

CNN - ПРАКТИКА

КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ

Рассмотрим задачу классификации изображений.



КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ

Каждое изображение представлено тремя матрицами, по одной на каждый цветовой канал (RGB):



(a) Red

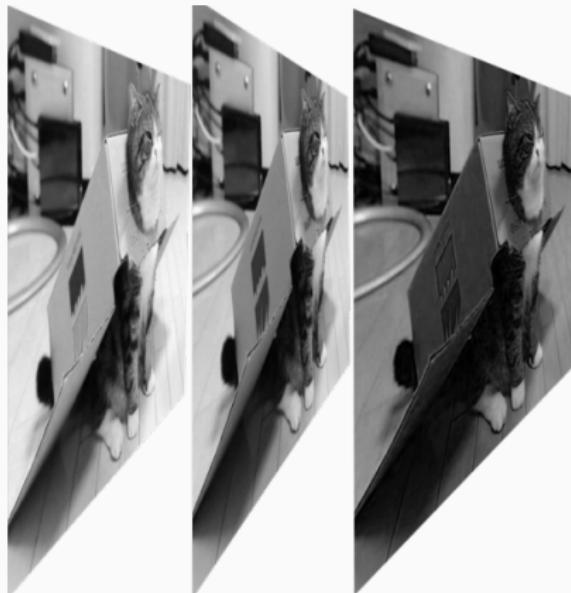
(b) Green

(c) Blue

Элементы матриц принимают значения от 0 до 255 и обозначают интенсивность одного из трёх цветов.

КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ

Для удобства мы будем хранить все три матрицы вместе в одном трёхмерном массиве, где третий индекс отвечает за номер канала. Такая структура является частным случаем **тензора**.



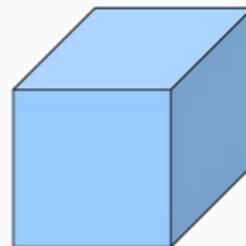
КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ



1d-tensor



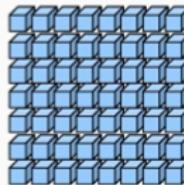
2d-tensor



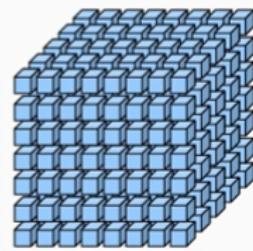
3d-tensor



4d-tensor



5d-tensor

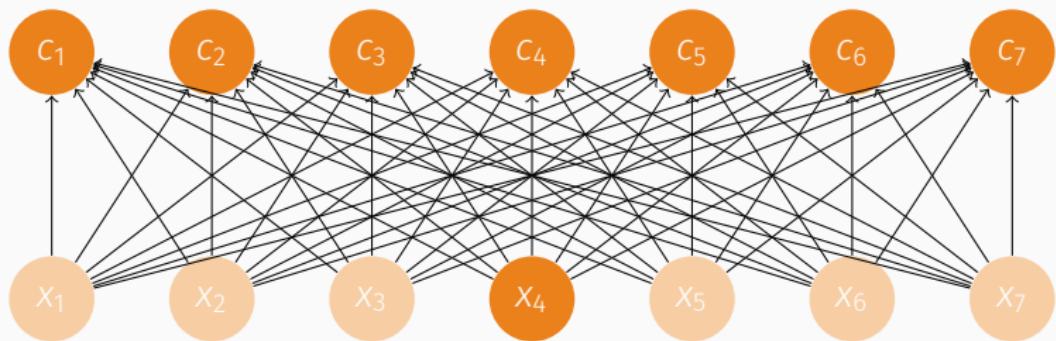


6d-tensor

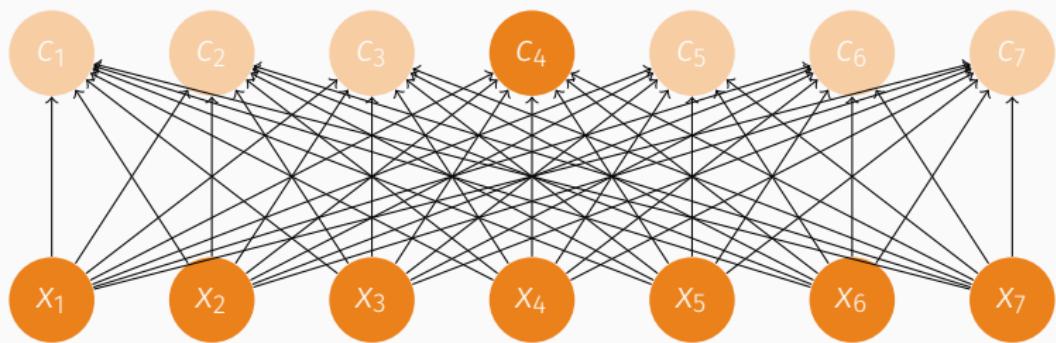
КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ

- Как нам работать с такими данными?
- Будет ли эффективна полносвязная сеть?
- Почему?

ПОЛНОСВЯЗНЫЙ СЛОЙ



ПОЛНОСВЯЗНЫЙ СЛОЙ



КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ

- Каждый вход влияет на каждый узел слоя и наоборот
- Зависит от положения объектов в кадре
- Большое количество параметров. Например для изображения 800×600 только одно измерение матрицы весов составит 480000

СВЁРТКИ

$$(f * K)(x) = \sum_{\tau} f(\tau)K(x - \tau) = \sum_{\tau} f(x - \tau)K(\tau)$$

Где f называется *оригиналом*, а K **ядром** свёртки. Можно сказать, что ядро присваивает вес каждому значению функции.

СВЁРТКИ

Пример:

- пусть $f(i)$ возвращает значения функции потерь на i -м батче при тренировке нейронной сети
- Из-за шума график из таких значений выглядит не очень красиво
- Необходимо применить сглаживание

