

Основы глубинного обучения

Лекция 6

Регуляризация в глубинных моделях. Архитектуры свёрточных сетей.

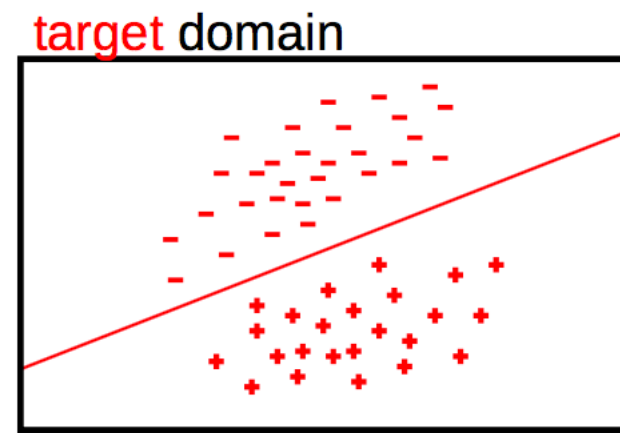
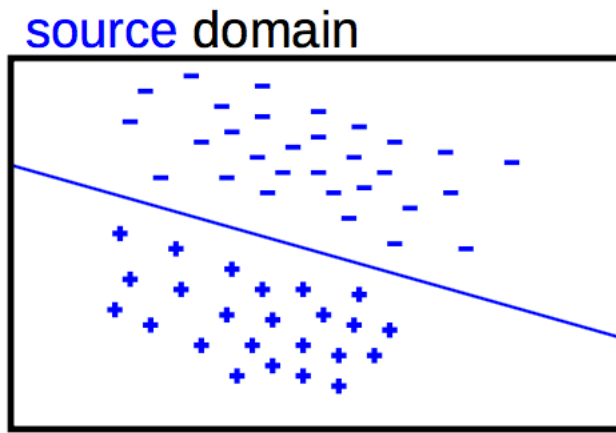
Евгений Соколов

esokolov@hse.ru

НИУ ВШЭ, 2022

Нормализации

Covariate shift



Covariate shift

- В классическом машинном обучении — изменение распределения данных
- Много методов решения

Domain adaptation

- Объекты по-разному распределены на обучении и на контроле
- Идея: взвешивать объекты при обучении

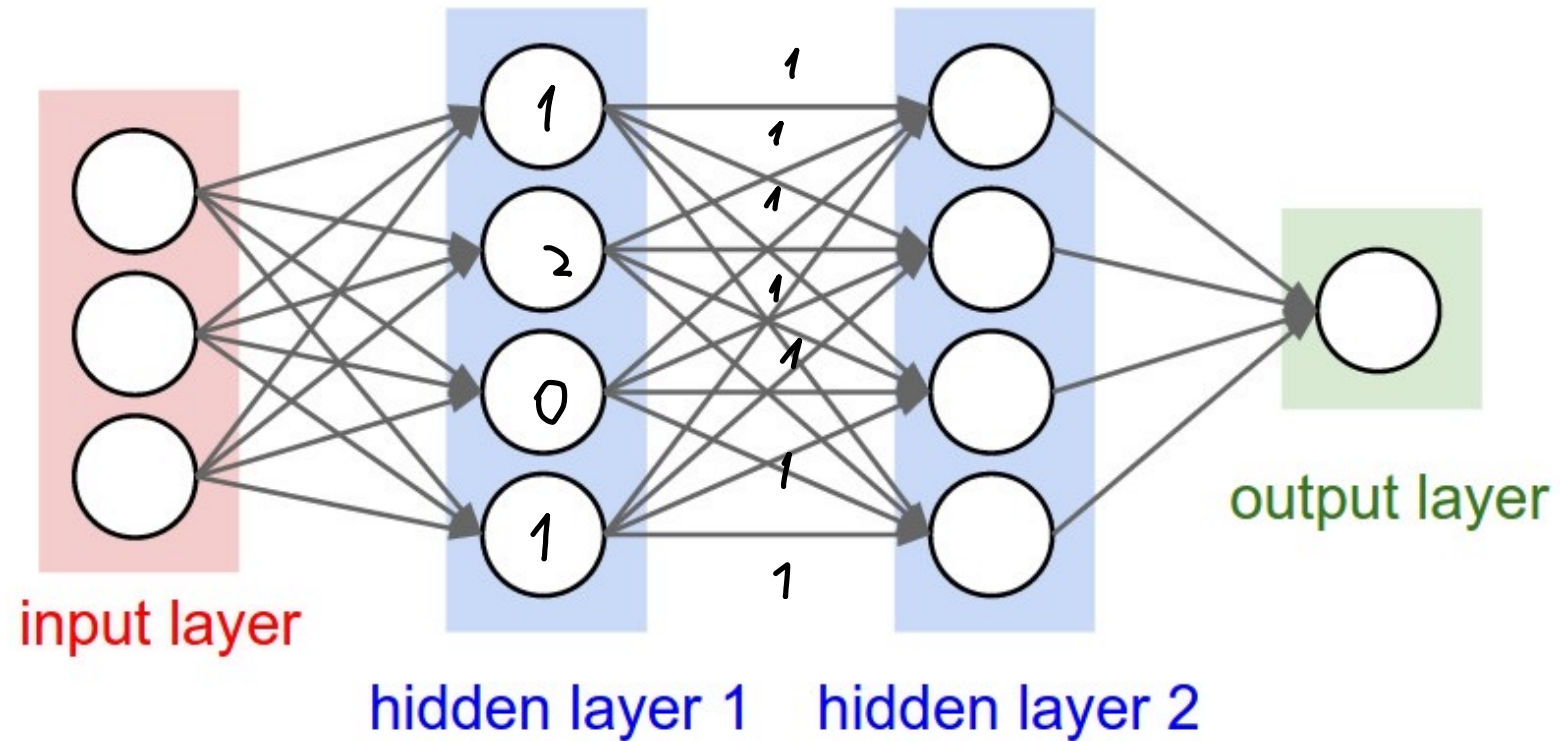
$$\sum_{i=1}^{\ell} s_i (a(x_i) - y_i)^2 \rightarrow \min$$

