



AI Community

10. Тренировка нейронных сетей

Transfer Learning



“Вам нужно очень много данных, чтобы обучить и использовать свёрточные нейронные сети (CNN)”

“Вам нужно очень много данных, чтобы обучить
и использовать свёрточные нейронные сети”

BUSTED

“Transfer Learning” с нейронными сетями



1. Обучаем на Imagenet

FC-1000
FC-4096
FC-4096

MaxPool
Conv-512
Conv-512

MaxPool
Conv-512
Conv-512

MaxPool
Conv-256
Conv-256

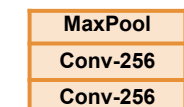
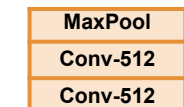
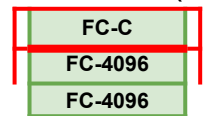
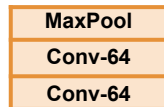
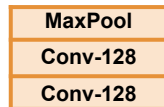
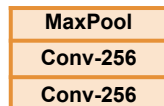
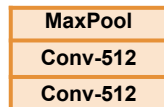
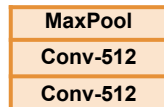
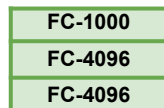
MaxPool
Conv-128
Conv-128

MaxPool
Conv-64
Conv-64

“Transfer Learning” с нейронными сетями



1. Обучаем на Imagenet
2. Обучаем на маленьком датасете (C классов)



Изменяем этот
слой и учим
только его

А эти слои
замораживаем

Обучаем с
более низким
LR (1/10 от
исходного
подойдёт)

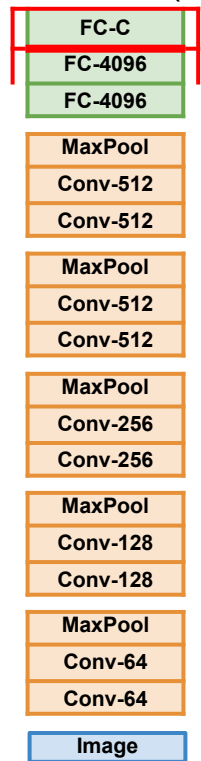
“Transfer Learning” с нейронными сетями



1. Обучаем на Imagenet



2. Обучаем на маленьком датасете (C классов)

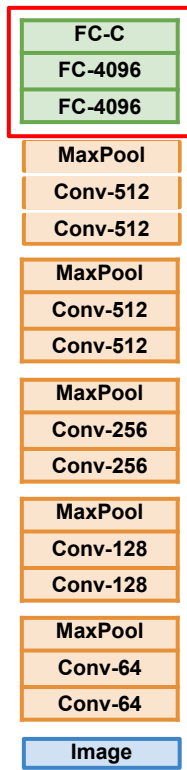


Изменяем этот
слой и учим
только его

А эти слои
замораживаем

Обучаем с
более низким
LR (1/10 от
исходного
подойдёт)

3. Обучаем на большем датасете



Тренируе
м эти слои

Чем больше датасет,
тем больше слоёв
нужно обучать

А эти слои
замораживаем

“Transfer Learning” с нейронными сетями



FC-1000
FC-4096
FC-4096

MaxPool
Conv-512
Conv-512

MaxPool
Conv-512
Conv-512

MaxPool
Conv-256
Conv-256

MaxPool
Conv-128
Conv-128

MaxPool
Conv-64
Conv-64

Image

Специфичные
признаки

Общие признаки

	Похожий датасет	Непохожий датасет
Маленький тренировочный датасет		
Большой тренировочный датасет		

“Transfer Learning” с нейронными сетями



FC-1000
FC-4096
FC-4096

MaxPool
Conv-512
Conv-512

MaxPool
Conv-512
Conv-512

MaxPool
Conv-256
Conv-256

MaxPool
Conv-128
Conv-128

MaxPool
Conv-64
Conv-64

Image

Специфичные
признаки

Общие признаки

	Похожий датасет	Непохожий датасет
Маленький тренировочный датасет	Учите только последний слой	
Большой тренировочный датасет	Учите сначала половину слоёв, после тюните всю модель	