СВЁРТКИ

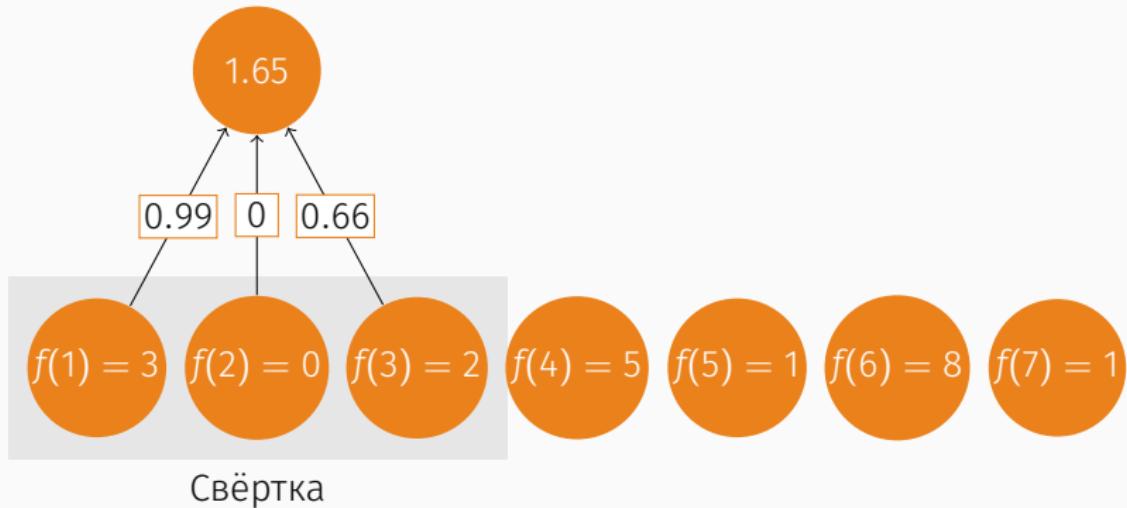
Пример:

- Ядро $K(i)$ возвращает значения 0.33 при значениях $i \in \{-1, 0, 1\}$
- И 0 во всех остальных случаях

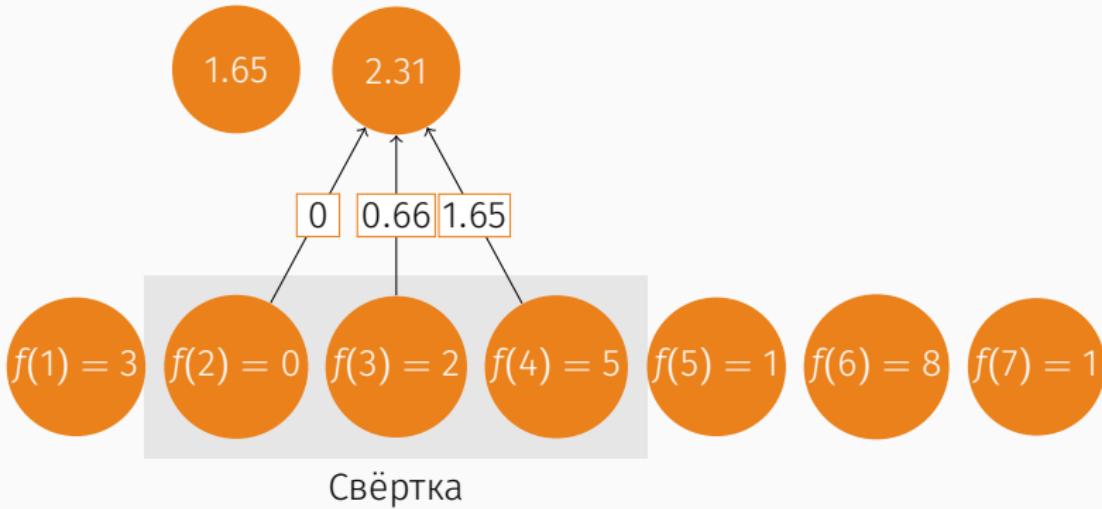
Таким образом можно представить операцию свёртки, как скользящее окно длины 3, на значениях функции оригинала f .

Визуализируем это:

