

# Глубинное обучение

Перенос стиля. Интерпретация свёрточных сетей. Автокодировщики

Даниил Водолазский

ВШЭ

1 сентября 2021 г.



НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ

# Содержание

- ① Перенос стиля
- ② Интерпретация свёрточных нейронных сетей
- ③ Автокодировщики  
Типы автокодировщиков
- ④ Что узнали

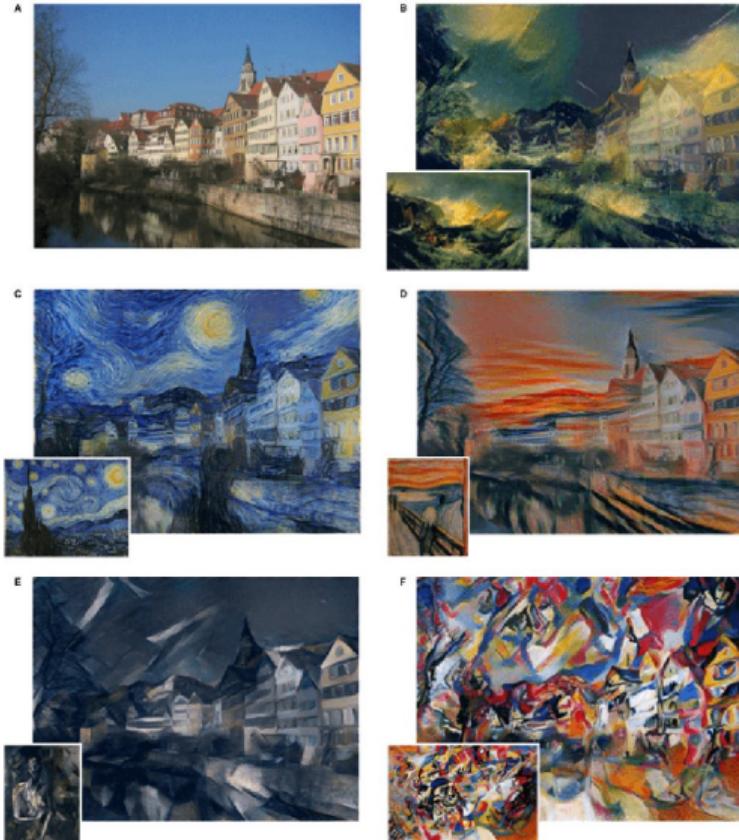
# Приложение Prisma



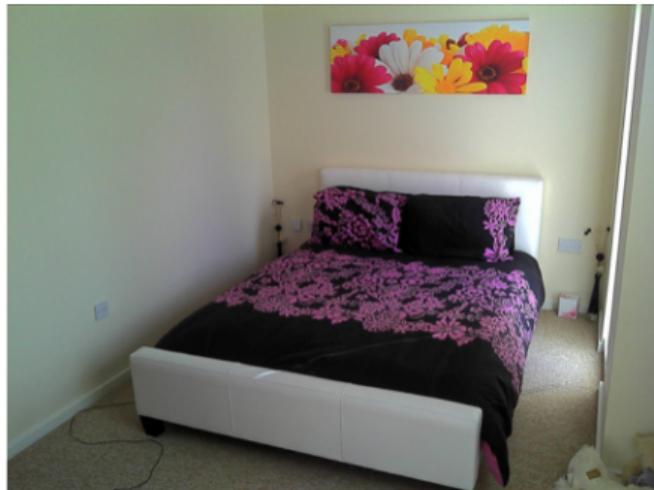
# Перенос стиля



# Перенос стиля



# Ваша комната



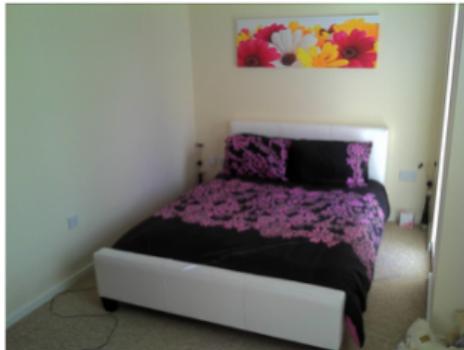
# Комната сына маминой подруги



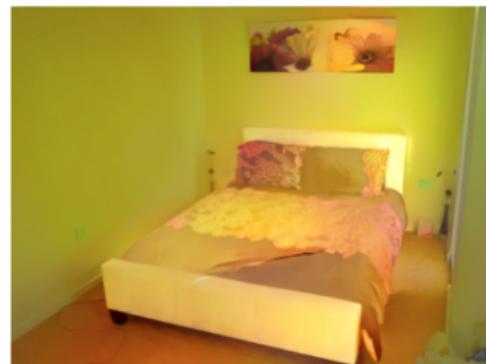
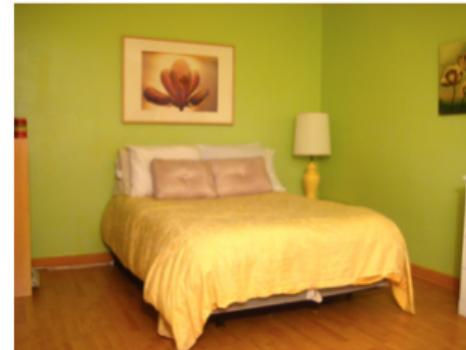
<https://habr.com/ru/post/402665/>