“Transfer Learning” с нейронными сетями



FC-1000
FC-4096
FC-4096

MaxPool
Conv-512
Conv-512

MaxPool
Conv-512
Conv-512

MaxPool
Conv-256
Conv-256

MaxPool
Conv-128
Conv-128

MaxPool
Conv-64
Conv-64

Image

Специфичные
признаки

Общие признаки

	Похожий датасет	Непохожий датасет
Маленький тренировочный датасет	Учите только последний слой	Трудный случай. Пробуйте учить всю модель с маленьким LR
Большой тренировочный датасет	Учите сначала половину слоёв, после тюните всю модель	Учите всю модель

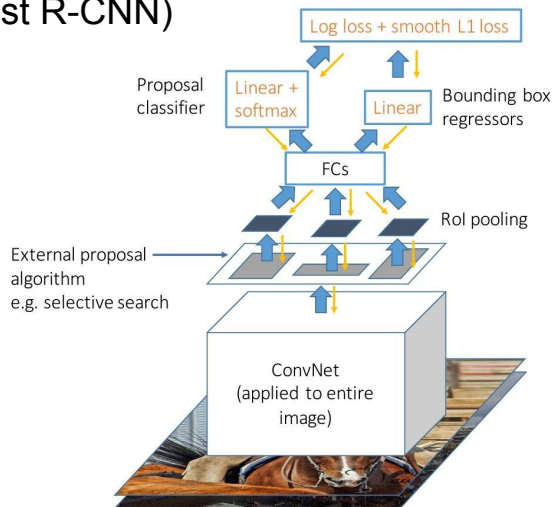
“Transfer Learning” с нейронными сетями



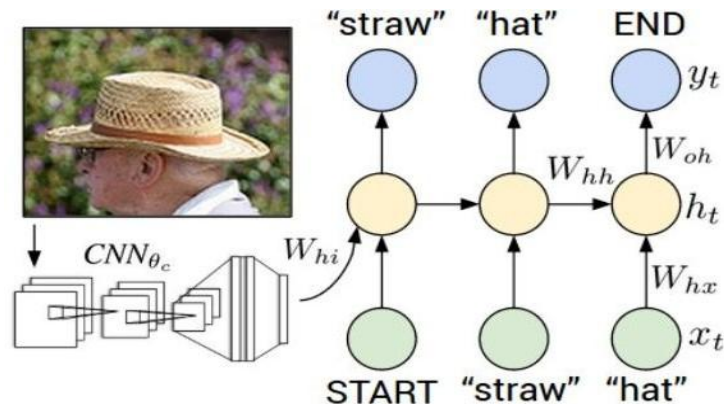
Общепринятая практика

Детекция объектов

(Fast R-CNN)