- Большие веса будем ставить объектам, которые похожи на объекты из тестовой выборки

Internal covariate shift

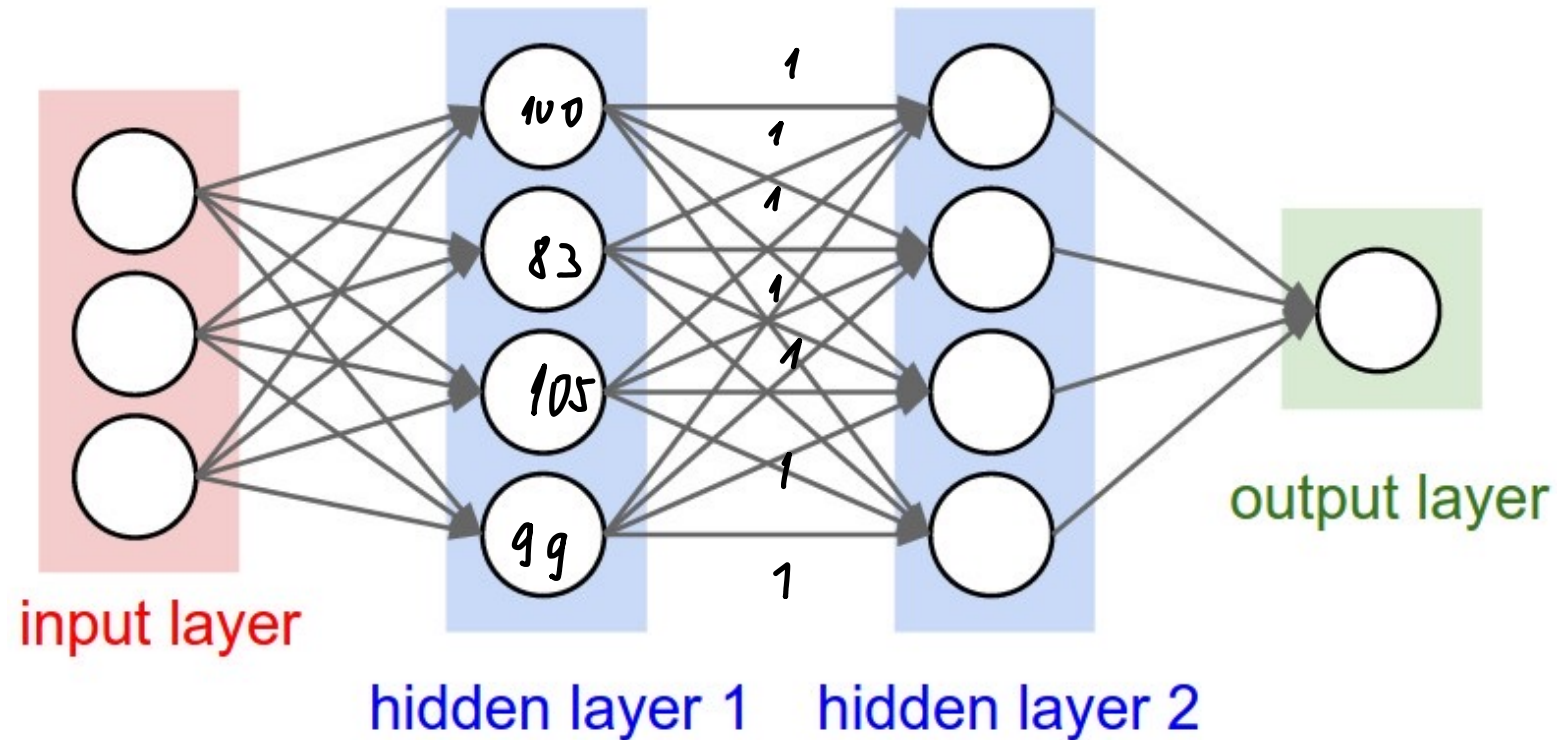
- В нейронной сети каждый слой обучается на выходах предыдущих слоёв
- Если слой в начале сильно меняется, то все следующие слои надо переделывать

Internal covariate shift



Internal covariate shift

Допустим, веса первого слоя сильно поменялись после градиентного шага



Internal covariate shift

- Идея: преобразовывать выходы слоёв так, чтобы они гарантированно имели фиксированное распределение


Batch Normalization

- Реализуется как отдельный слой
- Вычисляется для текущего батча
- Оценим среднее и дисперсию каждой компоненты входного вектора:

$$\mu_B = \frac{1}{n} \sum_{j=1}^n x_{B,j}$$

$$\sigma_B^2 = \frac{1}{n} \sum_{j=1}^n (x_{B,j} - \mu_B)^2$$

покоординатно



$x_{B,j}$ — j -й объект в батче B

Batch Normalization

- Отмасштабируем все выходы:

$$\tilde{x}_{B,j} = \frac{x_{B,j} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