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

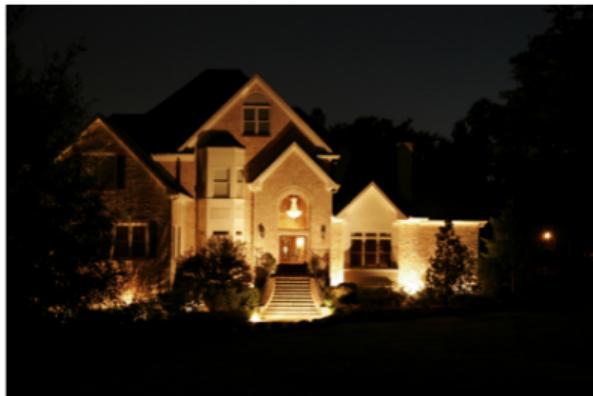
Ваша комната



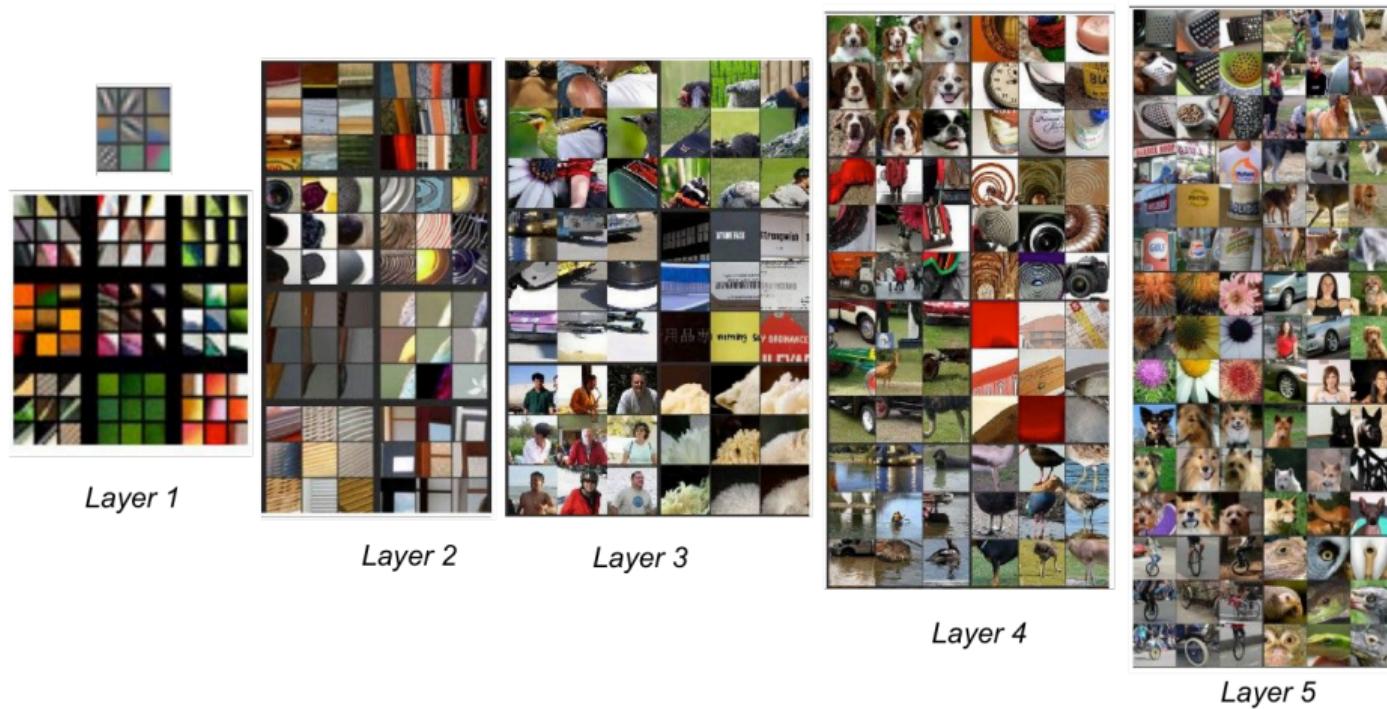