Описание изображений: CNN + RNN

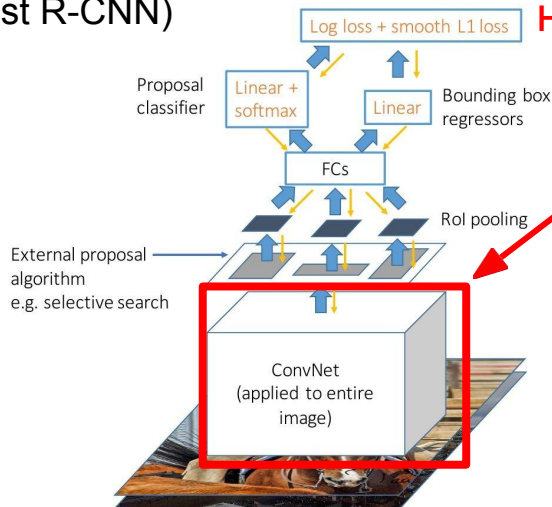


“Transfer Learning” с нейронными сетями



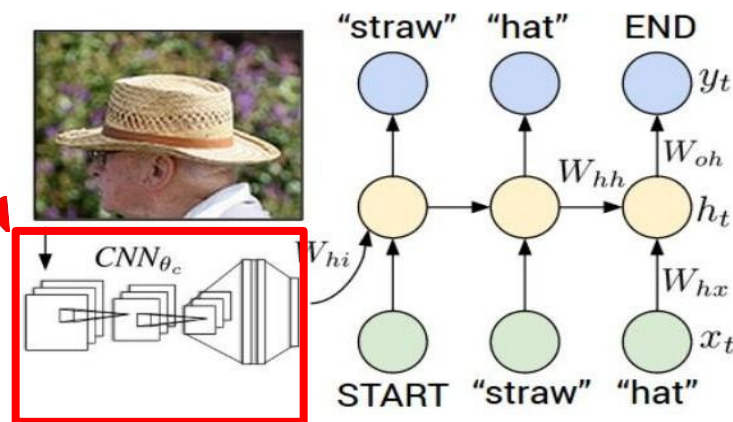
Общепринятая практика

Детекция объектов
(Fast R-CNN)



CNN предобучена
на ImageNet

Описание изображений: CNN + RNN

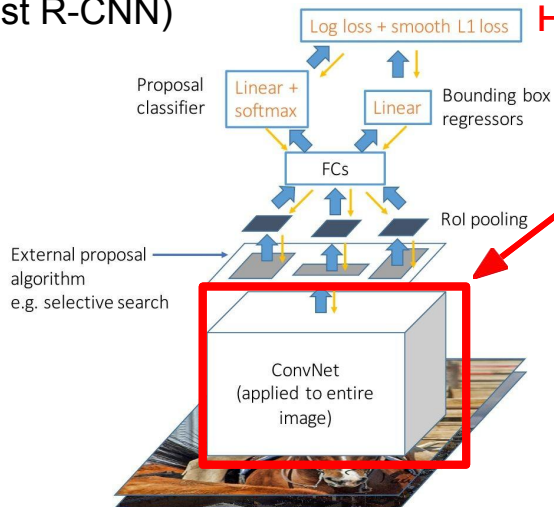


“Transfer Learning” с нейронными сетями



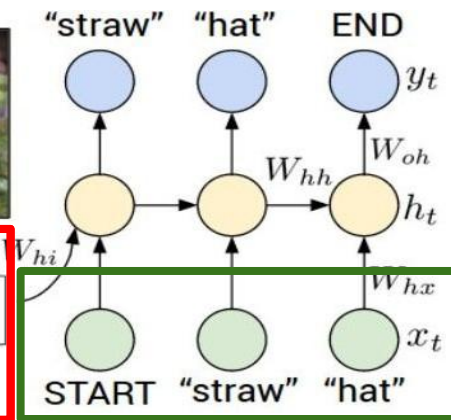
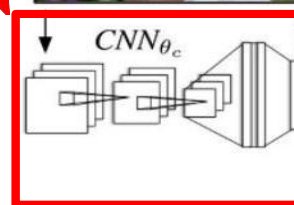
Общепринятая практика

Детекция объектов
(Fast R-CNN)



