Глубокие нейронные сети

Александр Сарачаков, Анастасия Кишкун,

Сколтех, Москва, Россия

Skoltech
Skolkovo Institute of Science and Technology

1 Первое поколение Нейронных сетей

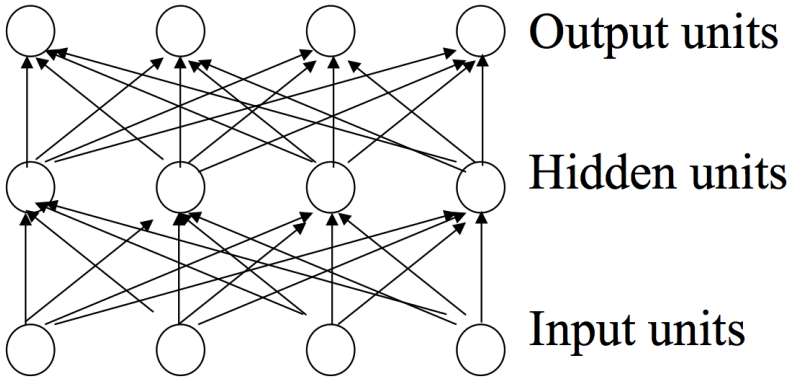
2 Современные нейронные сети

3 Свёрточные нейронные сети

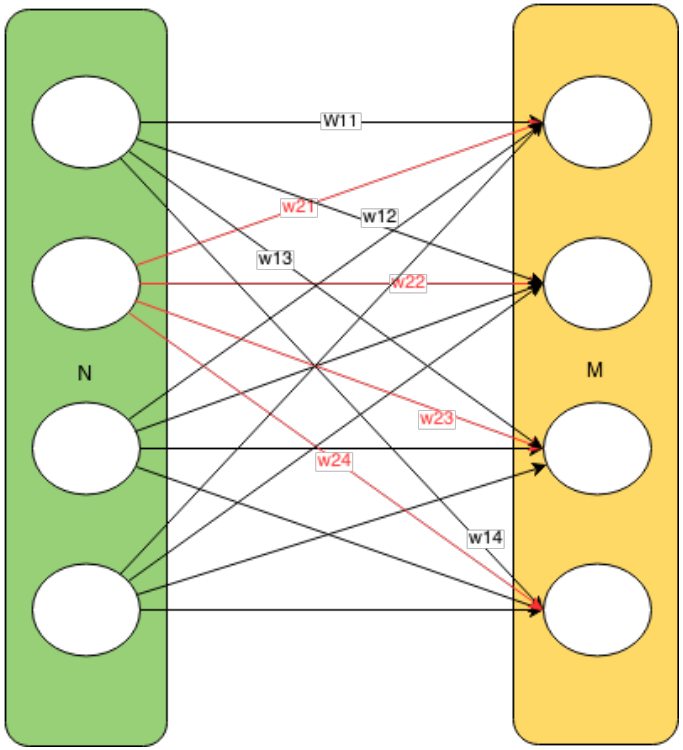
4 Регуляризация нейронных сетей

- Изначально самыми популярными были многослойные полносвязные нейросети прямого распространения

Полносвязная нейронная сеть

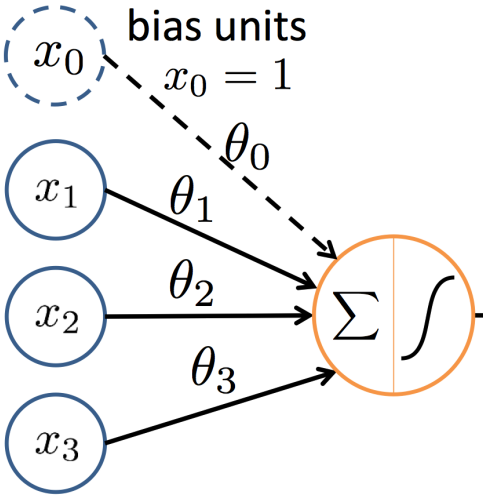


Полносвязный слой



Нейронные сети: логистическая функция (сигмоида)

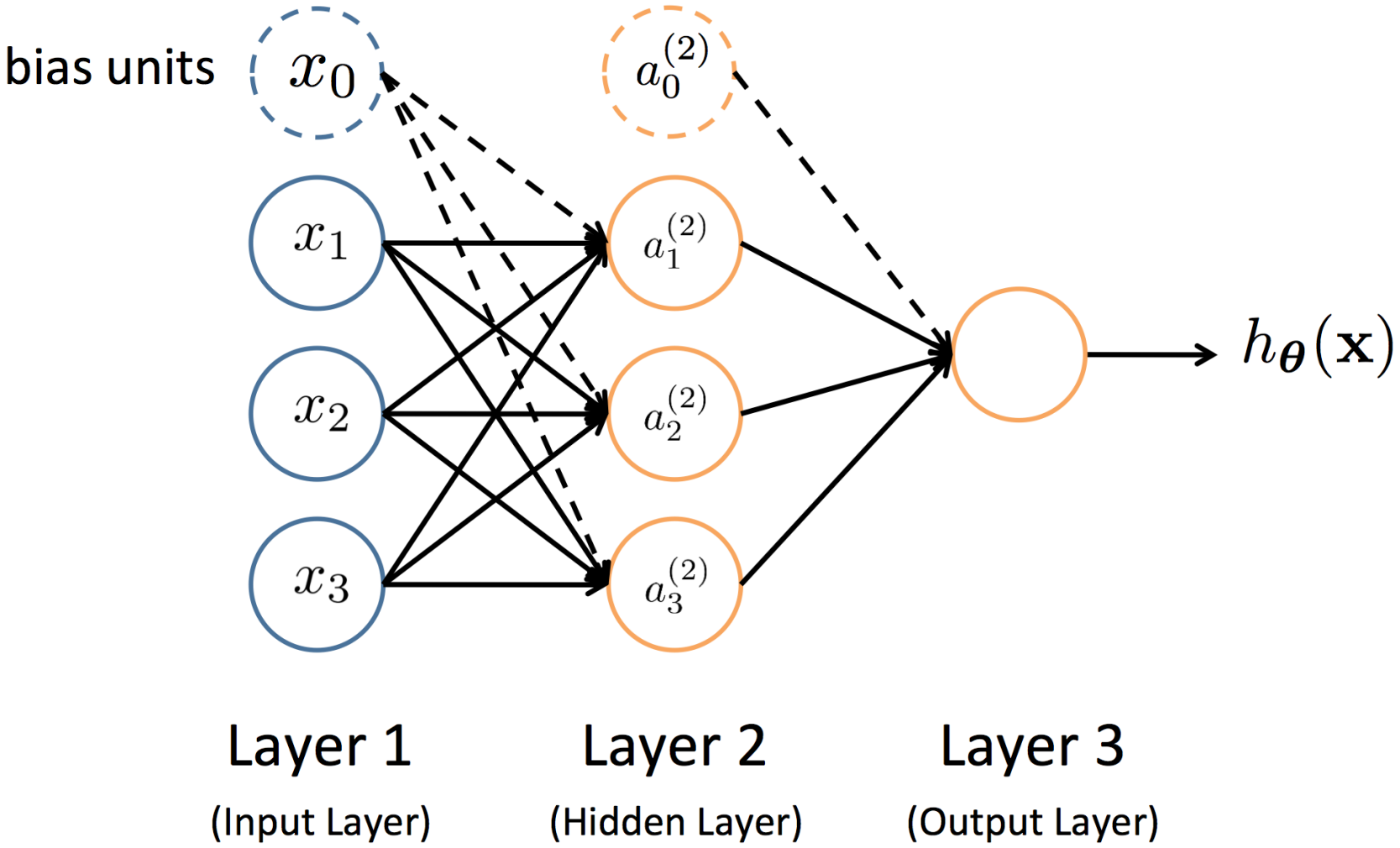
- Определим $\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$ и $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$



$$h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}\right) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}}}$$

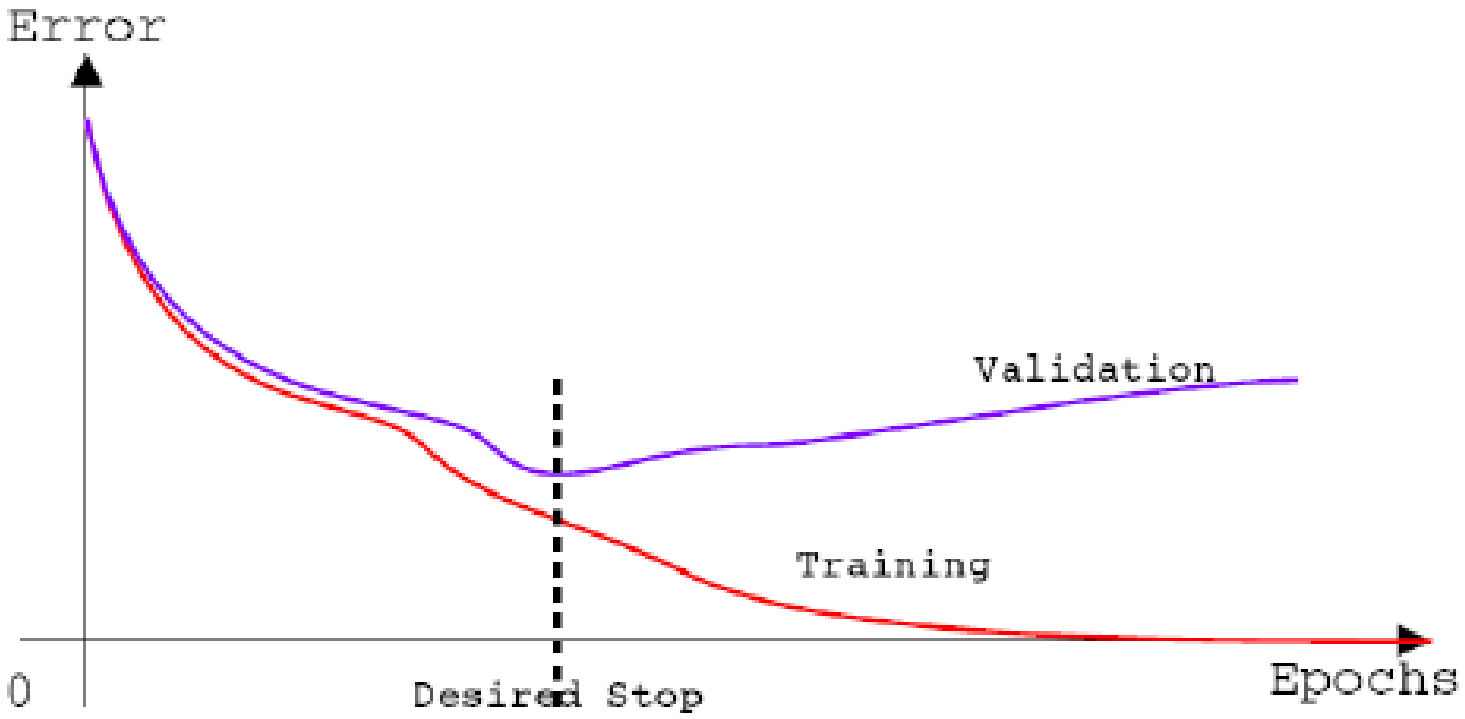
- Сигмоидная (логистическая) функция активации $g(z) = \frac{1}{1+e^{-z}}$

Многослойная нейронная сеть



Проблемы реального использования

- Обратное распространение ошибки (backpropagation) для обучения: градиентный спуск + эффективные вычисления (на основе представления в виде графов)
- Локальные минимумы
- Случайные инициализации
- Случайное разделение на данные для обучения и валидации
- Методы оптимизации второго порядка
- Ранняя остановка



1 Первое поколение Нейронных сетей

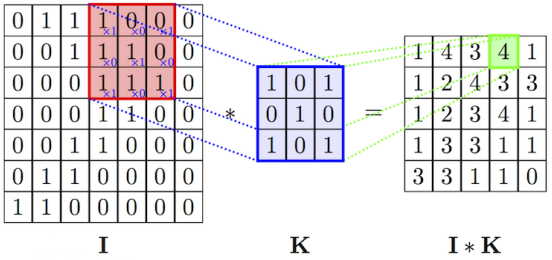
2 Современные нейронные сети

3 Свёрточные нейронные сети

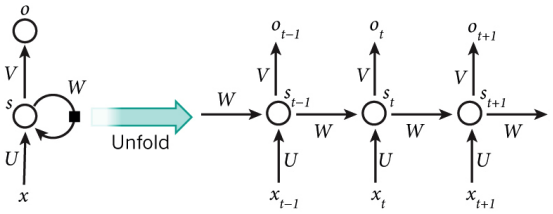
4 Регуляризация нейронных сетей

Различные архитектуры

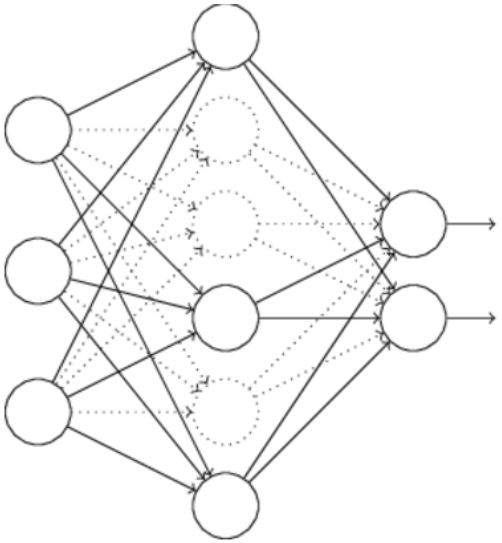
Свёрточный слой



Рекуррентный слой



Dropout слой



Вычислительные графы

- Наилучшее качество для многих задач компьютерного зрения и обработки естественного языка
- Иногда превосходит по качеству человека!
- Ключевые компоненты:
  - Вычислительная мощность и GPU
  - Большое количество данных
  - Стохастическая оптимизация
  - Регуляризация

Теорема об универсальной аппроксимации

Пусть $\phi(\cdot)$ — непостоянная ограниченная и монотонно-возрастающая непрерывная функция. Пусть I_m означает m-мерный единичный гиперкуб $[0,1]^m$. Пространство непрерывных функций на I_m обозначим $C(I_m)$. Затем, для любой $\varepsilon > 0$ и любой функции $f \in C(I_m)$, существует целое число p, действительные константы $v_i, b_i \in \mathbb{R}$ и действительные вектора $\mathbf{w}_i \in \mathbb{R}^m$, где $i = 1, \ldots, p$, такие, что мы можем определить:

$$F(\mathbf{x}) = \sum_{i=1}^{p} v_i \phi(\mathbf{w}_i^{\mathrm{T}} \mathbf{x} + b_i)$$

аппроксимированная реализация функции f, где f не зависит от ϕ; так, что,

$$|F(\mathbf{x}) - f(\mathbf{x})| < \varepsilon$$

для любых $\mathbf{x} \in I_m$. Другими словами, функции вида $F(\mathbf{x})$ плотны в $C(I_m)$.

Глубокие архитектуры лучше

- Глубокие нейросети можно обучать и регуляризовывать эффективно
- Глубокие нейросети могут достигать лучшего качества с тем же количество параметров
- Глубокие архитектуры более интуитивны и могут быть интерпретированы как последовательность простых преобразований в данных

Обучение графа вычислений

- $f_j(\mathbf{x}, \mathbf{w}_j)$ — j-й слой сети
- $v_j(\mathbf{x}) = f_j(v_{j-1}(\mathbf{x}), \mathbf{w}_j)$
- $a(\mathbf{x}, \mathbf{w}) = v_n(\mathbf{x})$ — выход
- Задача оптимизации

$$Q(\mathbf{w}) = \sum_{i=1}^{l} L(y_i, a(\mathbf{x}_i, \mathbf{w})) \to \min_{\mathbf{w}}$$

Метод обратного распространения ошибки

$$\frac{\partial Q}{\partial w_i} = \frac{\partial Q}{\partial f_{i+1}} \frac{\partial f_{i+1}}{\partial f_i} \frac{\partial f_i}{\partial w_i}$$

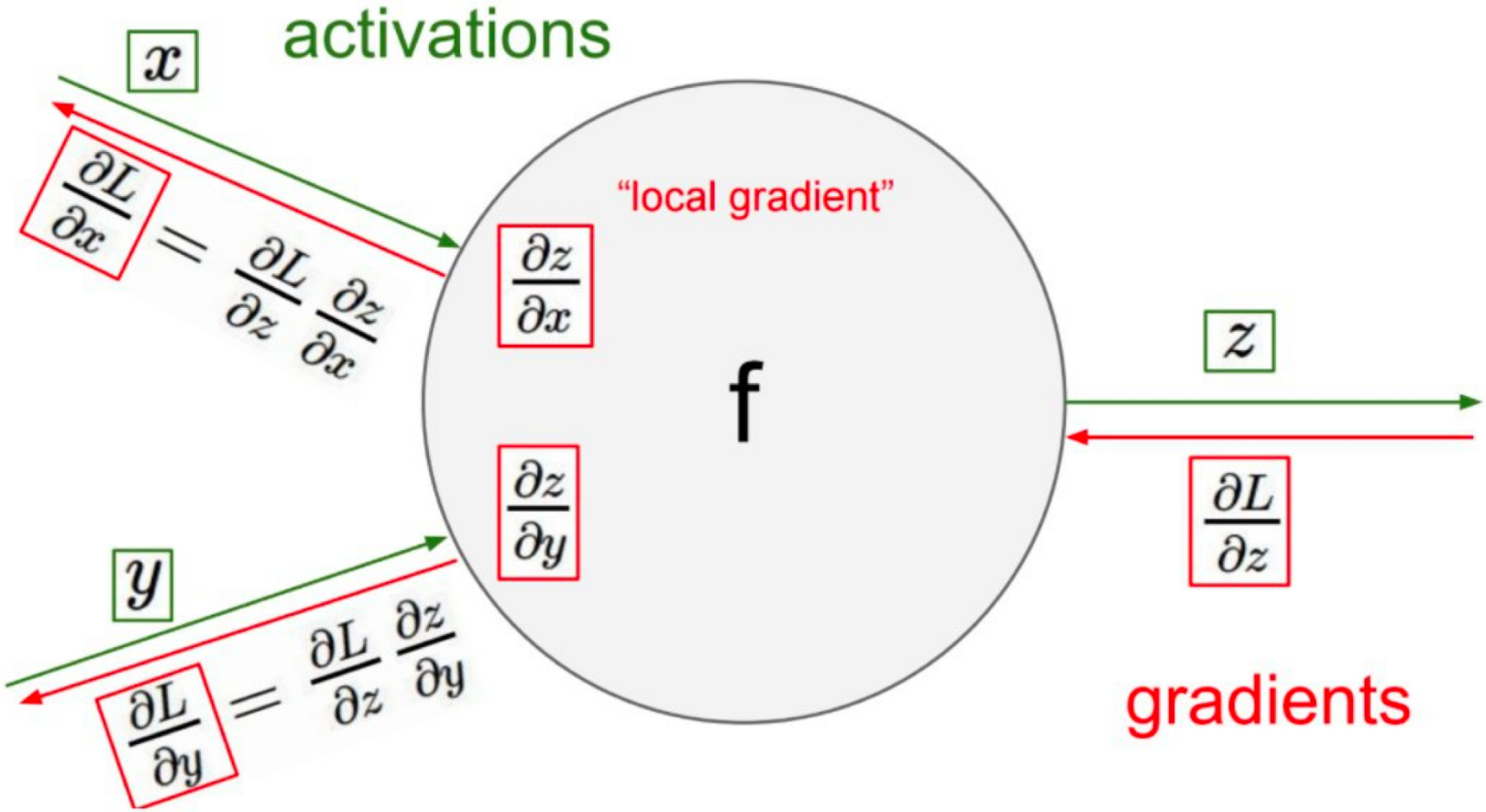
Comes from previous steps

Easy for differentiable layers

$$\frac{\partial Q}{\partial f_{ij}} = \sum_k \frac{\partial Q}{\partial f_{i+1,k}} \frac{\partial f_{i+1,k}}{\partial f_{ij}}$$

$$\frac{\partial Q}{\partial w_{ij}} = \sum_k \frac{\partial Q}{\partial f_{ik}} \frac{\partial f_{ik}}{\partial w_{ij}}$$

