

Глубокое обучение

Коломейцева Катерина

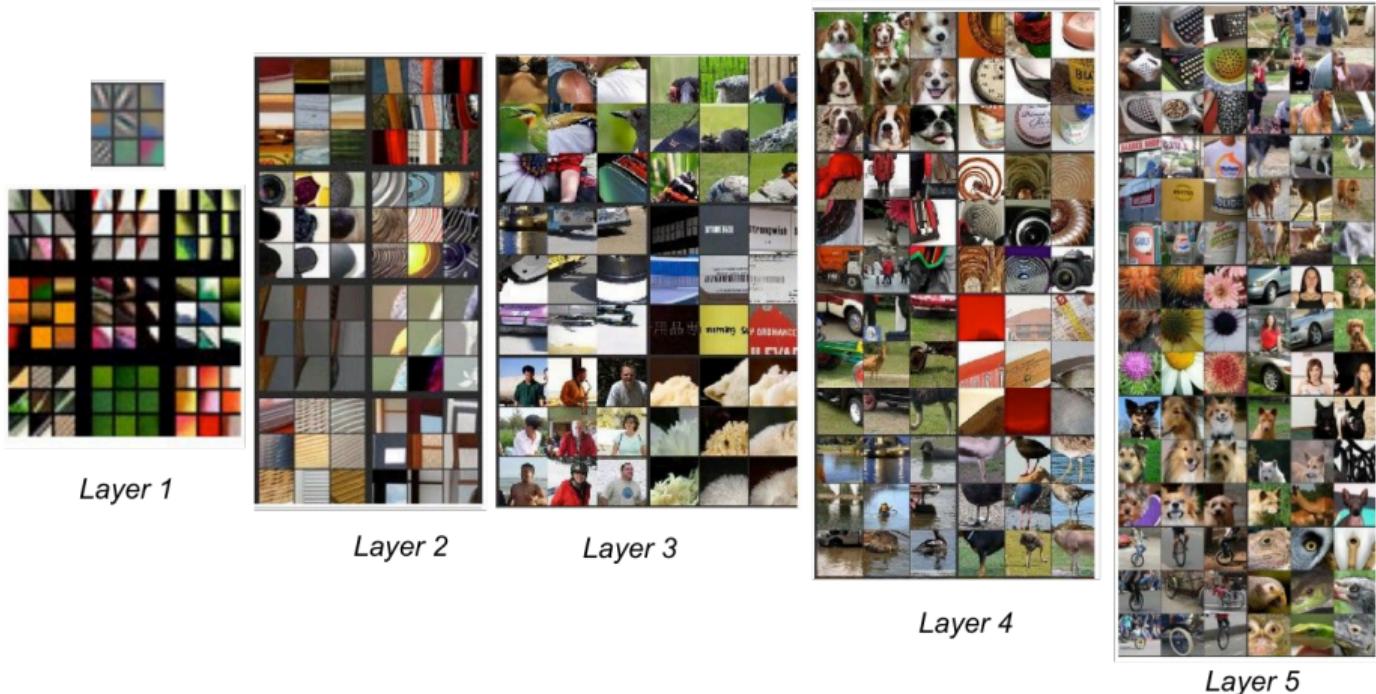
Лекция 6: Что видят сетки, перенос стиля
Автоэнкодеры