CNN предобучена
на ImageNet

Описание изображений: CNN + RNN

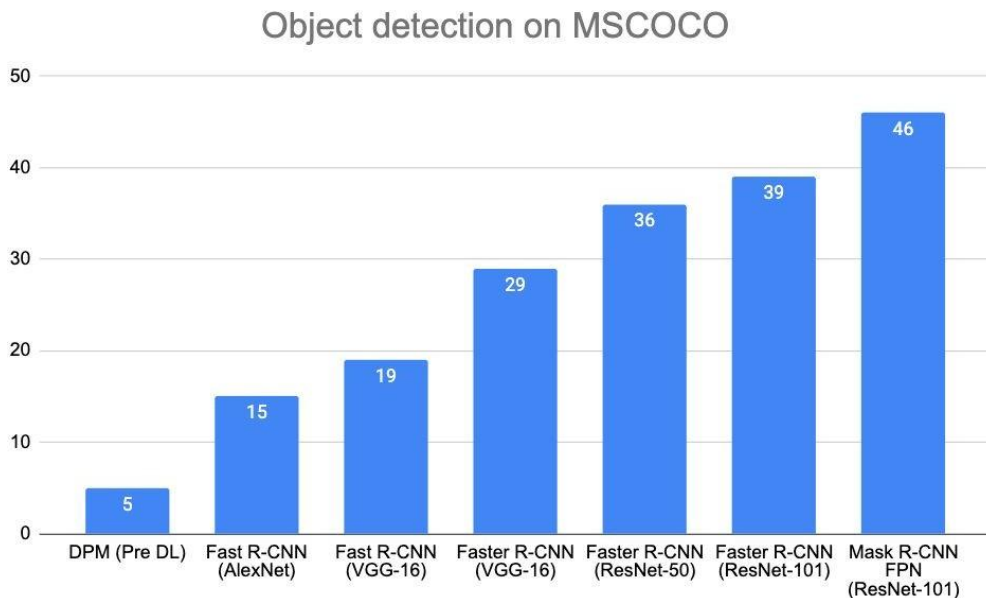


Предобучены вектора
слов на word2vec

“Transfer Learning” с нейронными сетями



Архитектура нейронной сети имеет большое значение как для обучения с нуля, так и для Transfer Learning



“Transfer Learning” с нейронными сетями



Transfer learning be like



“Transfer Learning” с нейронными сетями



Многие предобученные веса можно найти в GitHub-репозиториях или на официальных сайтах библиотек глубокого обучения

TensorFlow: <https://github.com/tensorflow/models>

PyTorch: <https://github.com/pytorch/vision>,
<https://github.com/rwightman/pytorch-image-models>

Оптимизация

Оптимизация

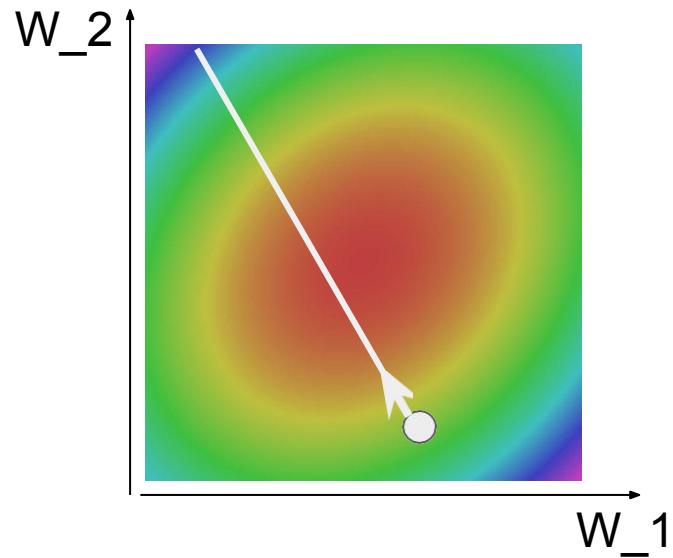


```
# Vanilla Gradient Descent
```

```
while True:
```

```
    weights_grad = evaluate_gradient(loss_fun, data, weights)
```

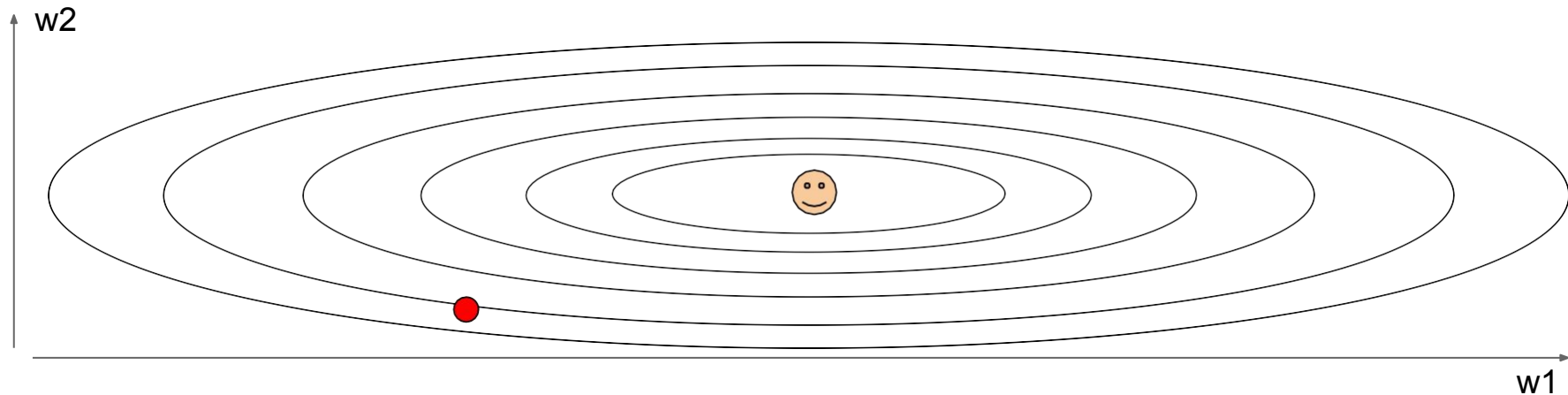
```
    weights += - step_size * weights_grad # perform parameter update
```