Метод обратного распространения ошибки



1 Первое поколение Нейронных сетей

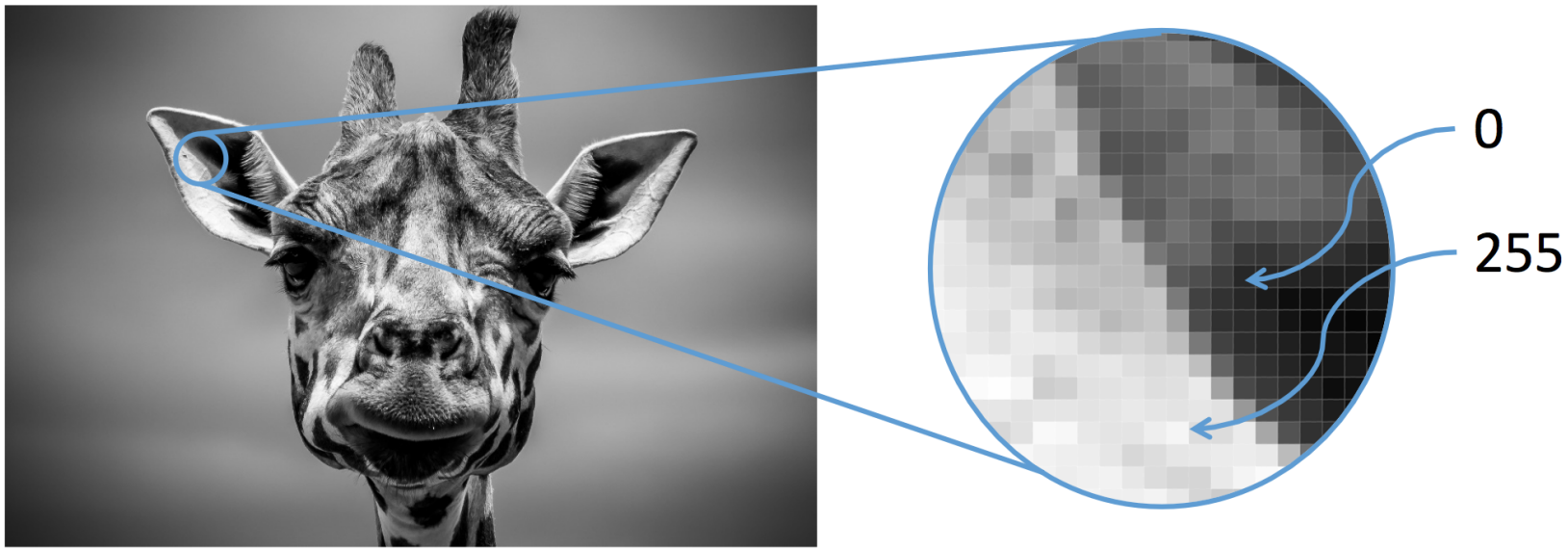
2 Современные нейронные сети

3 Свёрточные нейронные сети

4 Регуляризация нейронных сетей

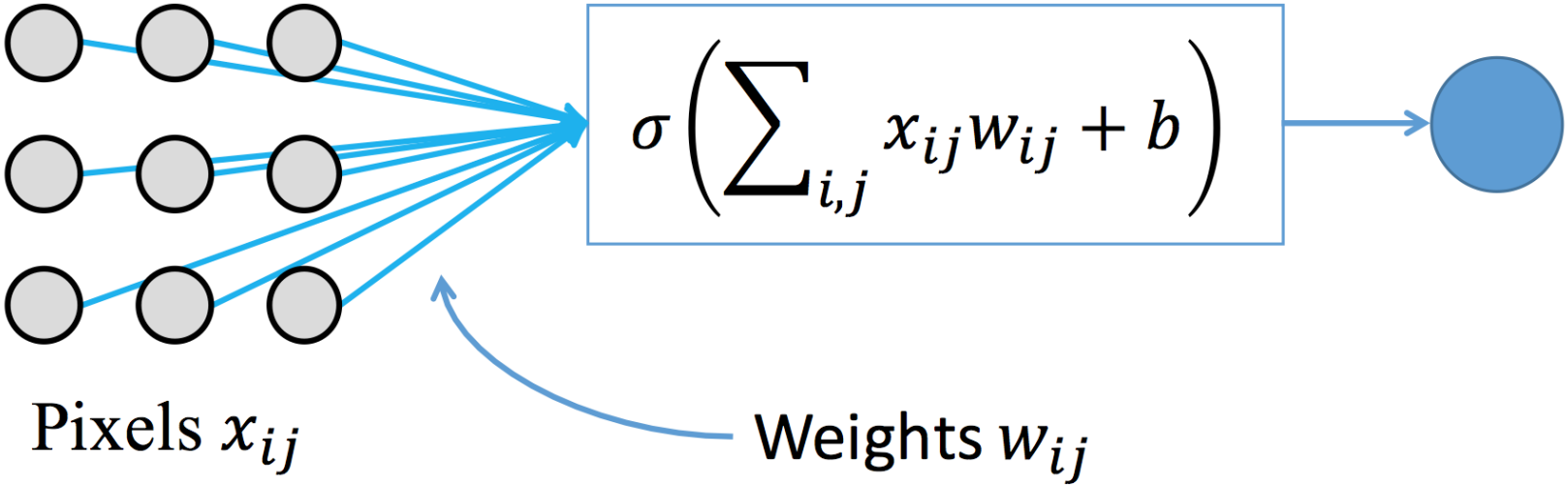
Представление изображений

- В каждом пикселе находится его яркость (или **интенсивность**) — число от 0 до 255, интенсивность 0 соответствует чёрному цвету
- В цветных изображениях в каждом пикселе содержатся интенсивности для 3-х каналов: **красного**, **зелёного** и **синего**



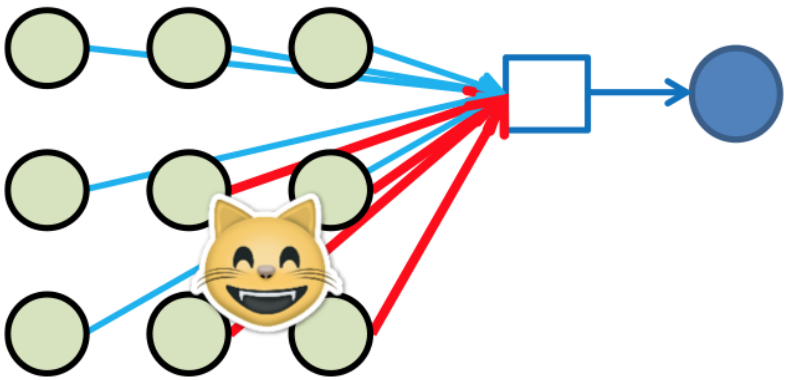
Многослойный персептрон (МП) для классификации изображений

- Нормализация входных пикселей: $\mathbf{x}_{\text{norm}} = \frac{\mathbf{x}}{255} - 0.5$
- Возможно МП будет работать?



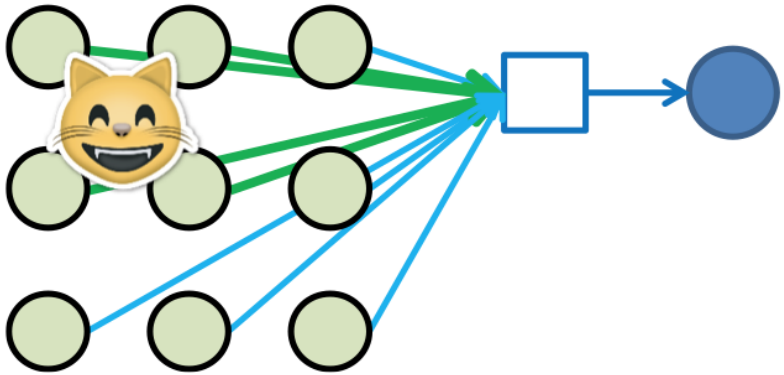
Многослойный персептрон (МП) для классификации изображений

Полносвязная нейросеть прямого распространения



На этом тренировочном изображении красные веса w_{ij} будут немного изменены для лучшего распознавания кота

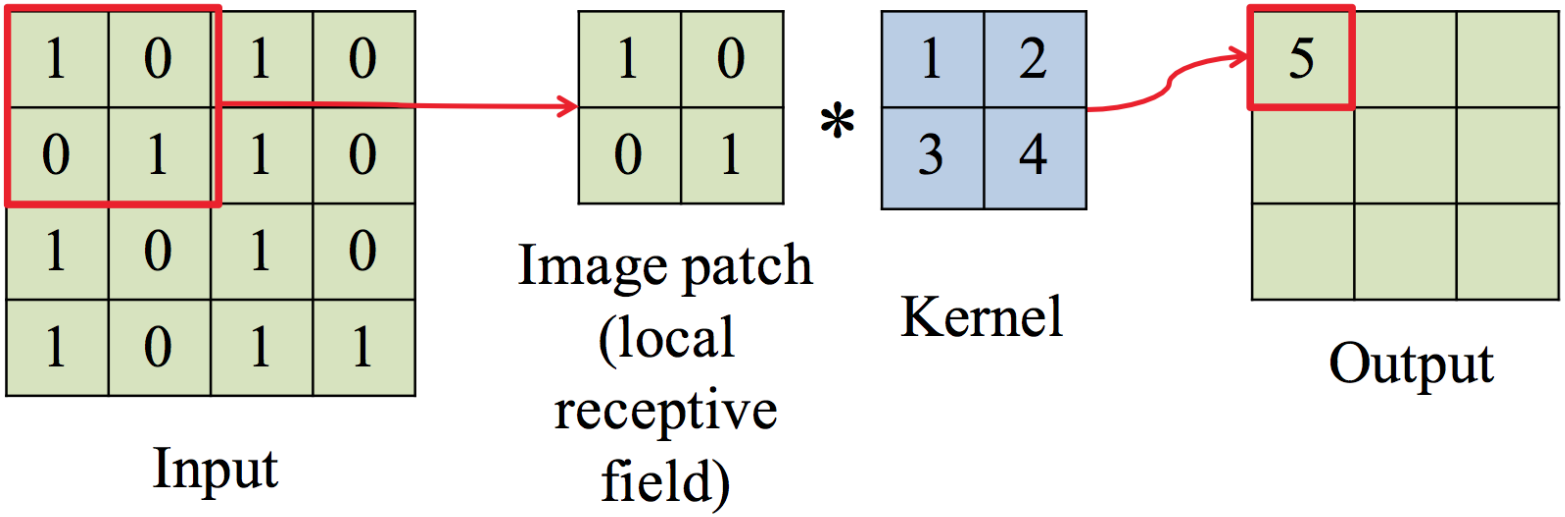
Полносвязная нейросеть прямого распространения



На этом тренировочном изображении зелёные веса w_{ij} будут немного изменены для лучшего распознавания кота

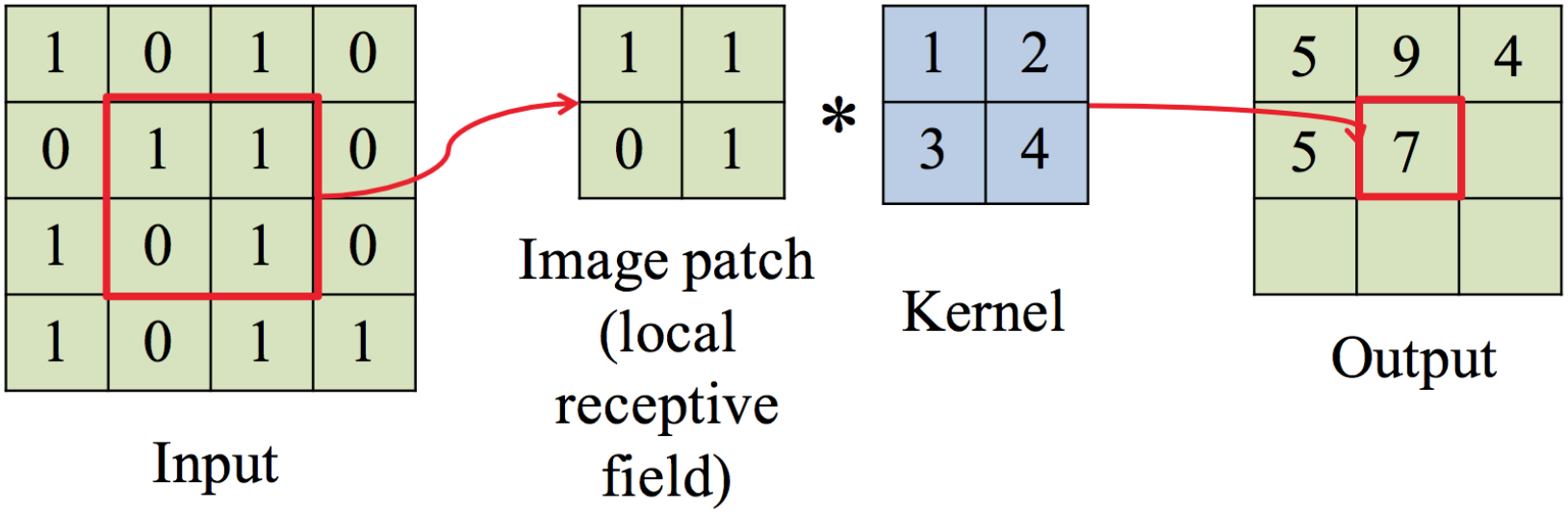
Свёртки

- Свёртка — матричное произведение **ядра** (или фильтра) и куска изображения (**локальное рецептивное поле**) того же размера



Свёртки

- Свёртка — матричное произведение **ядра** (или фильтра) и куска изображения (**локальное рецептивное поле**) того же размера



Раньше свёртки так же использовались

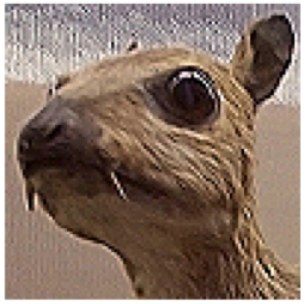
Original image

Kernel

$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$

Edge detection

$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$

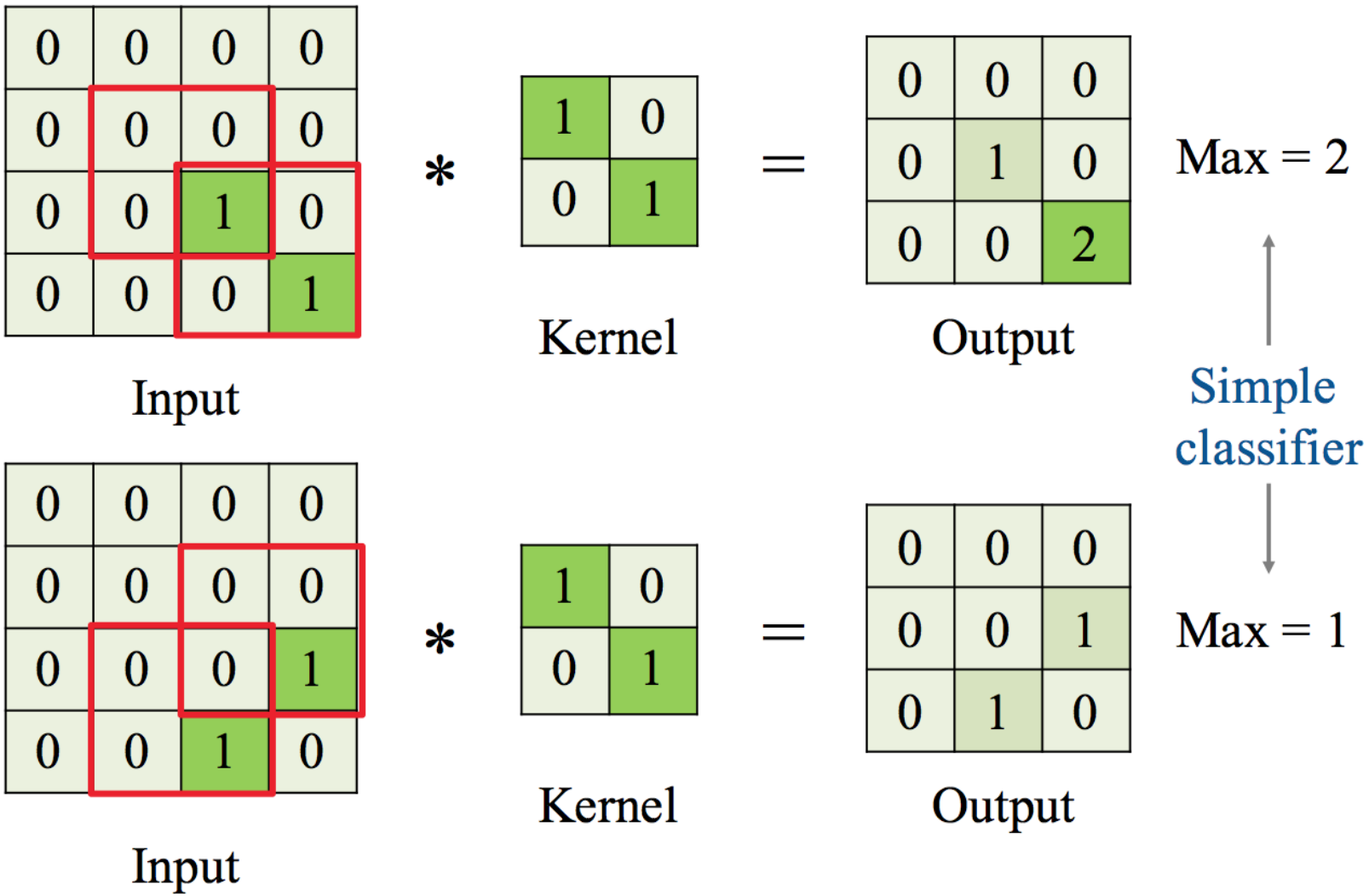


Sharpening

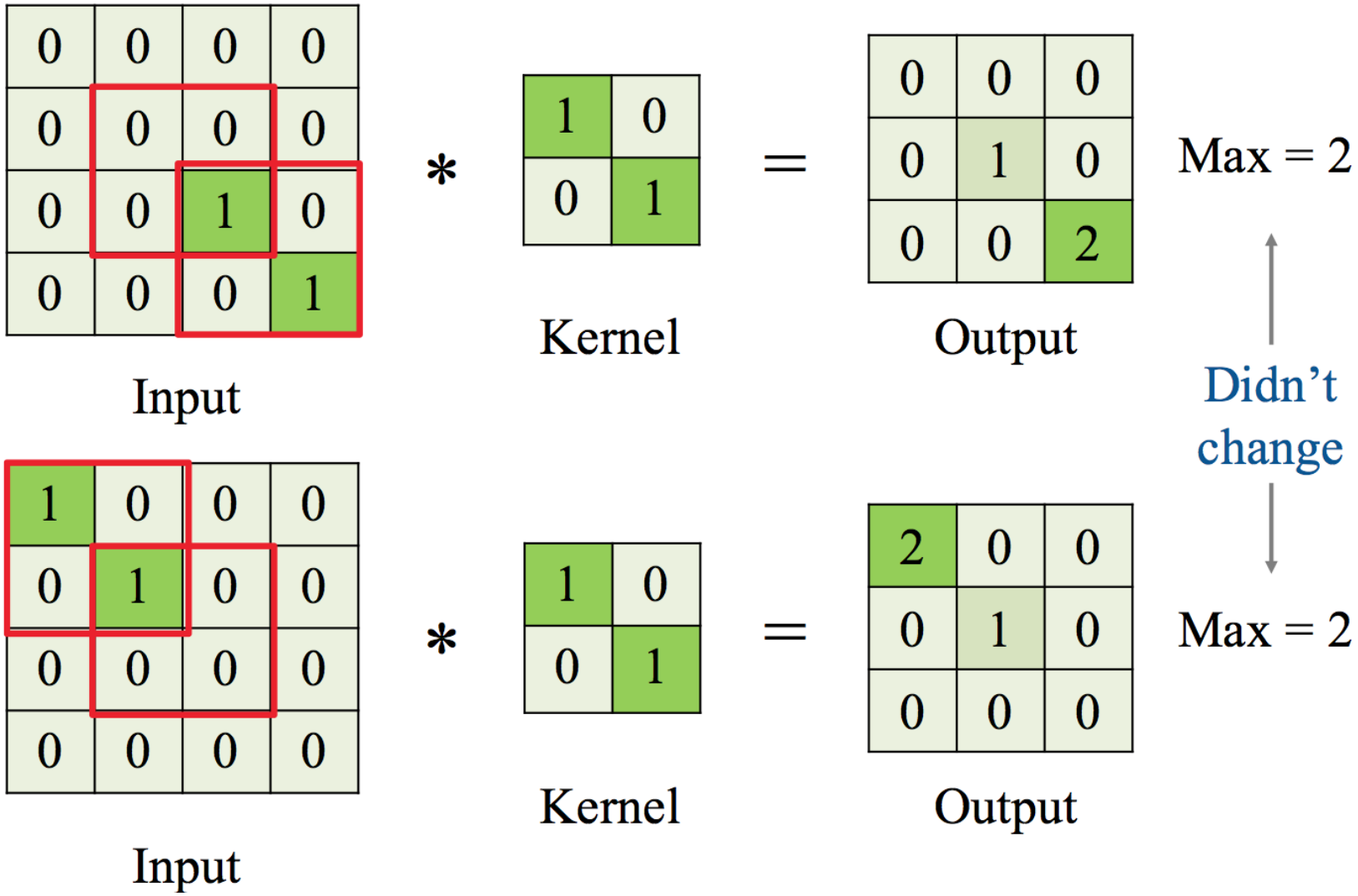
$$* \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

