

DEEP LEARNING

Неделя 6

Борис Коваленко, Артем Грачев, Святослав Елизаров

15 декабря 2017

Высшая школа экономики

ИНТЕРПРЕТАЦИЯ

ИНТЕРПРЕТАЦИЯ

Для того, чтобы интерпретировать полученную модель нам важно знать:

1. Какая область изображения (или любых других данных с пространственной структурой) была значимой при принятии решения
2. Какие признаки выбрала модель, как характерные для класса (в случае задачи классификации). Или более общо, какие признаки вызывают активацию конкретных нейронов.

Как нам это узнать?

ИНТЕРПРЕТАЦИЯ

Для того, чтобы интерпретировать полученную модель нам важно знать:

1. Какая область изображения (или любых других данных с пространственной структурой) была значимой при принятии решения
2. Какие признаки выбрала модель, как характерные для класса (в случае задачи классификации). Или более общо, какие признаки вызывают активацию конкретных нейронов.

Как нам это узнать? При помощи оптимизации!

ОПРЕДЕЛЕНИЕ ЗНАЧИМОЙ ОБЛАСТИ

Остановимся на первом пункте.

- Обозначим $\tilde{f}(x)$ интересующую нас модель
- Где x – это изображение, подаваемое на вход

Найдём градиент $\tilde{f}(x)$ по x :

$$\nabla_x \tilde{f} = \frac{\partial \tilde{f}(x)}{\partial x}$$

ОПРЕДЕЛЕНИЕ ЗНАЧИМОЙ ОБЛАСТИ

Остановимся на первом пункте.

- Обозначим $\tilde{f}(x)$ интересующую нас модель
- Где x – это изображение, подаваемое на вход

Найдём градиент $\tilde{f}(x)$ по x :

$$\nabla_x \tilde{f} = \frac{\partial \tilde{f}(x)}{\partial x}$$

Как мы можем интерпретировать данный градиент?

ОПРЕДЕЛЕНИЕ ЗНАЧИМОЙ ОБЛАСТИ

Из определения производной очевидно, что:

- Большие величины достигаются в точках, небольшое изменение которых приводит к большому изменению функции
- Градиент близок к 0 в точках, значение которых слабо влияет на результат

ОПРЕДЕЛЕНИЕ ЗНАЧИМОЙ ОБЛАСТИ

Из определения производной очевидно, что:

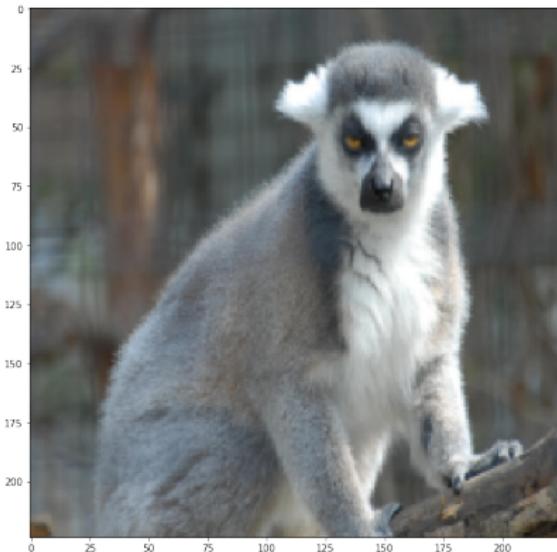
- Большие величины достигаются в точках, небольшое изменение которых приводит к большому изменению функции
- Градиент близок к 0 в точках, значение которых слабо влияет на результат

Следовательно значимость пикселя пропорциональна величине градиента.

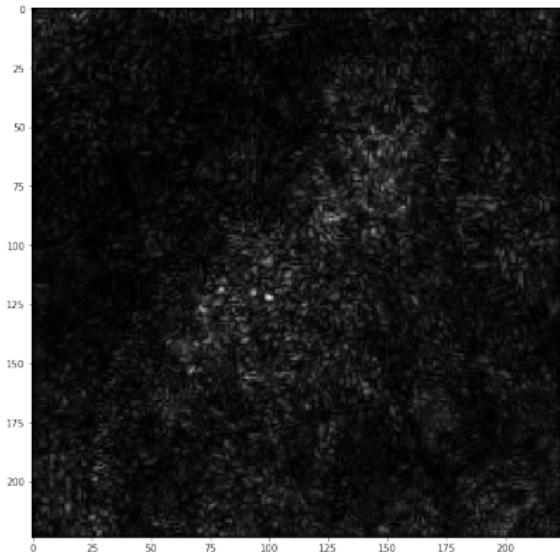
ОПРЕДЕЛЕНИЕ ЗНАЧИМОЙ ОБЛАСТИ

- Таким образом, отнормировав получившийся градиент в диапазоне от 0 до 255, можно легко визуализировать значимые объекты на изображении
- Такие визуализации носят название **saliency map**

ОПРЕДЕЛЕНИЕ ЗНАЧИМОЙ ОБЛАСТИ

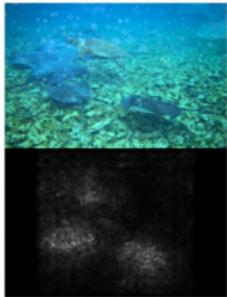


(a) Исходное изображение



(b) Saliency map

ОПРЕДЕЛЕНИЕ ЗНАЧИМОЙ ОБЛАСТИ



ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

- Каждый элемент в модели активируется при появлении определенных фич
- Значения вектора оценок зависят от присутствия фич, характерных для каждого из классов

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

- Каждый элемент в модели активируется при появлении определенных фич
- Значения вектора оценок зависят от присутствия фич, характерных для каждого из классов

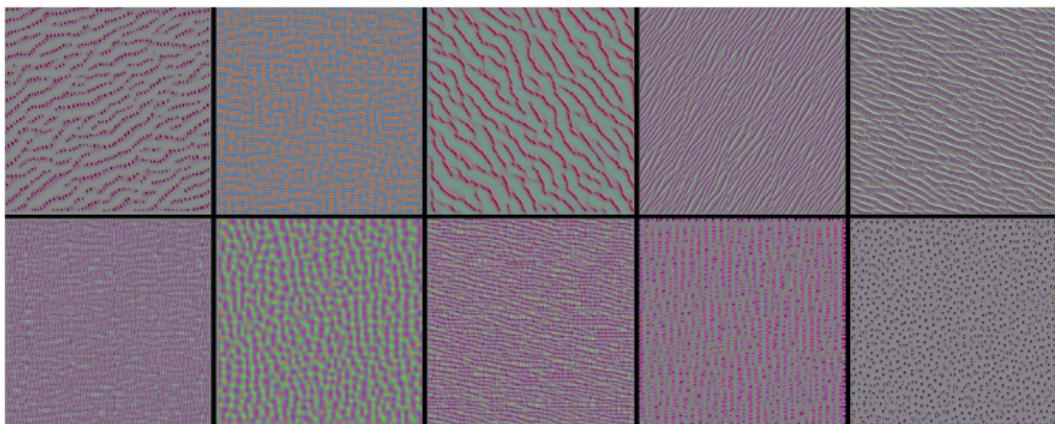
Идея: Подадим на вход шум и будем изменять его так, чтобы максимизировать значения выбранных активаций

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

Визуализируем VGG-16, обученный на ImageNet, предложенным способом.