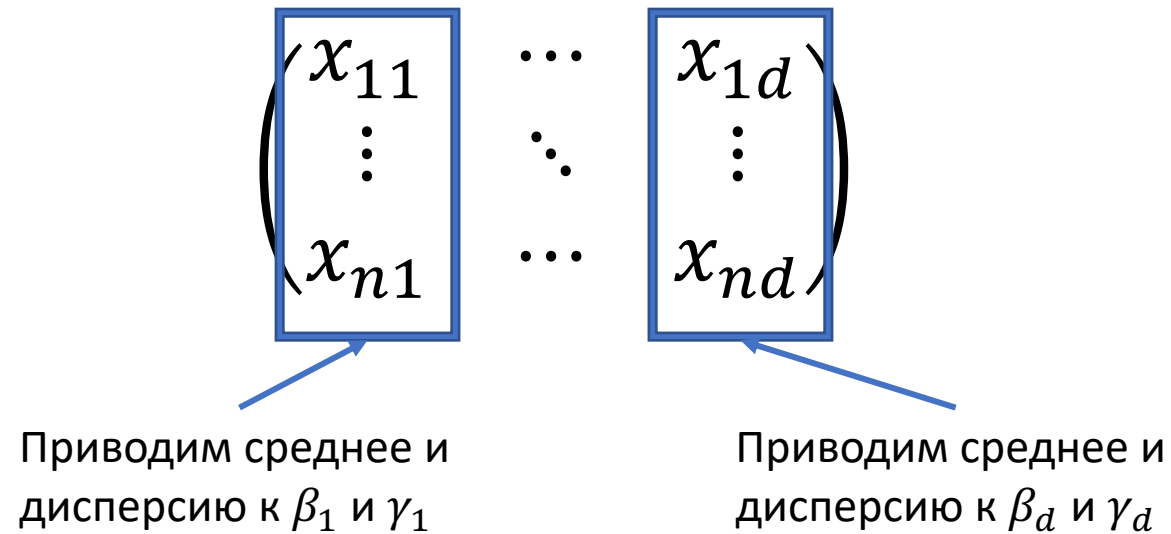
- Зададим нужные нам среднее и дисперсию:

$$z_{B,j} = \gamma \circ \tilde{x}_{B,j} + \beta$$



обучаемые параметры (векторы, размерность
равна размерности входных векторов)

Batch Normalization



- n — размер батча
- d — размерность входного вектора

Batch Normalization

Важно: после BatchNorm среднее и дисперсия каждого выхода зависят только от параметров нормализации, но не от параметров прошлых слоёв!

Batch Normalization

Во время применения нейронной сети:

- Те же самые формулы, но вместо μ_B и σ_B^2 используем их средние значения по всем батчам

Batch Normalization

- Обычно вставляется между полносвязным/свёрточным слоём и нелинейностью
- Позволяет увеличить длину шага в градиентном спуске
- Не факт, что действительно устраняет covariance shift

В чём польза от BatchNorm?

How Does Batch Normalization Help Optimization?