Комната сына  
маминой подруги



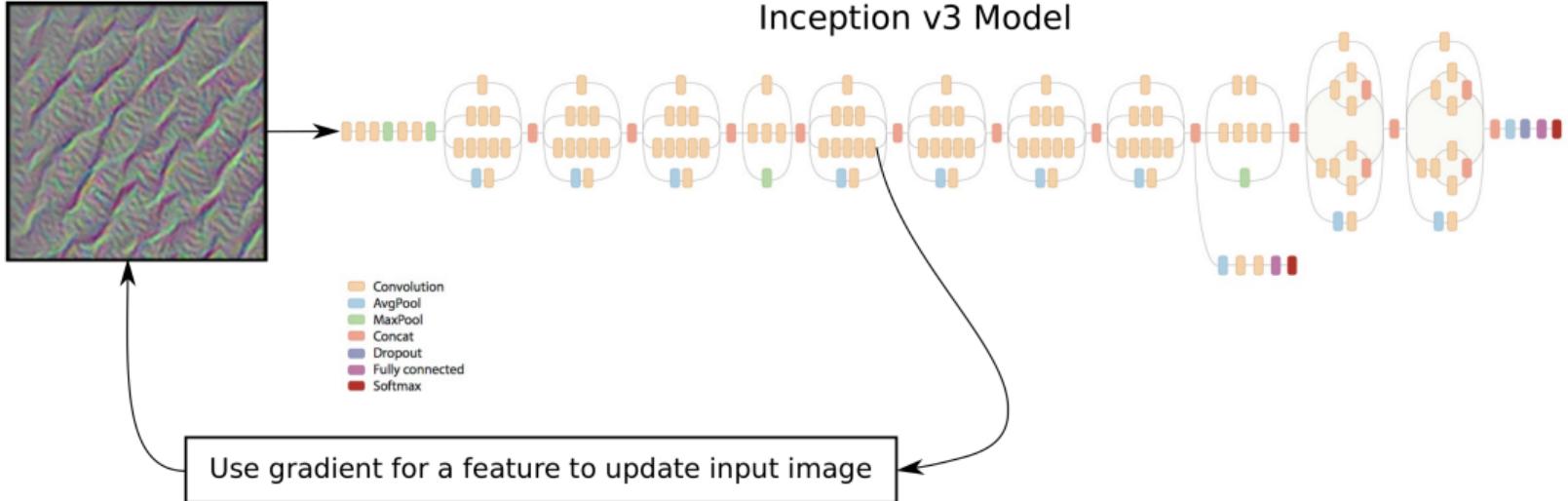




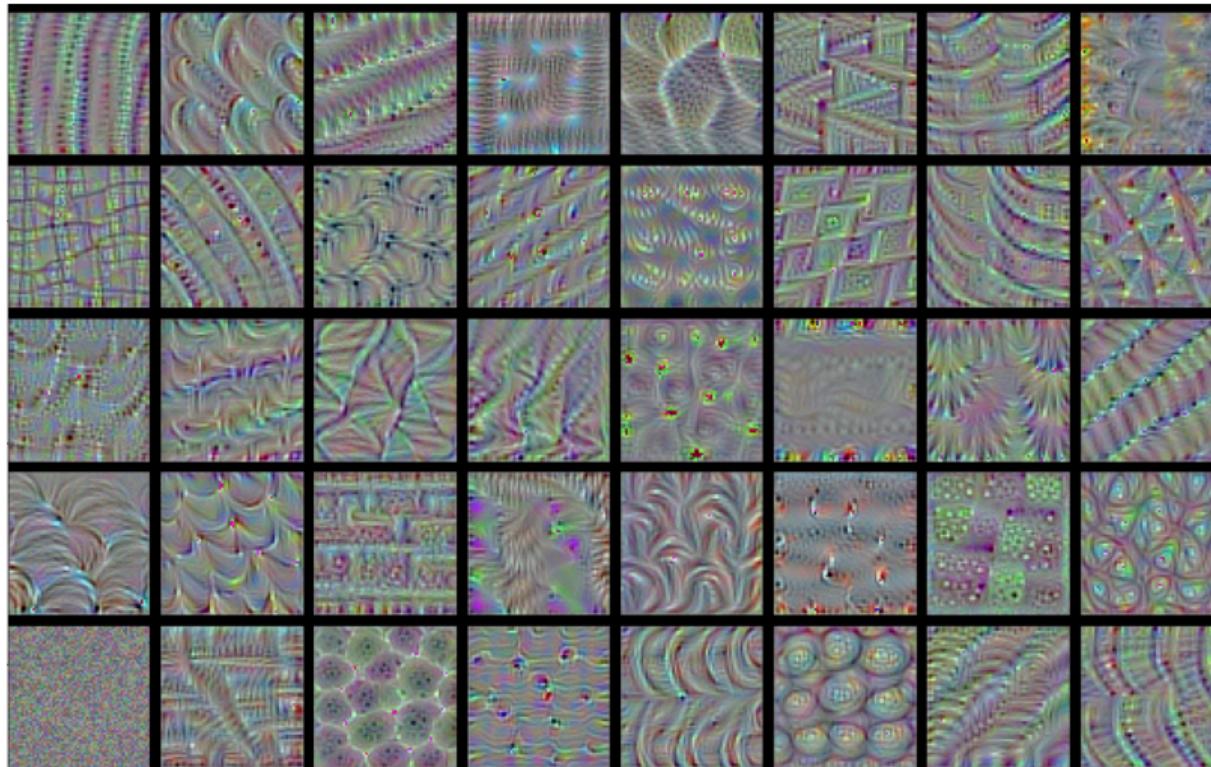
# Опять этот слайд!