Blurring

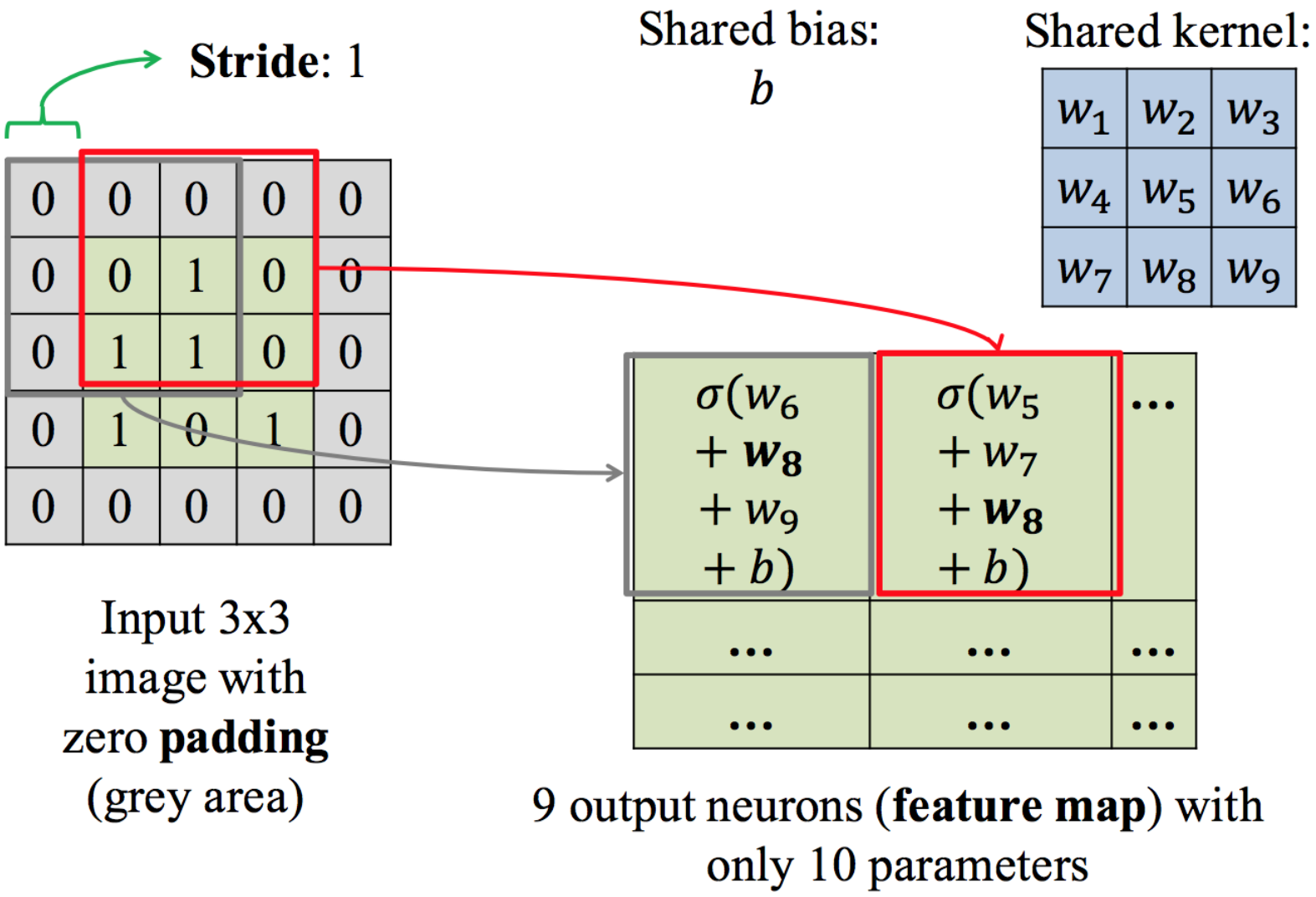
Свёртки схожи с корреляцией



Свёртки инвариантны к преобразованиям

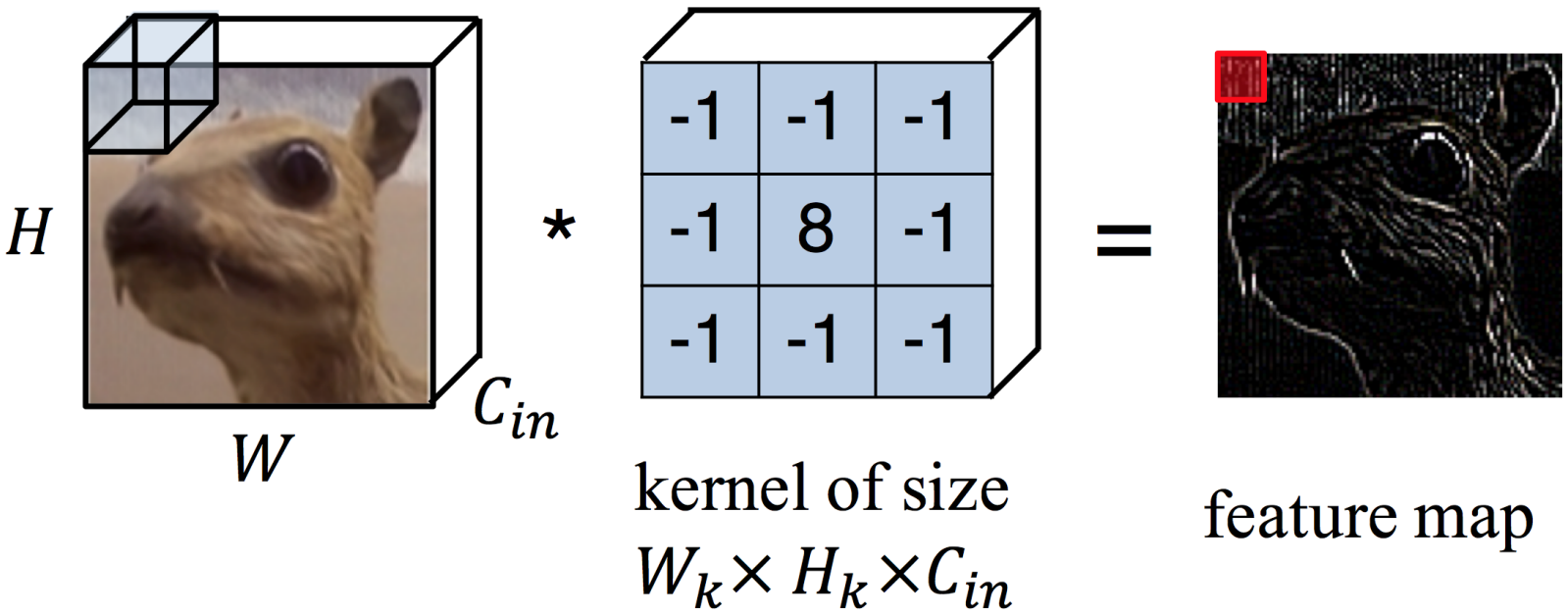


Свёрточные слои



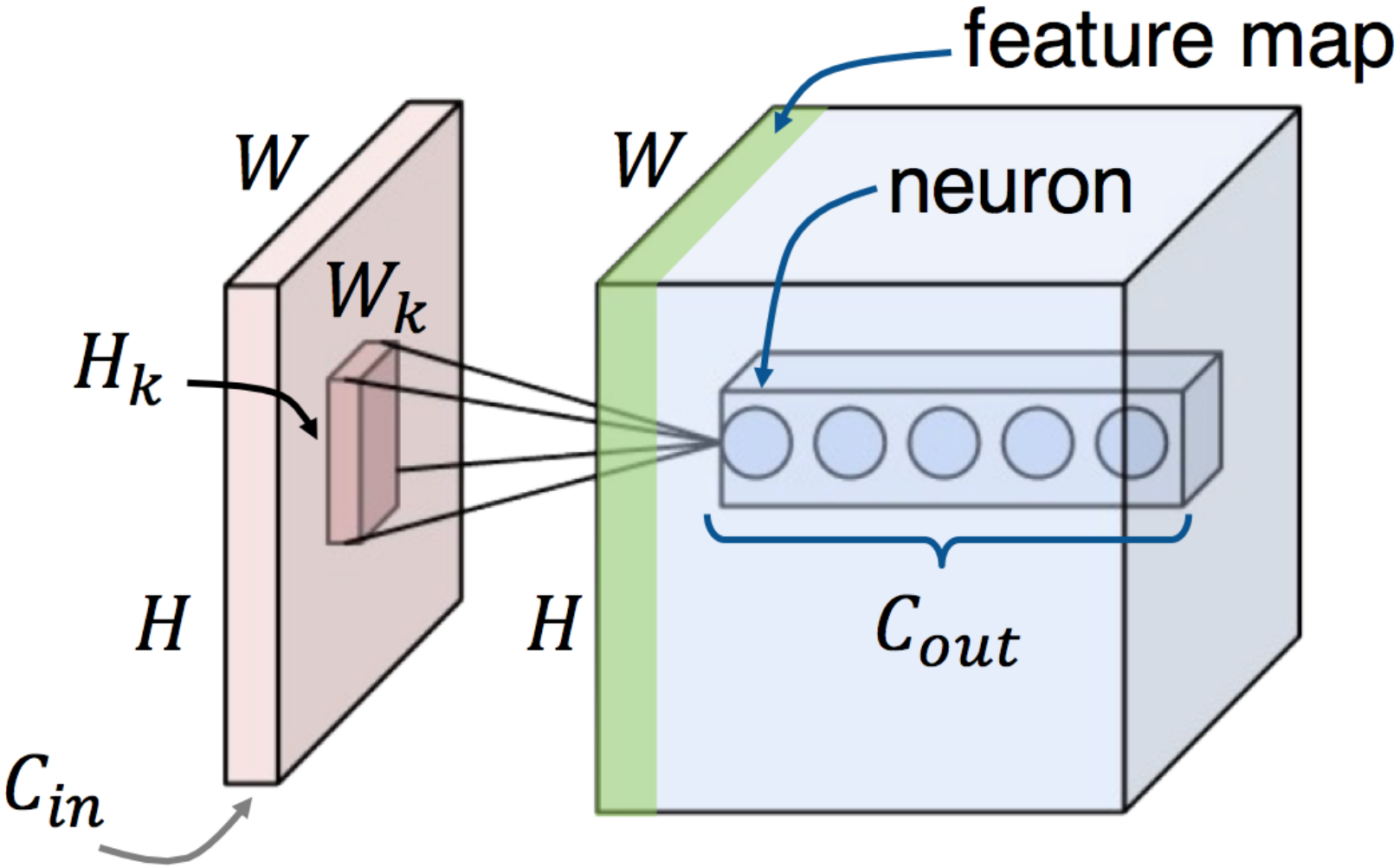
Допустим у нас есть цветное изображение на входе — $W \times H \times C_{in}$ **тензор** (многомерный массив), где

- W — ширина изображения
- H — высота изображения
- C_{in} количество каналов на входе (например, 3 RGB канала)



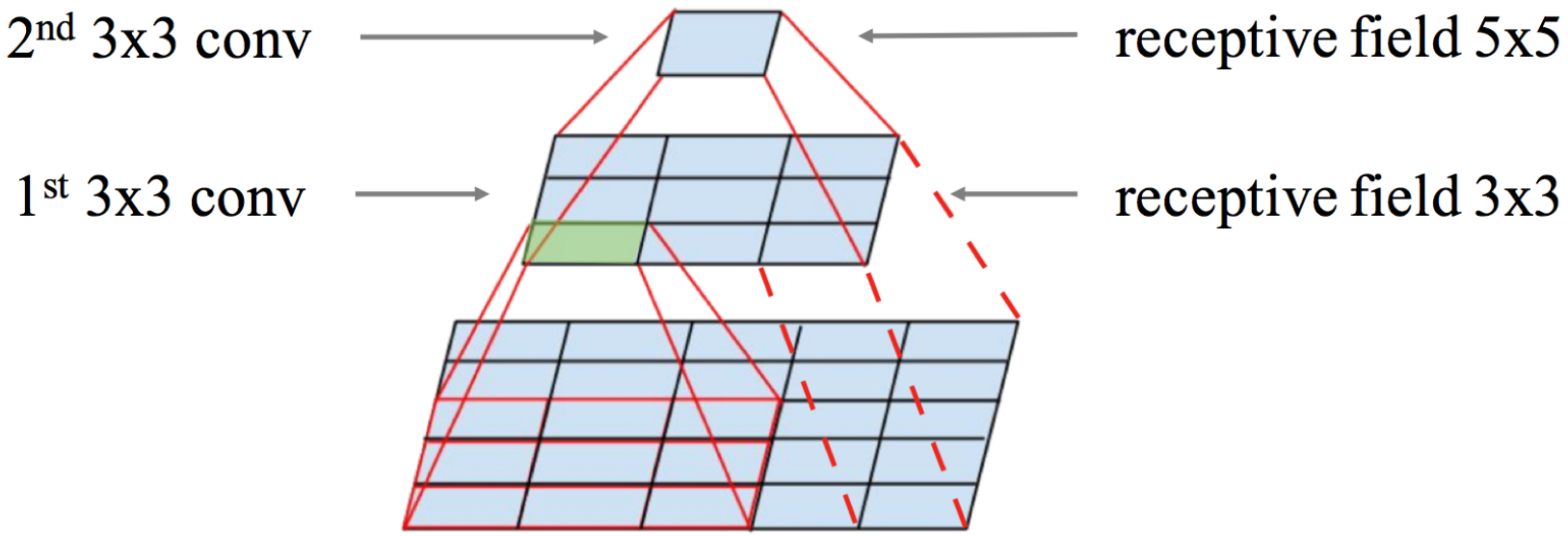
Свёрточный слой

- Мы хотим обучить ядро C_{out} размера $W_k \times H_k \times C_{in}$
- Задав шаг 1 и окаймление входного изображения нулями (zero-padding), на выходе мы получим $W \times H \times C_{out}$ нейронов



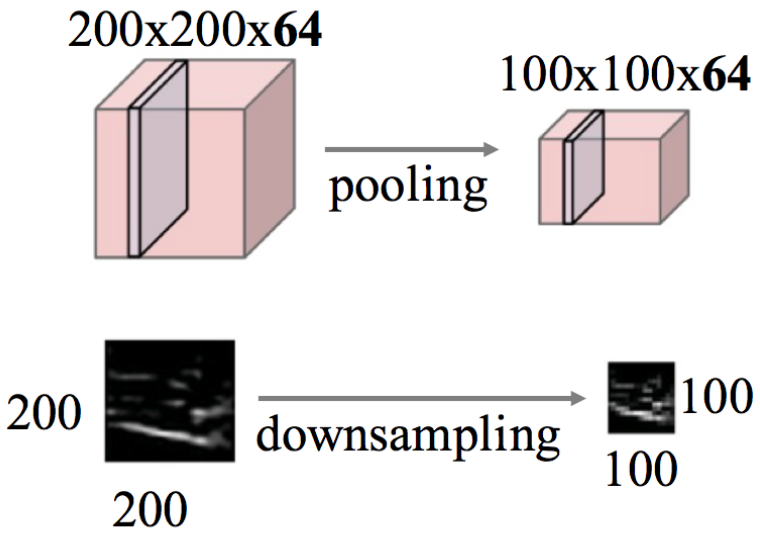
Одного свёрточного слоя не достаточно

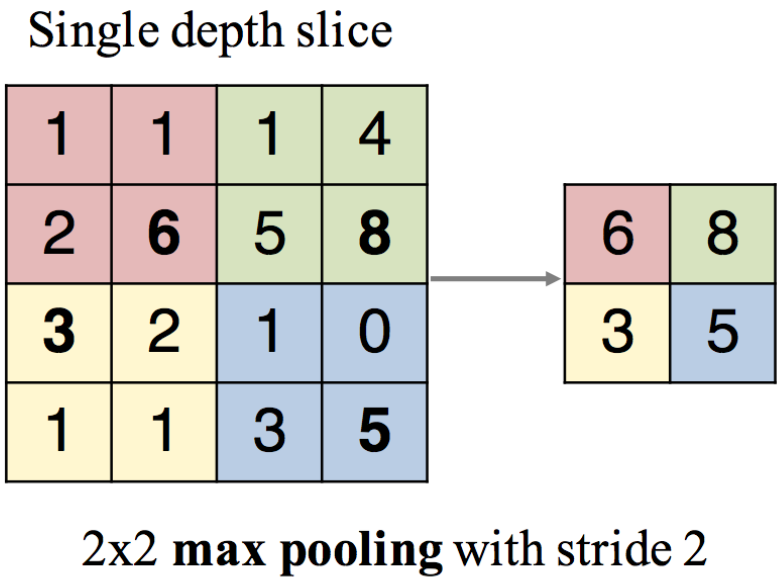
- Допустим нейроны 1-го свёрточного слоя смотря на патчи (куски) изображения размера 3×3
- Что, если объект нашего интереса большего размера
- Нам нужен 2-й свёрточный слой поверх 1-го!