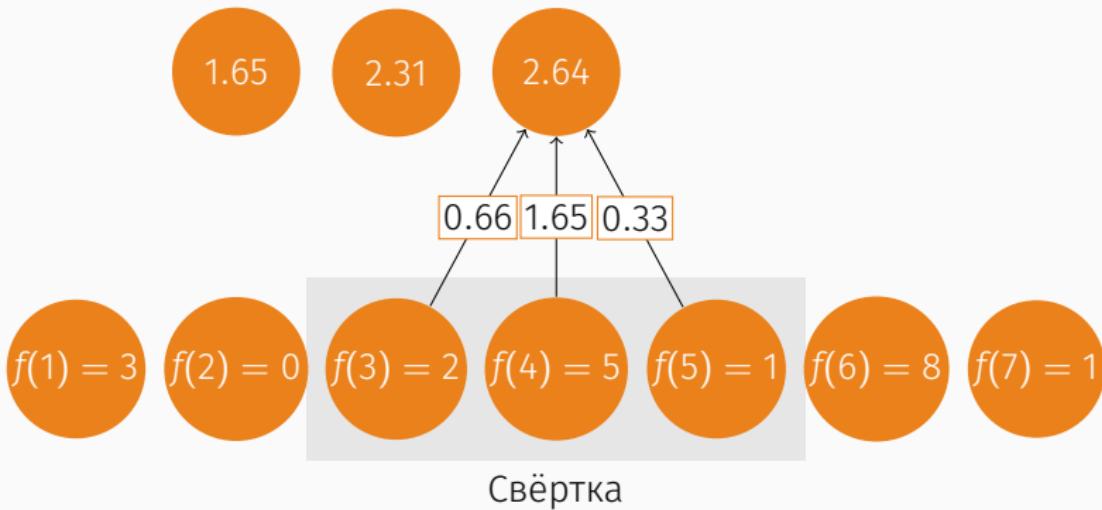
СВЁРТКИ



СВЁРТКИ



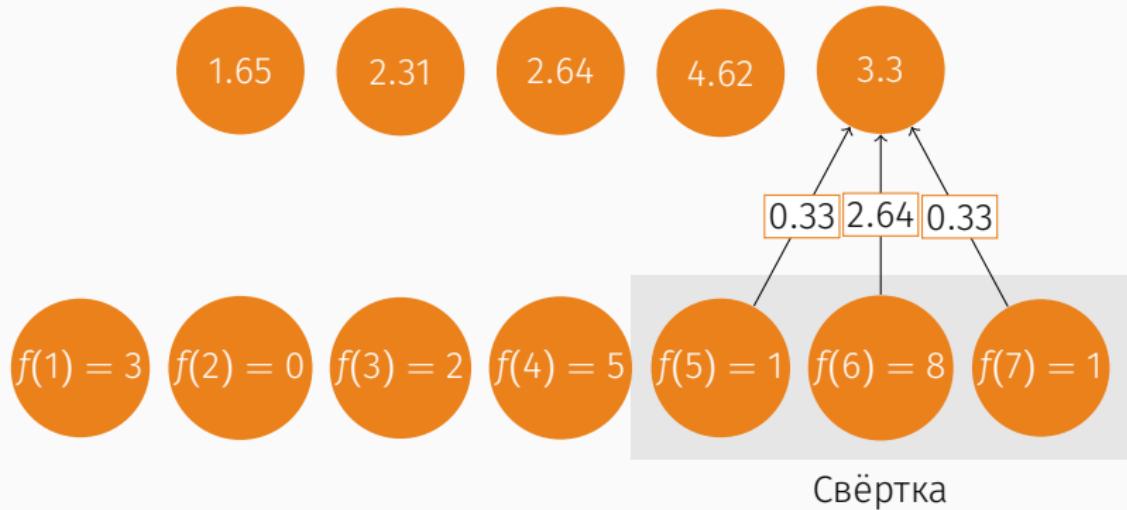
СВЁРТКИ



СВЁРТКИ



СВЁРТКИ



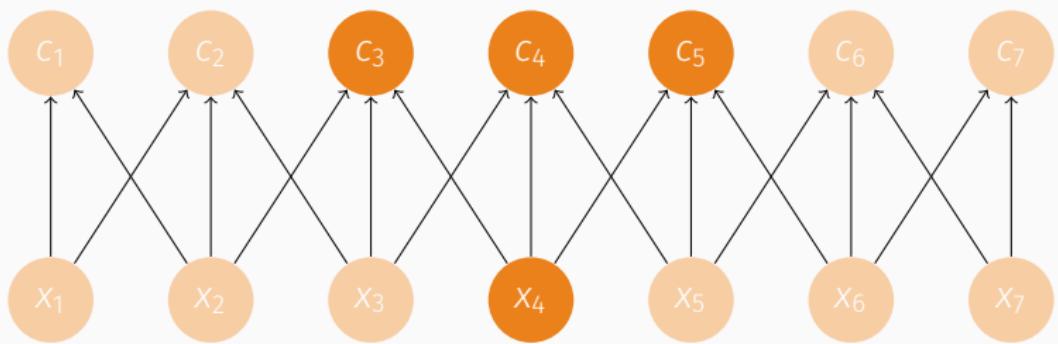
СВЁРТКИ

- Ядро в данном случае задаёт обычную линейную комбинацию и полностью определено вектором $(0.33, 0.33, 0.33)$
- Среди значений свёртки не хватает двух элементов. Это вызвано тем, что окно не выходит за пределы имеющегося ряда. Такой тип свёрток обычно называют **valid convolution**
- Так же часто используются **same convolution**. В этом случае "недостающие" элементы заполняются нулями, и результирующий ряд имеет такое же количество элементов.
- В данном примере использовался единичный шаг окна, однако этот параметр может меняться. Шаг обычно обозначается **stride**.

СВЁРТКИ

Так зачем это нужно? Чем это лучше полносвязного слоя?

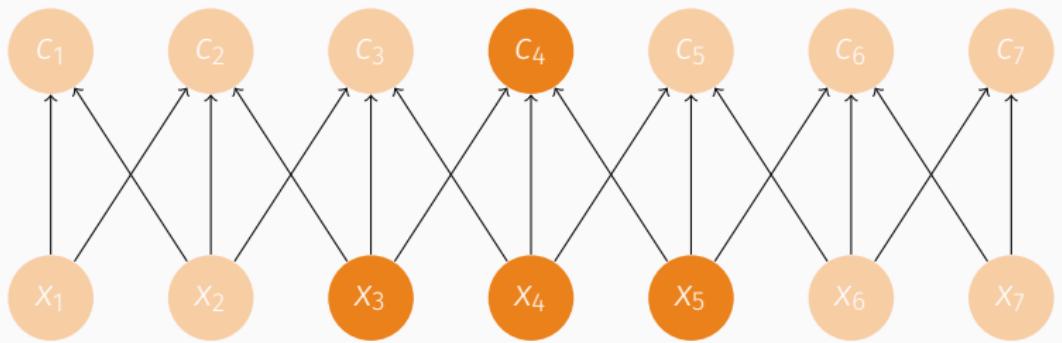
ОБЩИЕ ВЕСА



ОБЩИЕ ВЕСА

- Каждый вход влияет только на три узла (зависит от размера ядра свёртки)
- Притом веса используются повторно. Например, вес "соединяющий" x_4 и x_5 равен весу связи x_6 и x_7 .
- Свёртки действуют локально

