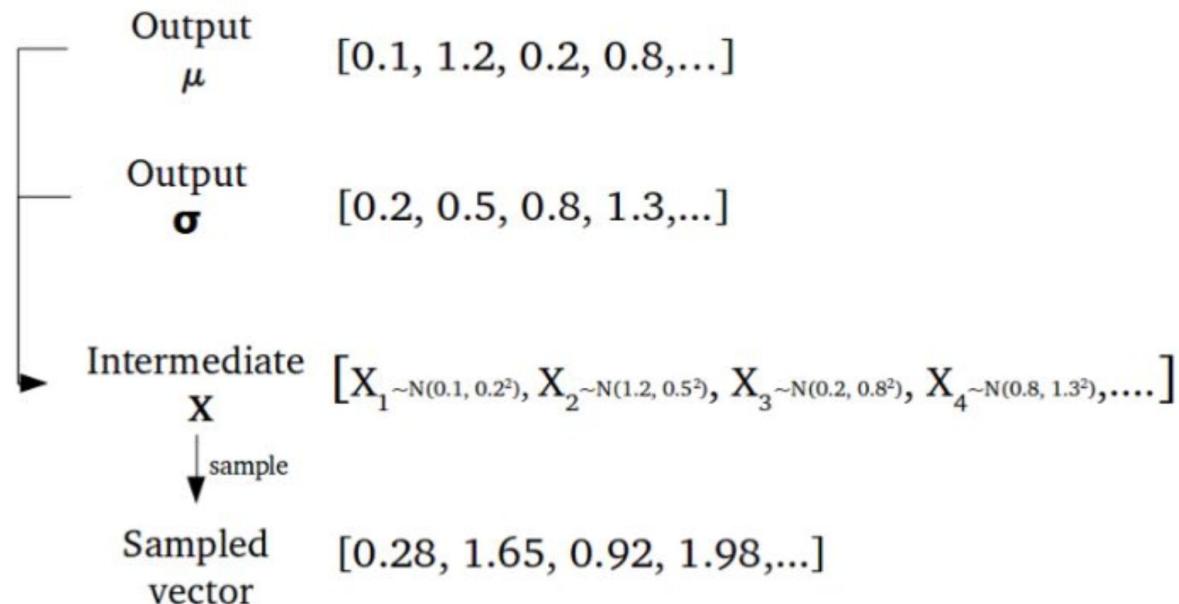
Autoencoder: минусы

Если случайно генерировать z ,
нейросеть не поймет что генерировать на выходе

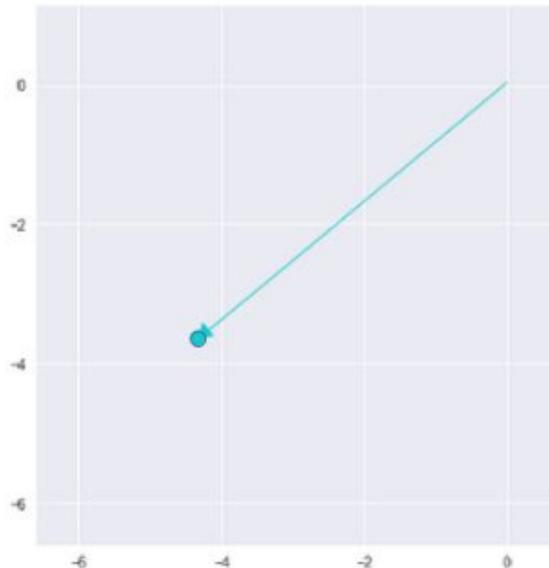


VAE

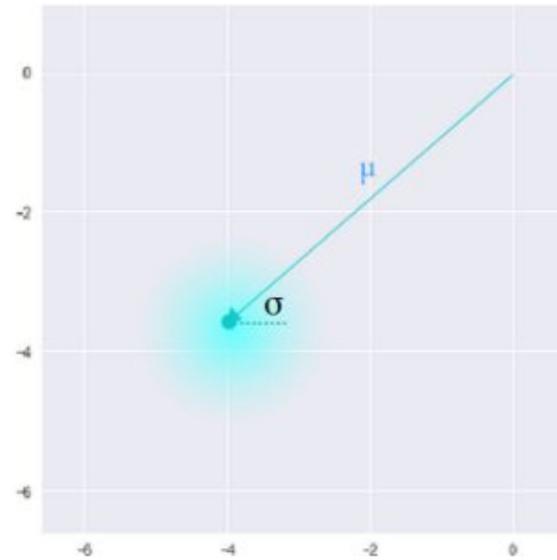
- Encoder дает на выходе два вектора размерности n, один соответствует mean, второй std
- Стохастическая генерация, для фиксированного x, mean и std одинаковые, латентный вектор z разный в результате сэмплинга