Pooling слой

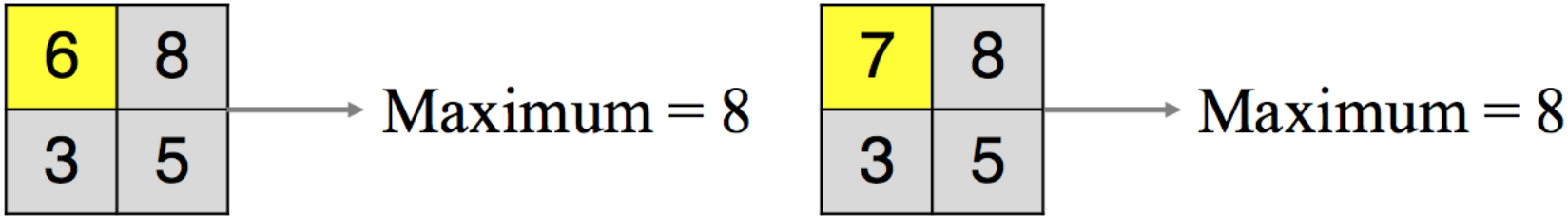
- Этот слой работает как свёрточный слой, но не имеет ядра, вместо этого он вычисляет **максимум** или **среднее** входных значений



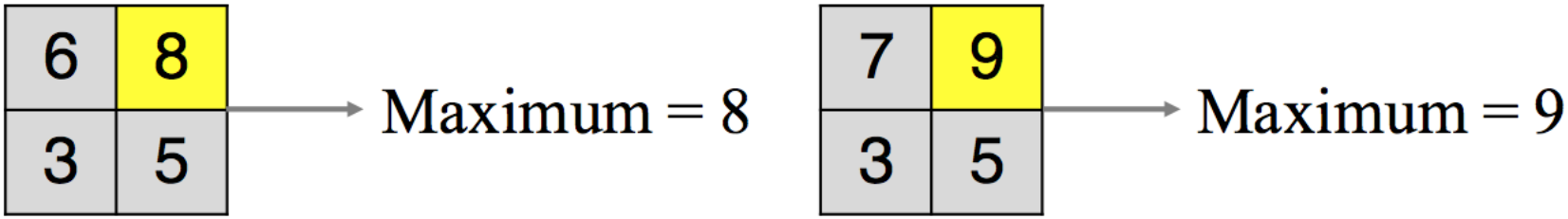


Pooling слой

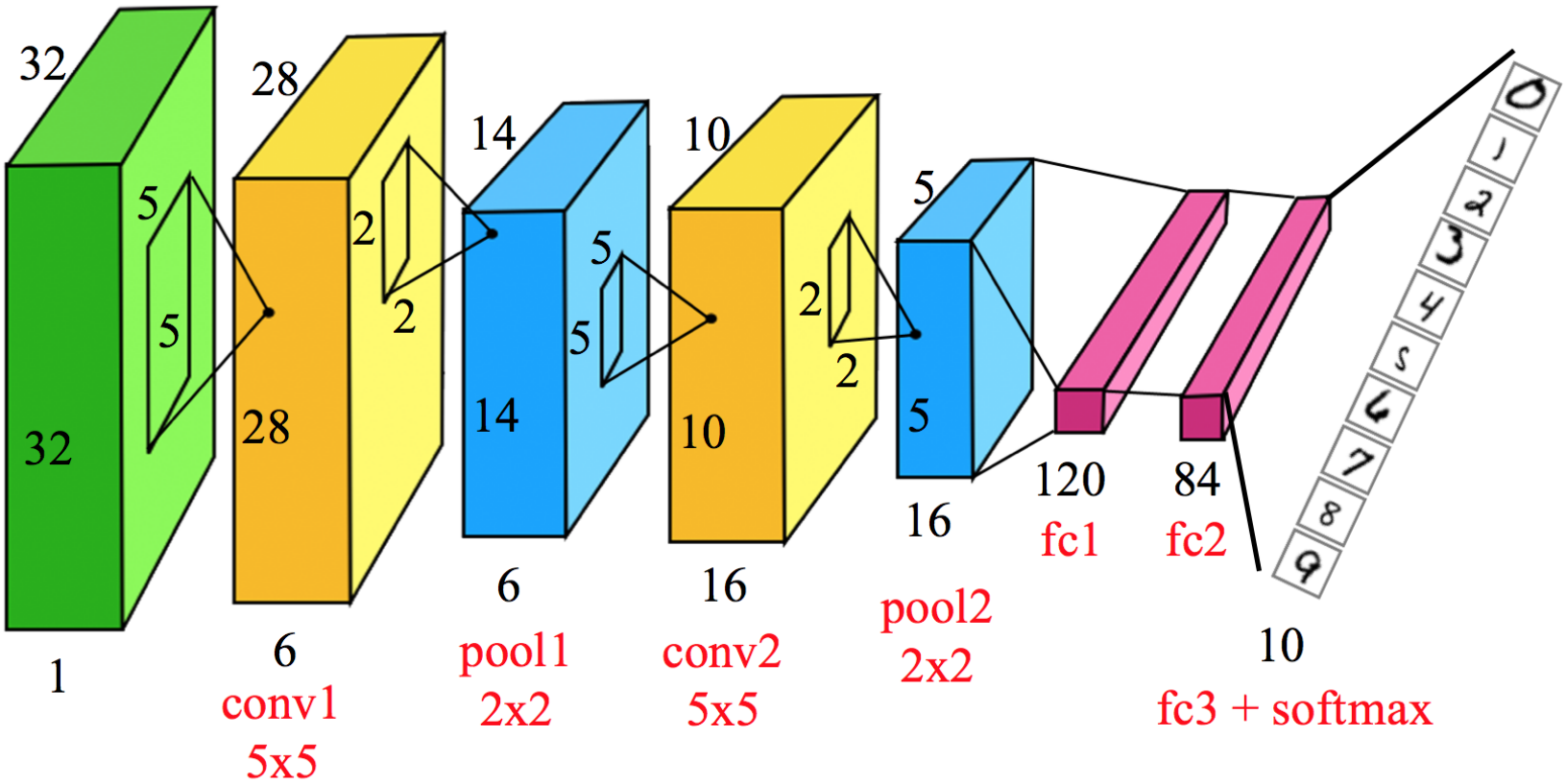
- Строго говоря, максимум — недифференцируемая функция



- Нет градиента по отношению к не максимальным нейронам, так как небольшое их изменение не влияет на выходные данные.



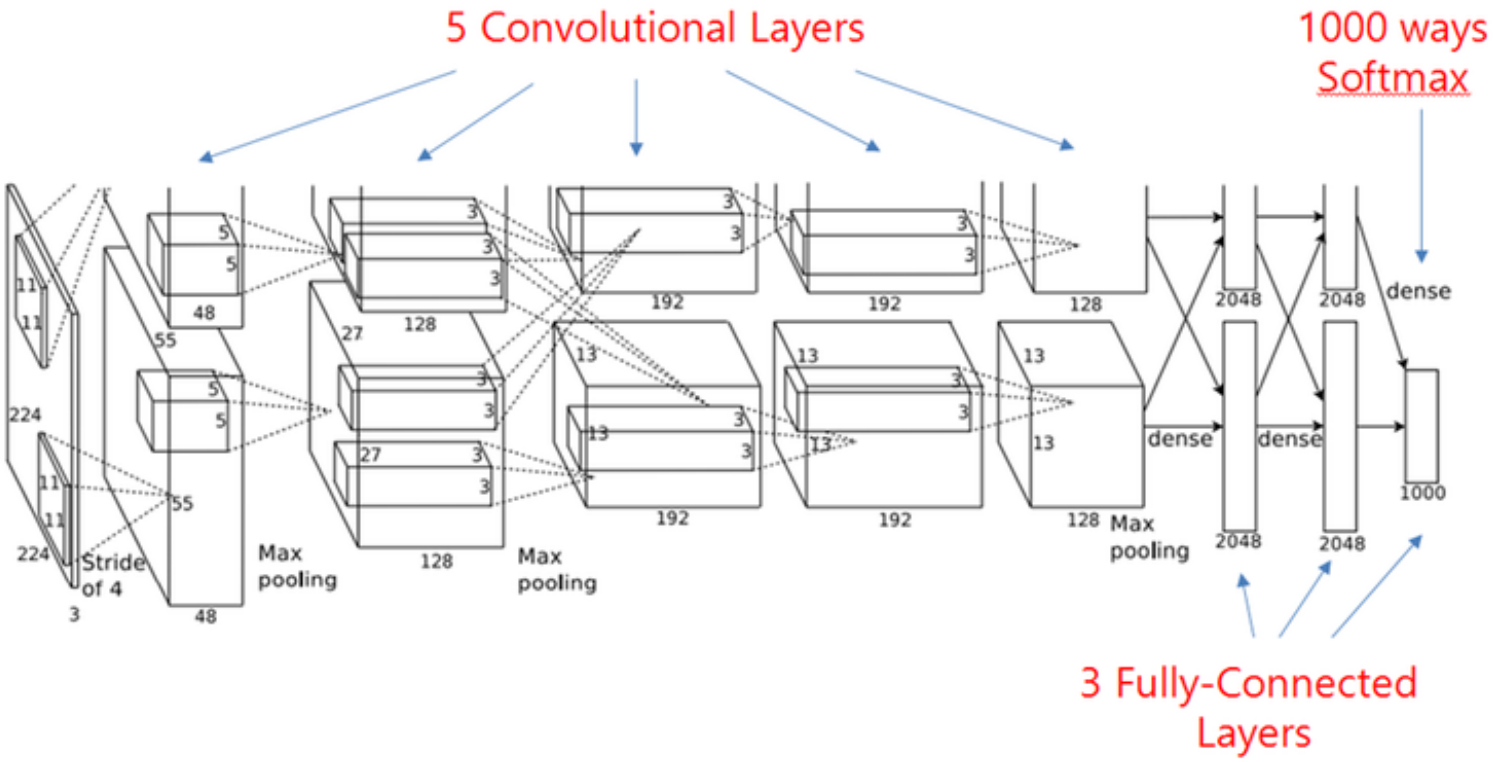
Свёрточные нейросети



http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

AlexNet (2012)

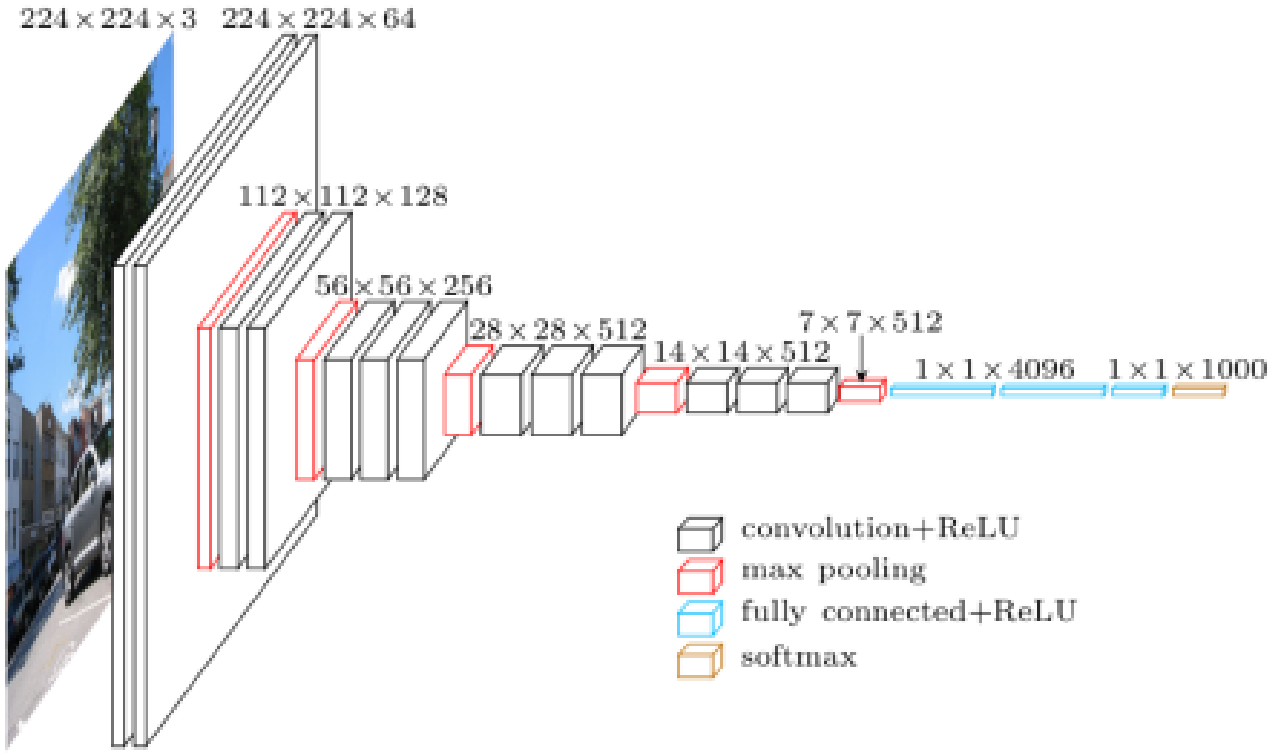
- Первая глубокая нейросеть для ImageNet
- Значительно уменьшила топ 5 ошибку с 26% до 15%



- 11×11, 5×5, 3×3 свёртки, max pooling, dropout, аугментация данных, ReLU активации, SGD с Momentum
- 60 миллионов параметров
- Обучение на 2 GPUs в течении 6 дней

VGG (2015)

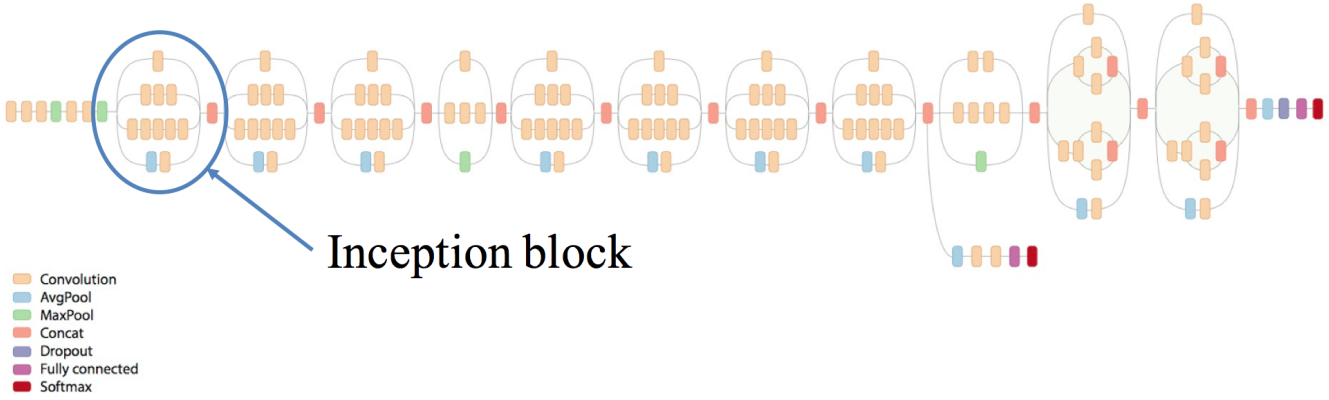
- Похожа на AlexNet, только 3×3 свёртки, но большое количество фильтров!
- ImageNet топ 5 ошибка: 8.0% (одна модель)



- Обучение похоже на AlexNet с дополнительным обрезанием на разных масшабах (multi-scale cropping)
- 138 миллионов параметров
- Обучается на 4 GPUs в течении $2 - 3$ недель

Inception V3 (2015)

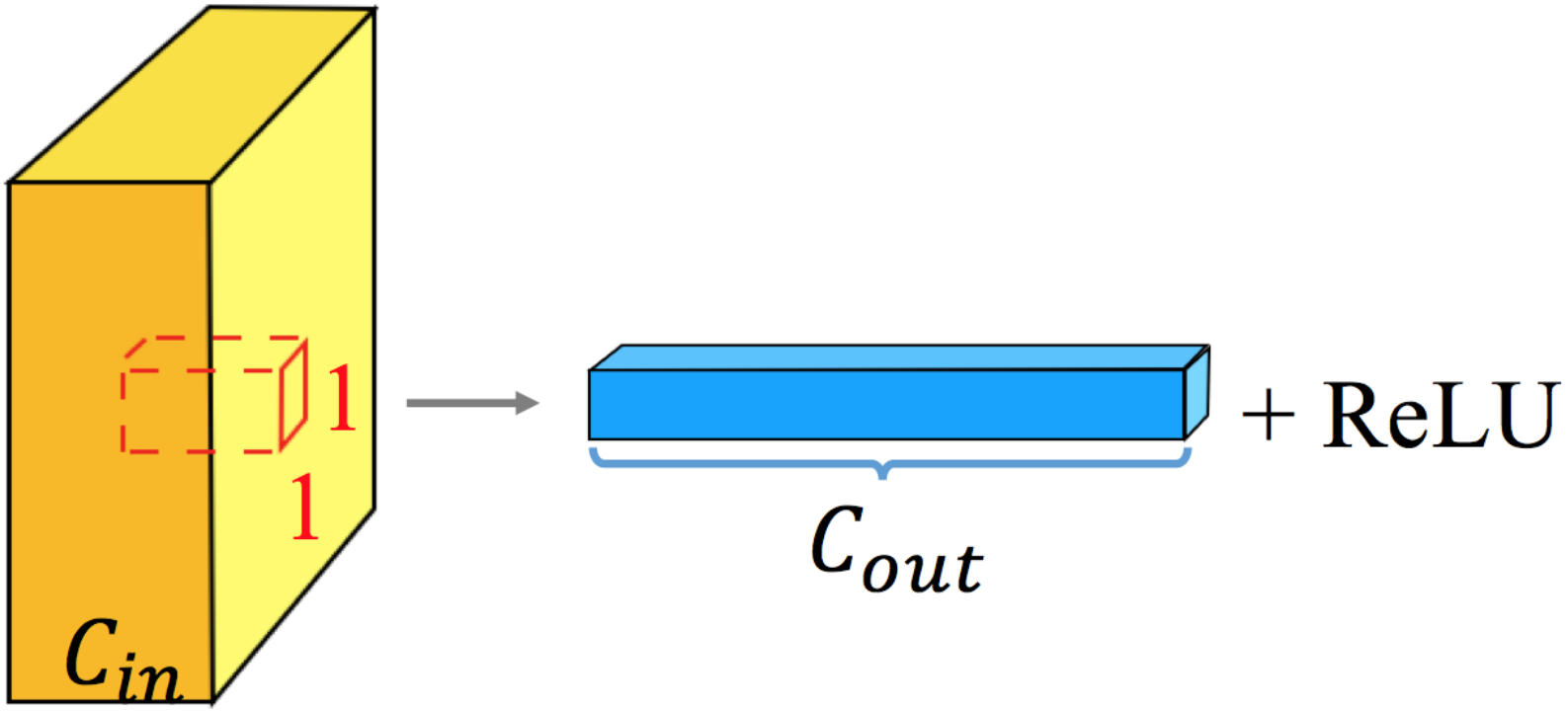
- Похожа на AlexNet? Не совсем, использует Inception блоки, представленные в GoogLeNet (a.k.a. Inception V1)
- ImageNet топ 5 ошибка: 5.6% (одна модель), 3.6% (ансамбль)



- Batch normalization, искажения изображений, RMSProp
- 25 миллионов параметров!
- Обучение на 8 GPUs в течении 2 недель