Оптимизация: проблемы с SGD



Что произойдёт, если градиент будет большим по одному признаку, но маленьким по другому? Как поведёт себя градиентный спуск?

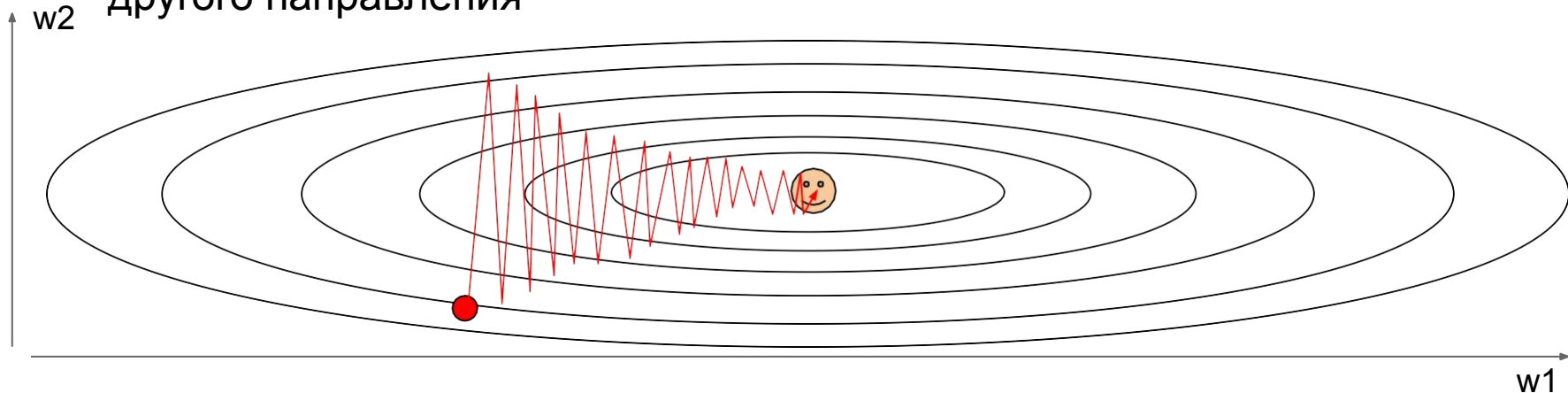


Оптимизация: проблемы с SGD



Что произойдёт, если градиент будет большим по одному признаку, но маленьким по другому? Как поведёт себя градиентный спуск?