VAE vs AE

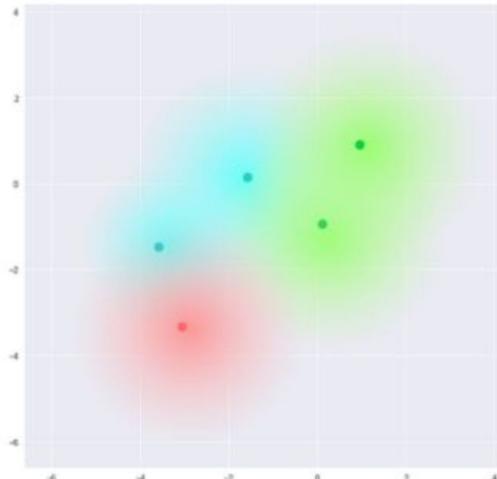


Standard Autoencoder
(direct encoding coordinates)

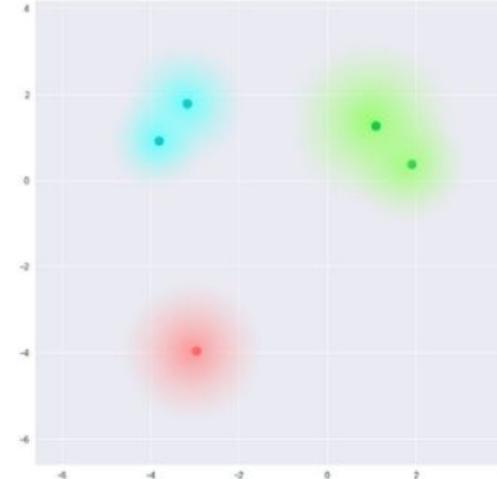


Variational Autoencoder
(μ and σ initialize a probability distribution)

Проблема все равно остается



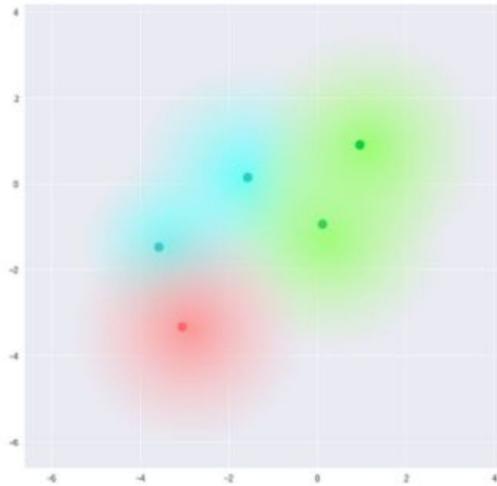
What we require



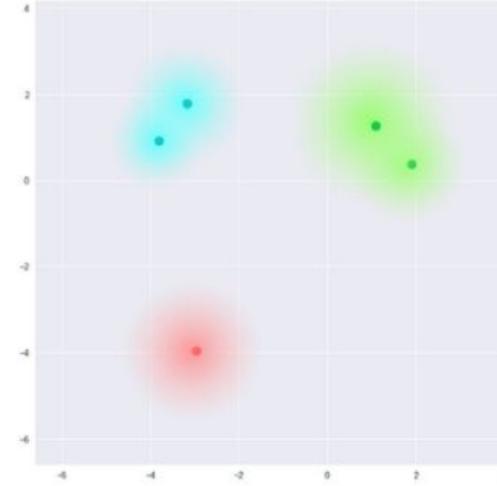
What we may inadvertently end up with

- все равно имеем пустоты между кластерами
- остается шанс, что сеть не понимает что ей генерить