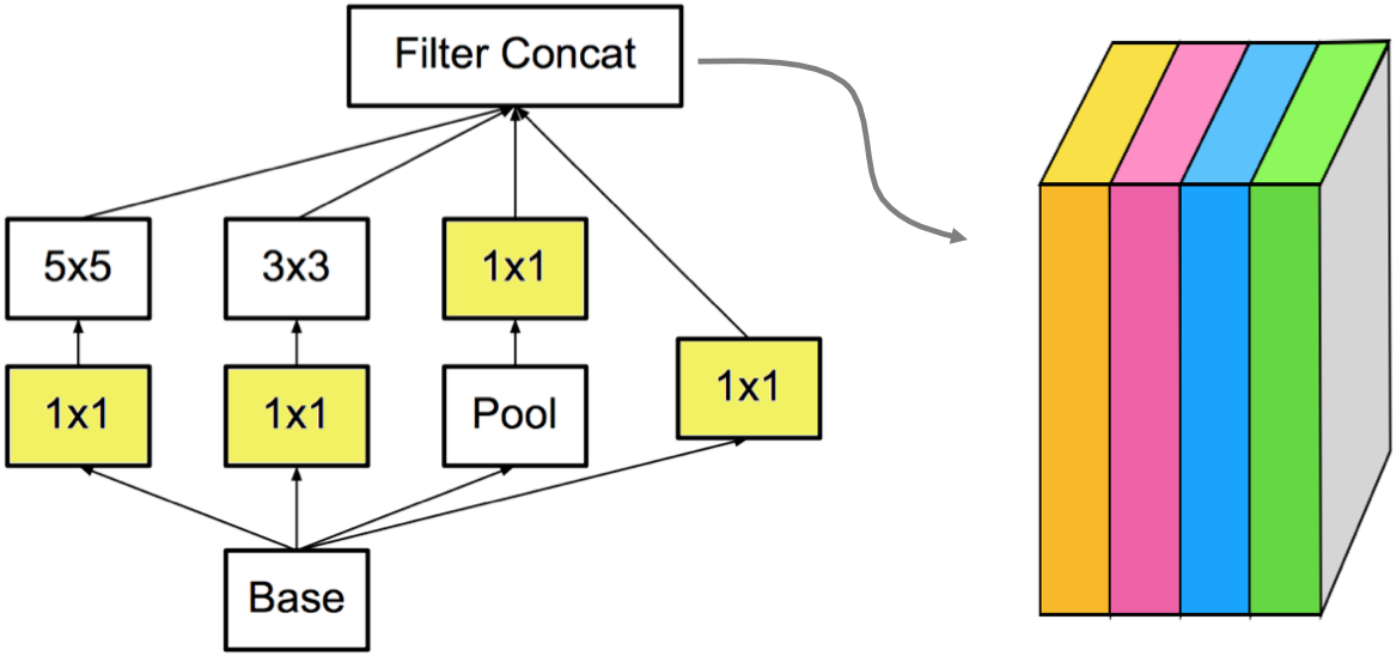
Свёртки 1×1

- Такие свёртки отлавливают взаимодействия входных каналов в одном “пикселе” карты признаков
- Они могут уменьшить количество каналов без ущерба для качества модели из-за того, что различные каналы могут коррелировать
- По сути это снижение размерности с добавлением активации ReLU



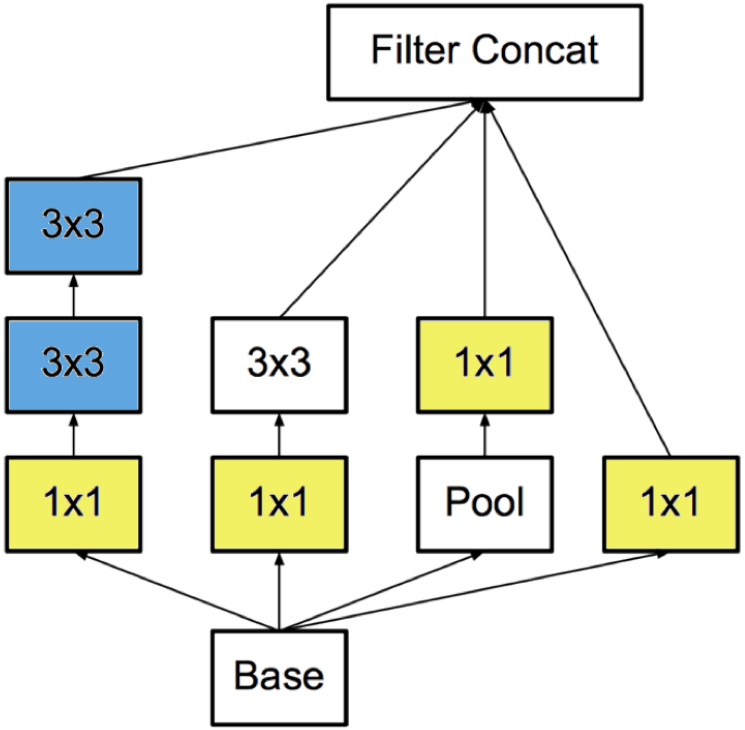
Inception блок

- Все операции внутри блока используют шаг 1 и достаточный padding для вывода тех же пространственных измерений $(W \times H)$ на карте признаков.
- 4 различные карты признаков конкатенируются по глубине на выходе из блока

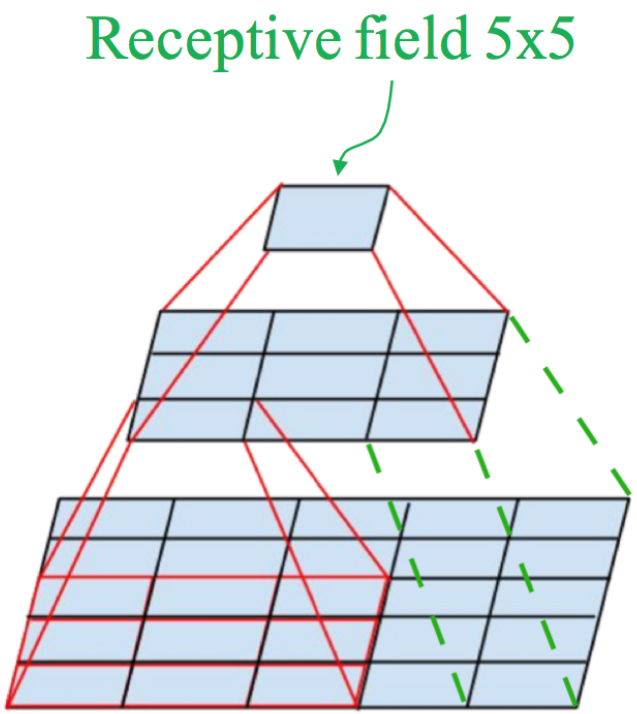


Christian Szegedy, https://arxiv.org/pdf/1512.00567.pdf

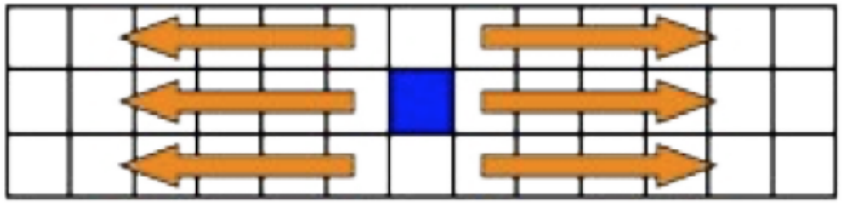
Замена свёрток 5×5



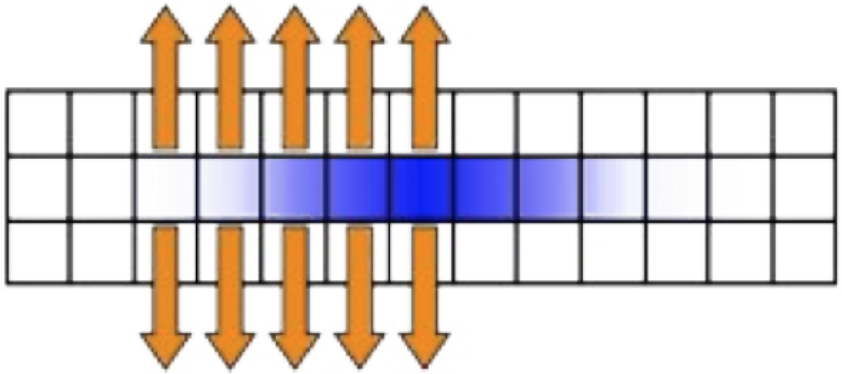
Christian Szegedy, https://arxiv.org/pdf/1512.00567.pdf



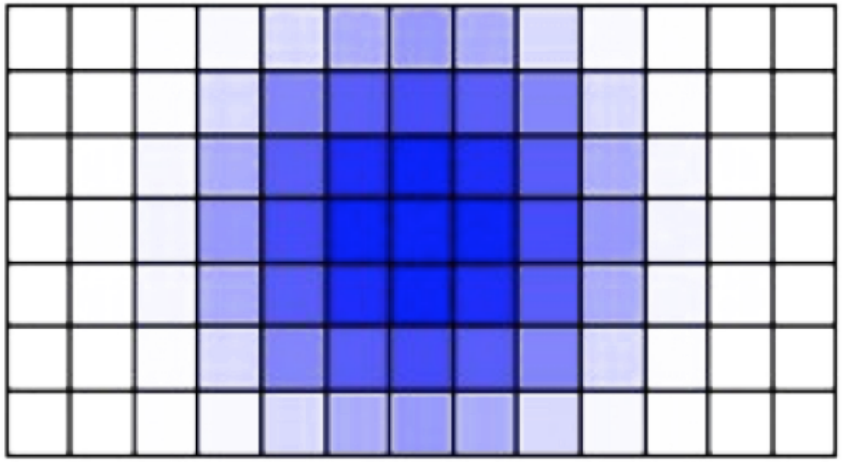
Разложение фильтров



Blur the source horizontally

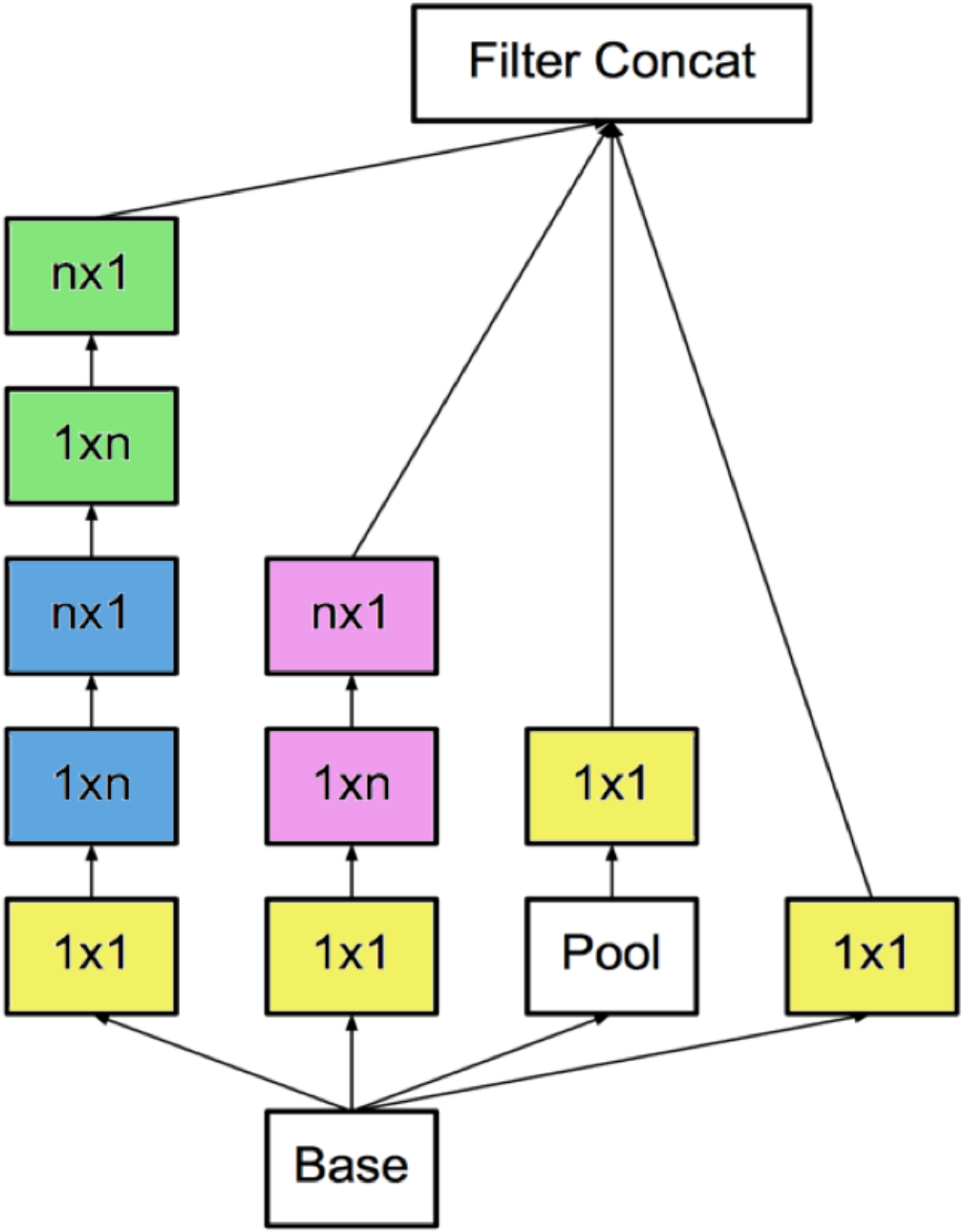


Blur the blur verticaly



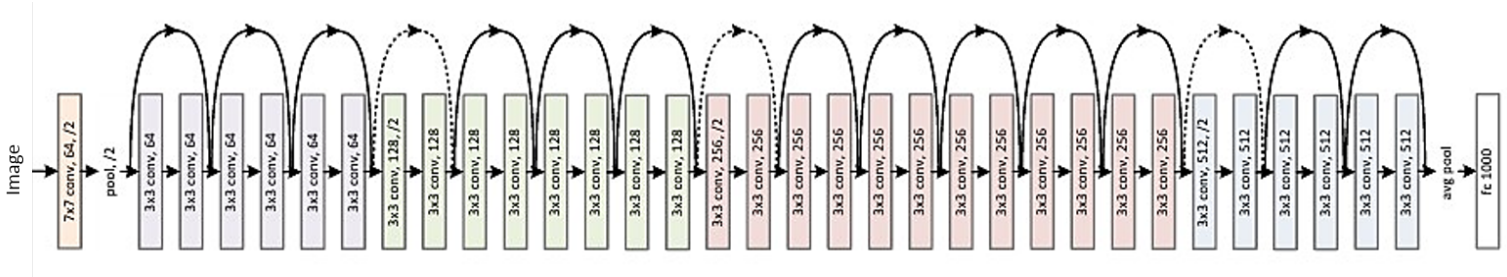
Result

Inception блок



ResNet (2015)

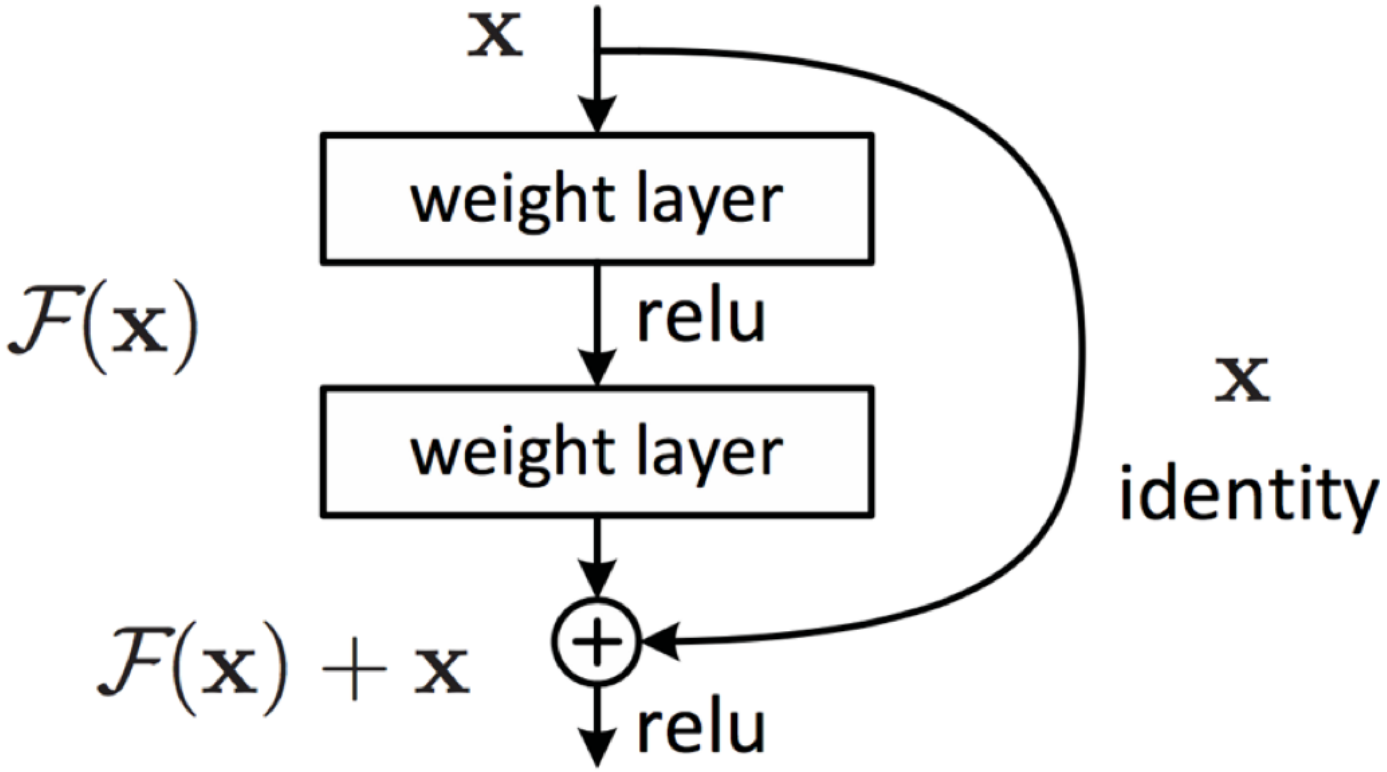
- Представили residual связи
- ImageNet топ 5 ошибка: 4.5% (одна модель), 3.5% (ансамбль)



- 152 слоёв, несколько 7×7 свёрток, остальные 3×3, batch normalization, max и average pooling
- 60 миллионов параметров
- Обучалось на 8 GPUs в течении $2 - 3$ недель

Residual связи

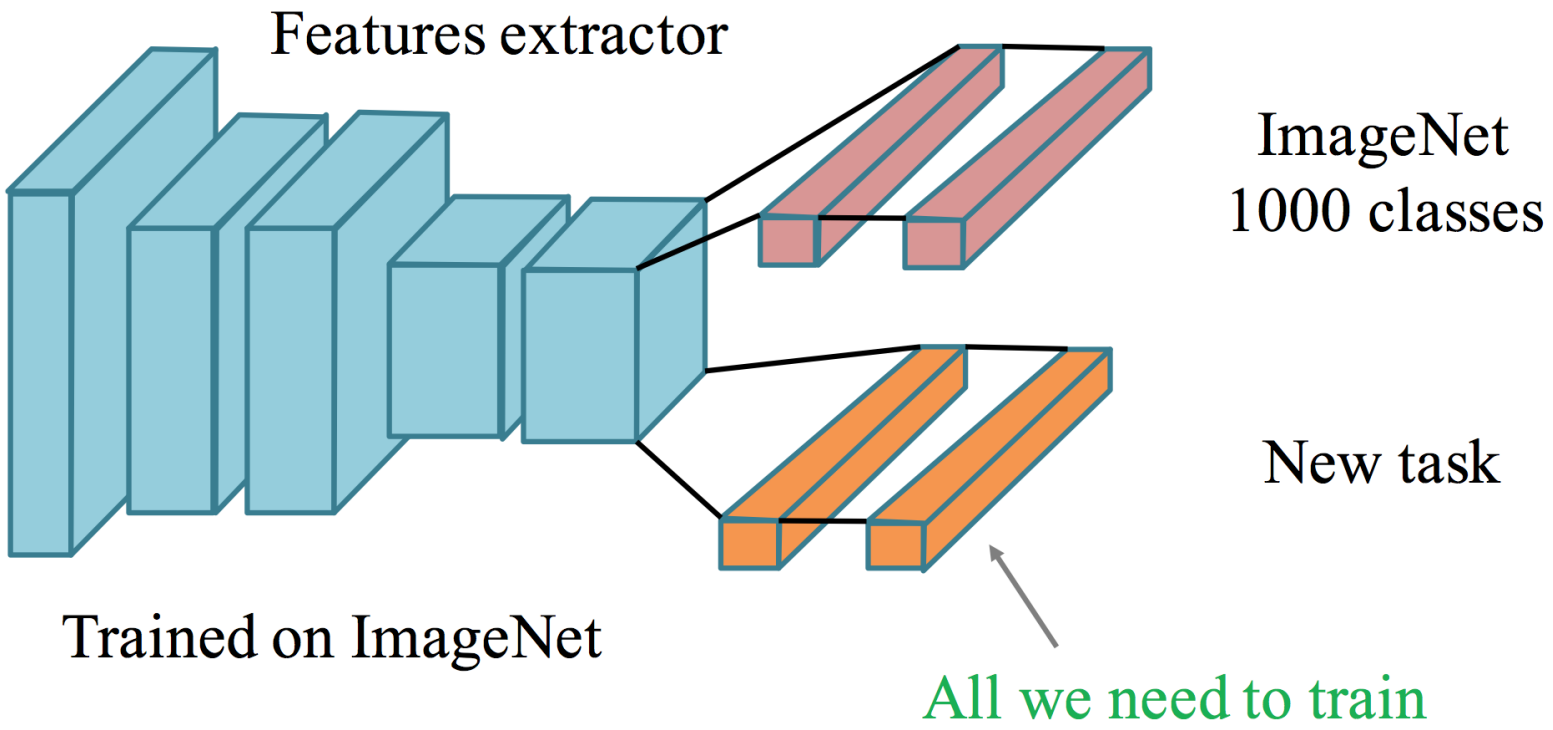
- Мы создаем выходные каналы, добавляя небольшую дельту $F(\mathbf{x})$ ко входным каналам $\mathbf{x}$



- Таким образом мы можем пристыковывать тысячи слоев, и градиенты не затухают благодаря residual связям

Transfer learning

- Глубокие нейросети становятся выучивают сложные представления (feature extraction), но нам необходимо много данных для их обучения!
- Что, если мы сможем переиспользовать существующий извлекатель признаков для новой задачи?