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



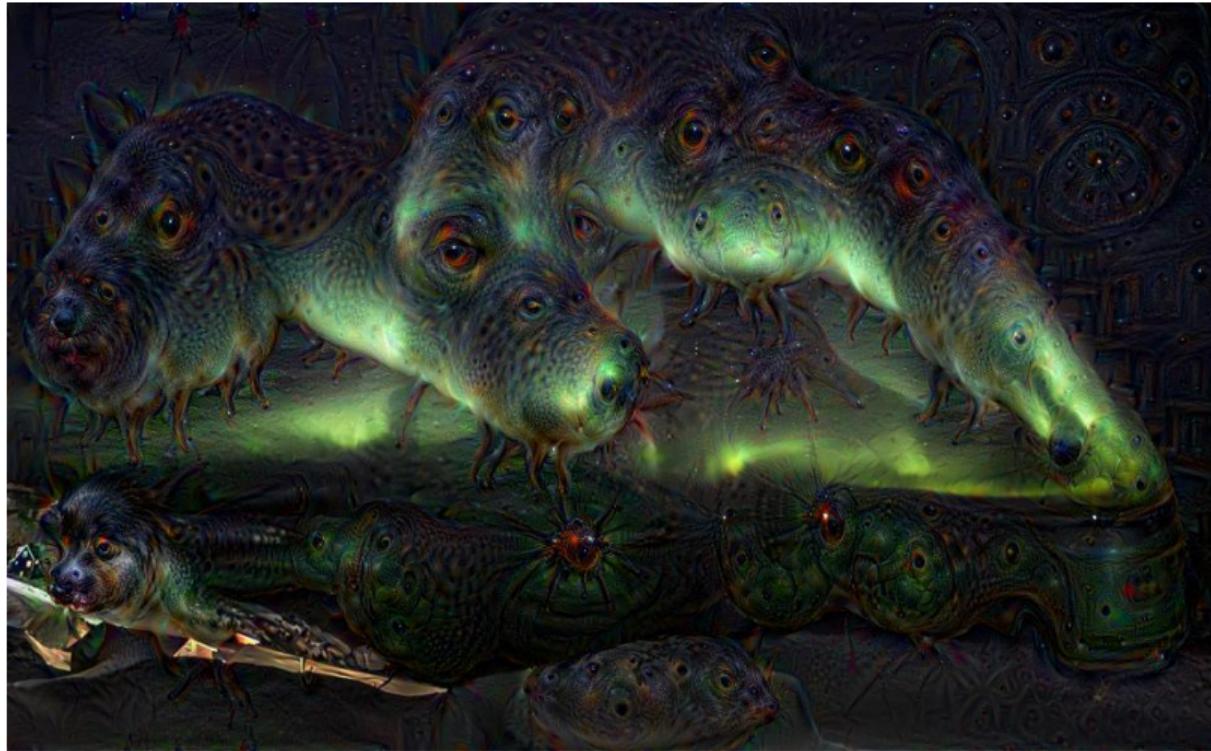
# Пример из вашей домашки



# 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 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



<https://nplus1.ru/material/2015/07/13/use>

# Перенос стиля

Впервые такую штуку провернули в 2015 году, в статье **A Neural Algorithm of Artistic Style** (<https://arxiv.org/abs/1508.06576>).

**Важно:** в задаче переноса стиля мы берём **готовую** нейронку, веса внутри неё мы никак не меняем. Мы изменяем **пиксели исходного изображения!** Держите эту мысль у себя в голове до победного конца. Пиксели изображения в этой задачке — это и есть переменные, которые мы будем изменять в ходе градиентного спуска. **Игра перевернулась.**

# Перенос стиля: стиль и контент

Мы будем пытаться раскладывать изображение на две составляющие:

- **стиль** — текстуры, цвета, визуальные узоры на изображении в различных пространственных масштабах (детали);
- **содержание, контент** — макроструктура изображения (верхнеуровневый контур).

Например: сине-желтые круглые мазки кисти на картине — это стиль, дома на фотографии — это контент.

Мы хотим сохранить контент, но заменить стиль. Так и запишем:

$$L_{total} = \alpha L_{content} + \beta L_{style}.$$

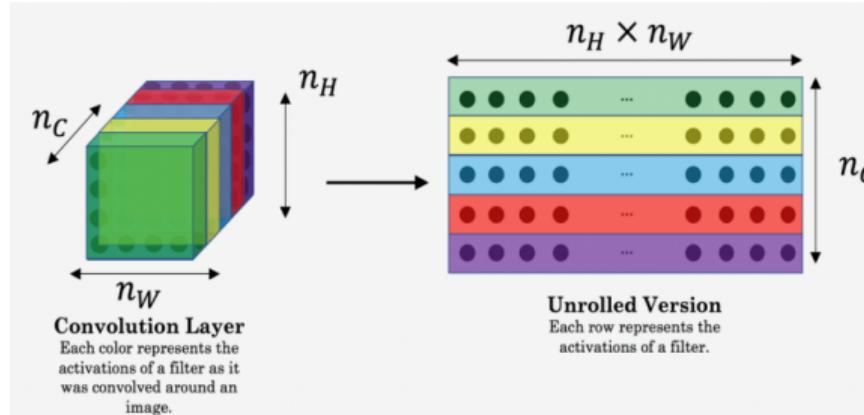
# Перенос стиля: контент

Как понять, насколько сильно контент исходного изображения отличается от контента сгенерированного? Ответ прост: найти **поэлементную невязку** всех наших активаций.

$$B_l = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2, L_{content} = \sum_{l=0}^L u_l B_l,$$

где

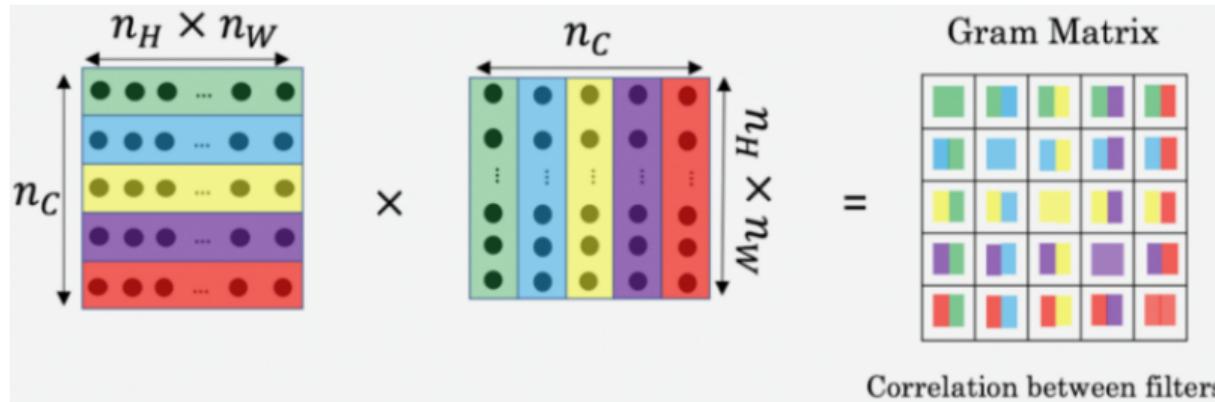
- $F_{ij}^l$  — активация пикселя  $i, j$  на слое  $l$  для исходного изображения,
- $P_{ij}^l$  — активация пикселя  $i, j$  на слое  $l$  для нового изображения.