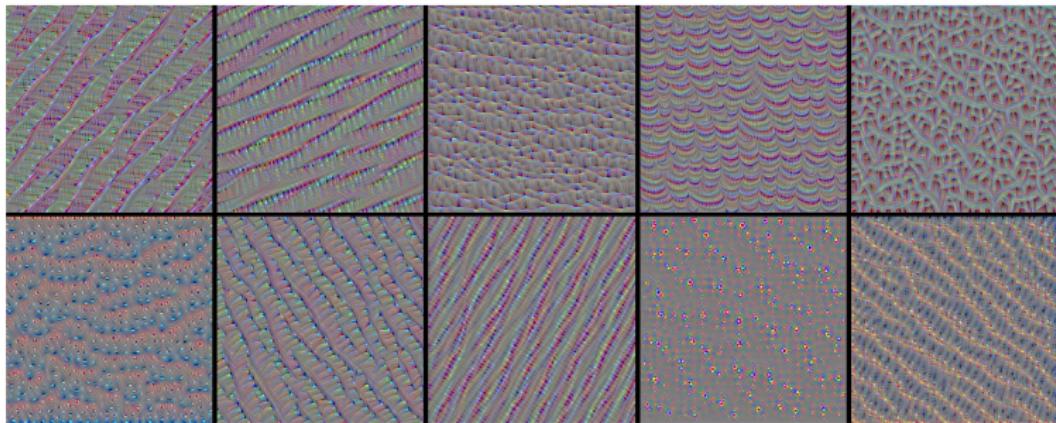
ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

Блок 2, второй свёрточный слой



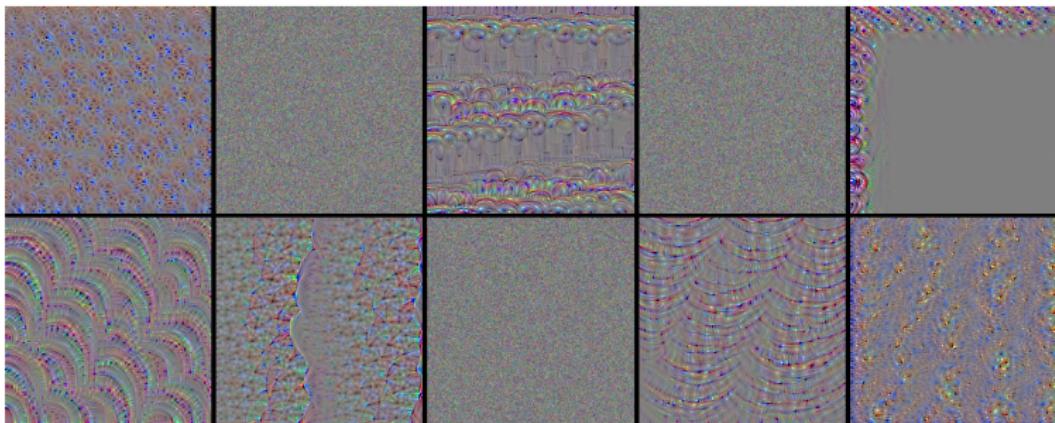
ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

Блок 3, третий свёрточный слой



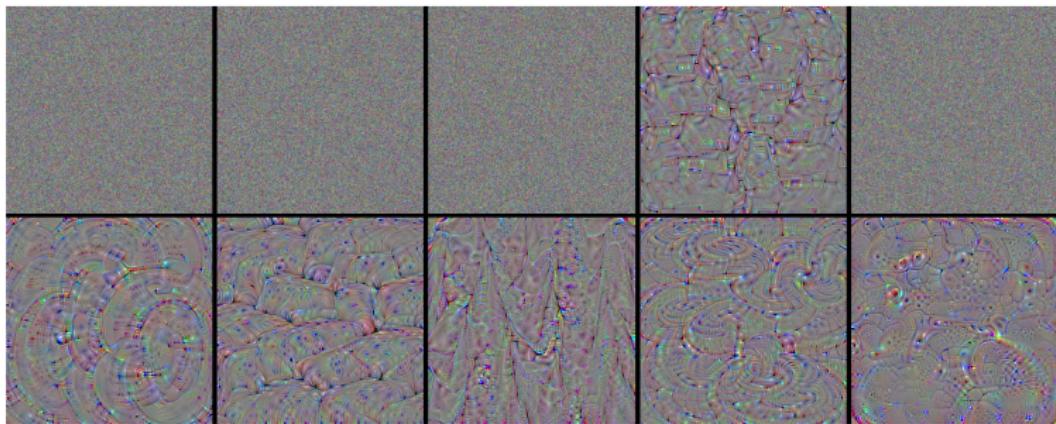
ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