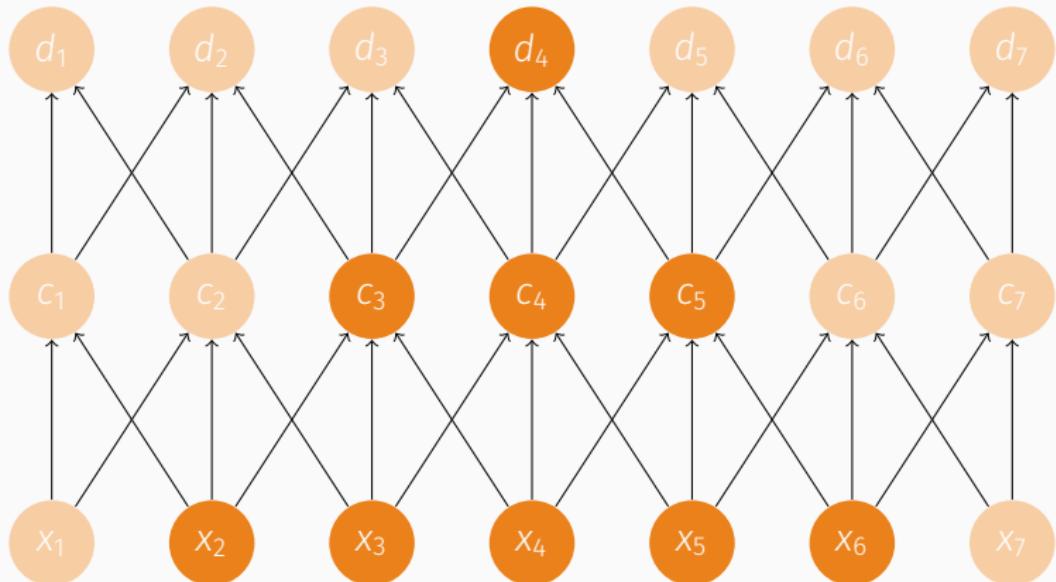
ЗОНА ВИДИМОСТИ



ЗОНА ВИДИМОСТИ

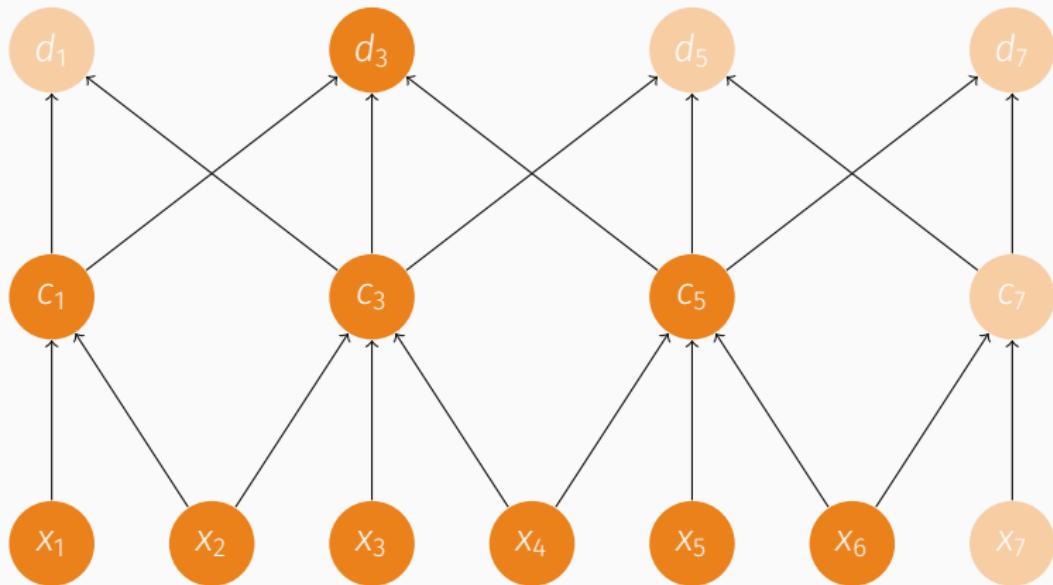
- Каждый узел изменяется под действием только трёх входов (зависит от размера ядра свёртки)
- Входы x_3, x_4, x_5 называются **зоной видимости или рецептивным полем** (receptive field) узла c_4
- Зона видимости зависит от глубины

ЗОНА ВИДИМОСТИ



ЗОНА ВИДИМОСТИ

Зона видимости так же зависит от размера шага. Чем больше шаг, тем больше зона.



СВЁРТКИ

Свёртки легко обобщить на двухмерный (n -мерный) случай:

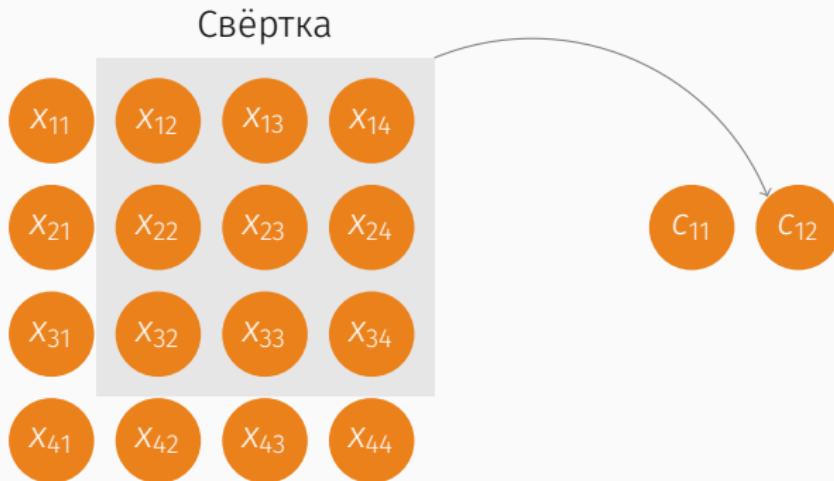
$$(f * K)(x, y) = \sum_{\tau, \eta} f(\tau, \eta)K(x - \tau, y - \eta) = \sum_{\tau, \eta} f(x - \tau, y - \eta)K(\tau, \eta)$$

Теперь ядро свёртки задаётся матрицей (n -мерным тензором).
Рассмотрим пример.

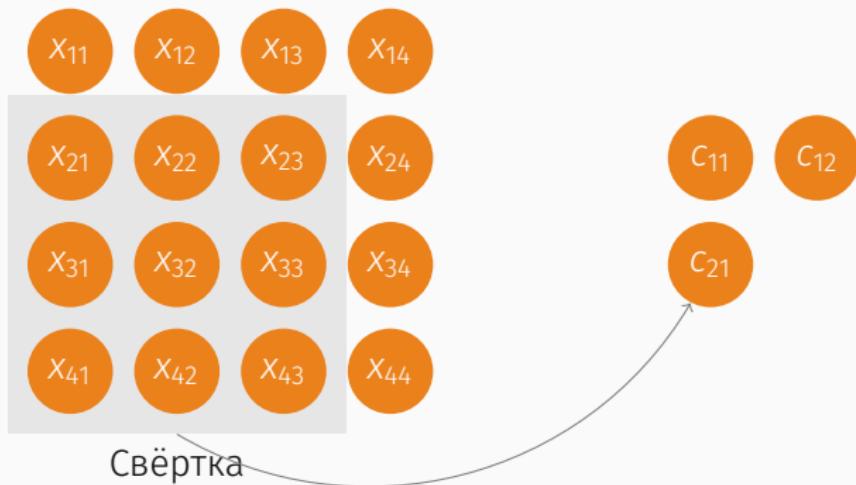
СВЁРТКИ



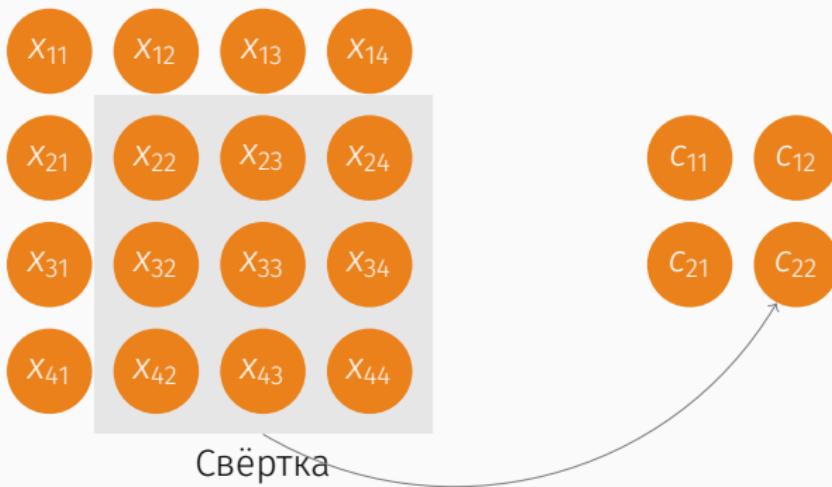
СВЁРТКИ



СВЁРТКИ



СВЁРТКИ



СВЁРТКИ

Как рассчитать размер результирующего слоя?