Agenda

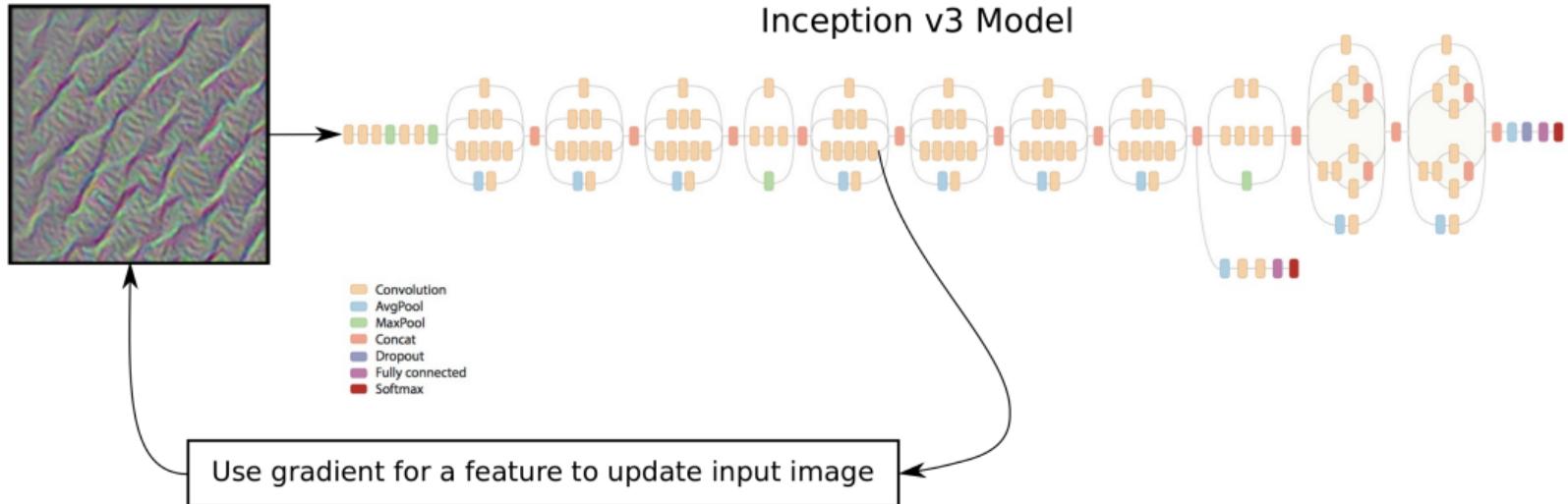
- Что видят свёрточные сетки?
- Перенос стиля

Что видят нейросетки

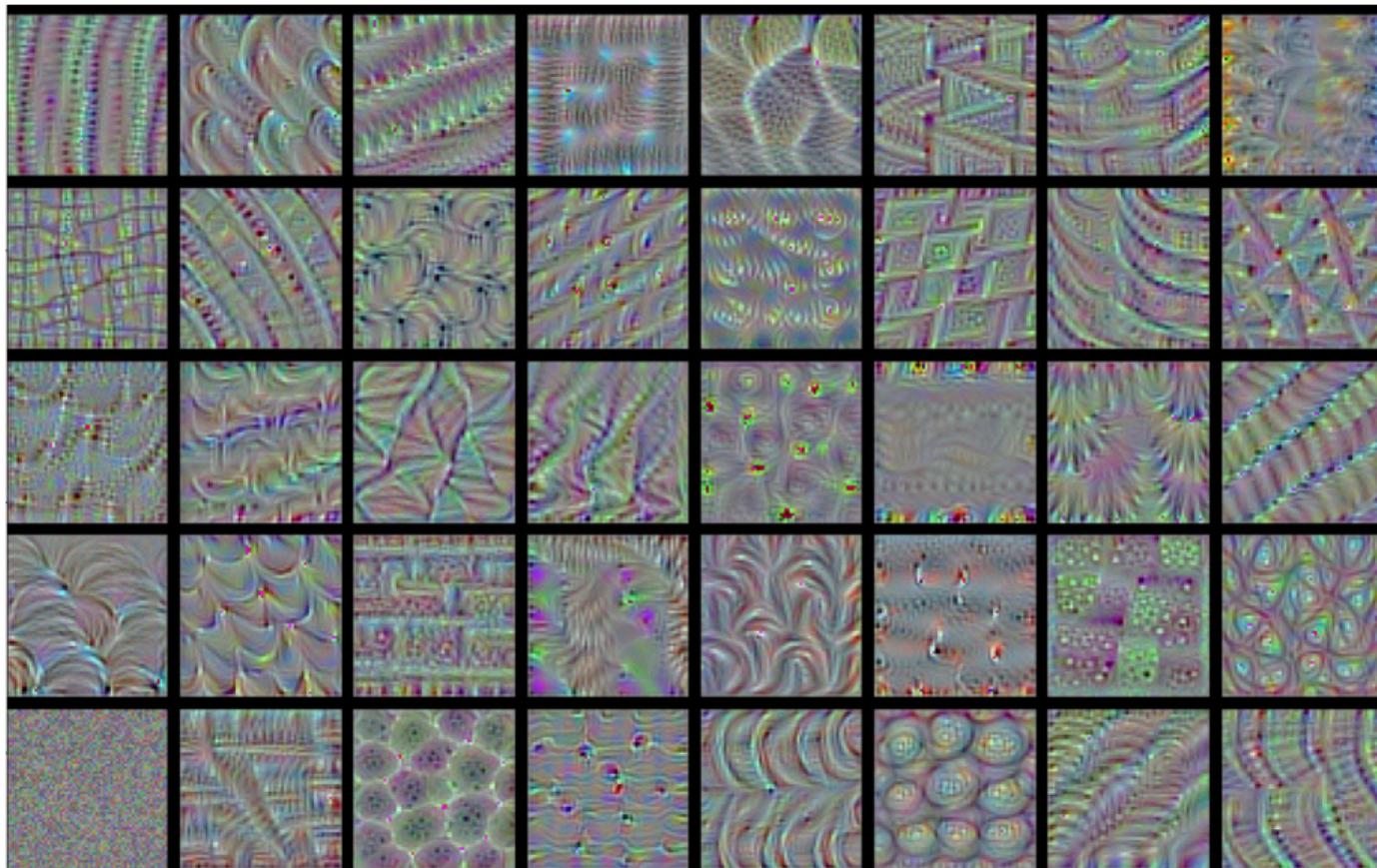
Опять этот слайд!