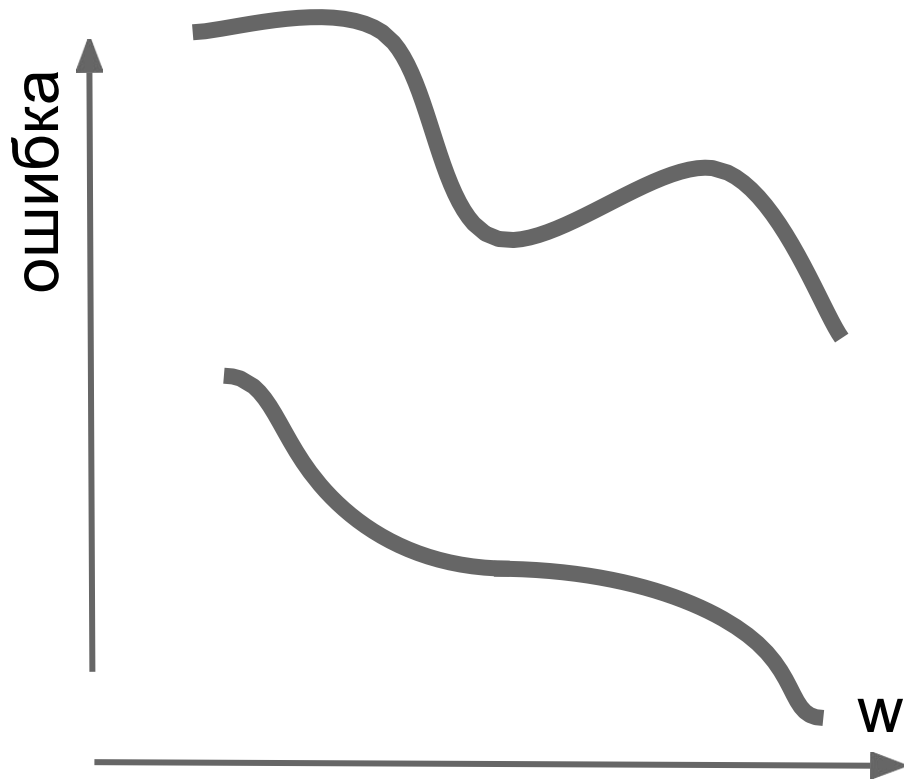
Очень медленный прогресс вдоль одного измерения, скачки вдоль другого направления



Оптимизация: проблемы с SGD



Как будет работать алгоритм
в **локальном минимуме** или
в **седловой точке**?

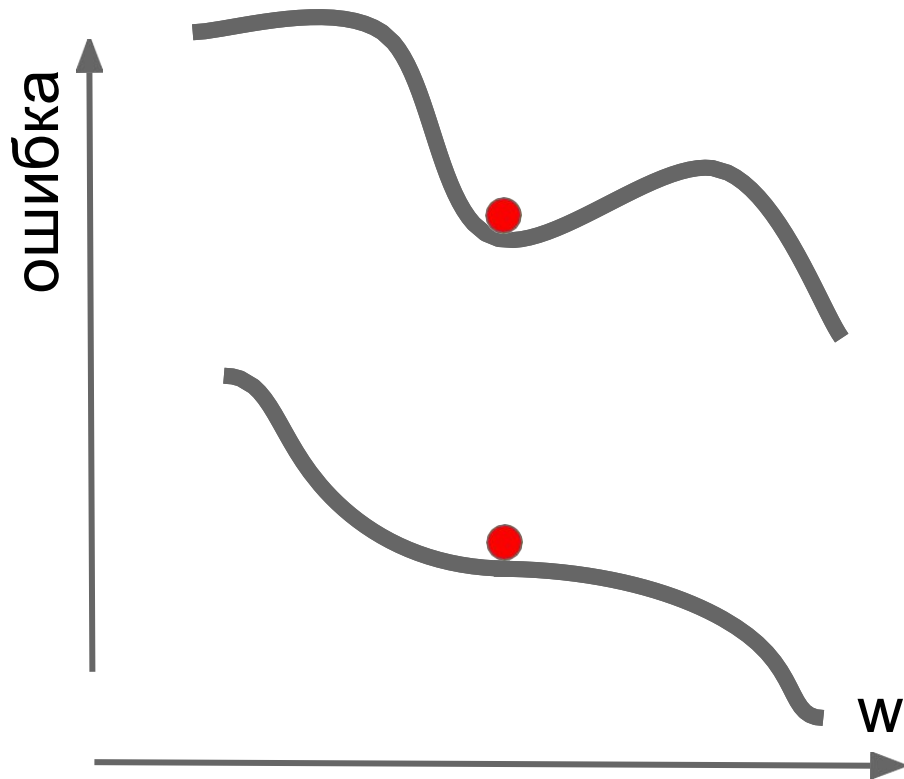


Оптимизация: проблемы с SGD



Как будет работать алгоритм
в **локальном минимуме** или
в **седловой точке**?

Градиент равен нулю.
Алгоритм остановится.