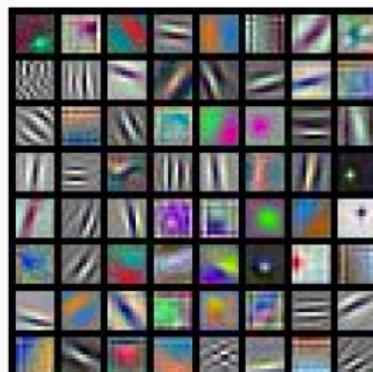
$$width_{result} = (width_{input} - width_{kernel} + 2padding)/stride + 1$$

$$height_{result} = (height_{input} - height_{kernel} + 2padding)/stride + 1$$

ВИЗУАЛИЗАЦИЯ

ВИЗУАЛИЗАЦИЯ

Визуализация фильтров 1 слоя



AlexNet:
 $64 \times 3 \times 11 \times 11$



ResNet-18:
 $64 \times 3 \times 7 \times 7$



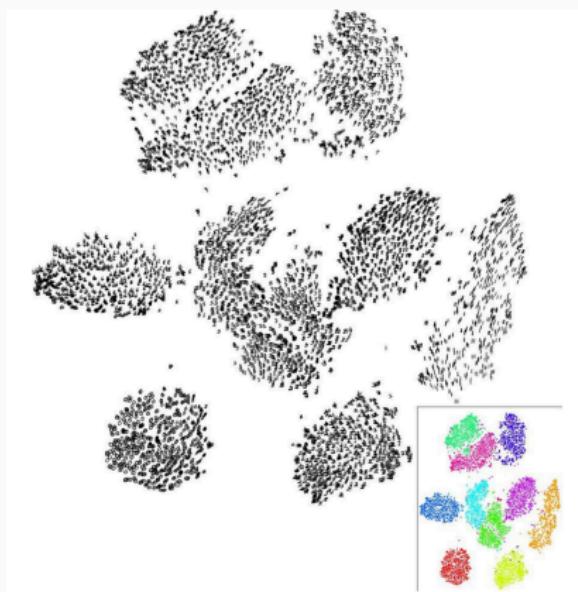
ResNet-101:
 $64 \times 3 \times 7 \times 7$



DenseNet-121:
 $64 \times 3 \times 7 \times 7$

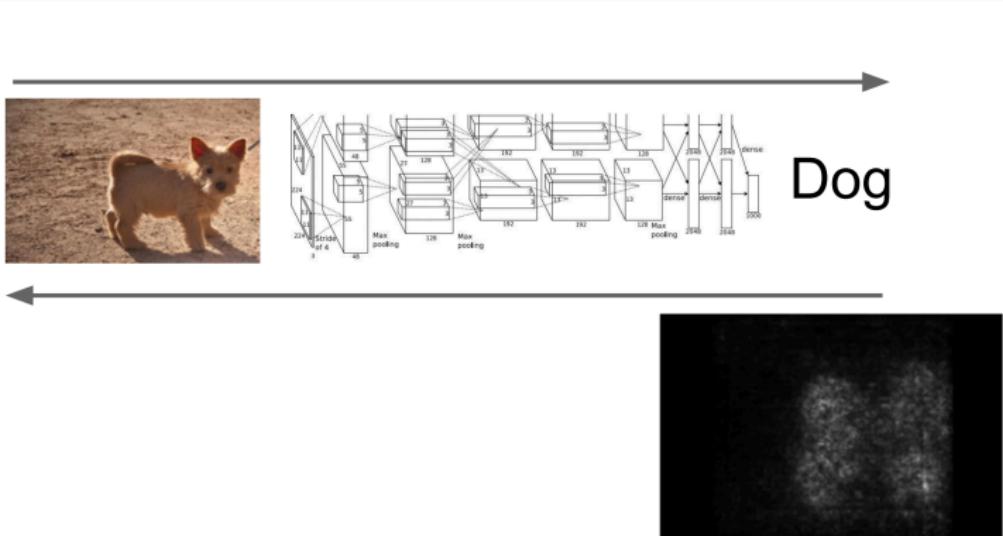
ВИЗУАЛИЗАЦИЯ

Визуализация извлеченных признаков



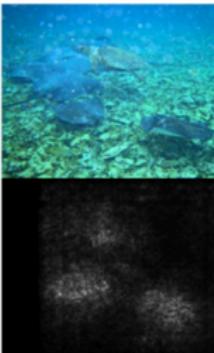
ВИЗУАЛИЗАЦИЯ

Saliency maps



ВИЗУАЛИЗАЦИЯ

Saliency maps



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:

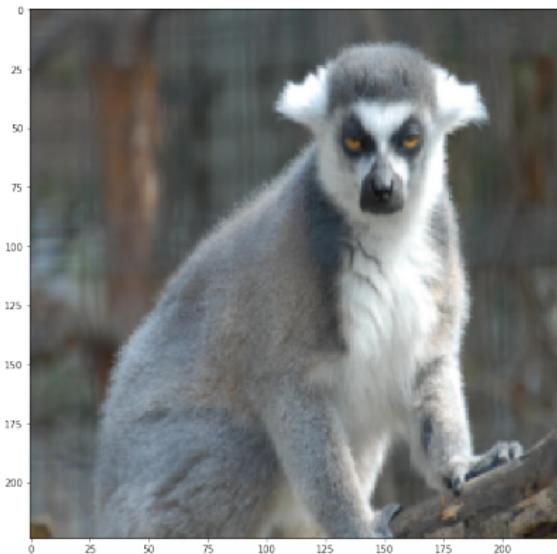
- Большие величины достигаются в точках, небольшое изменение которых приводит к большому изменению функции
- Градиент близок к 0 в точках, значение которых слабо влияет на результат

Следовательно значимость пикселя пропорциональна величине градиента.