Есть несколько способов заглянуть внутрь свёрточной сетки, мы сейчас посмотрим на один из них



Пример визуализации фильтров сверточных сетей



Что видит свёртка

- На вход идет пустое изображение, мы хотим изменить его пиксели так, чтобы активация конкретной свёртки была максимальной
- Максимизируем среднее значение свёртки по пикселям
- Шаг градиентного спуска: меняем пиксели так, чтобы свёртка выдавала на выход более большие значение
- На входной карточке постепенно прорисовывается шаблон, который возбуждает соответствующую свёртку
- Если на вход в сетку подсунуть не пустую карточку, а какое-то изображение, то фильтр отрисуется на нём. Если эту процедуру немного подправить, получится наркомания под названием **Deep dream**

Deep dream



Общая идея создания таких изображений основана на том, что в фотографии или картине усиливают те черты, которые напоминают системе что-то знакомое. Например, если система натренирована на распознавание лица, она даже в совершенно случайном изображении — например, облаков — увидит какие-то его фрагменты. Затем их можно будет усилить, получив изображения с лицами на облаках.

Перенос стиля

Приложение Prisma



Gothic



Candy



Monono



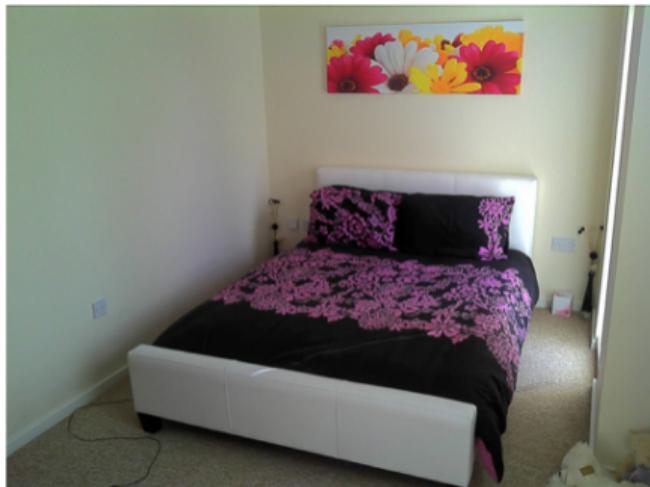
Перенос стиля



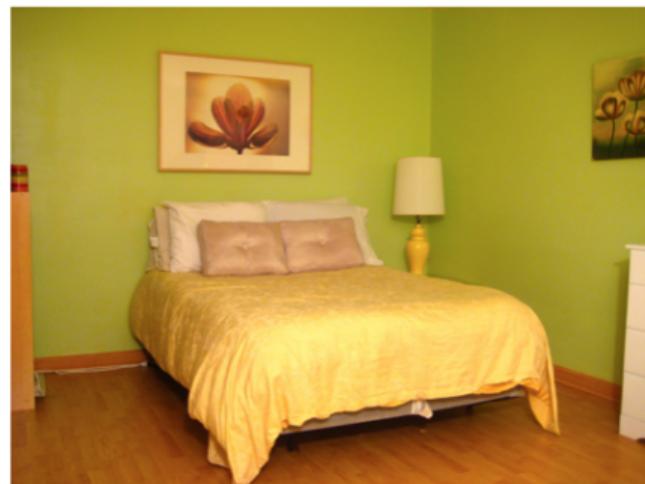
Перенос стиля



Ваша комната



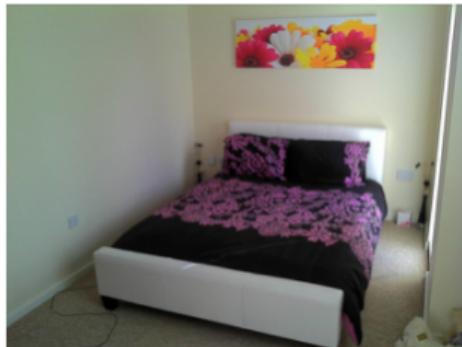
Комната сына
маминой подруги



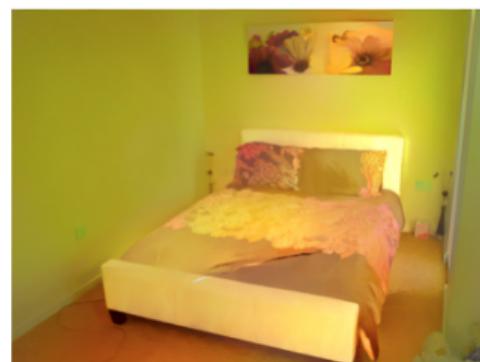
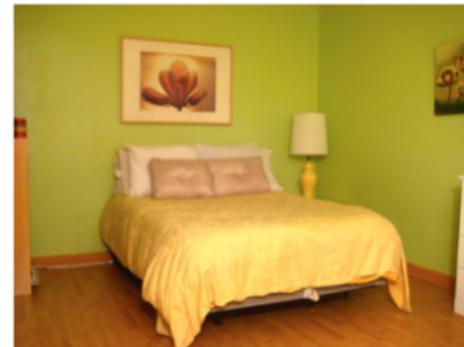
<https://habr.com/ru/post/402665/>

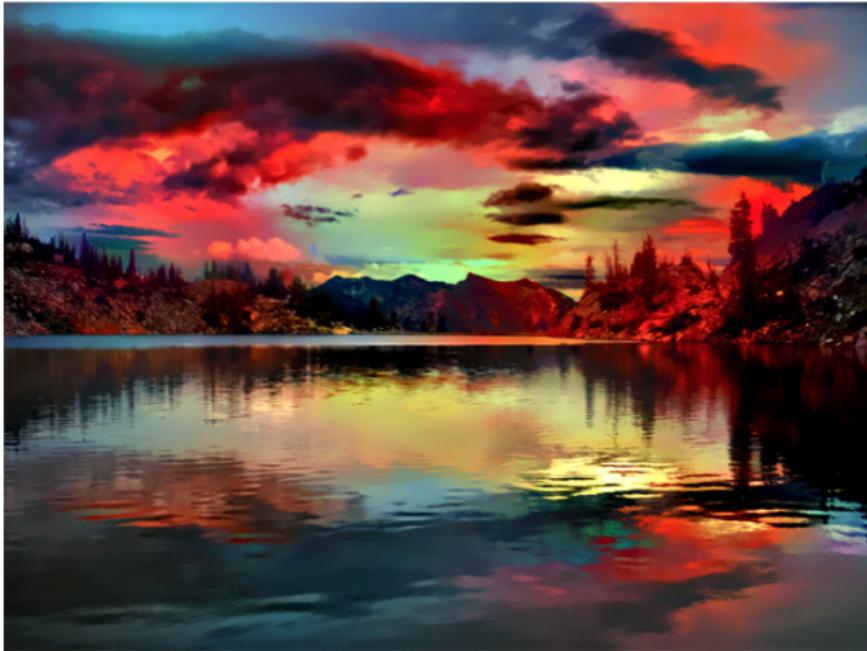
<https://github.com/LouieYang/deep-photo-styletransfer-tf>