Блок 4, третий свёрточный слой

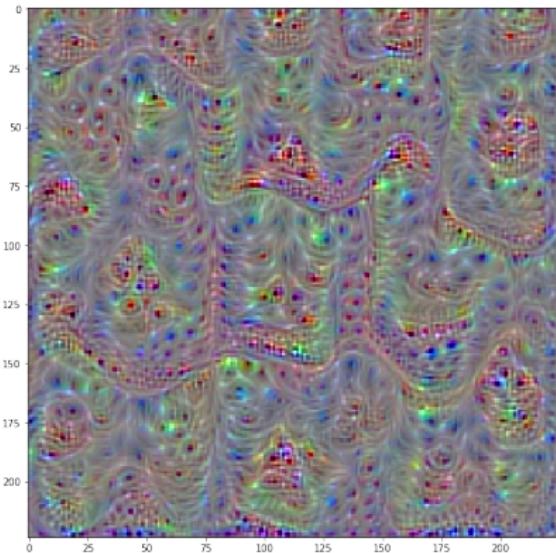


ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

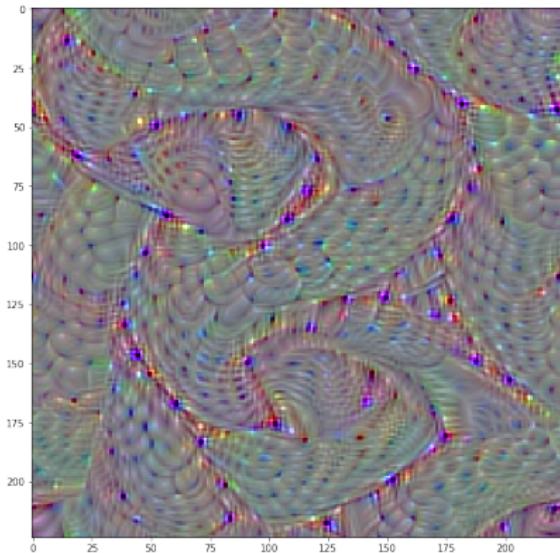
Блок 5, третий свёрточный слой



ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ



(c) Блок 5, свёртка 1, канал 66

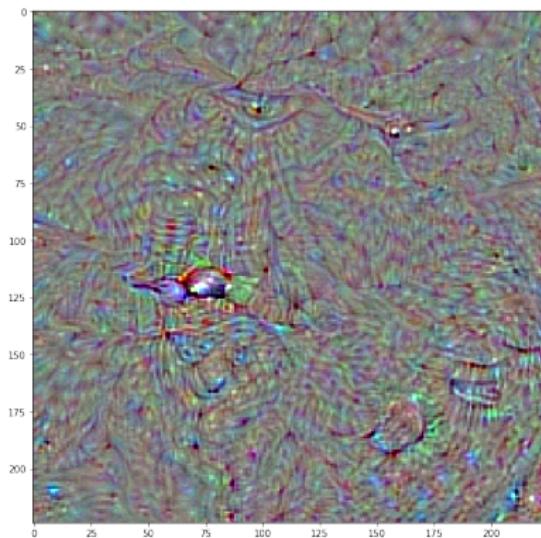


(d) Блок 5, свёртка 3, канал 66

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

Помимо каналов карты активации, мы можем визуализировать отдельные нейроны или целые классы.

Выведем визуализацию класса "дрозд"

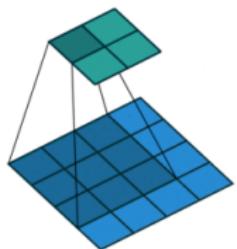


ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

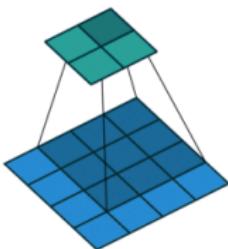
- Мы видим наглядное подтверждение нашей теории о том, что с глубиной увеличивается сложность выученных признаков
- Многие изображения зашумлены, а некоторые и вовсе состоят только из шума. С чем это связано? Как с этим бороться?

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

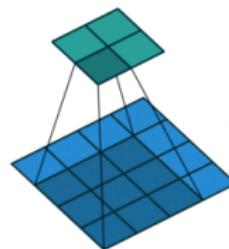
Рассмотрим простой пример свёртки:



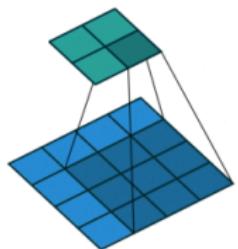
(e) шаг 1



(f) шаг 2



(g) шаг 3



(h) шаг 4

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

Легко угадать параметры:

- Ядро 3×3
- $stride = 1$
- $padding = 0$

Обозначим ядро:

$$K = \begin{pmatrix} k_{0,0} & k_{0,1} & k_{0,2} \\ k_{1,0} & k_{1,1} & k_{1,2} \\ k_{2,0} & k_{2,1} & k_{2,2} \end{pmatrix}$$

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

- Операция свёртки может быть представлена, как умножение матриц
- Для этого размотаем оригинал $I_{4 \times 4}$ (аналог операции Flatten в Keras), запишем как I_{flat}
- Запишем свёртку, как матрицу специального вида

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

$$C = \begin{pmatrix} k_{0,0} & k_{0,1} & k_{0,2} & 0 & k_{1,0} & k_{1,1} & k_{1,2} & 0 & k_{2,0} & k_{2,1} & k_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & k_{0,0} & k_{0,1} & k_{0,2} & 0 & k_{1,0} & k_{1,1} & k_{1,2} & 0 & k_{2,0} & k_{2,1} & k_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & k_{0,0} & k_{0,1} & k_{0,2} & 0 & k_{1,0} & k_{1,1} & k_{1,2} & 0 & k_{2,0} & k_{2,1} & k_{2,2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & k_{0,0} & k_{0,1} & k_{0,2} & 0 & k_{1,0} & k_{1,1} & k_{1,2} & 0 & k_{2,0} & k_{2,1} & k_{2,2} & 0 \end{pmatrix}$$

Теперь мы можем вычислить операцию свёртки следующим образом:

$$C \cdot I_{flat} = O_{flat}$$

Где O_{flat} является плоской версией карты активации

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

- Такое представление позволяет рассчитывать свёртки очень быстро
- Оно используется в современных фреймворках
- Особую актуальность такое представление приобретает при рассчёта backpropagation: необходимо домножить значение лежащего выше узла на C^T . Такая операция называется **transposed convolution** или **deconvolution**

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

- **transposed convolution** являются не только артефактом вычисления градиента, но и широко применяются в нейронных сетях, для увеличения пространственных размерностей. Это особенно актуально при синтезе изображений
- Градиент транспонированной свёртки вычисляется так же просто, при помощи домножения на $(C^T)^T = C$

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

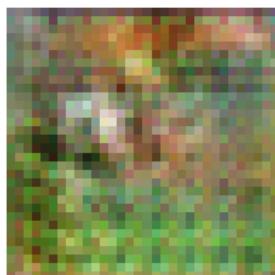
Изучая результаты работы разных алгоритмов, или, возможно, самостоятельно обучая модели, вы могли столкнуться так называемым эффектом шахматной доски (checkerboard artifacts)



(i) птица



(j) лодка



(k) олень

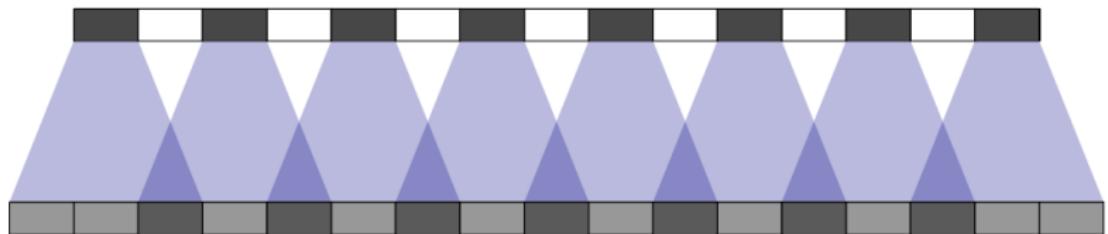


(l) самолёт

Откуда они берутся?

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

Odena, et al., "Deconvolution and Checkerboard Artifacts Distill, 2016.



Мы видим, что результаты применения каждой транспонированной свёртки налагаются. Отсюда и возникает эффект шахматной доски.

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

- Существует несколько способов избежать эффекта шахматной доски в изображениях, сгенерированных нейронной сетью (будет в следующих лекциях)
- Но какое отношение это имеет к нашей проблеме?