# Перенос стиля: стиль

Стиль, в свою очередь, никак не должен зависеть от расположения объектов. Поэтому нам нужно избавиться в функции потерь от пространственной информации. В частном случае на слое  $l$  для сгенерированного изображения  $G$  используется матрица Грэма  $G^l$  (аналогично вычисляется  $A^l$  для оригинального изображения):

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l, E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2, L_{style} = \sum_{l=0}^L w_l E_l.$$

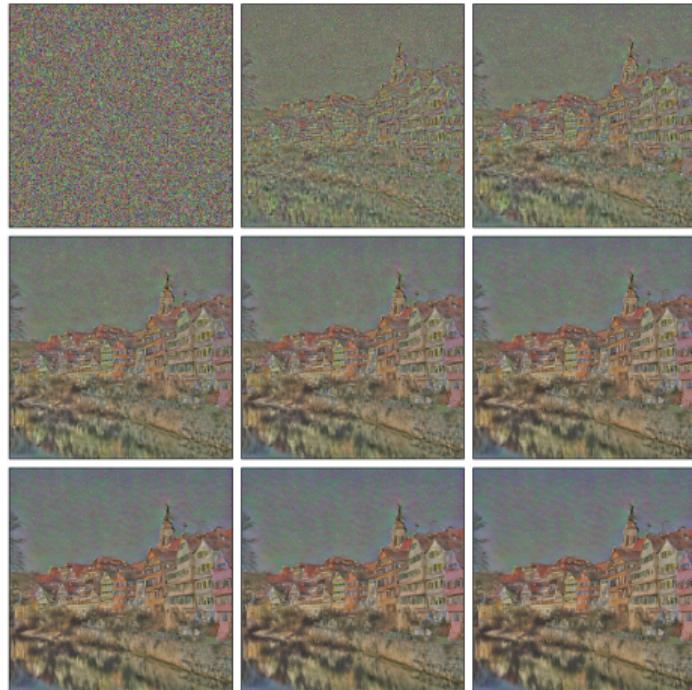


# Content loss



<https://habr.com/ru/company/mailru/blog/306916/>

# Content loss

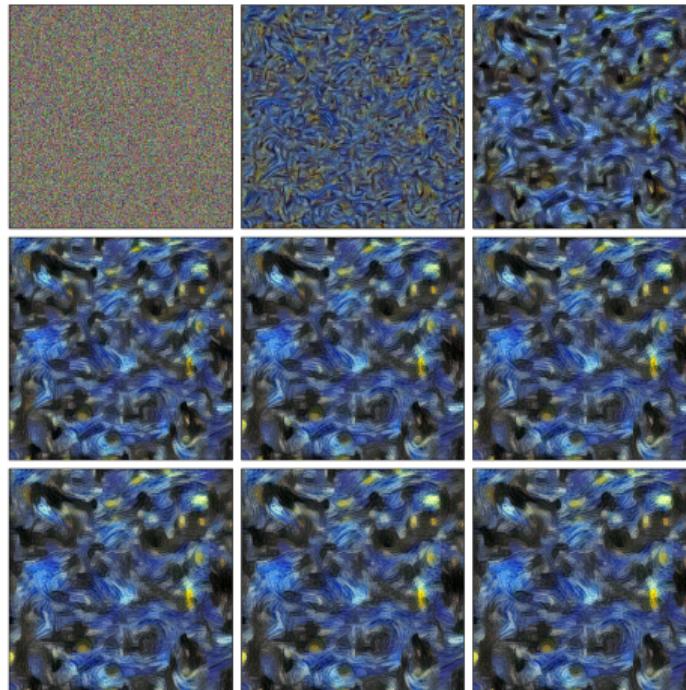


<https://habr.com/ru/company/mailru/blog/306916/>

# Style loss



# Style loss



<https://habr.com/ru/company/mailru/blog/306916/>

# Смесь нескольких стилей



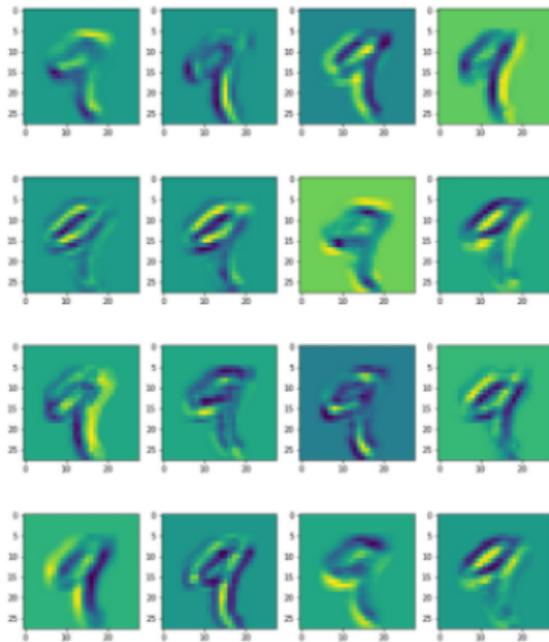
<https://arxiv.org/pdf/1610.07629.pdf>

# Содержание

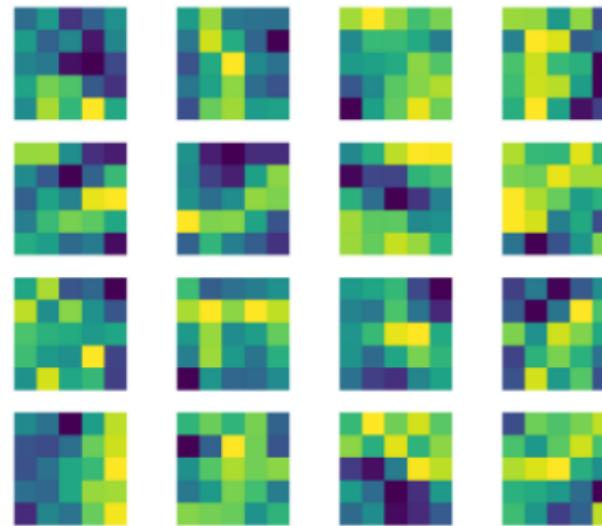
- ① Перенос стиля
- ② Интерпретация свёрточных нейронных сетей
- ③ Автокодировщики  
Типы автокодировщиков
- ④ Что узнали