Ваша комната



Комната сына
маминой подруги





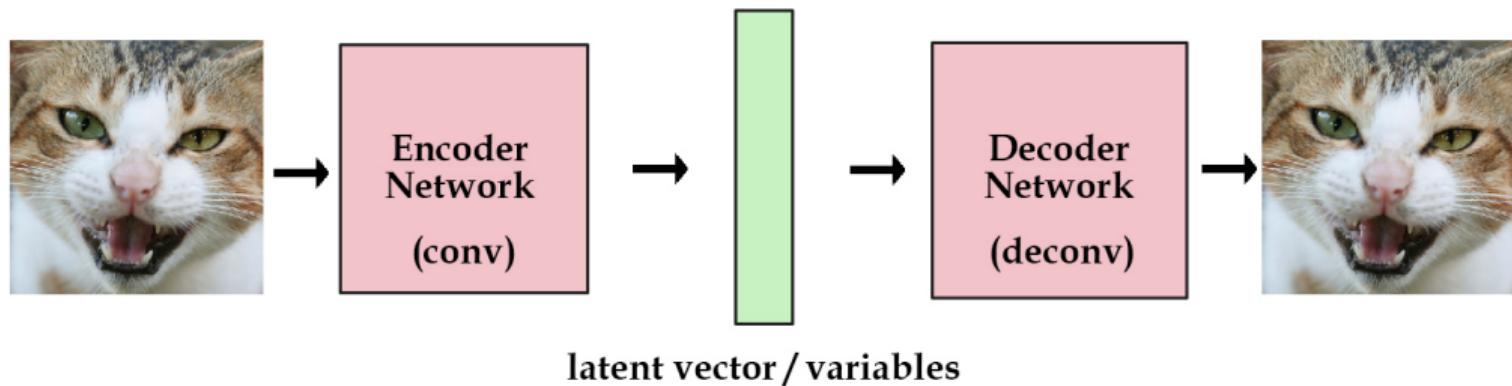


Автокодировщики

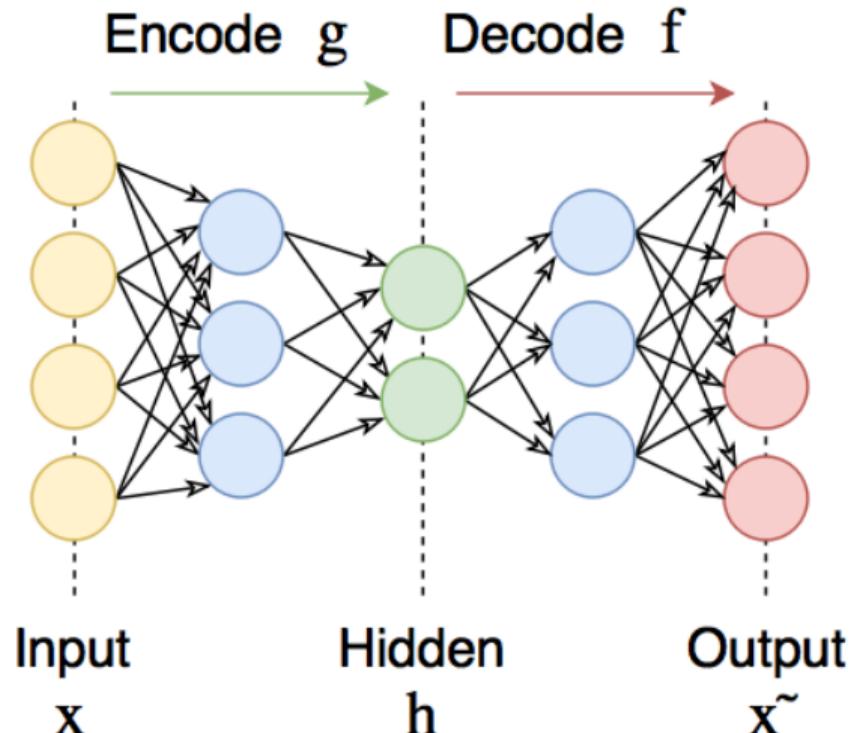
Автокодировщики

- Когда данных много, мы хотим понизить их размерность. Классическое машинное обучение позволяет делать это с помощью метода главных компонент, tsne и других методов. Нейросети также