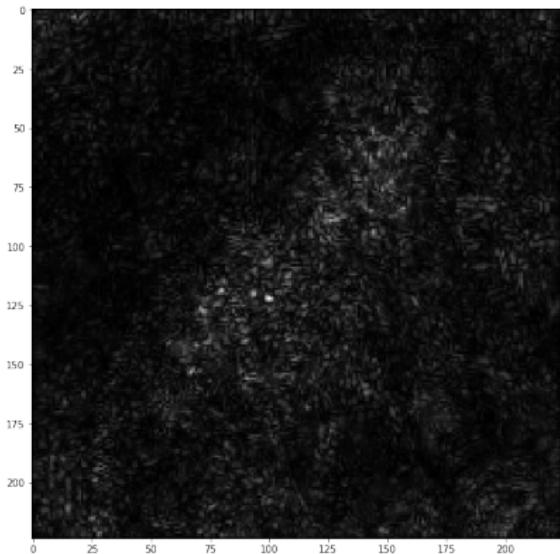
ОПРЕДЕЛЕНИЕ ЗНАЧИМОЙ ОБЛАСТИ

- 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

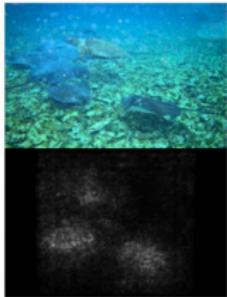


(a) Исходное изображение



(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

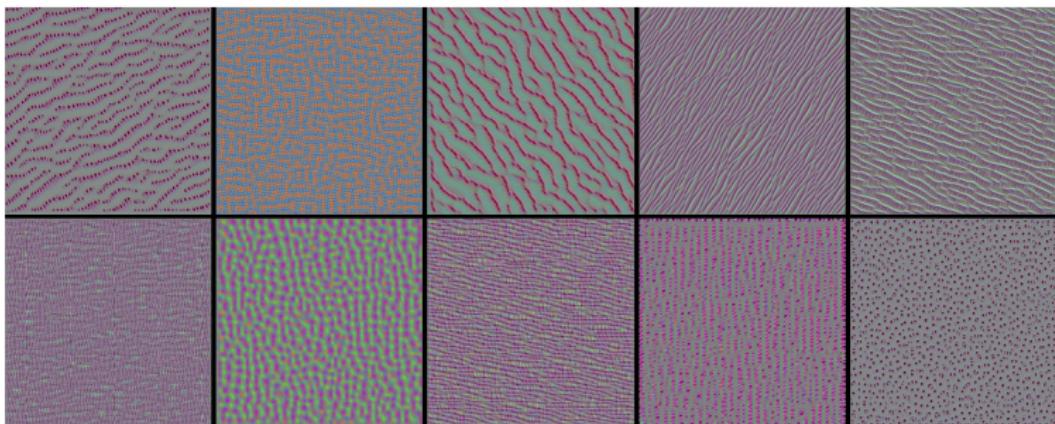
Идея: Подадим на вход шум и будем изменять его так, чтобы максимизировать значения выбранных активаций

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

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

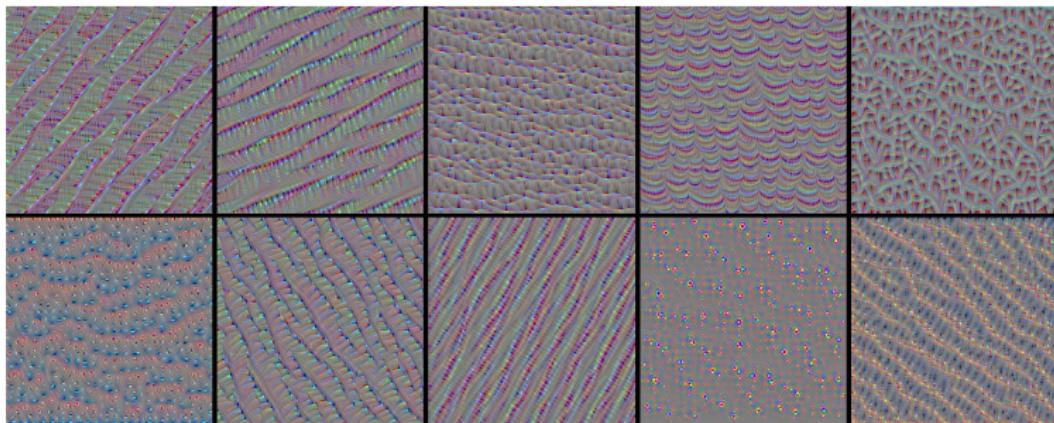
ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

Блок 2, второй свёрточный слой



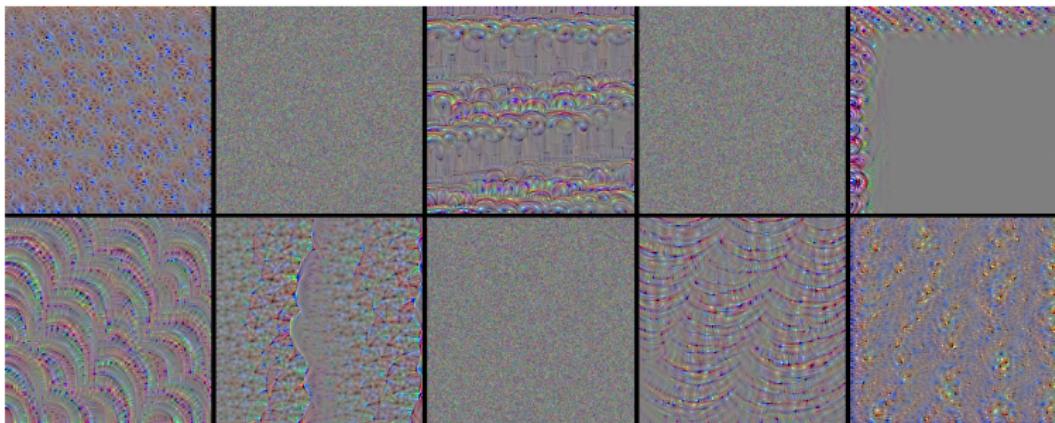
ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

Блок 3, третий свёрточный слой



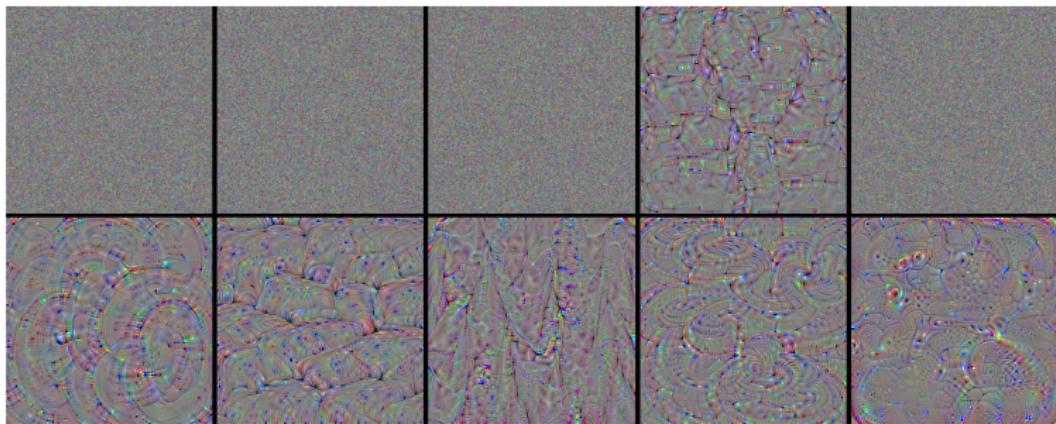
ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