# Интерпретация свёрточных нейронных сетей

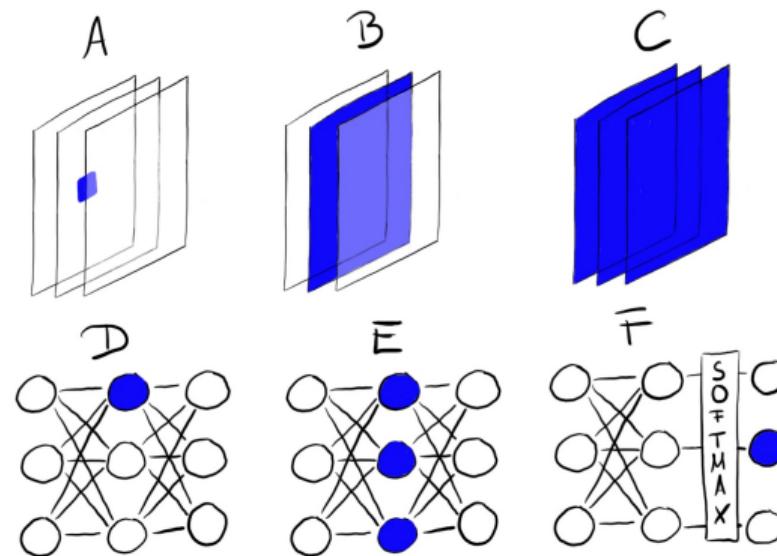
Feature map



Filters



# Интерпретация свёрточных нейронных сетей



Можно визуализировать А) свёрточный нейрон, В) свёрточный канал, С) свёрточный слой, Д) нейрон, Е) скрытый слой, Ф) нейрон для вероятности класса.

# Интерпретация свёрточных нейронных сетей

В математической постановке задача визуализации — это оптимизационная задача. Мы предполагаем, что веса сети фиксированы, то есть сеть уже обучена. Мы ищем изображение, которое максимизирует активацию юнита (в данном случае одного нейрона):

$$img^* = \operatorname{argmax}_{img} h_{n,x,y,z}(img),$$

где  $h$  — функция активации нейрона.

Для каналов берут усреднение по всем пространственным координатам:

$$img^* = \operatorname{argmax}_{img} \sum_{x,y} h_{n,x,y,z}(img).$$

Вместо максимизации можно минимизировать активацию и получать интересные результаты.

# Интерпретация свёрточных нейронных сетей

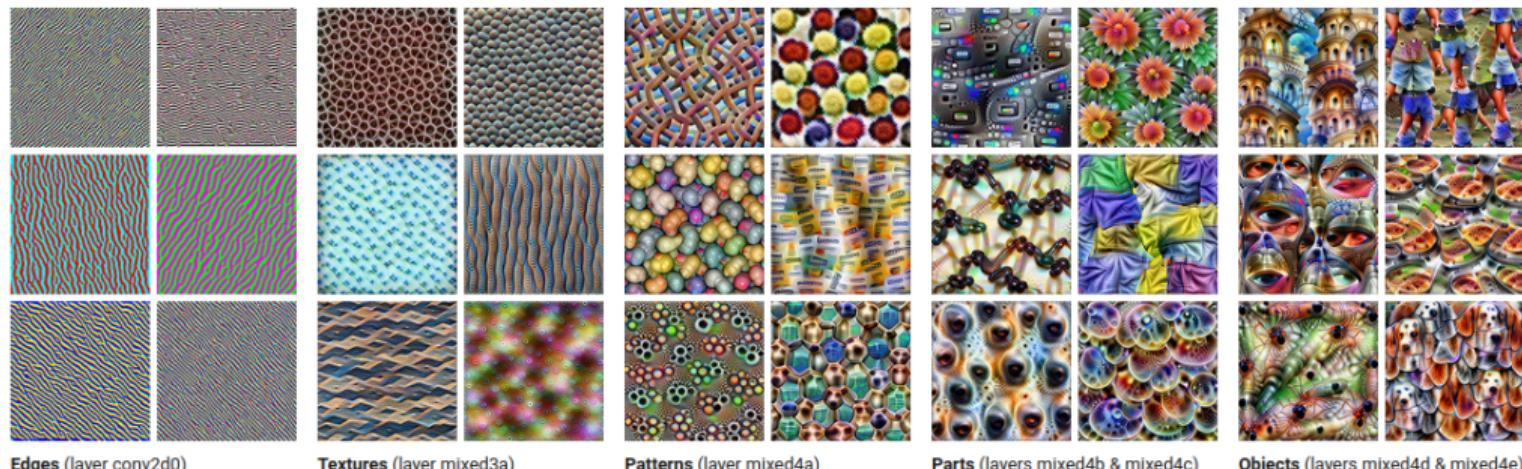


Положительные (слева) и отрицательные (справа) активации Inception V1, нейрон 484 со слоя mixed4d pre relu.<sup>1</sup>