позволяют решать подобную задачу сжатия с минимальными потерями.
- Понижение размерности — задача обучения без учителя
- Давайте превратим её в обучение с учителем!

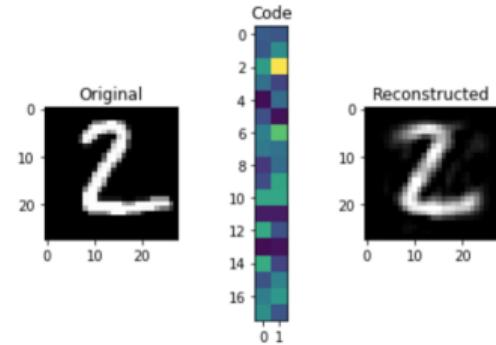
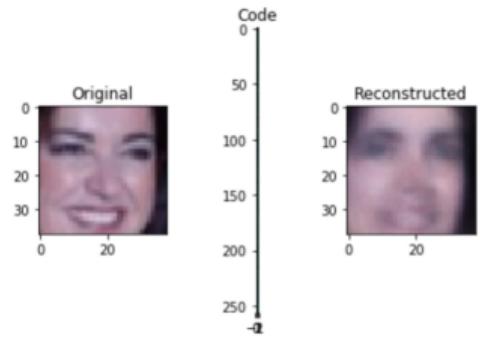
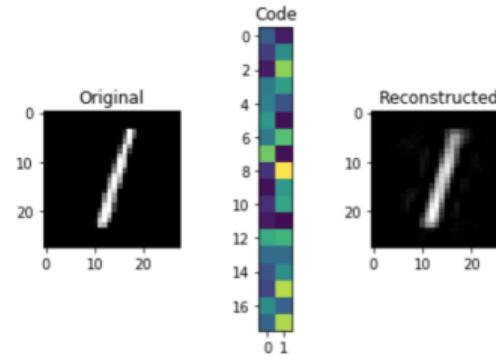
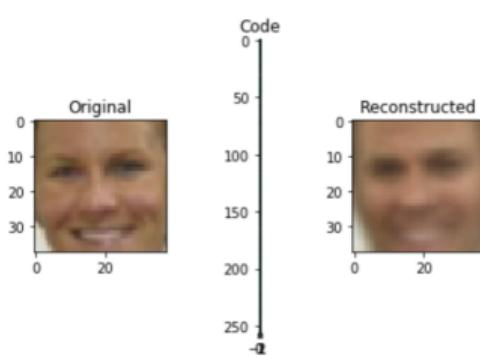
Автокодировщики