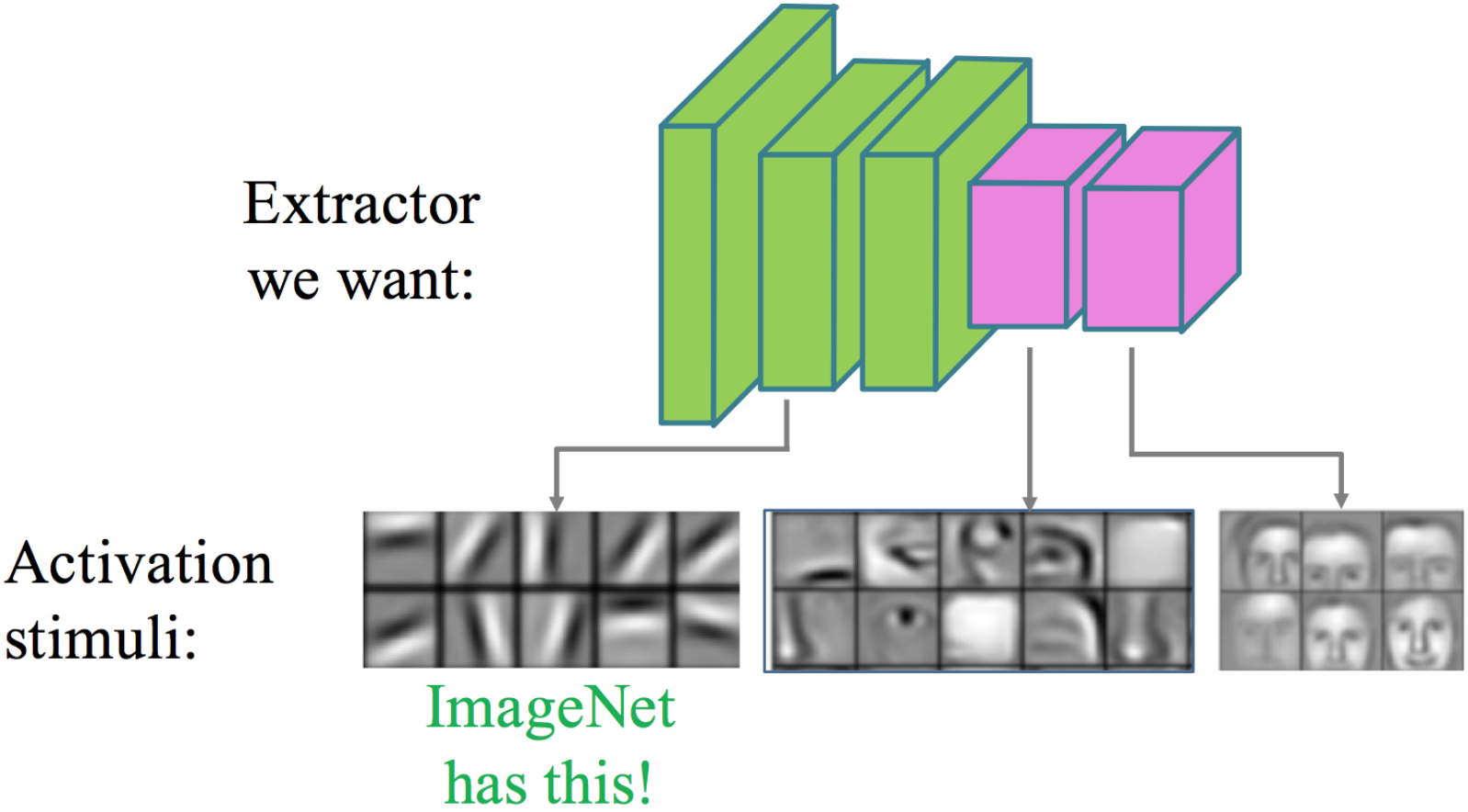


Transfer learning

- Вам нужно меньше данных для обучения (нужны данные только для обучения последнего МП)
- Это работает, если новая область схожа с ImageNet's
- Например, это не будет работать для классификации человеческих эмоций, в ImageNet нет фотографий людей!

Transfer learning

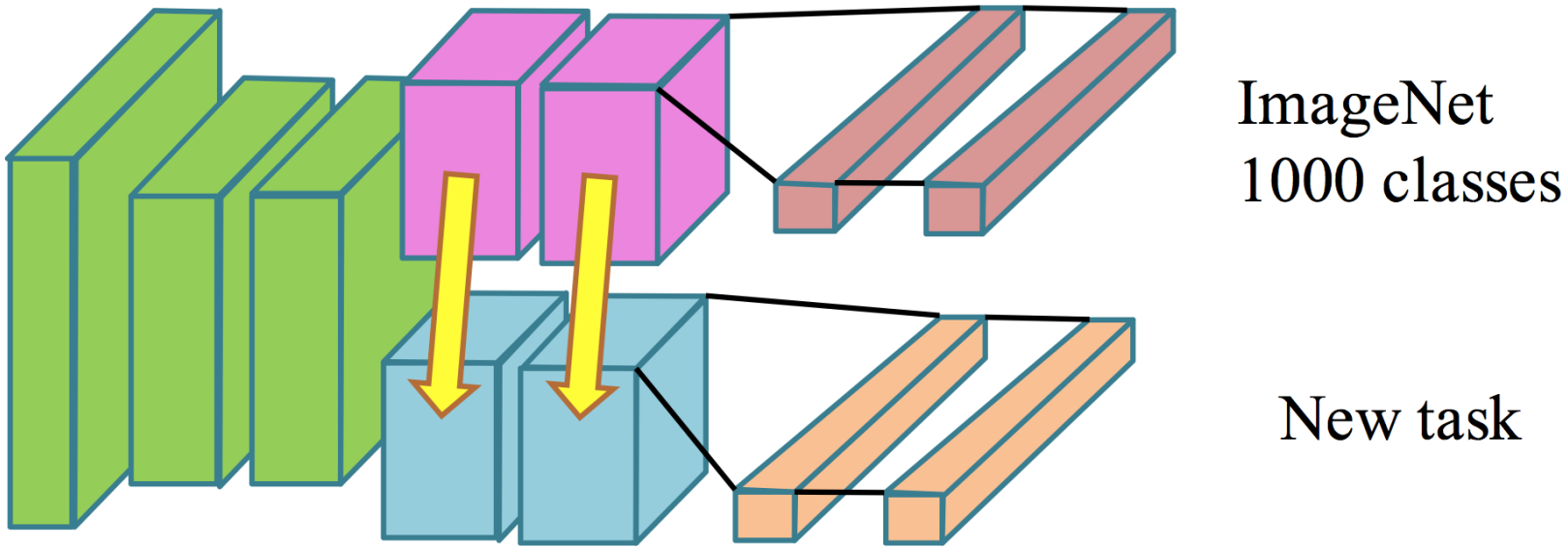
- Но что, если нам нужно классифицировать человеческие эмоции?
- Может мы можем частично переиспользовать извлечение признаков из ImageNet?



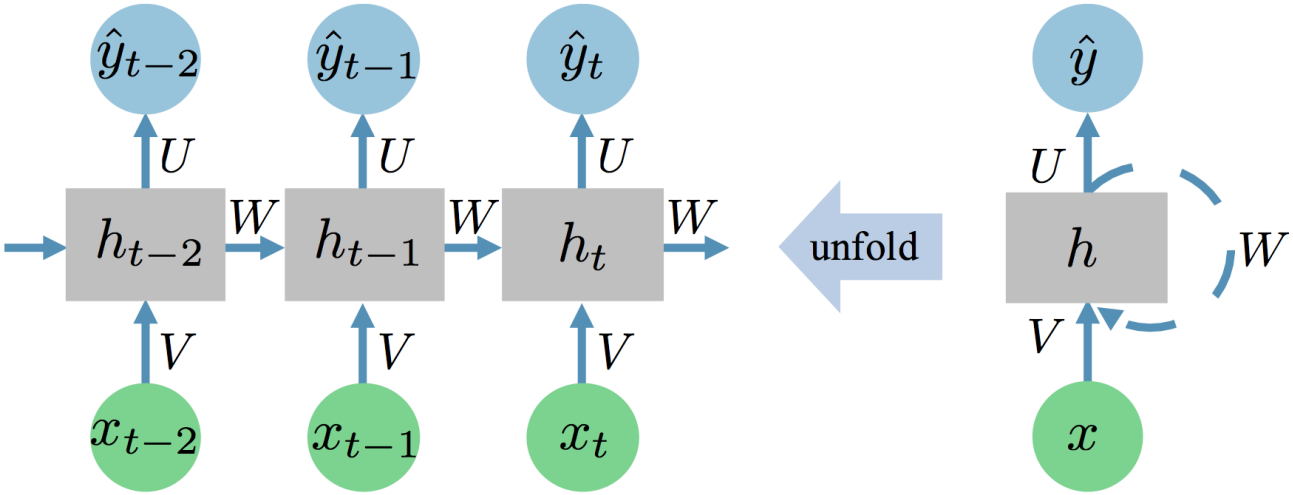
Honglak Lee, http://web.eecs.umich.edu/~honglak/icml09-ConvolutionalDeepBeliefNetworks.pdf

Transfer learning

- Вы можете инициализировать глубокие слои значениями для ImageNet
- Это называется **fine-tuning**, поскольку вы начинаете не со случайной инициализации весов
- Обучаете все веса с меньшим learning rate



Рекуррентные нейронные сети



- $\mathbf{x}$ — вход
- $\widehat{y}$ — выход (предсказание)
- h — скрытое состояние

- $h_t = f_h(V\mathbf{x}_t + Wh_{t-1} + b_h)$
- $\widehat{y}_t = f_y(Uh_t + b_y)$

Заключение

- Простые дифференцируемые вычислительные блоки могут быть объединены в графы и по-прежнему могут быть эффективно обучены
- Сверточные архитектуры достигают производительности, превосходящей человека, в некоторых проблемах компьютерного зрением
- Графы, обученные на больших наборах данных, могут быть повторно использованы для других задач
- Рекуррентные нейронные сети достаточно эффективны для последовательных данных

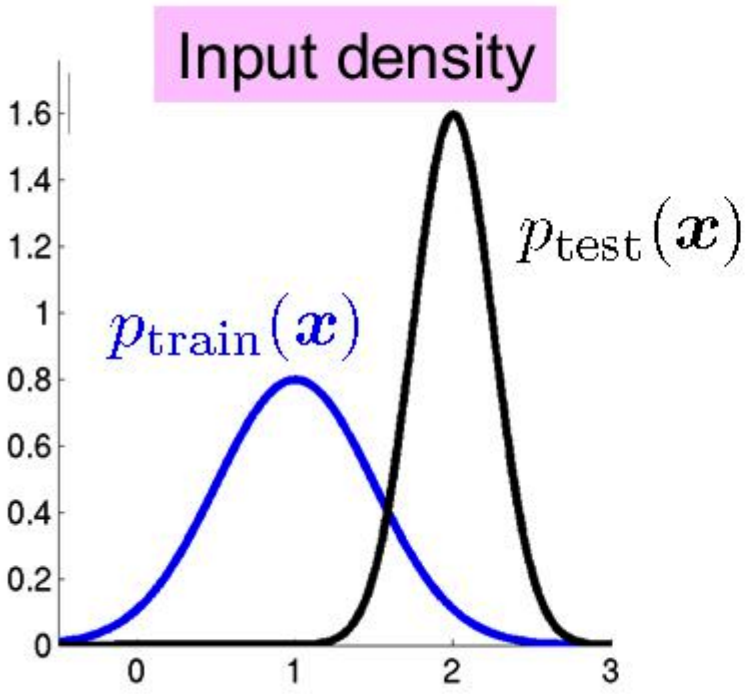
1 Первое поколение Нейронных сетей

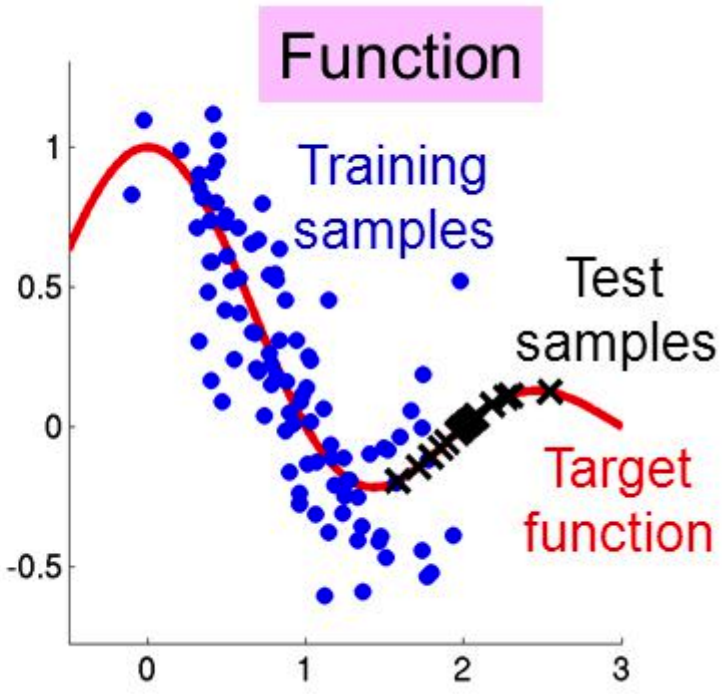
2 Современные нейронные сети

3 Свёрточные нейронные сети

4 Регуляризация нейронных сетей

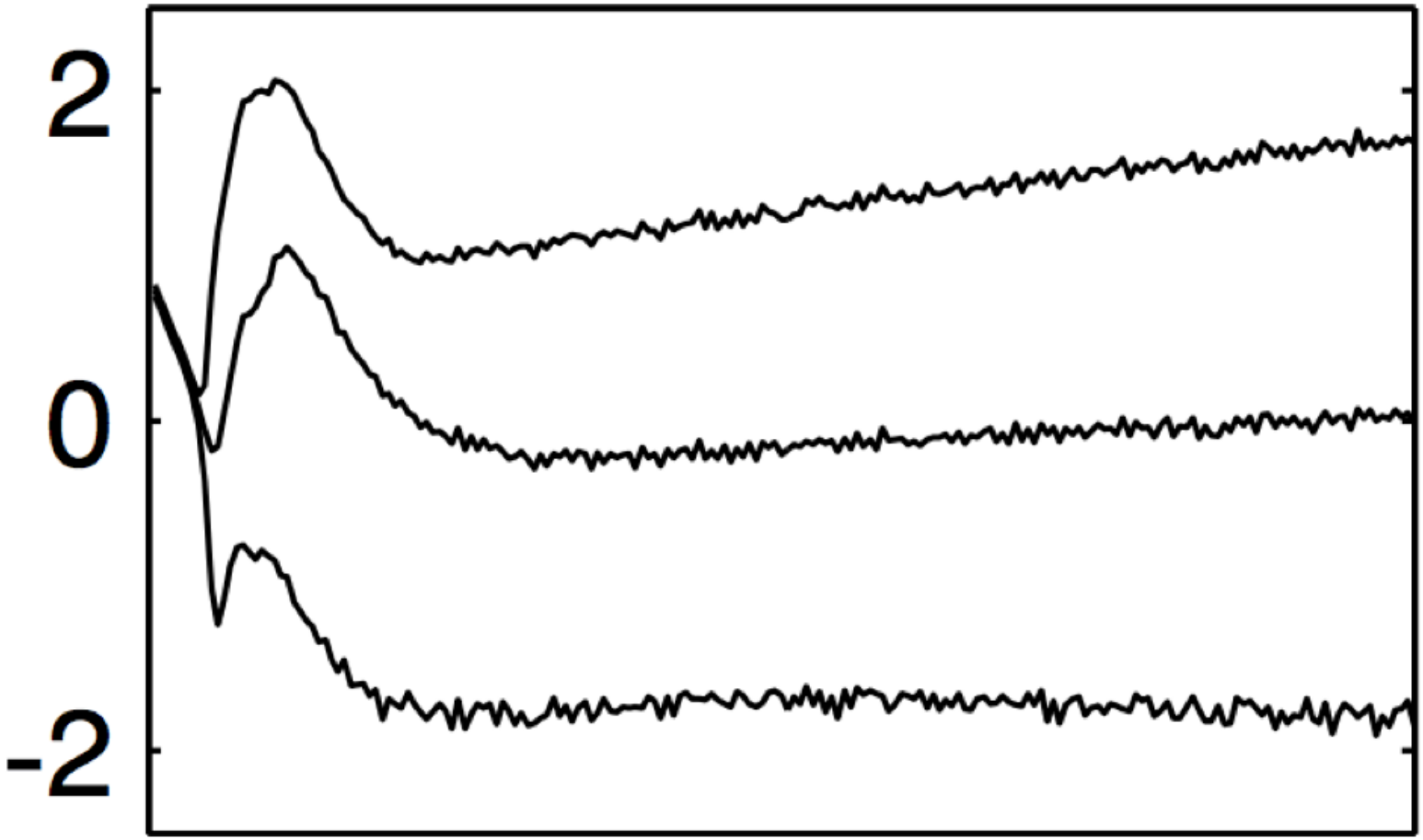
Covariate shift





Входное распределение в нейронной сети

- Входное распределение меняется по мере обучения
- Пример: 3 полносвязных слоя, 100 активаций в каждом
- Сильный внутренний covariate shift