Блок 4, третий свёрточный слой

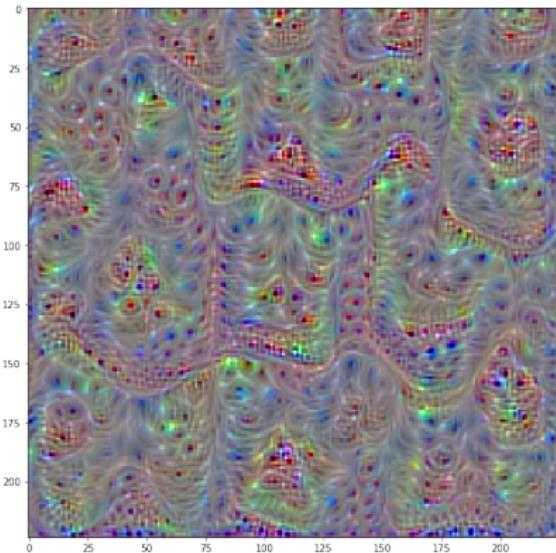


ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

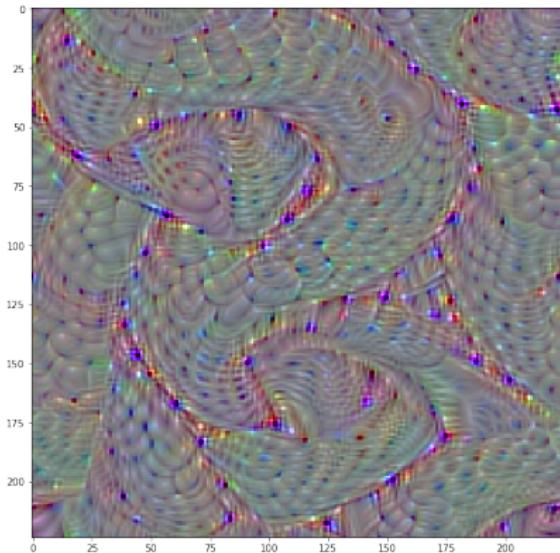
Блок 5, третий свёрточный слой



ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ



(c) Блок 5, свёртка 1, канал 66

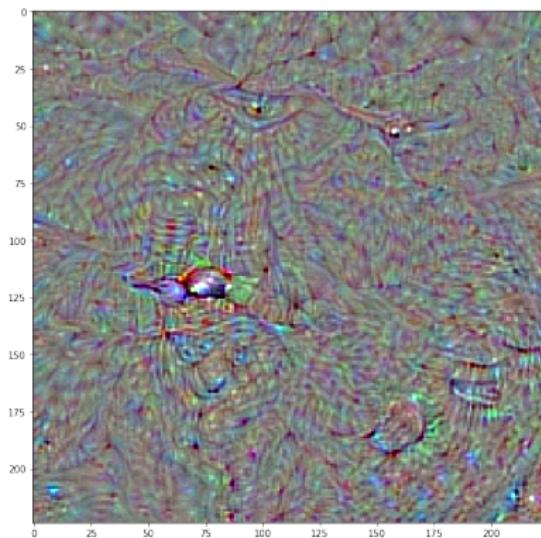


(d) Блок 5, свёртка 3, канал 66

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

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

Выведем визуализацию класса "дрозд"

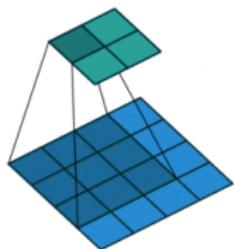


ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

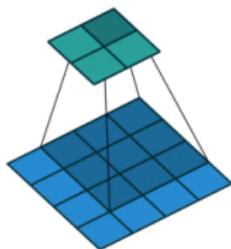
- Мы видим наглядное подтверждение нашей теории о том, что с глубиной увеличивается сложность выученных признаков
- Многие изображения зашумлены, а некоторые и вовсе состоят только из шума. С чем это связано? Как с этим бороться?

ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

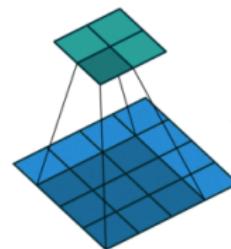
Рассмотрим простой пример свёртки:



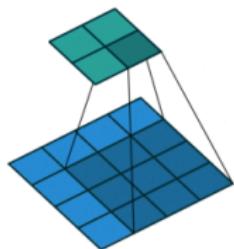
(e) шаг 1



(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

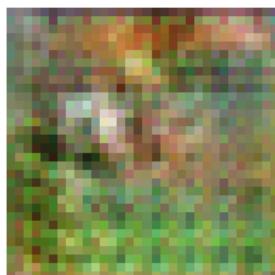
Изучая результаты работы разных алгоритмов, или, возможно, самостоятельно обучая модели, вы могли столкнуться так называемым эффектом шахматной доски (checkerboard artifacts)



(i) птица



(j) лодка



(k) олень

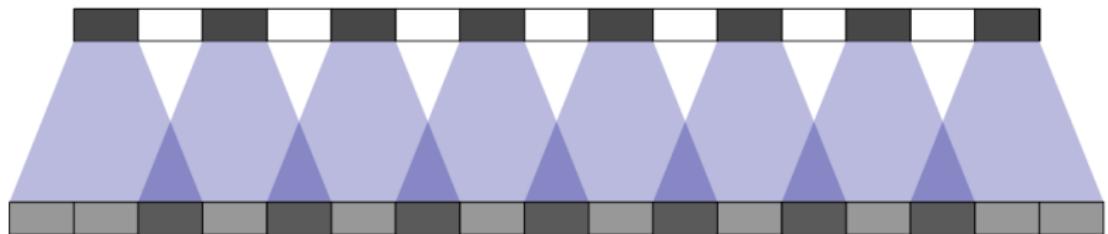


(l) самолёт

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?