Shibani Santurkar*
MIT
shibani@mit.edu

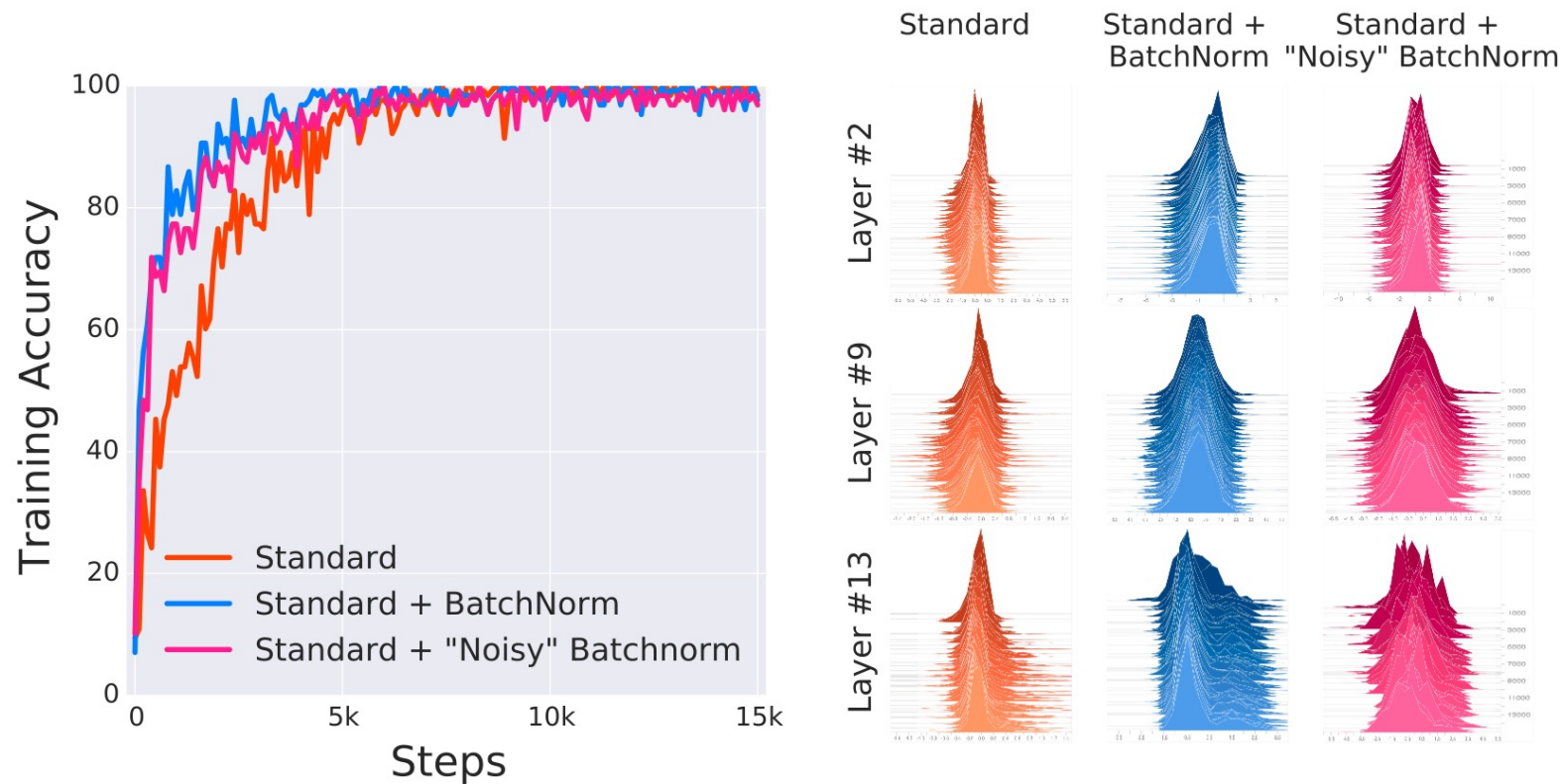
Dimitris Tsipras*
MIT
tsipras@mit.edu

Andrew Ilyas*
MIT
ailyas@mit.edu

Aleksander Madry
MIT
madry@mit.edu

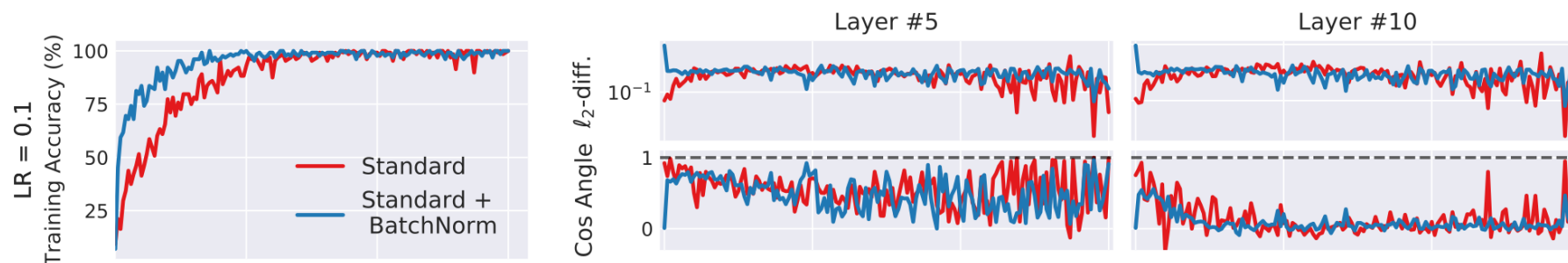
В чём польза от BatchNorm?

- Добавим шум после нормализации — хуже не становится!

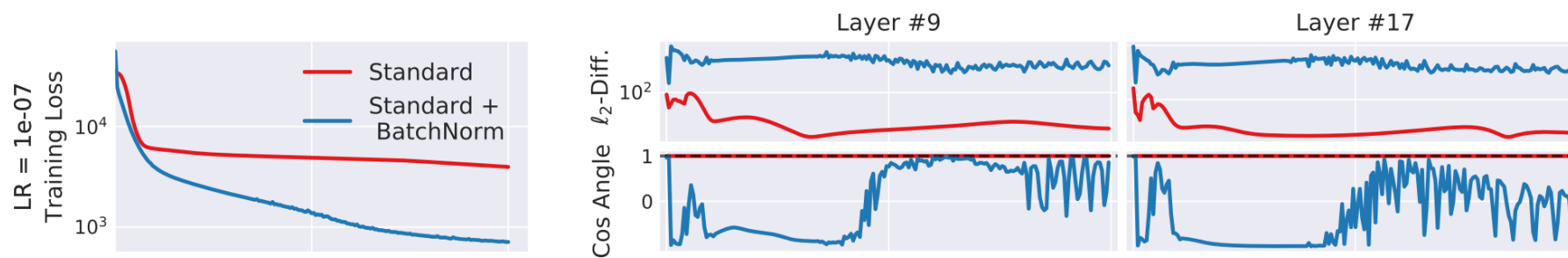


В чём польза от BatchNorm?

- Как связаны градиенты на соседних итерациях?