Простая нормализация

- Первая идея: стандартизация выхода слоя
- $f(\mathbf{x}; b) = \mathbf{x} + b - \mathbb{E}(\mathbf{x} + b)$
- Если обратное распространение ошибки игнорирует зависимость среднего в данных от b, тогда после шага $b + g$:

$$f(\mathbf{x}; b) = \mathbf{x} + b + g - \mathbb{E}(\mathbf{x} + b + g) = \mathbf{x} + b - \mathbb{E}(\mathbf{x} + b)$$

- Параметр b может вырасти в бесконечность
- Градиенты должны принимать во внимание нормализацию

Полная нормализация

- Слой нормализации:

$$\widehat{\mathbf{x}} = \mathrm{Norm}(\mathbf{x}, \mathcal{X})$$

- Мы должны уметь считать градиенты:

$$\frac{\partial \mathrm{Norm}(\mathbf{x}, \mathcal{X})}{\partial \mathbf{x}} \quad \text{and} \quad \frac{\partial \mathrm{Norm}(\mathbf{x}, \mathcal{X})}{\partial \mathcal{X}}$$

- Стандартизация требует обратного квадратного корня ковариационной матрицы:

$$\{\mathrm{Cov}(\mathbf{x})\}^{-1/2} (\mathbf{x} - \mathbb{E}\mathbf{x})$$

Упрощения

- Нормализовать активацию каждого слоя независимо:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathbb{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

- Оценка среднего и дисперсии входных данных на основе текущего мини-батча

BatchNorm

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Градиенты для слоя BatchNorm

$$\frac{\partial \ell}{\partial \widehat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum\nolimits_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum\nolimits_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum\nolimits_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum\nolimits_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

BatchNorm во время тестирования

- Оценки среднего и дисперсии по всем батчам обучения:

for $k = 1 \dots K$ **do**
// For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.
Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$\mathrm{E}[x] \leftarrow \mathrm{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\mathrm{Var}[x] \leftarrow \frac{m}{m-1}\mathrm{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

- Используйте эти новые оценки для тестирования:

$$y = \frac{\gamma}{\sqrt{\mathrm{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\mathrm{E}[x]}{\sqrt{\mathrm{Var}[x]+\epsilon}}\right)$$

Использование BatchNorm

- Обычно BatchNorm добавляется до нелинейности
- Learning rate может быть увеличен
- Оптимизация становится робастной к масштабу параметров:

$$\frac{\partial \mathrm{BN}((aW)u)}{\partial u} = \frac{\partial \mathrm{BN}(Wu)}{\partial u}$$
$$\frac{\partial \mathrm{BN}((aW)u)}{\partial (aW)} = \frac{1}{a} \cdot \frac{\partial \mathrm{BN}(Wu)}{\partial W}$$

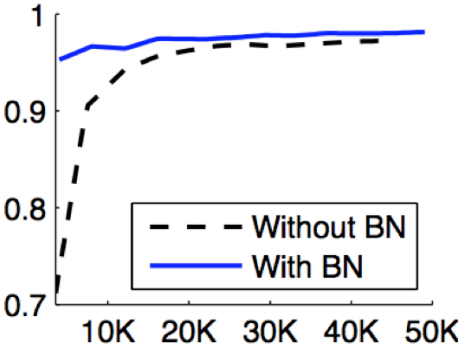
- Dropout слои можно убрать
- Регуляризация весов может быть уменьшена

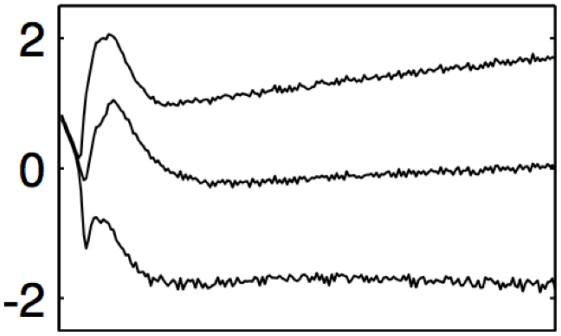
BatchNorm пример

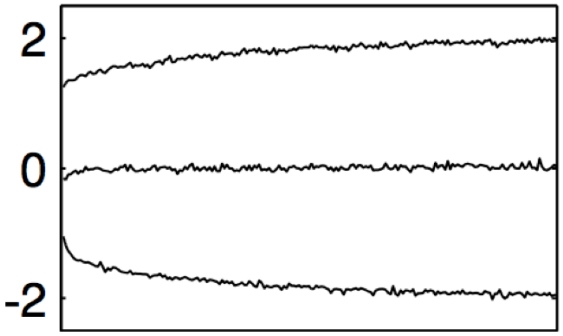
Training BatchNorm and Only BatchNorm: On the Expressive Power of Random Features in CNNs CODE

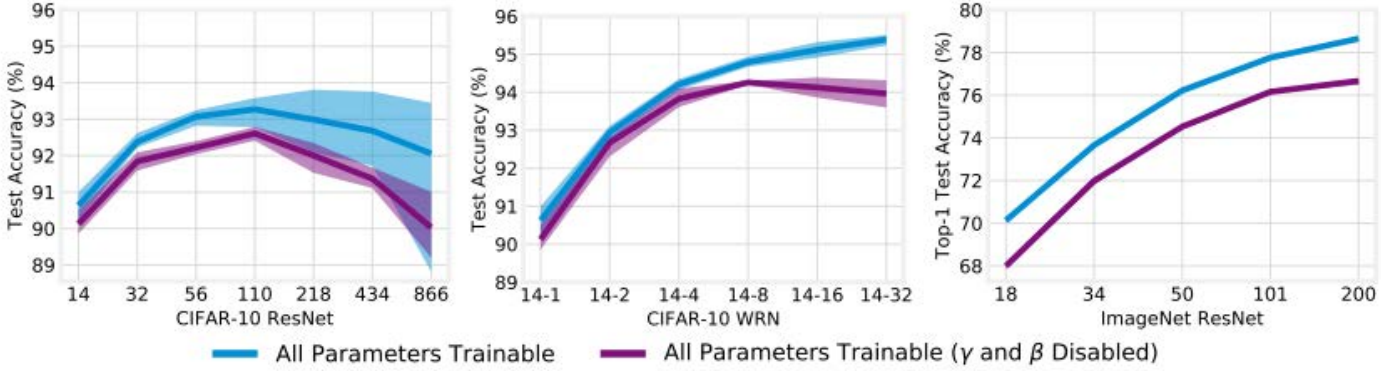
Jonathan Frankle, David J. Schwab, Ari S. Morcos

A wide variety of deep learning techniques from style transfer to multitask learning rely on training affine transformations of features. Most prominent among these is the popular feature normalization technique BatchNorm, which normalizes activations and then subsequently applies a learned affine transform. In this paper, we aim to understand the role and expressive power of affine parameters used to transform features in this way. To isolate the contribution of these parameters from that of the learned features they transform, we investigate the performance achieved when training only these parameters in BatchNorm and freezing all weights at their random initializations. Doing so leads to surprisingly high performance considering the significant limitations that this style of training imposes. For example, sufficiently deep ResNets reach 82% (CIFAR-10) and 32% (ImageNet, top-5) accuracy in this configuration, far higher than when training an equivalent number of randomly chosen parameters elsewhere in the network. BatchNorm achieves this performance in part by naturally learning to disable around a third of the random features. Not only do these results highlight the expressive power of affine parameters in deep learning, but - in a broader sense - they characterize the expressive power of neural networks constructed simply by shifting and rescaling random features.

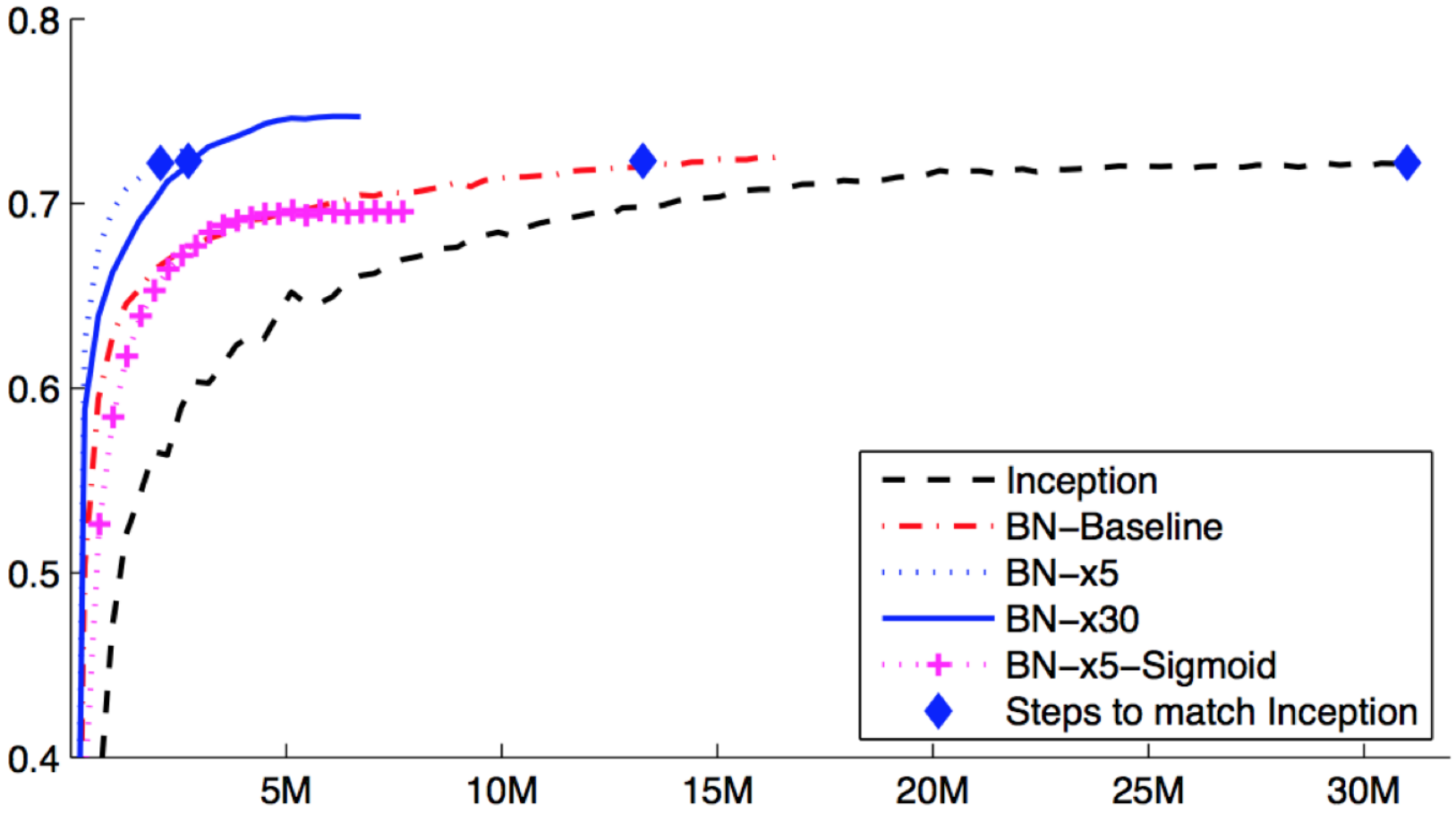








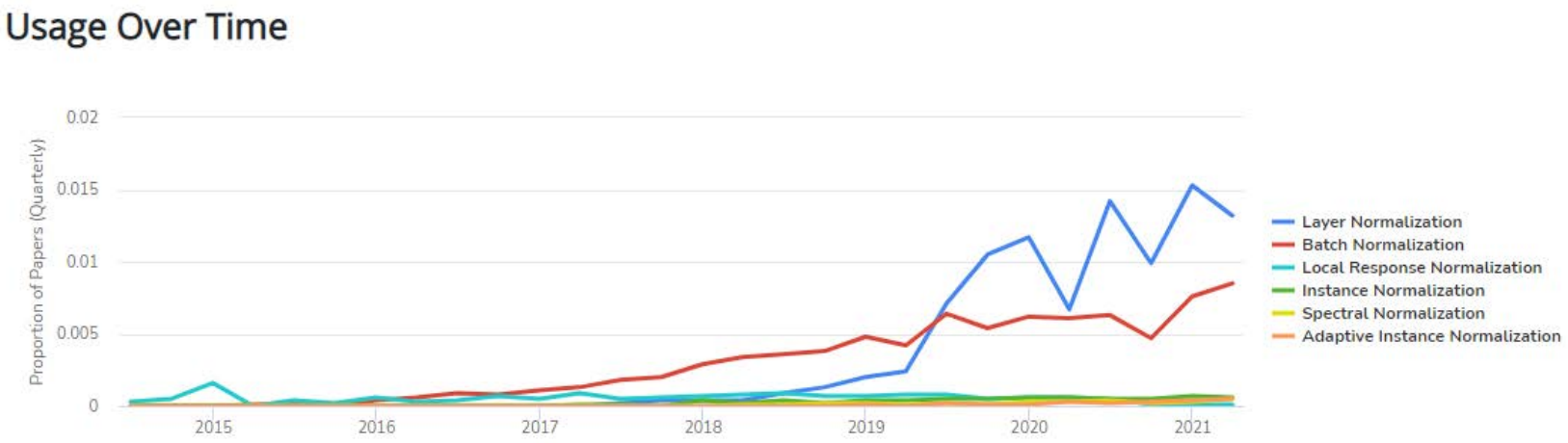
BatchNorm пример



Layer Normalization

- В отличие от Batch Norm, Layer Norm напрямую оценивает статистику нормализации по суммированным входным данным нейронов в скрытом слое, поэтому нормализация не вводит никаких новых зависимостей между обучающими примерами.

- Он хорошо работает для RNN и улучшает как время обучения, так и производительность обобщения нескольких существующих моделей RNN. Активно применяется в моделях типа Transformer.



Layer Normalization

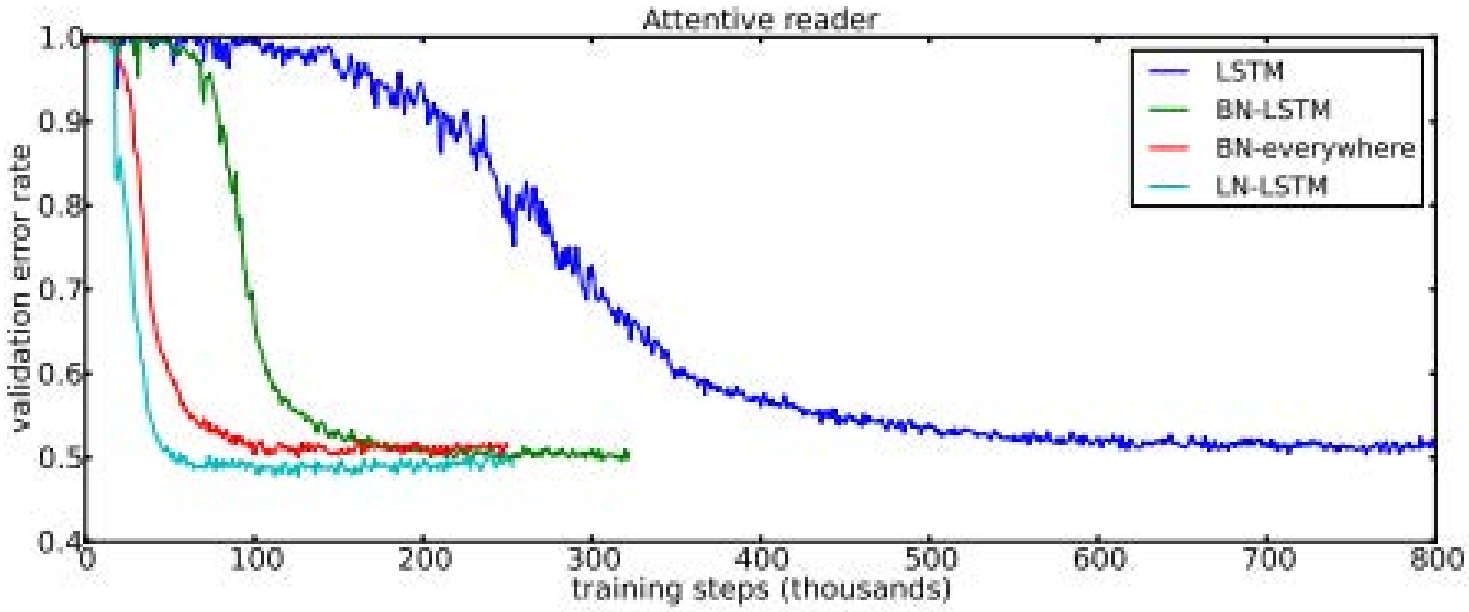
- Мы вычисляем статистику нормализации слоя по всем скрытым юнитам в одном слое следующим образом:

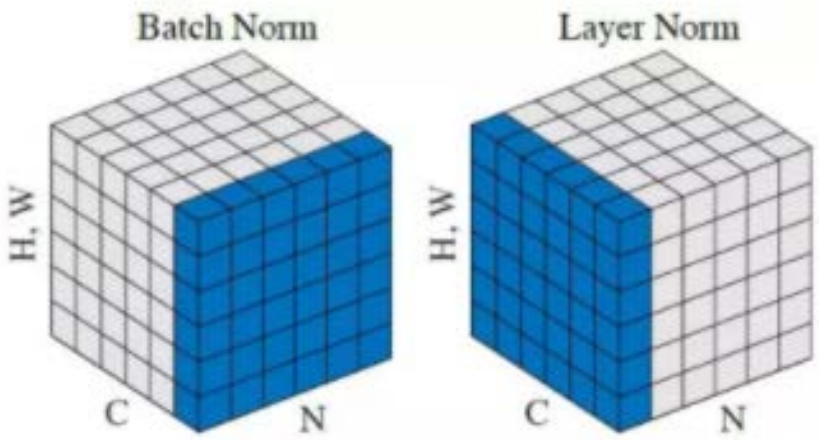
$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l$$

$$\sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2}$$

- где **H** обозначает количество скрытых нейронов в слое. При нормализации уровня все скрытые нейроны в слое используют одни и те же параметры нормализации μ и σ, но разные обучающие примеры имеют разные параметры нормализации. В отличие от Batch Norm, Layer Norm не накладывает никаких ограничений на размер батча, и ее можно использовать в онлайн-обучении с размером пакета 1.