Автокодировщики



Пример сжатия

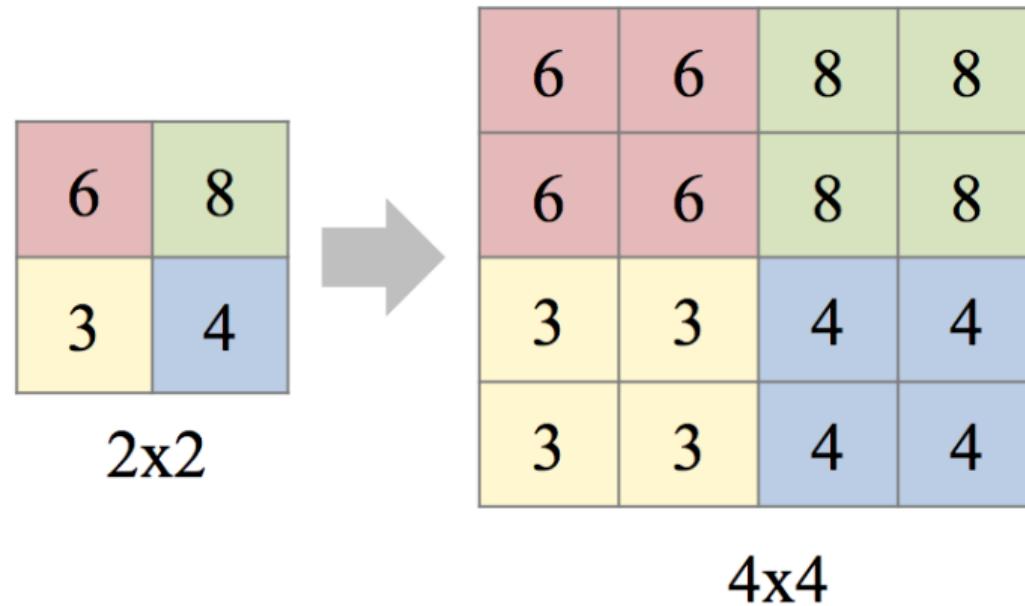


Как используют

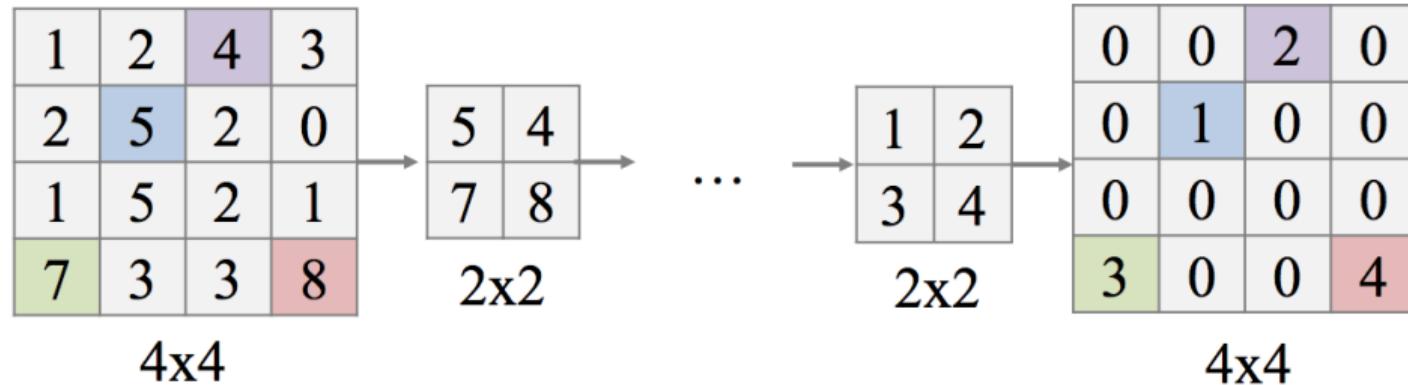
- Для предобучения сетей. Именно так в 2005 началась революция.
- Скрытое представление признаков можно использовать в других моделях в качестве фичей.
- Конструкцию автокодировщика можно немного модернизировать для решения других задач, например для генерирования подписи по картинке (про это позже)