Проблема все равно остается



What we require



What we may inadvertently end up with

- нет ограничений на mean и variance
- encoder может выучить разные mean для разных классов, а потом минимизировать variance
- в результате меньше неопределенность для декодера

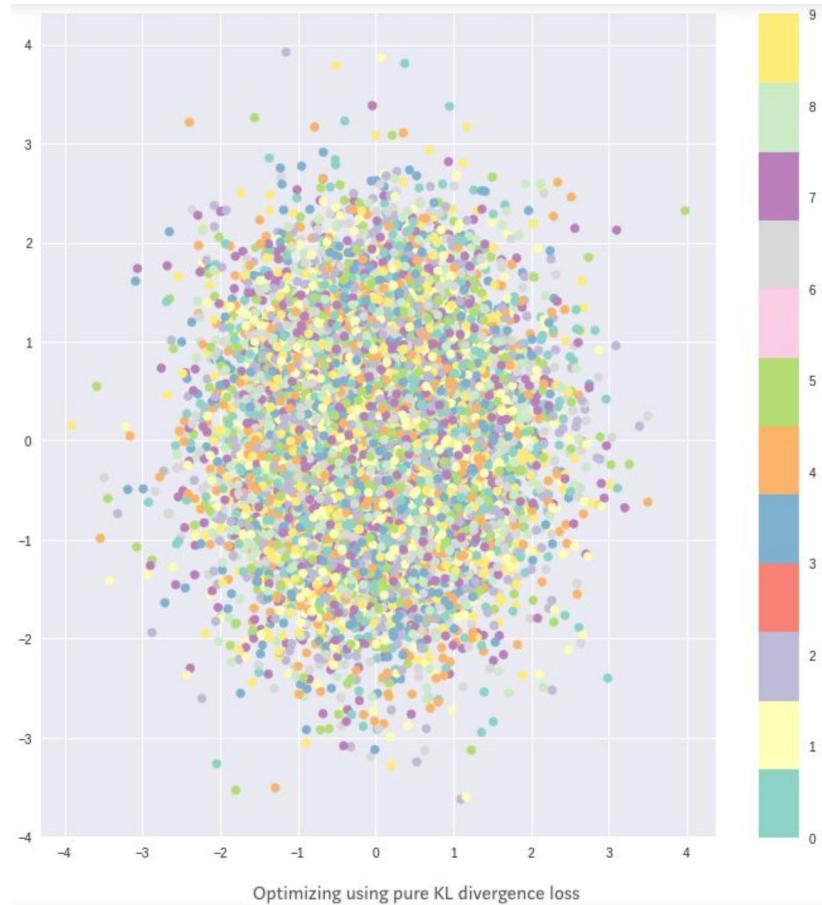
KL divergence

- Измеряет расстояние между двумя распределениями
- Оптимизация по KL divergence означает оптимизацию параметров распределения таким образом, что оно станет похожим на target распределение
- В случае если у X компоненты $x_i \sim N(\mu_i, \sigma_i^2)$ и target распределение имеет вид стандартного нормального KL имеет вид:

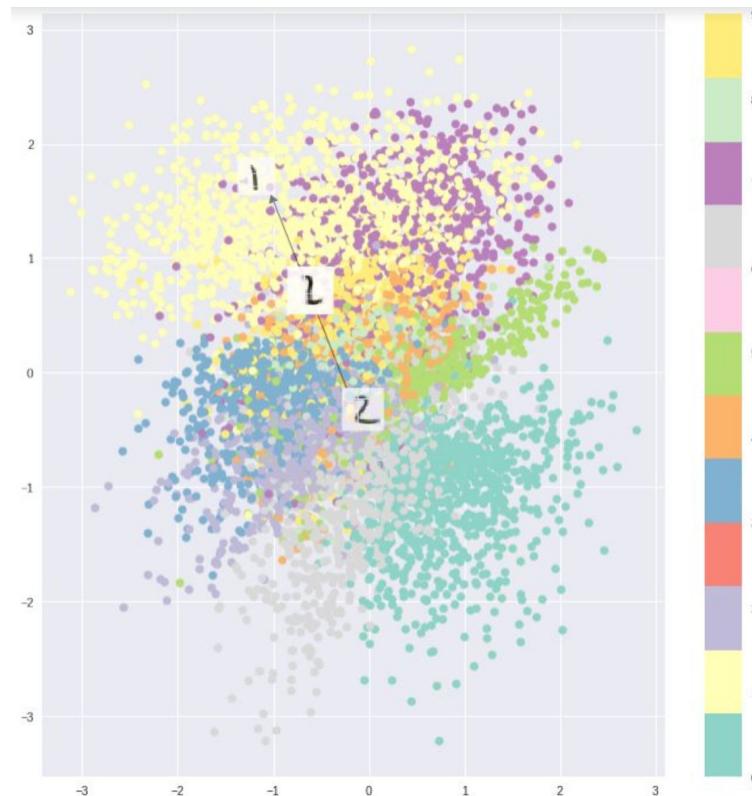
$$\sum_{i=1}^n \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1$$