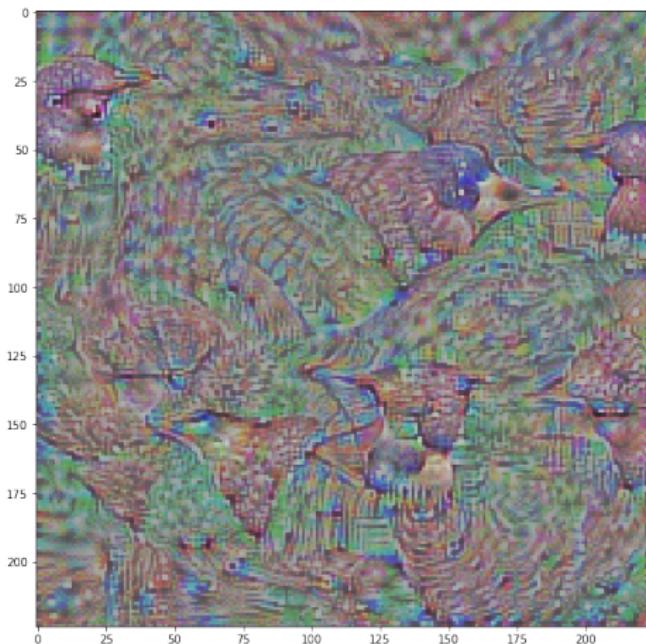
ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

- Существует несколько способов избежать эффекта шахматной доски в изображениях, сгенерированных нейронной сетью (будет в следующих лекциях)
- Но какое отношение это имеет к нашей проблеме?
- Поскольку градиент считается при помощи транспонированной свёртки, шумы которые в нём возникают, пораждают высокочастотный шум, который мы наблюдаем на наших визуализациях.

НАИВНАЯ РЕГУЛЯРИЗАЦИЯ

- Нам бы хотелось, чтобы значения пикселей отличались не очень сильно, не было больших перепадов, а как следствие и шума (спорное утверждение)
- Применим L_2 регуляризацию к изображению, или L более высоких порядков плоть до L_∞

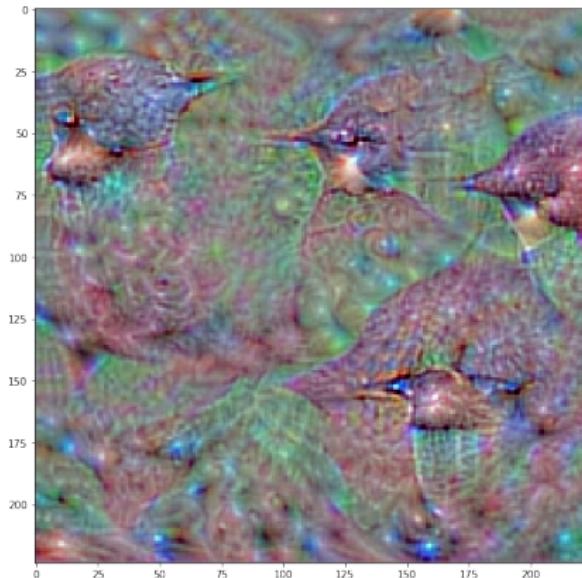
НАИВНАЯ РЕГУЛЯРИЗАЦИЯ



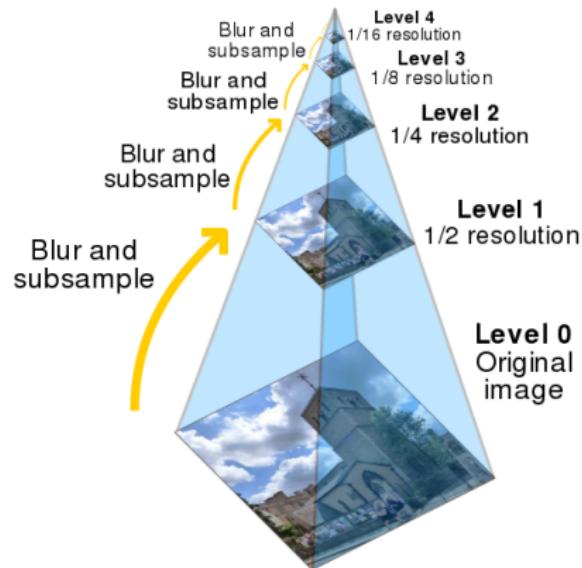
Дрозду стало лучше, но не настолько, насколько бы нам хотелось

НАИВНАЯ РЕГУЛЯРИЗАЦИЯ

- Чтобы избавиться от высокочастотного шума можно регулярно размывать изображение
- Применим Гауссовское размытие



ПИРАМИДЫ



ПИРАМИДЫ

Необходимо повысить низкие частоты, и понизить высокие.
Воспользуемся, классом методов на основе пирамид.

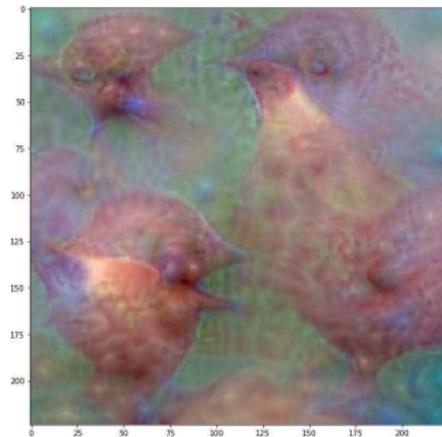
Пирамида лапласса:

- Применяем gaussian blur к изображению, затем вычитаем размытое изображение из исходного (получаем высокочастотные признаки)
- Затем размытое изображение уменьшается вдвое и процедура повторяется

ПИРАМИДЫ

Склейв пирамиду, мы повысим роль низких частот (в зависимости от грубины пирамиды).