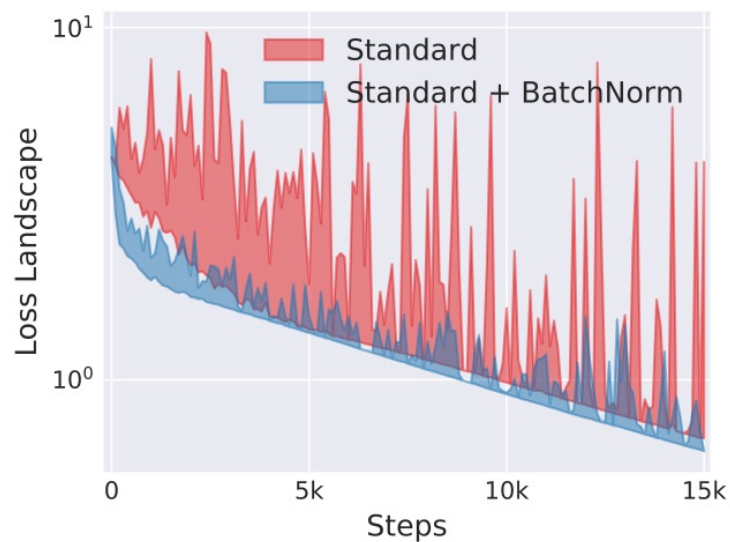
(a) VGG



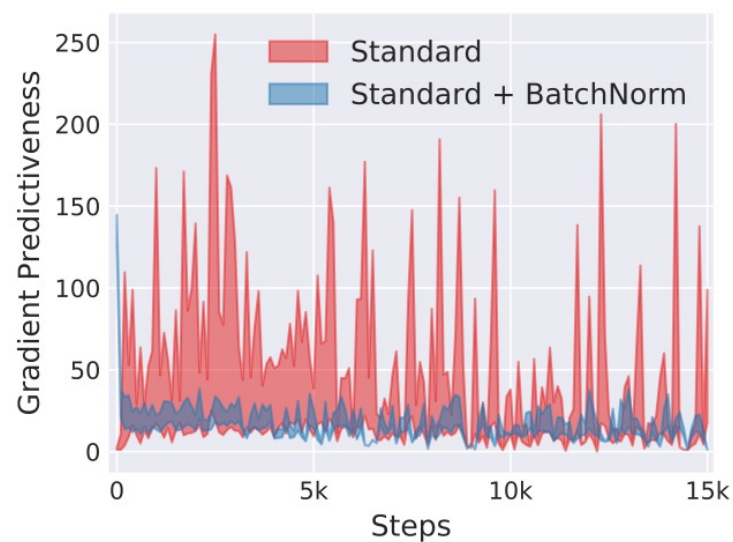
(b) DLN

В чём польза от BatchNorm?

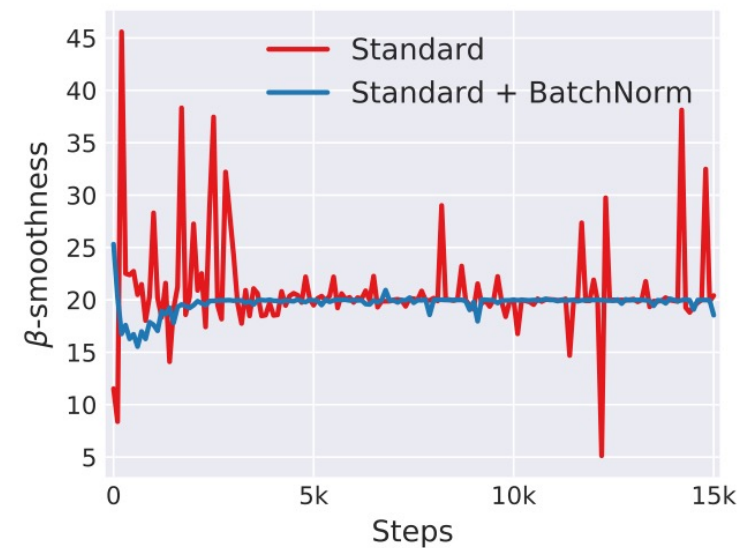
- Функционал ошибки становится более «гладким»!