Сверточные слои декодера

Nearest neighbor unpooling



Max unpooling

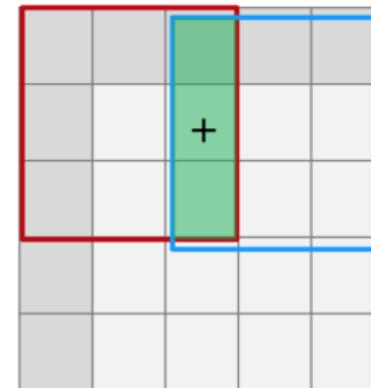


Learnable unpooling: Transpose convolution

Input: 2x2



Input gives
weight for
filter

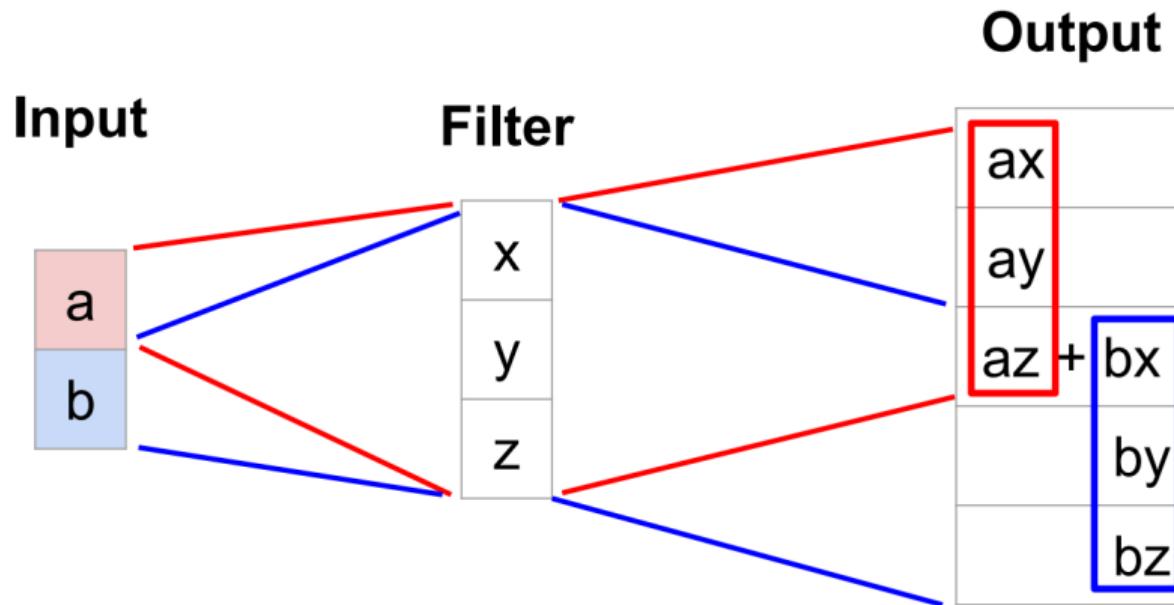


Stride: 2

Output: 4x4

- Каждую клетку надо распаковать в 4 клетки \Rightarrow свёртка 3×3 со сдвигом 2

Пример:



Типы автокодировщиков

Обычный автокодировщик

В обычном автокодировщике восстанавливаем следующую последовательность: $x = f(g(x))$, изменяя 2 функции $f(x)$ и $g(x)$. Loss в нашем случае - $L(x, f(g(x)))$ Тождественно не можем выучить из-за искусственного ограничения количества нейронов в середине.

- Однослойной автокодировщик по своему действию совпадает с PCA
- Если задаем многослойный автокодировщик может находить достаточно сложные особенности в данных (по сути, правильной архитектуре - любые)
- Можно делать и сверточные, если работаем с изображениями

Denoise autoencoder

Мы пытаемся не просто восстановить выход по входу, но и ещё искусственно добавляем шум к входным данным.

Последовательно решим следующую задачу $x = f(g(\hat{x}))$, где \hat{x} зашумленные входные данные. Для изображений шум можно задавать 2-мя способами - затемнять какую-часть изображения или добавлять шум к каждому пикселию. Весь остальной процесс обучения совпадает с обычным автокодировщиком.

Разреженный автокодировщик

Разреженный автокодировщик Теперь мы к нашему лосу добавляем регуляризатор. $L(x, g(f(x))) + \omega(h)$, где $g(h)$ - выход декодера, $h = f(x)$ - выход энкодера, и ограничение накладывается на энкодер. Как ограничения обычно использую L1 или L2 норму. Такой автокодировщик не сможет полностью выучить картинку из-за штрафа при любой архитектуре. Он может расширяться к выходу, пытаясь разложить сигнал на множество статистически независимых сигналов. Используют его также как и обычный автокодировщик, если требуется чтобы получающиеся латентные векторы были более линейно-независимые. Из-за того, что он пытается разложить один сигнал на множество, иногда его для разложения сигнала на составляющие - аналог вайвлет преобразований для аудио.

Собираем свои автокодировщики