<sup>1</sup>[https://colab.research.google.com/github/tensorflow/lucid/blob/master/notebooks/feature-visualization/negative\\_neurons.ipynb](https://colab.research.google.com/github/tensorflow/lucid/blob/master/notebooks/feature-visualization/negative_neurons.ipynb)

## Feature Visualization

How neural networks build up their understanding of images



Edges (layer conv2d0)

Textures (layer mixed3a)

Patterns (layer mixed4a)

Parts (layers mixed4b & mixed4c)

Objects (layers mixed4d & mixed4e)

Feature visualization allows us to see how GoogLeNet<sup>[1]</sup>, trained on the ImageNet<sup>[2]</sup> dataset, builds up its understanding of images over many layers. Visualizations of all channels are available in the [appendix](#).

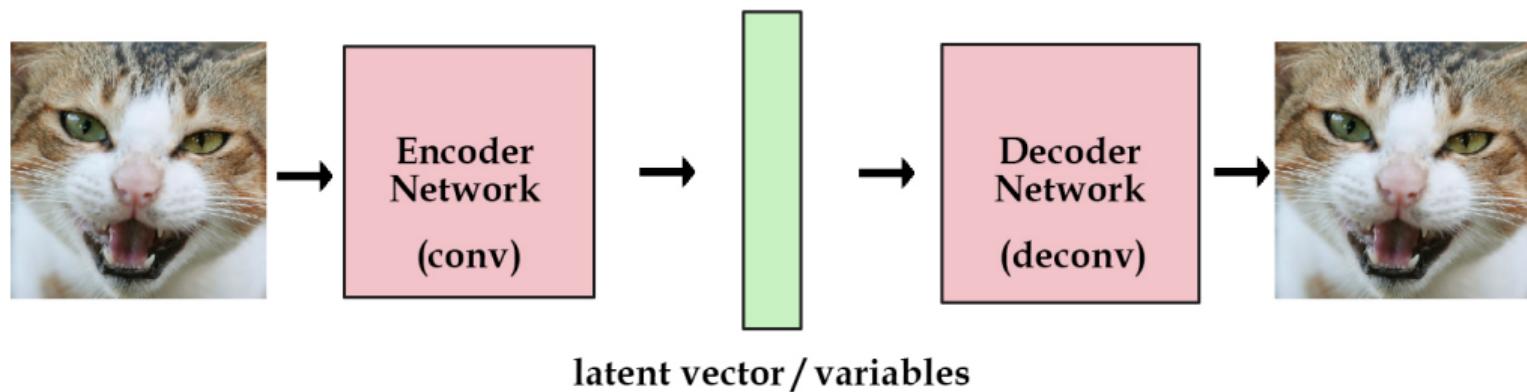
# Содержание

- ① Перенос стиля
- ② Интерпретация свёрточных нейронных сетей
- ③ Автокодировщики  
Типы автокодировщиков
- ④ Что узнали