KL divergence

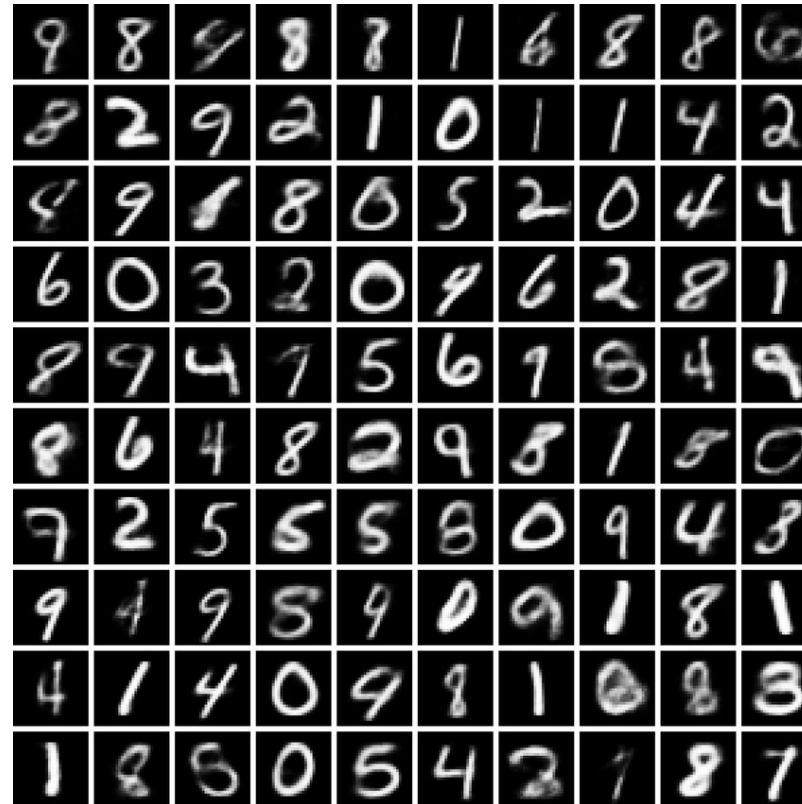
- Заставляет encoder распределять векторы з равномерно между модами target распределения
- Не различает разные классы, не учитывает похожесть объектов одного класса



KL + reconstruction loss



VAE: пример генерации



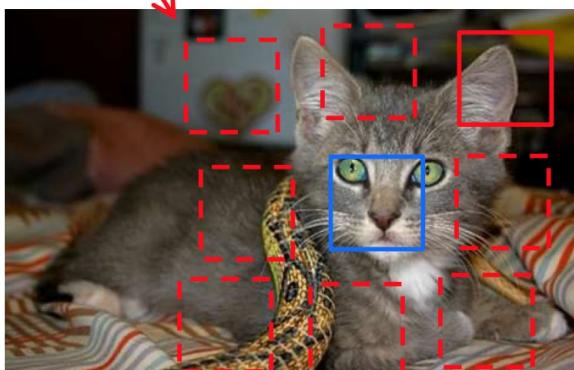
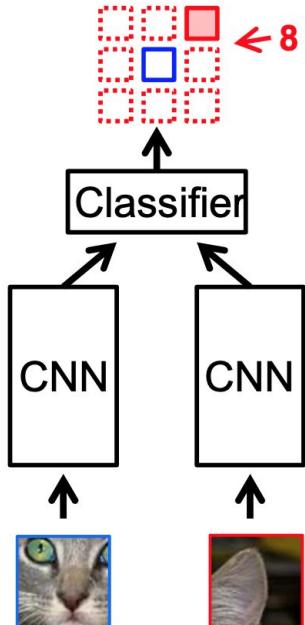
VAE: пример генерации



Что еще?