Применим эту технику к градиенту



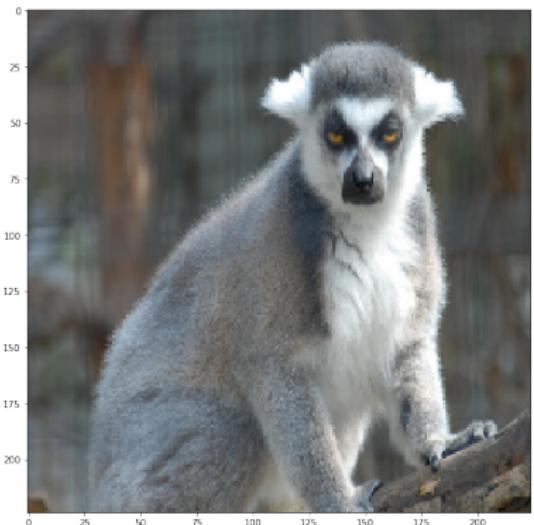
ADVERSARIAL EXAMPLES

ADVERSARIAL EXAMPLES

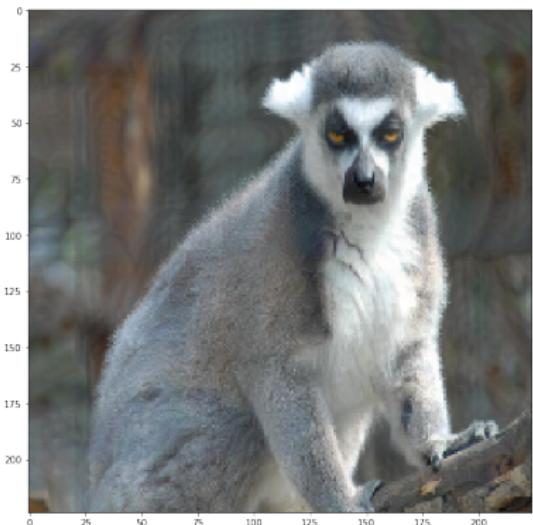
- Что будет если подать на вход изображение и попытаться максимизировать вероятность класса, к которому оно не принадлежит?
- Какое-то время визуально ничего не меняется, но как на это реагирует сеть?

ADVERSARIAL EXAMPLES

Не перепутайте!



(m) Лемур



(n) Буритто

ADVERSARIAL EXAMPLES

Будем искать adversarial пример \hat{x} , максимизирующий вероятность класса y , следующим образом:

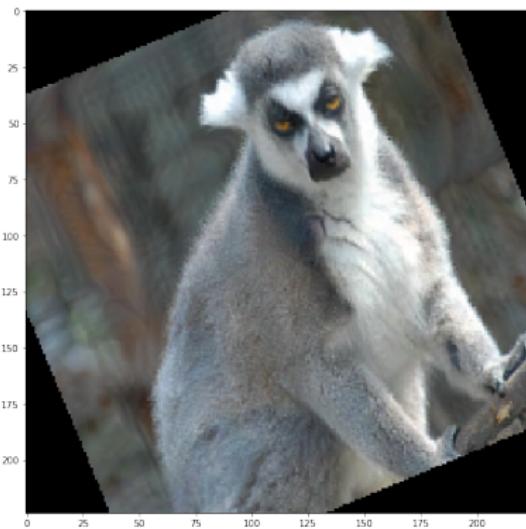
$$\log(P(y|\hat{x})) \rightarrow \max$$

$$\|x - \hat{x}\|_\infty \leq \epsilon$$

Где ϵ некоторое заданное небольшое число.

ADVERSARIAL EXAMPLES

Всё ещё лемур



ADVERSARIAL EXAMPLES

Athalye, Anish, and Ilya Sutskever. "Synthesizing robust adversarial examples." arXiv preprint arXiv:1707.07397 (2017).



■ classified as turtle

■ classified as rifle

■ classified as other

ADVERSARIAL EXAMPLES

Вводится понятие ожидания по трансформации (expectation over transformation):

$$\delta = \mathbb{E}_t[d(t(x) - t(\hat{x}))]$$

Где T – это распределение всех возможных функций трансформации.

Теперь задачу можно записать следующим образом:

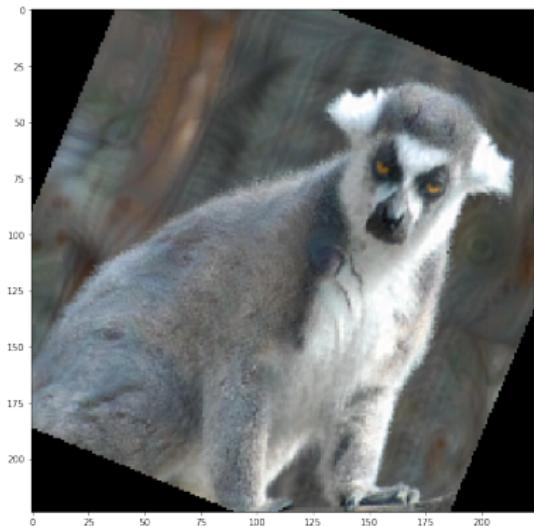
$$\mathbb{E}_t[\log(P(y|\hat{x}))] \rightarrow \max$$

$$\mathbb{E}_t[d(t(x) - t(\hat{x}))] \leq \epsilon$$

ADVERSARIAL EXAMPLES



(o) Буритто



(p) Буритто

STYLE TRANSFER

STYLE TRANSFER

У нас есть предобученная сверточная сеть и изображение.

Пропустив изображение через сеть, мы получим представление изображения в виде тензоров признаков.

Давайте найдем изображение, которое имеет аналогичные тензоры признаков и "выглядит естественно"

Решим задачу:

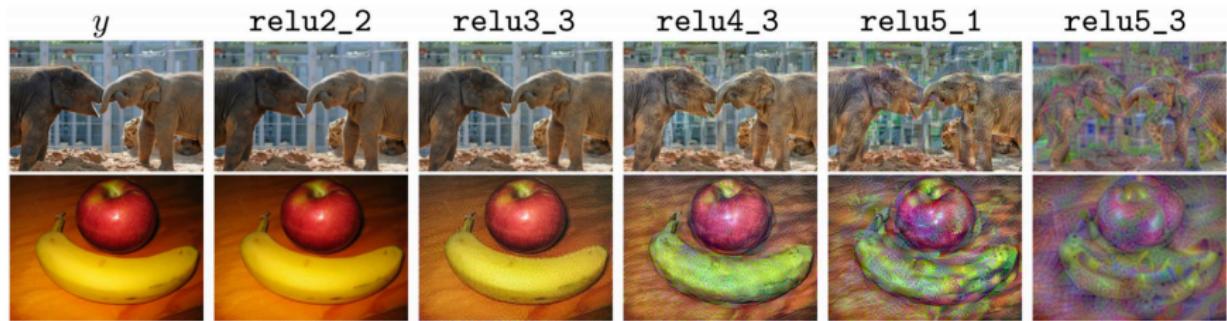
$$x^* =_{x \in R^{H \times W \times C}} L(\Phi(x), \Phi_0) + \lambda R(x)$$

$$L(\Phi(x), \Phi_0) = \|\Phi(x) - \Phi_0\|_2$$

$$R(x) = \sum_{i,j} ((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2)^{\beta/2}$$

STYLE TRANSFER

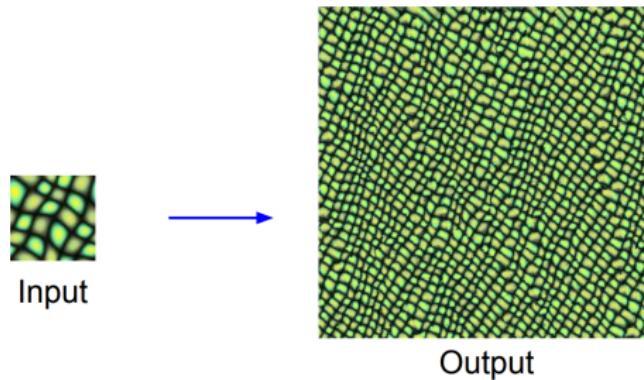
Для VGG-16 пример изображений будет выглядеть так:



Johnson, Alahi, and Fei-Fei "Perceptual Losses for Real-Time Style Transfer and Super-Resolution" ECCV 2016

STYLE TRANSFER

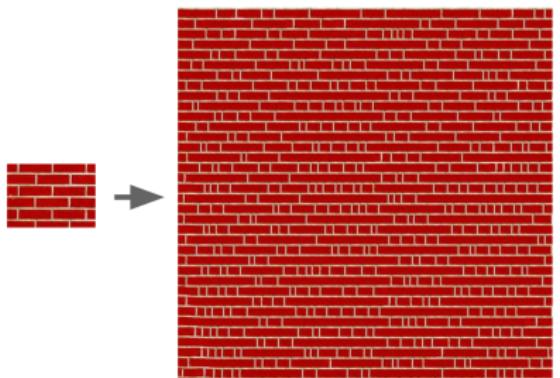
Решим немного другую задачу. Будем генерировать текстуру заданного размера из маленького примера.



Как это сделать?

STYLE TRANSFER

Простой подход не сработает



При этом мы можем заметить, что изображение получилось не очень хорошо. И это неудивительно, так как мы просто перенесли красный цвет с одного изображения на другое. Для этого нам нужно использовать специальные алгоритмы, которые могут извлечь стиль из одного изображения и применять его к другому. Одним из таких алгоритмов является GAN (Generative Adversarial Network). В GAN есть две сети: генератор и дискриминатор. Генератор создает изображение, а дискриминатор определяет, является ли оно реальным или фальшивым. Алгоритм GAN обучается так, чтобы генератор создавал изображения, которые были бы почти идентичны изображениям из стиля, но при этом выглядели как новые изображения. Таким образом, GAN может перенести стиль из одного изображения на другое, создавая при этом новые и интересные изображения.

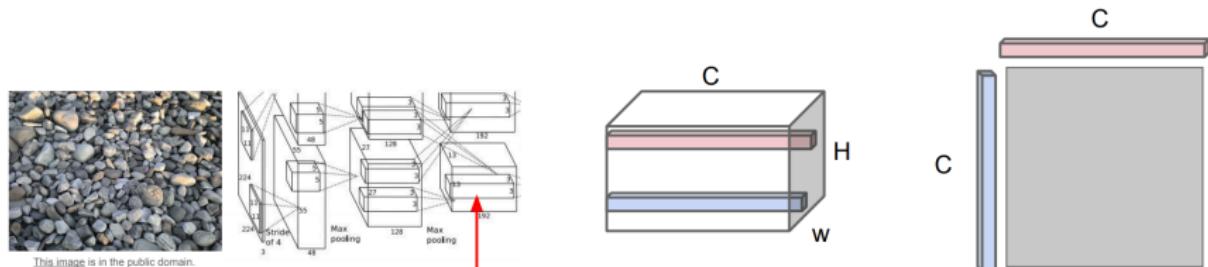
STYLE TRANSFER

Вспомним что такое матрица Грамма

$$G(x_1, \dots, x_n) = \begin{vmatrix} \langle x_1, x_1 \rangle & \langle x_1, x_2 \rangle & \dots & \langle x_1, x_n \rangle \\ \langle x_2, x_1 \rangle & \langle x_2, x_2 \rangle & \dots & \langle x_2, x_n \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x_n, x_1 \rangle & \langle x_n, x_2 \rangle & \dots & \langle x_n, x_n \rangle \end{vmatrix}.$$

STYLE TRANSFER

Будем считать матрицу Грамма для каждой пары признаков

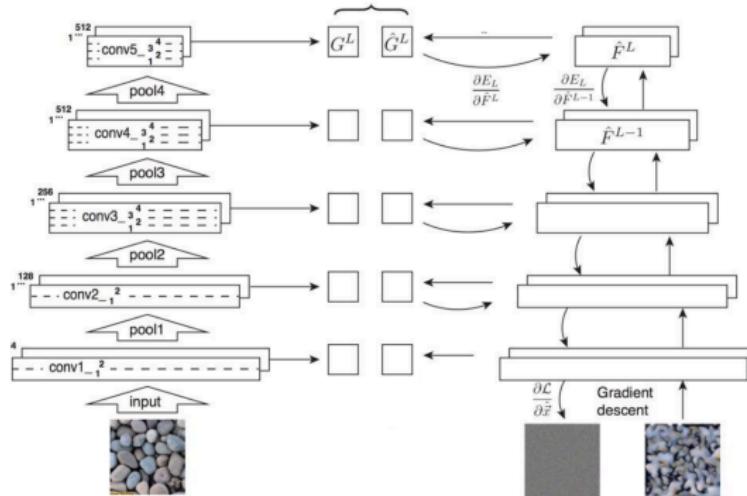


Конечная матрица это среднее по всем полученным матрицам Грамма

Эффективно выполнить расчет можно с помощью небольшого трюка:

- Reshape $F \in R^{CxHxW} \rightarrow F \in R^{CxHW}$
- $G = FF^T$

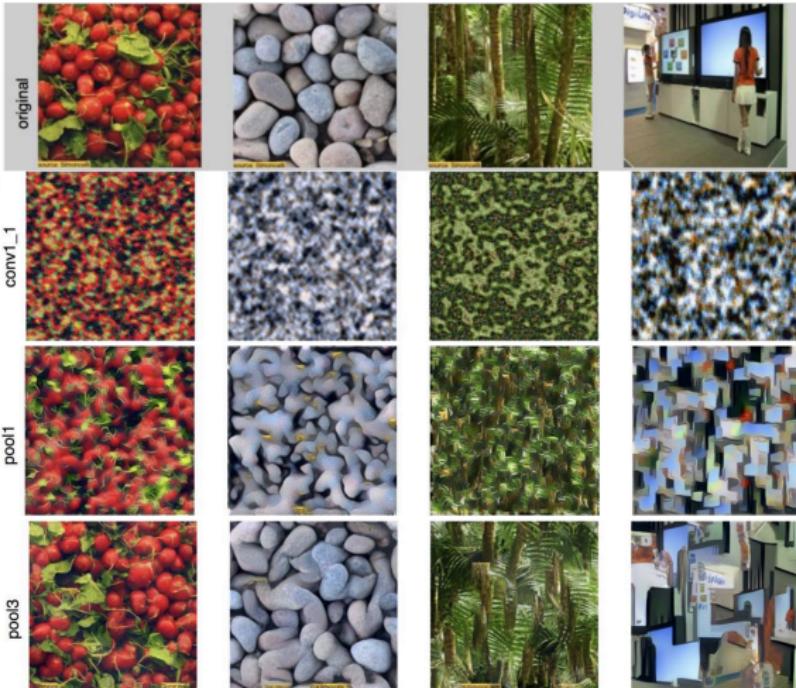
STYLE TRANSFER



$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - \hat{G}_{ij}^l \right)^2 \quad \mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^L w_l E_l$$