Layer Normalization





$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l$$

$$\sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2}$$

WeightNorm

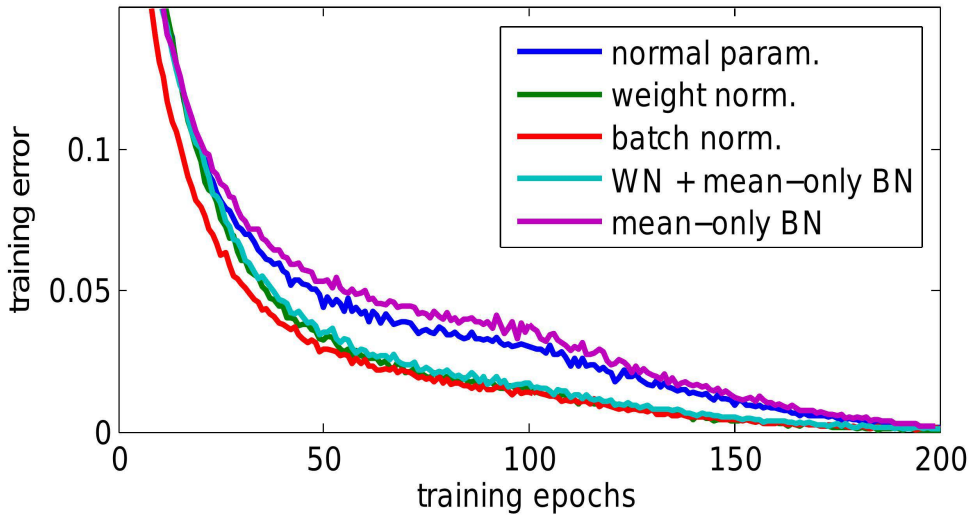
Репараметризация весов:

$$\mathbf{w} = \frac{g}{\|\mathbf{v}\|}\mathbf{v}$$

- Фиксирует Евклидову норму $\mathbf{w}$ на g

  Градиенты:

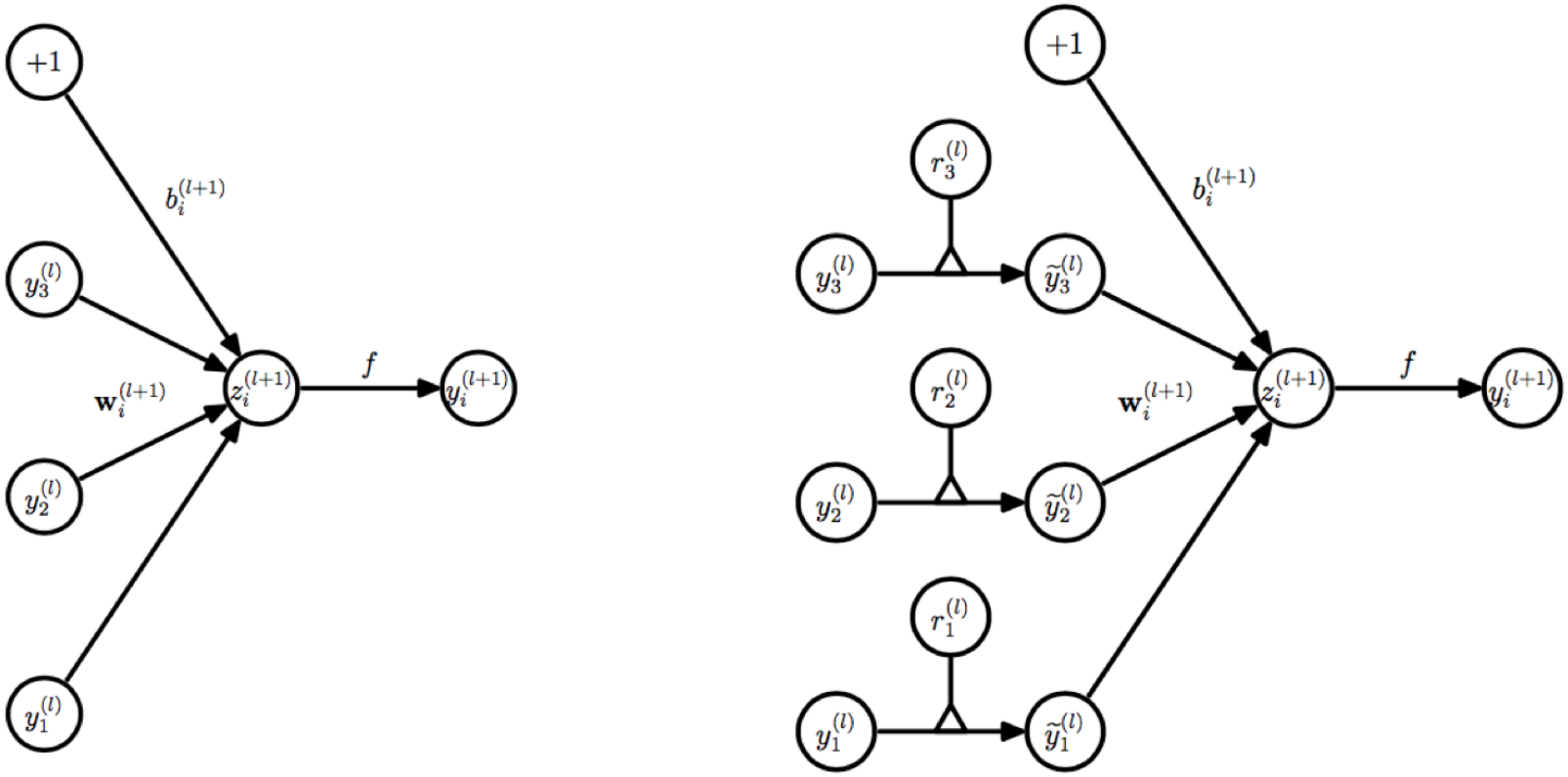
- $$\nabla_g L = \frac{\nabla_{\mathbf{w}} L \cdot \mathbf{v}}{\|\mathbf{v}\|}, \quad \nabla_{\mathbf{v}} L = \frac{g}{\|\mathbf{v}\|}\nabla_{\mathbf{w}} L - \frac{g \nabla_g L}{\|\mathbf{v}\|^2}\mathbf{v}$$
-



WeightNorm результаты на CIFAR-10

Model	Test Error
Maxout [6]	11.68%
Network in Network [17]	10.41%
Deeply Supervised [16]	9.6%
ConvPool-CNN-C [26]	9.31%
ALL-CNN-C [26]	9.08%
our CNN, mean-only B.N.	8.52%
our CNN, weight norm.	8.46%
our CNN, normal param.	8.43%
our CNN, batch norm.	8.05%
ours, W.N. + mean-only B.N.	**7.31%**

Dropout

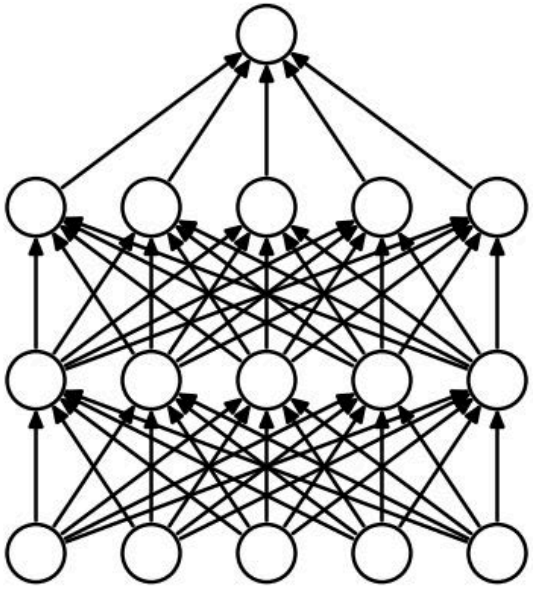

+1
$b_i^{(l+1)}$
$y_3^{(l)}$
$\mathbf{w}_i^{(l+1)}$
$z_i^{(l+1)}$
f
$y_i^{(l+1)}$
$y_2^{(l)}$
$y_1^{(l)}$
+1
$r_3^{(l)}$
$b_i^{(l+1)}$
$y_3^{(l)}$
$\tilde{y}_3^{(l)}$
$r_2^{(l)}$
$\mathbf{w}_i^{(l+1)}$
$z_i^{(l+1)}$
f
$y_i^{(l+1)}$
$y_2^{(l)}$
$\tilde{y}_2^{(l)}$
$r_1^{(l)}$
$y_1^{(l)}$
$\tilde{y}_1^{(l)}$

Dropout

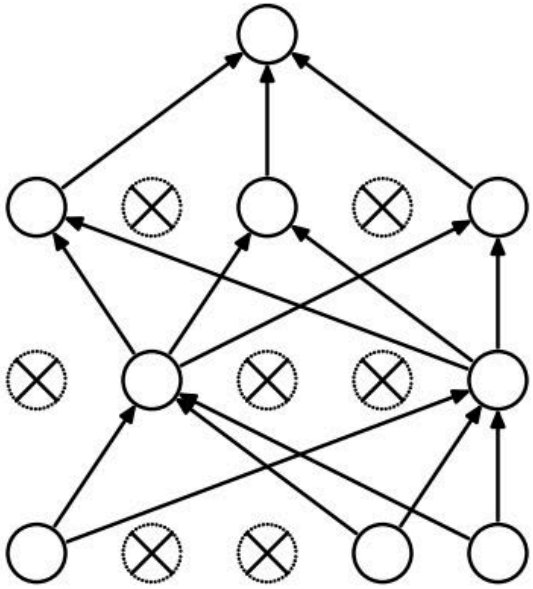
Случайно выключаем (зануляем) выходы слоёв:

$$r_{jk} \sim \text{Bernoulli}(p)$$

- $$v_{j+1}(\mathbf{x}) = f_{j+1}(v_j(\mathbf{x}) \otimes r_j; w_j)$$



(a) Standard Neural Net



(b) After applying dropout.

Dropout

- Предотвращает адаптацию нейронов и делает их более устойчивыми к случайным возмущениям
- Аналогично обучению 2^n моделей с общими весами
- Во время тестирования: умножьте веса dropout слоя на p
- Хорошо работает с регуляризацией максимальной нормы
- Вариационный dropout может выучить отдельный коэффициент dropout для каждого выхода

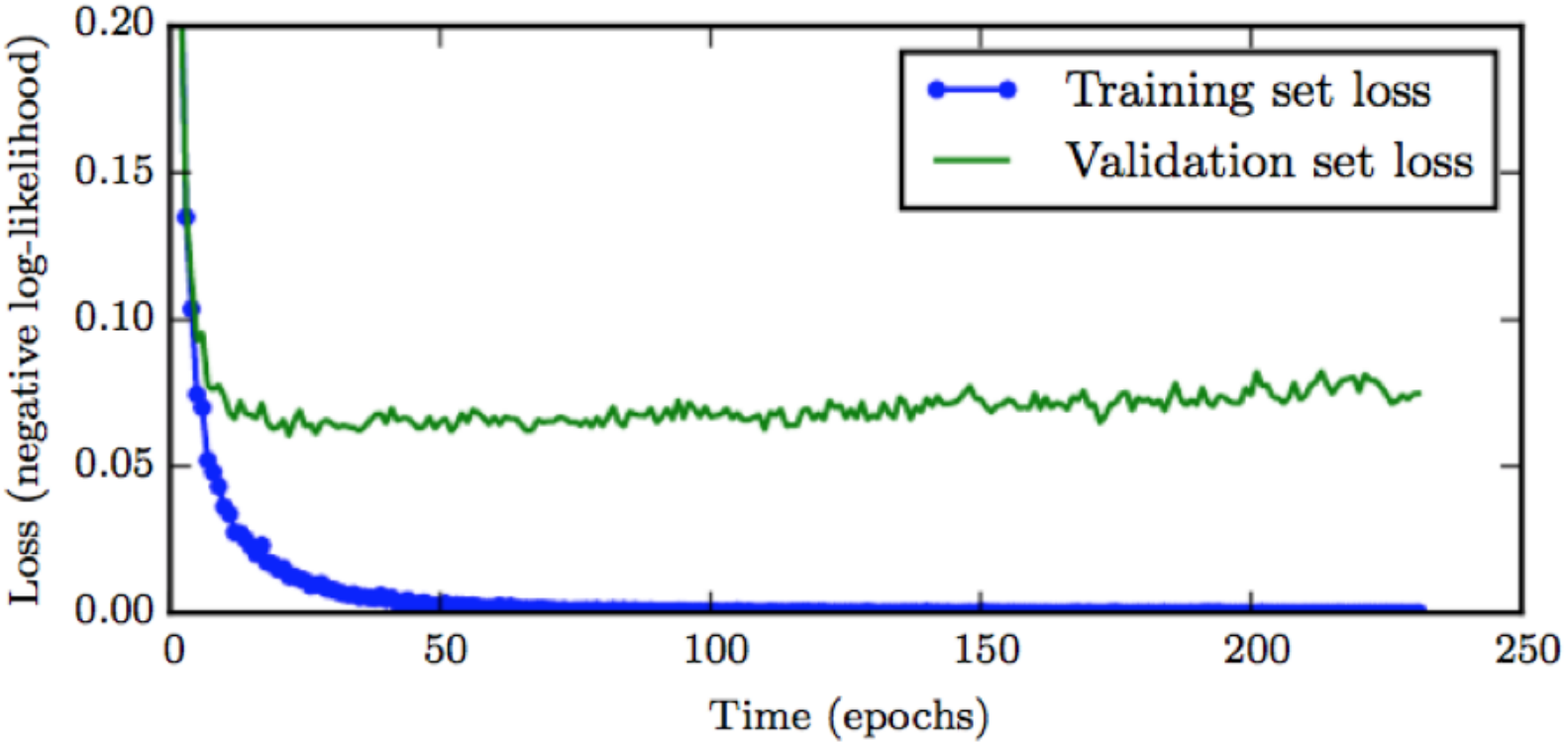
Регуляризация весов

- L_2 регуляризация: добавляет $\lambda\|\mathbf{w}\|^2$ к функции потерь
- Штрафует пиковые веса векторов
- Предотвращает фокусировку нейронов на одном сильном входе
- Похоже на dropout
- Wager, Wang, Liang. Dropout Training as Adaptive Regularization:

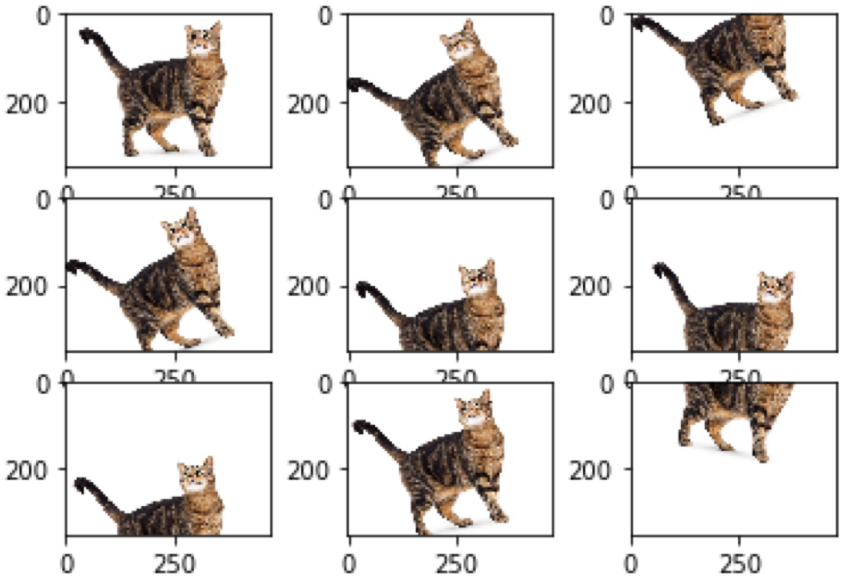
  "We show that the dropout regularizer is first-order equivalent to an L2 regularizer applied after scaling the features by an estimate of the inverse diagonal Fisher information matrix"

Ранняя остановка

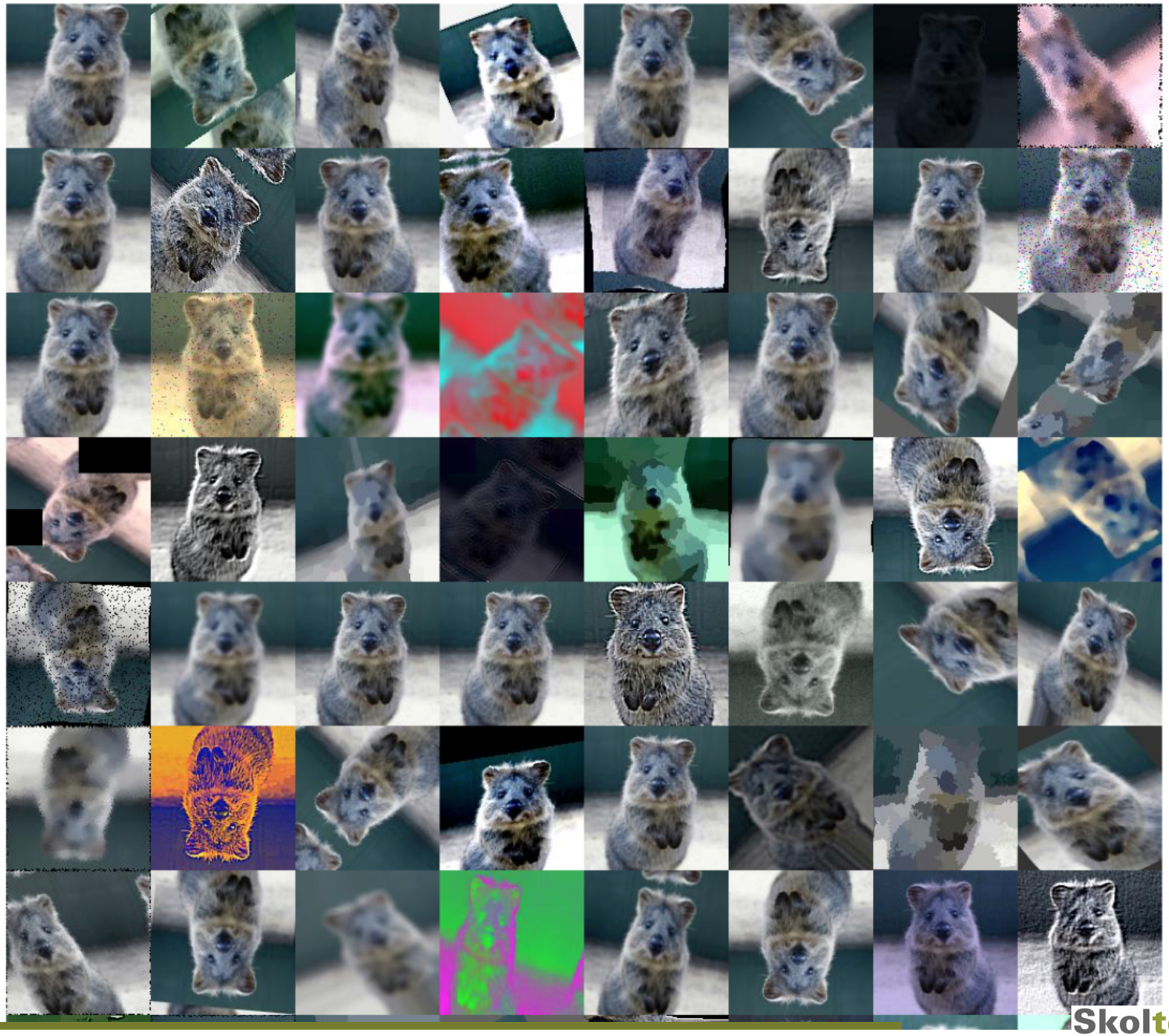
- Выберите количество шагов оптимизации на основе качества проверки
- Когда оптимальное количество шагов известно, можно обучить заново на данных обучения и валидации
- Для линейной модели с MSE loss и SGD оптимизацией, ранняя остановка эквивалентна L_2 регуляризации



Аугментация данных



Аугментация данных



Аугментация данных для изображений

- Сдвиг
- Приближение / отдаление
- Повороты
- Отражения
- Добавление шума
- Тени

Аугментация данных

- Можно даже использовать Style Transfer

Аугментация данных для аудио

- Растяжение времени
- Сдвиг высоты тона
- Сжатие динамического диапазона
- Фоновый шум
- ...

Time stretching

- Увеличивает робастность модели
- Полезно, когда мало тренировочных примеров
- Так же улучшает качество для больших датасетов

Выводы

- Нормализация улучшает производительность как с точки зрения скорости, так и качества
- Регуляризация (dropout, max-norm, L_2) предотвращает переобучение
- Аугментация данных полезна для улучшения генерализации
- Существуют другие методы регуляризации: обучение на нескольких задачах, соперничающие примеры, разреженность и т.д.