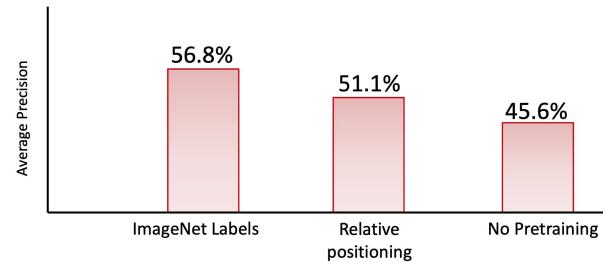
Relative positioning

Train network to predict relative position of two regions in the same image



**Randomly Sample Patch
Sample Second Patch**

Unsupervised visual representation learning by context prediction,
Carl Doersch, Abhinav Gupta, Alexei A. Efros, ICCV 2015



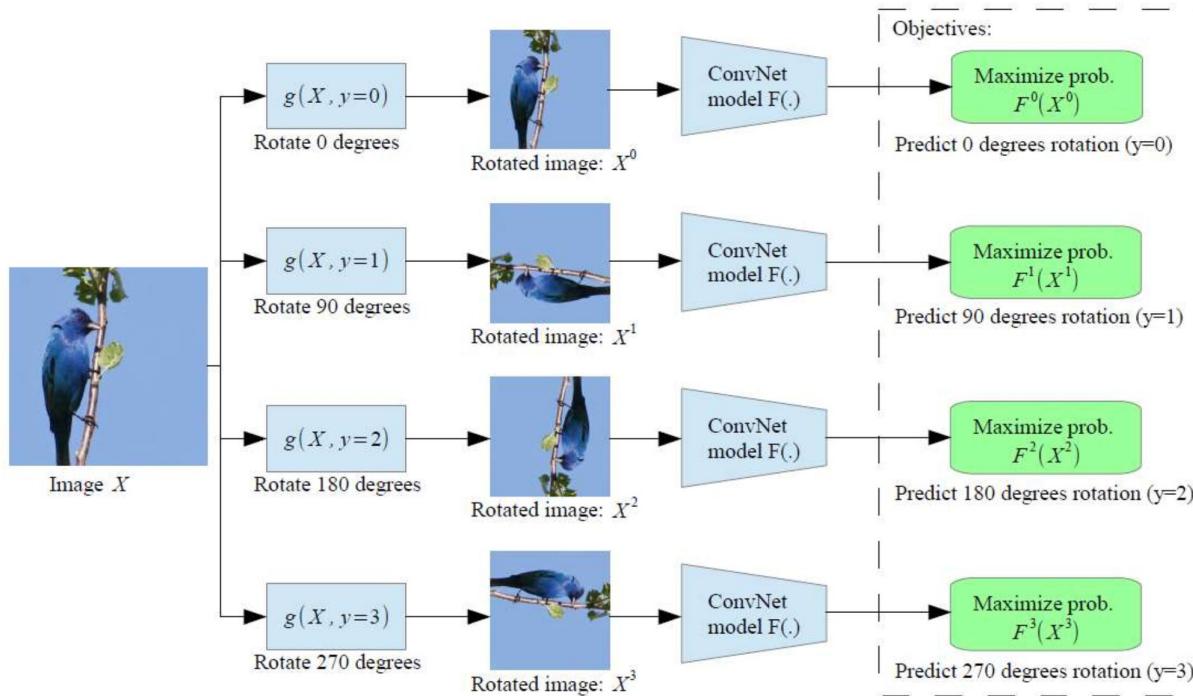
Колоризация

Train network to predict pixel colour from a monochrome input



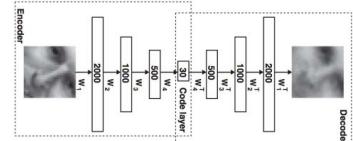
Colorful Image Colorization, Zhang et al., ECCV 2016