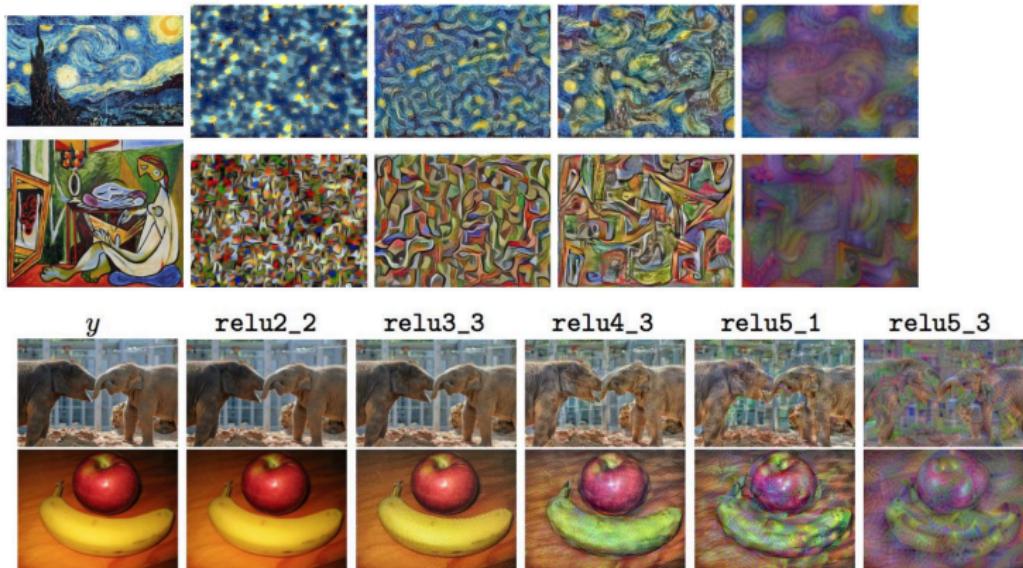
Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015

STYLE TRANSFER

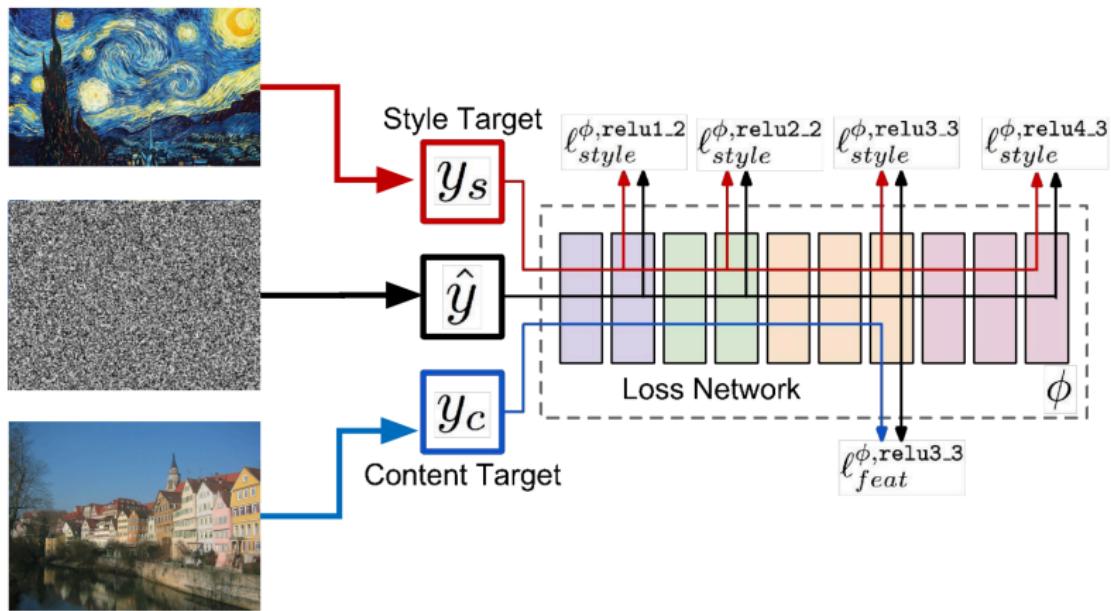


STYLE TRANSFER

Объединим идеи создания похожего на оригинал изображения с одинковой картой активаций и генерацию текстур по заданной текстуре.