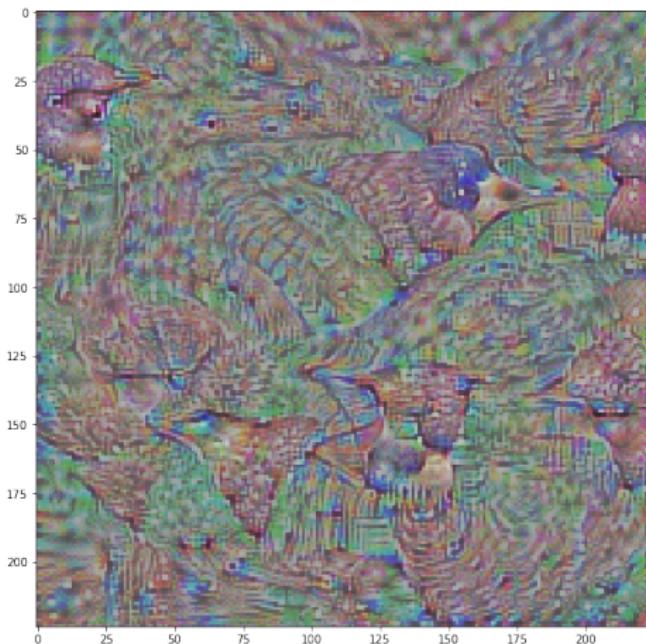
ВИЗУАЛИЗАЦИЯ ПРИЗНАКОВ

- 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_∞

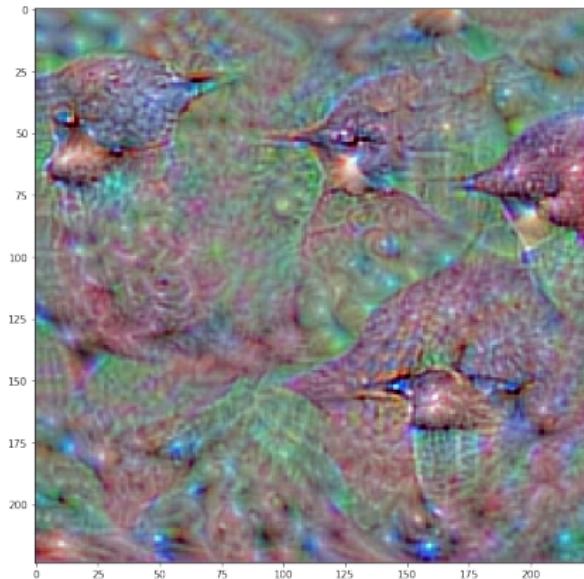
НАИВНАЯ РЕГУЛЯРИЗАЦИЯ



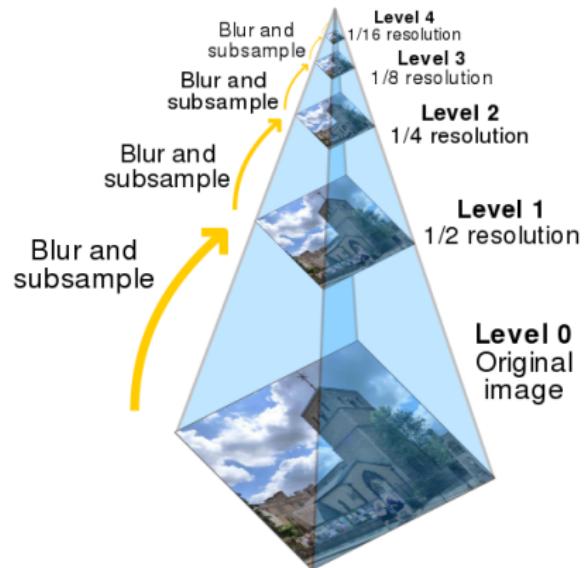
Дрозду стало лучше, но не настолько, насколько бы нам хотелось

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.

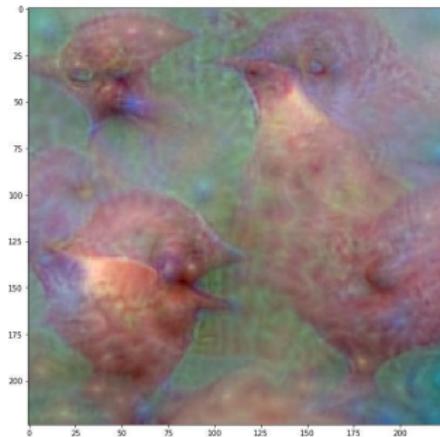
Пирамида лапласса:

- 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

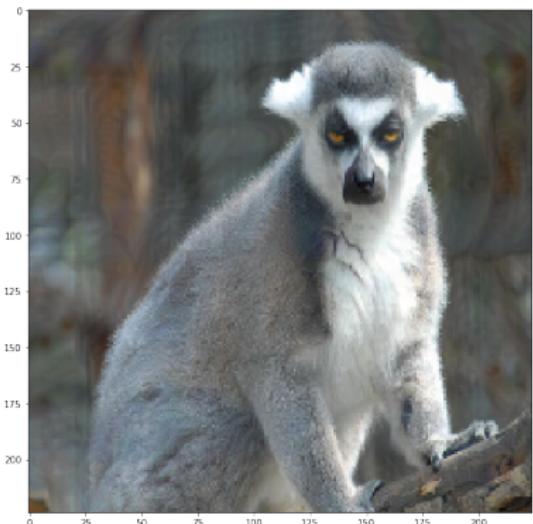
- Что будет если подать на вход изображение и попытаться максимизировать вероятность класса, к которому оно не принадлежит?
- Какое-то время визуально ничего не меняется, но как на это реагирует сеть?

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:

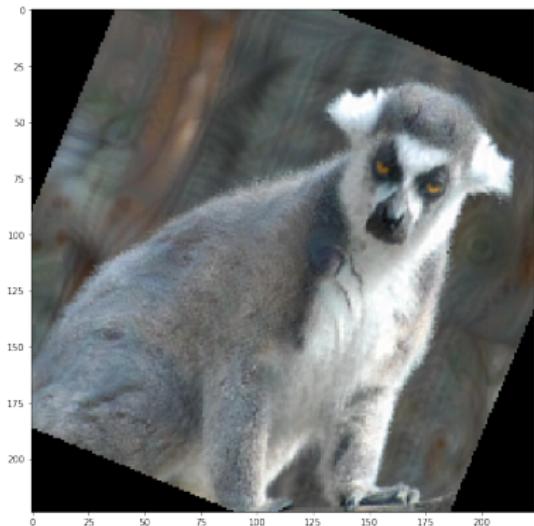
$$\mathbb{E}_t[\log(P(y|\hat{x}))] \rightarrow \max$$

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

ADVERSARIAL EXAMPLES



(o) Буритто



(p) Буритто