Image transformations



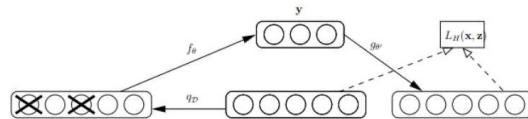
Unsupervised representation learning by predicting image rotations,
Spyros Gidaris, Praveer Singh, Nikos Komodakis, ICLR 2018

Autoencoders



Hinton & Salakhutdinov.
Science 2006.

Denoising Autoencoders



Vincent et al. ICML 2008.

Exemplar networks



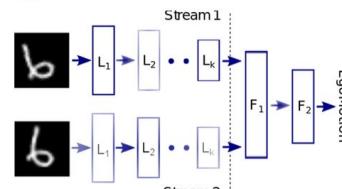
Dosovitskiy et al., NIPS 2014

Co-Occurrence



Isola et al. ICLR Workshop 2016.

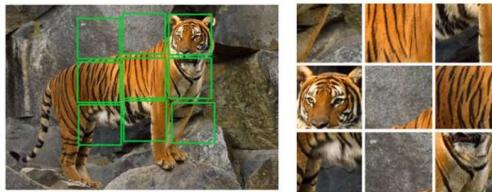
Egomotion



Agrawal et al. ICCV 2015 Jayaraman et al. ICCV 2015



Context

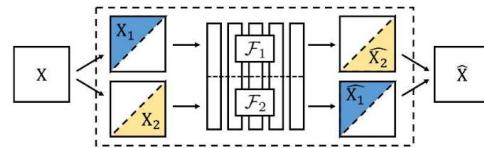


Norooz et al 2016



Pathak et al. CVPR 2016

Split-brain auto-encoders



Zhang et al. CVPR 2017

Procedure:

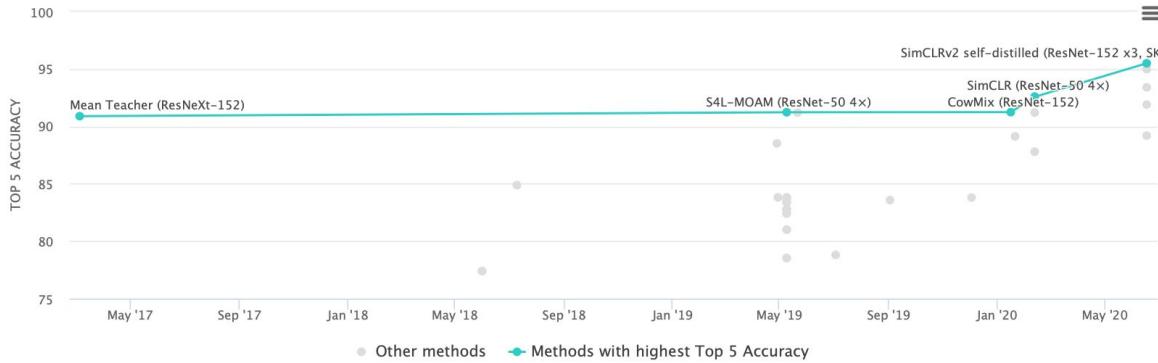
- ImageNet-frozen: self-supervised training, network fixed, classifier trained on features
- PASCAL: self-supervised pre-training, then train Faster-RCNN
- ImageNet labels: strong supervision

NB: all methods re-implemented on same backbone network (ResNet-101)

Self-supervision task	ImageNet Classification top-5 accuracy	PASCAL VOC Detection mAP
Rel. Pos	59.21	66.75
Colour	62.48	65.47
Exemplar	53.08	60.94
Rel. Pos + colour	66.64	68.75
Rel. Pos + Exemplar	65.24	69.44
Rel. Pos + colour + Exemplar	68.65	69.48
ImageNet labels	85.10	74.17

Что сейчас?