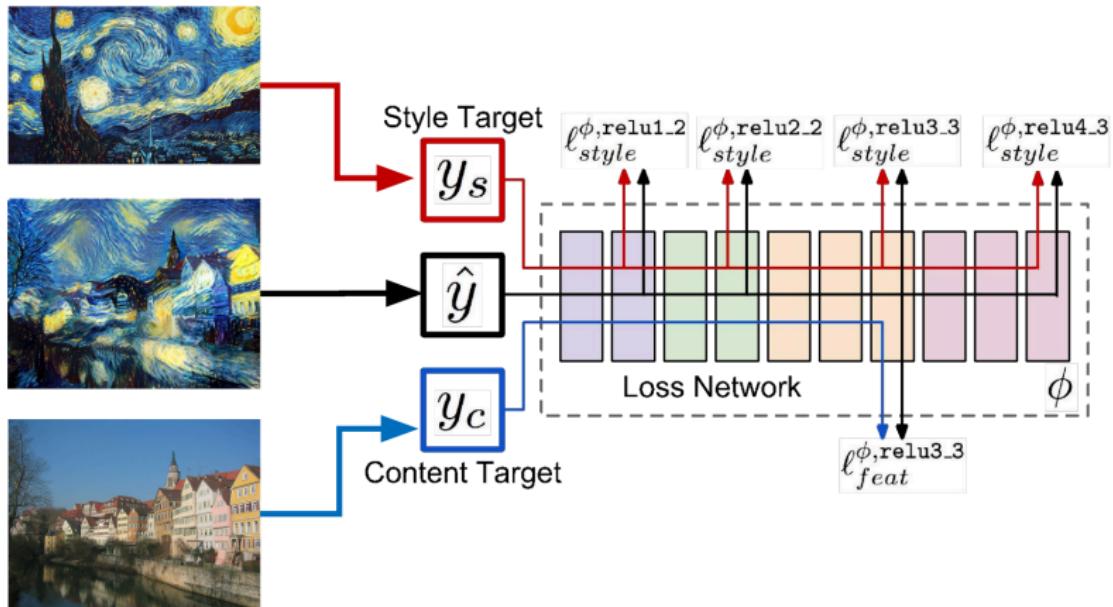


STYLE TRANSFER



Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016

STYLE TRANSFER



Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016

STYLE TRANSFER



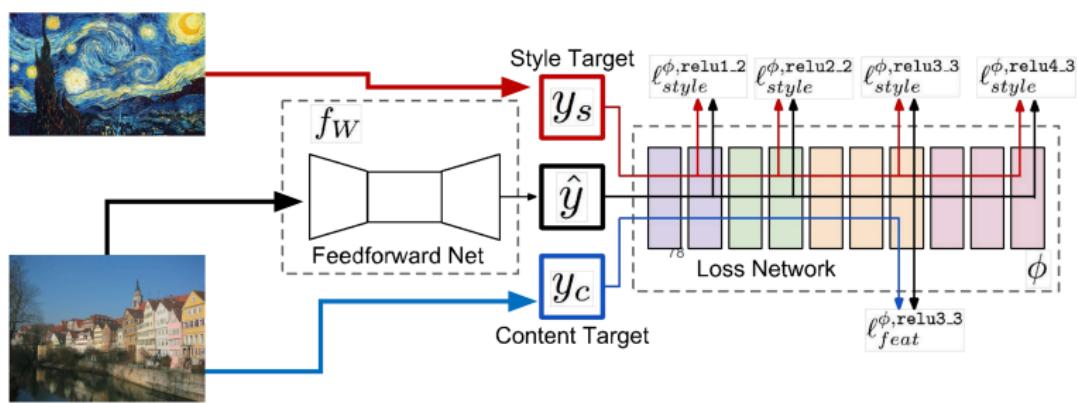
Отдавая предпочтение задаче "создание похожей картинки с такими же активациями" или задаче "создание картинки у которой матрица грамма признаков похожа на образец можно варьировать "силу" накладываемого стиля.

STYLE TRANSFER

Как сделать Style Transfer быстрее?

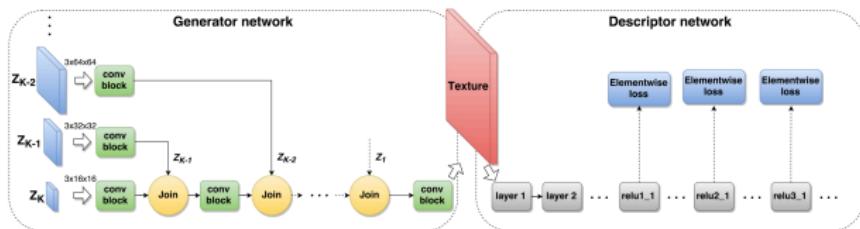
STYLE TRANSFER

Вместо того чтобы менять картинку, будем менять сверточную сеть, которая обрабатывает картинку



STYLE TRANSFER

Пример архитектуры:



"We found that training benefited significantly from inserting batch normalization layers right after each convolutional layer and, most importantly, right before the concatenation layers, since this balances gradients travelling along different branches of the network."

Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, Victor Lempitsky.
Texture Networks: Feed-forward Synthesis of Textures and Stylized Images

STYLE TRANSFER

Какие могут быть проблемы с Batch Normalization?

Как их решать?

STYLE TRANSFER

Batch normalization:

$$y_{tijk} = \frac{x_{tijk} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}, \quad \mu_i = \frac{1}{HWT} \sum_{t=1}^T \sum_{l=1}^W \sum_{m=1}^H x_{tilm}, \quad \sigma_i^2 = \frac{1}{HWT} \sum_{t=1}^T \sum_{l=1}^W \sum_{m=1}^H (x_{tilm} - \mu_i)^2.$$

...

STYLE TRANSFER

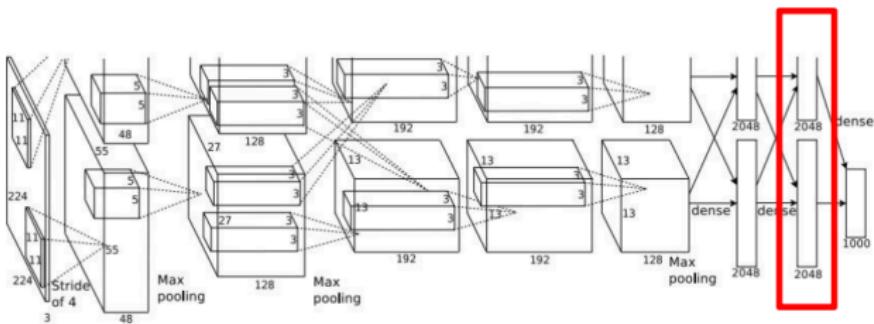
Batch normalization:

$$y_{tijk} = \frac{x_{tijk} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}, \quad \mu_i = \frac{1}{HWT} \sum_{t=1}^T \sum_{l=1}^W \sum_{m=1}^H x_{tilm}, \quad \sigma_i^2 = \frac{1}{HWT} \sum_{t=1}^T \sum_{l=1}^W \sum_{m=1}^H (x_{tilm} - mu_i)^2.$$

Instance Normalization:

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}, \quad \mu_{ti} = \frac{1}{HW} \sum_{l=1}^W \sum_{m=1}^H x_{tilm}, \quad \sigma_{ti}^2 = \frac{1}{HW} \sum_{l=1}^W \sum_{m=1}^H (x_{tilm} - mu_{ti})^2.$$

IMAGE EMBEDDINGS



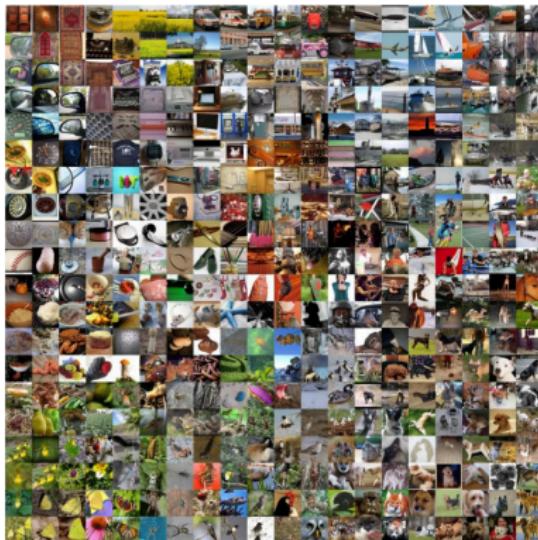
Для анализа изображений можно использовать вектор с последних слоев сети. Например для AlexNet можно взять вектор с предпоследнего слоя, его размерность 4096

IMAGE EMBEDDINGS



В пространстве этих векторов можно искать похожие картинки
Krizhevsky et al, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012.

IMAGE EMBEDDINGS



А можно понизить размерность с помощью t-SNE или других методов и визуализировать полученное облако.

<http://cs.stanford.edu/people/karpathy/cnnembed/>