(a) loss landscape

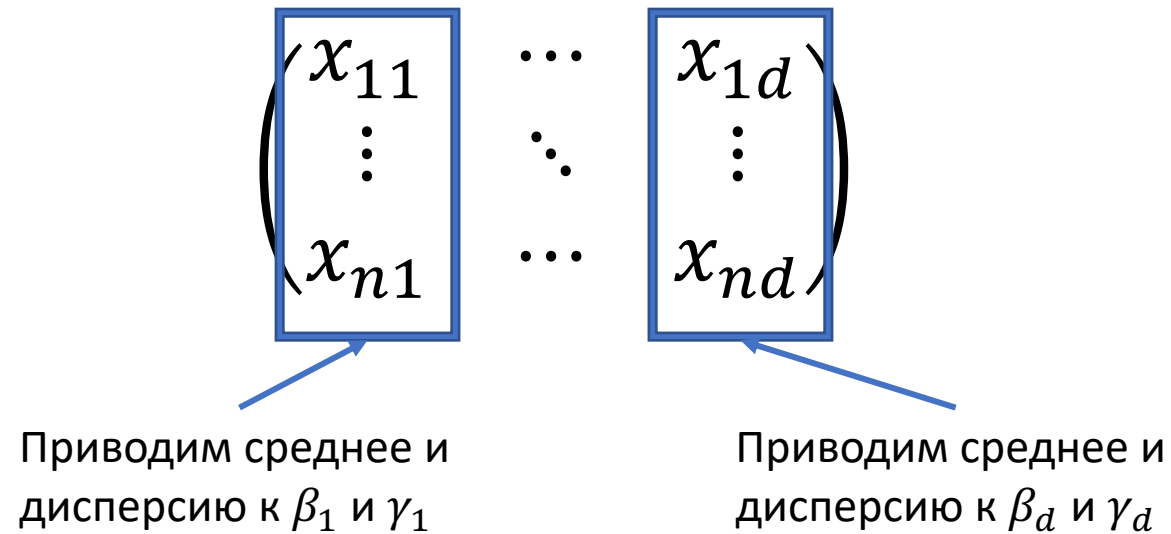


(b) gradient predictiveness



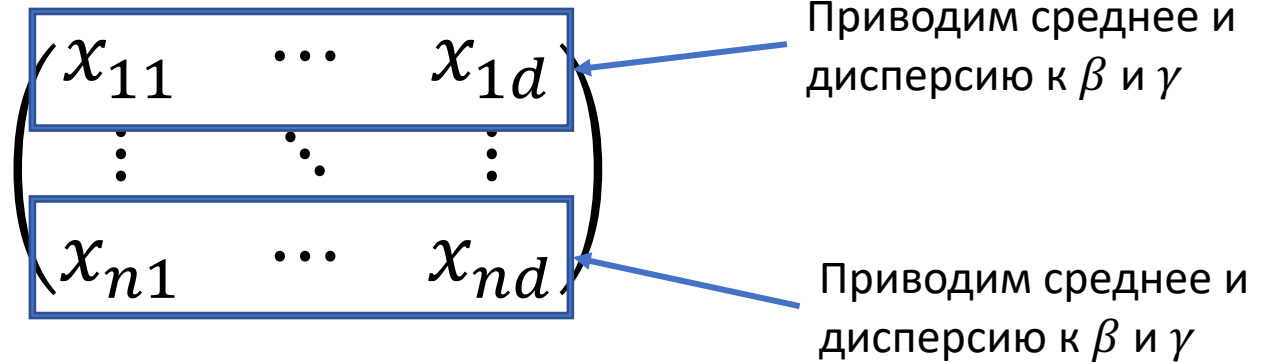
(c) “effective” β -smoothness

Batch Normalization



- n — размер батча
- d — размерность входного вектора

Layer Normalization



- n — размер батча
- d — размерность входного вектора

Layer Normalization

- Нормализуем распределение «признаков» одного объекта

$$\mu_i = \frac{1}{d} \sum_{j=1}^d x_{ij}$$

$$\sigma_i^2 = \frac{1}{d} \sum_{j=1}^d (x_{ij} - \mu)^2$$

x_{ij} — j -й признак i -го объекта

Layer Normalization

- Отмасштабируем все выходы:

$$\tilde{x}_{ij} = \frac{x_{ij} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$

- Зададим нужные нам среднее и дисперсию:

$$z_{ij} = \gamma \circ \tilde{x}_{ij} + \beta$$

↑ ↑
обучаемые параметры (скаляры)

Инициализации

Инициализация весов

- Не должно быть симметрий (плохо инициализировать всё одним числом)
- Хороший вариант:

$$w_j \sim \frac{2}{\sqrt{n}} \mathcal{N}(0, 1)$$

n — число входов

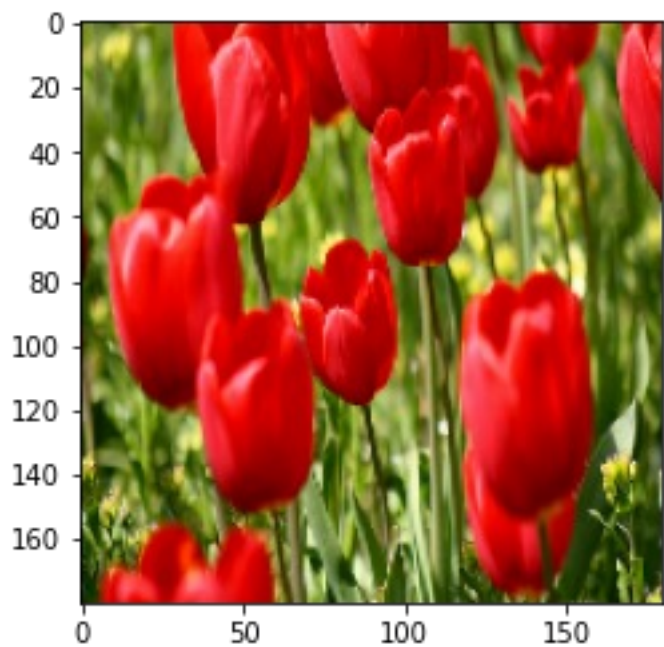
- Пытаемся сделать так, чтобы масштаб всех выходов был примерно одинаковым