Semi-Supervised Image Classification on ImageNet - 10% labeled data



View Top 5 Accuracy ▾

Edit

RANK	METHOD	TOP 5 ACCURACY	TOP 1 ACCURACY	PAPER	CODE	RESULT	YEAR
1	SimCLRv2 self-distilled (ResNet-152 x3, SK)	95.5%	80.9%	Big Self-Supervised Models are Strong Semi-Supervised Learners	🔗	🔗	2020
2	SimCLRv2 distilled (ResNet-50 x2, SK)	95.0%	80.2%	Big Self-Supervised Models are Strong Semi-Supervised Learners	🔗	🔗	2020
3	SimCLRv2 (ResNet-152 x3, SK)	95.0%	80.1%	Big Self-Supervised Models are Strong Semi-Supervised Learners	🔗	🔗	2020
4	SimCLRv2 distilled (ResNet-50)	93.4%	77.5%	Big Self-Supervised Models are Strong Semi-Supervised Learners	🔗	🔗	2020
5	SimCLR (ResNet-50 4x)	92.6%		A Simple Framework for Contrastive Learning of Visual Representations	🔗	🔗	2020
6	SimCLRv2 (ResNet-50 x2)	91.9%	73.9%	Big Self-Supervised Models are Strong Semi-Supervised Learners	🔗	🔗	2020

SimCLR (2020)



(a) Original



(b) Crop and resize



(c) Crop, resize (and flip)



(d) Color distort. (drop)



(e) Color distort. (jitter)



(f) Rotate $\{90^\circ, 180^\circ, 270^\circ\}$



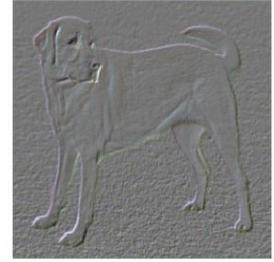
(g) Cutout



(h) Gaussian noise



(i) Gaussian blur



(j) Sobel filtering

Figure 4. Illustrations of the studied data augmentation operators. Each augmentation can transform data stochastically with some internal parameters (e.g. rotation degree, noise level). Note that we *only* test these operators in ablation, the *augmentation policy used to train our models* only includes *random crop (with flip and resize), color distortion, and Gaussian blur*. (Original image cc-by: Von.grzanka)

SimCLR (2020)

Algorithm 1 SimCLR's main learning algorithm.

```

input: batch size  $N$ , constant  $\tau$ , structure of  $f, g, \mathcal{T}$ .
for sampled minibatch  $\{\mathbf{x}_k\}_{k=1}^N$  do
    for all  $k \in \{1, \dots, N\}$  do
        draw two augmentation functions  $t \sim \mathcal{T}, t' \sim \mathcal{T}$ 
        # the first augmentation
         $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k)$ 
         $\mathbf{h}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1})$  # representation
         $\mathbf{z}_{2k-1} = g(\mathbf{h}_{2k-1})$  # projection
        # the second augmentation
         $\tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$ 
         $\mathbf{h}_{2k} = f(\tilde{\mathbf{x}}_{2k})$  # representation
         $\mathbf{z}_{2k} = g(\mathbf{h}_{2k})$  # projection
    end for
    for all  $i \in \{1, \dots, 2N\}$  and  $j \in \{1, \dots, 2N\}$  do
         $s_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$  # 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  $g(\cdot)$ 

```

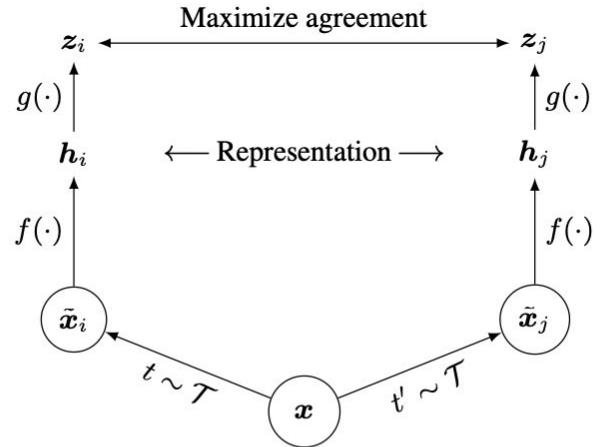


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation \mathbf{h} for downstream tasks.