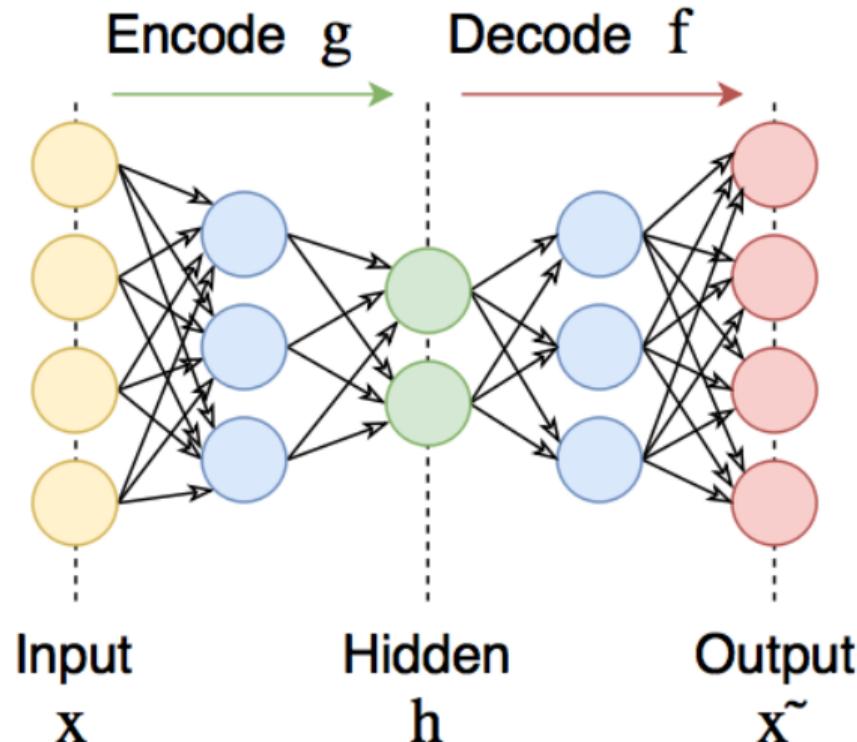
# Автокодировщики

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

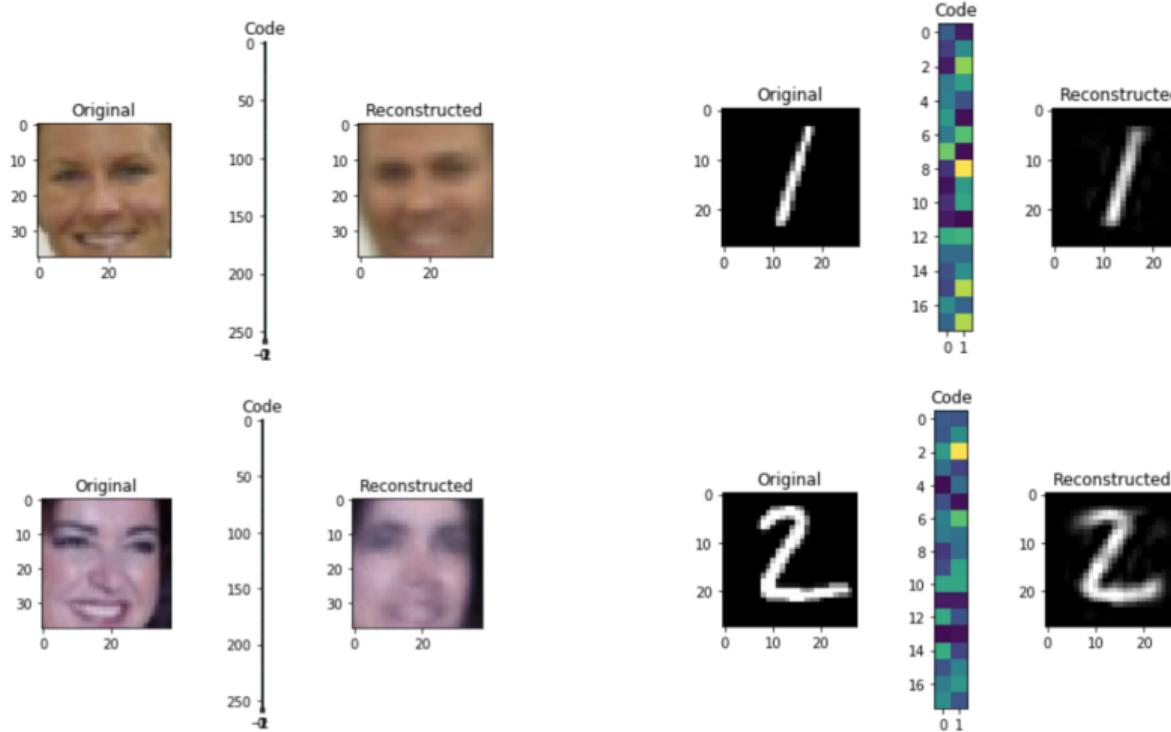
# Автокодировщики



# Автокодировщики



# Автокодировщики



# Автокодировщики. Применение

- Для предобучения сетей. Именно так в 2005 началась революция.
- Скрытое представление признаков можно использовать в других моделях в качестве признаков.
- Генерация похожих объектов.
- Хранение и обработка больших объемов данных.
- Сжатие медиафайлов (изображения, аудио, видео) и их декодирование как альтернатива классическим технологиям (JPEG, MP3, AVI, etc.).

# Типы автокодировщиков. Обычный

В обычном автокодировщике решаем следующую задачу:  $x = g(f(x))$ , где

- $f(x)$  — энкодер (сжимает данные),
- $g(h)$  — декодер (восстанавливает данные).

В глубинном обучении  $f$  и  $g$  — это две нейросети.

Функция потерь (например, MSE):

$$L(x, g(f(x))) \rightarrow \min_{f,g} .$$

Тождественно не можем выучить из-за искусственного ограничения количества нейронов в середине. Важные наблюдения:

- Однослойной автокодировщик по своему действию совпадает с PCA.
- Многослойный автокодировщик может находить достаточно сложные особенности в данных (по сути, при правильной архитектуре — любые).
- Можно использовать свёрточные слои, если работаем с изображениями.

# Типы автокодировщиков. Шумоподавляющий

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

Формально мы пытаемся решить следующую задачу  $x = g(f(\hat{x}))$ , где  $\hat{x}$  — зашумлённые входные данные. Для изображений шум можно задавать двумя способами:

- ① затемнять какую-часть изображения;
- ② добавлять шум к каждому пикселью.

Весь остальной процесс обучения совпадает с обычным автокодировщиком.

# Типы автокодировщиков. Разреженный

Добавляем регуляризацию:

$$L(x, g(f(x))) + \omega(f(x)) \rightarrow \min_{f,g} .$$

Ограничение накладывается на энкодер. Обычно берут  $L_1$ - или  $L_2$ -норму.

- Такой автокодировщик не сможет полностью выучить входные данные из-за штрафа при любой архитектуре. Он может расширяться к выходу, пытаясь разложить сигнал на множество статистически независимых сигналов.
- Использует 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 вейвлет-преобразований для аудио.

# Типы автокодировщиков. Вариационный (VAE)

О нём поговорим в следующий раз, когда будем разбирать генеративные модели.

# Содержание

- ① Перенос стиля
- ② Интерпретация свёрточных нейронных сетей
- ③ Автокодировщики  
Типы автокодировщиков
- ④ Что узнали

# Что узнали

- Как интерпретировать слои в свёрточных сетях (несколько способов).
- Как делать перенос стиля в изображениях.
- Как сжимать-разжимать данные в автокодировщиках.