Аугментация

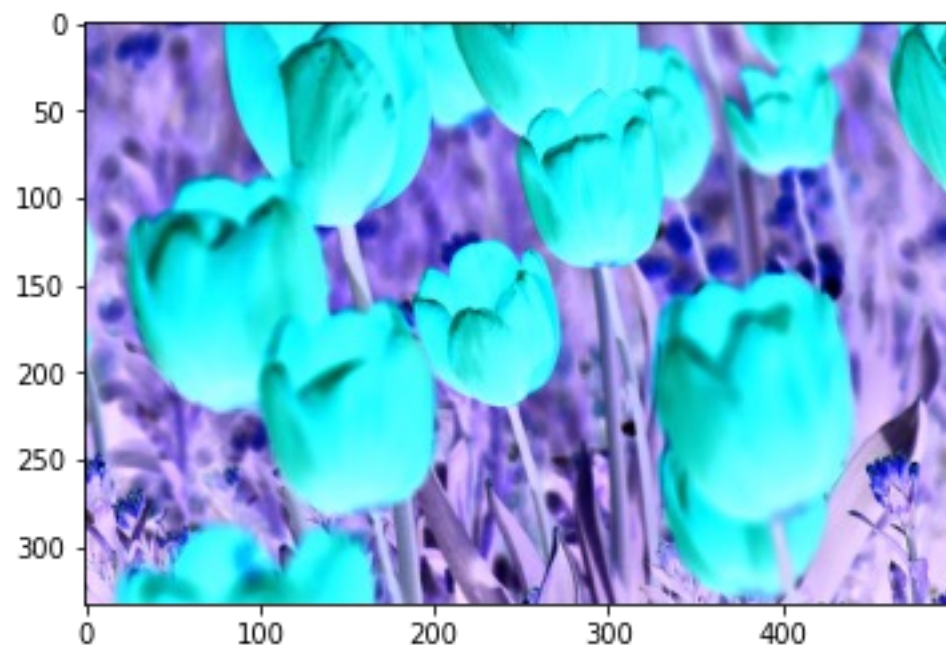
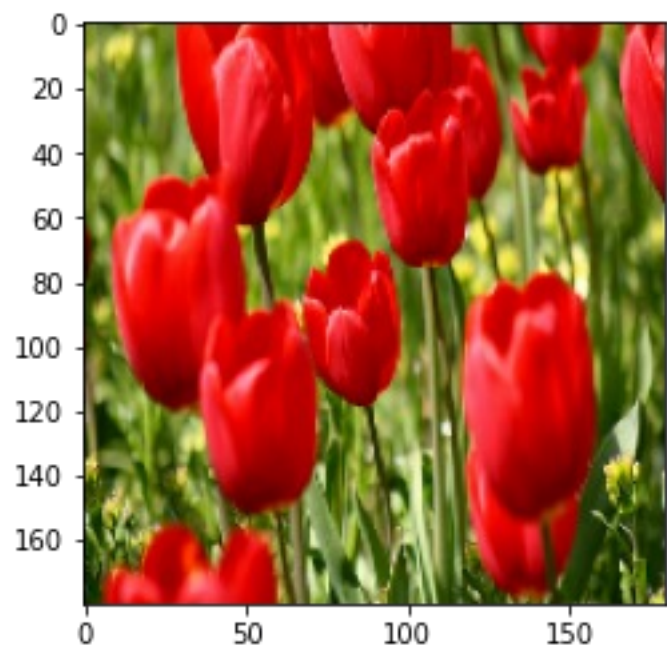
Аугментация



Аугментация



Аугментация



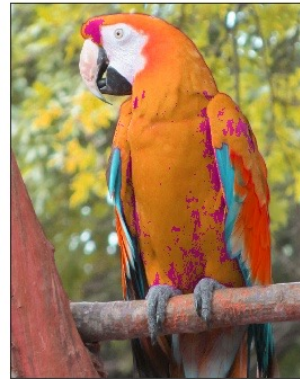
Original image



RGBShift



HueSaturationValue



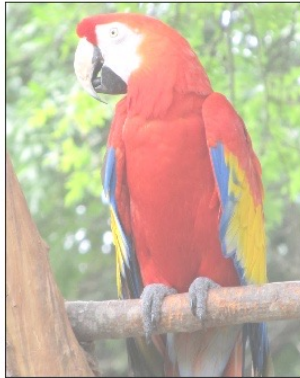
ChannelShuffle



CLAHE



RandomContrast



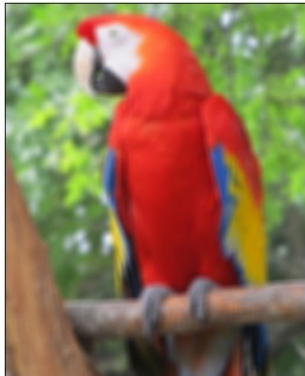
RandomGamma



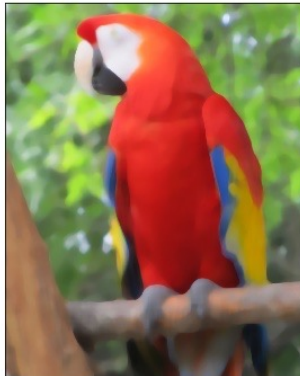
RandomBrightness



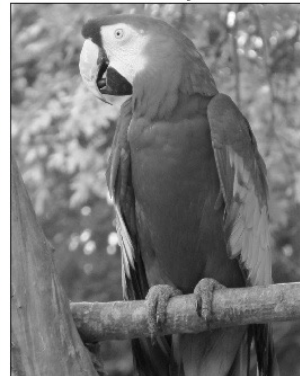
Blur



MedianBlur



ToGray



JpegCompression

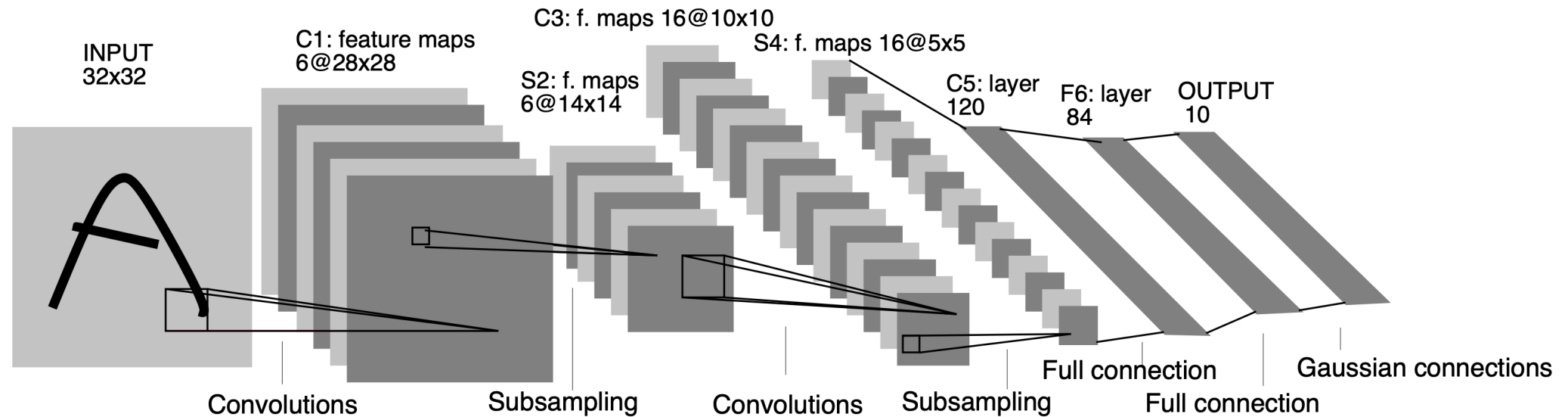


Аугментация

- Много разных вариантов
- «Бесплатное» расширение обучающей выборки
- В некотором смысле регуляризация модели
- Обычно аугментации случайно применяют к картинкам из текущего батча
- На этапе применения можно сделать несколько аугментаций картинки, применить сеть к каждой, усреднить предсказания

Архитектуры свёрточных сетей

LeNet (1998)



LeNet (1998)

- Для данных MNIST
- Идея end-to-end обучения
- Использовали аугментацию
- Около 60.000 параметров
- Доля ошибок на тесте 0.8%

ImageNet



- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- Около 1.000.000 изображений
- 1000 классов
- Обычно качество измерялось на основе лучшей гипотезы модели

AlexNet (2012)

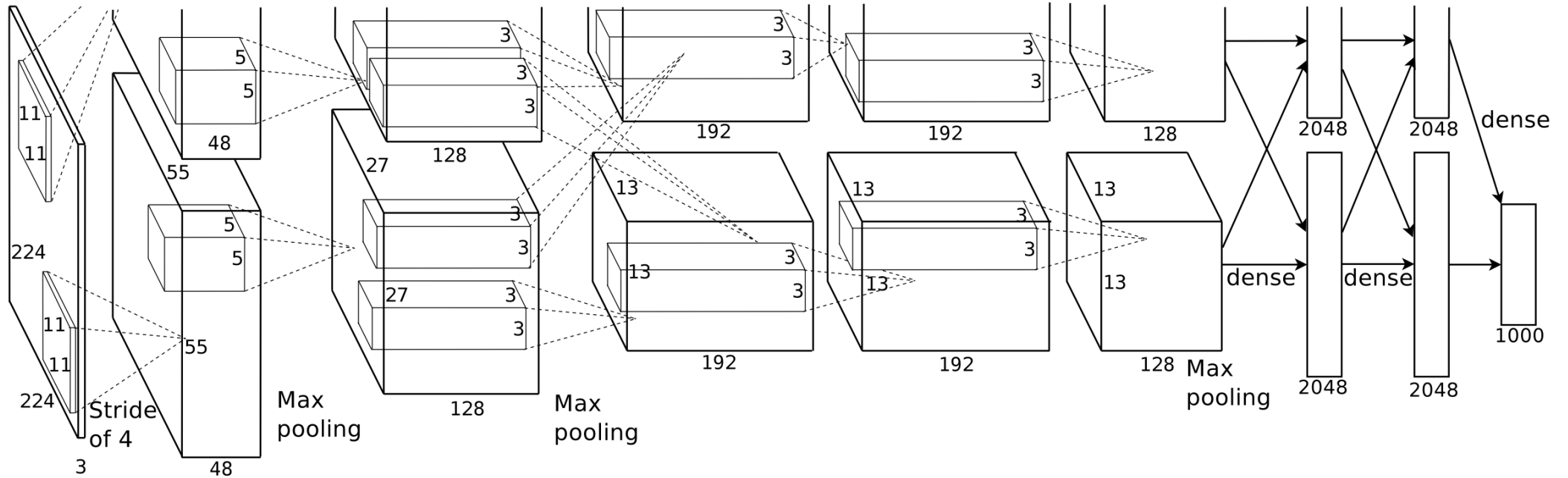
ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

AlexNet (2012)



AlexNet (2012)

- Используют ReLU, аугментацию, dropout
 - Градиентный спуск с инерцией (momentum)
 - Обучение на двух GPU (5-6 суток)
 - Около 60 миллионов параметров
-
- Ошибка около 17%