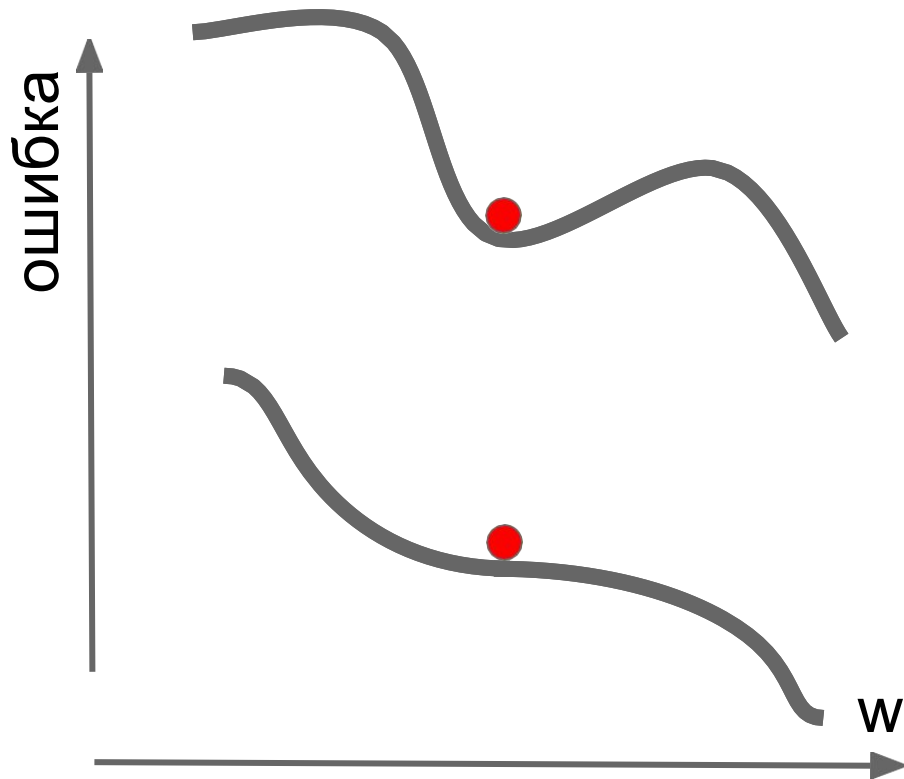


Оптимизация: проблемы с SGD



Как будет работать алгоритм
в **локальном минимуме** или
в **седловой точке**?

Градиент равен нулю.
Алгоритм остановится.
Локальные минимумы и
седловые точки - очень
частое явление в
многомерном пространстве



Используйте разные оптимизаторы



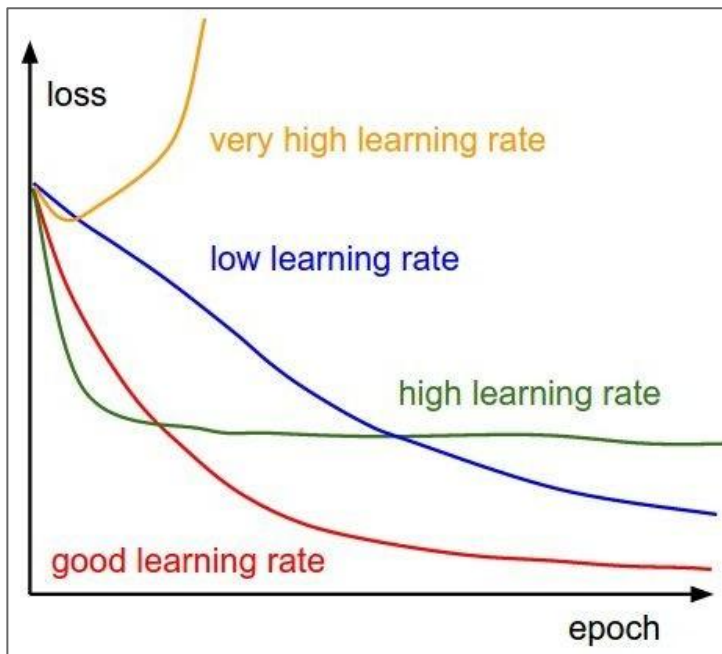
1. SGD + (Nesterov) Momentum
2. Adam
3. RMSProp
4. AdaGrad
5. NovoGrad
6. L-BFGS
7. И многие другие...

(хотя на чаще всего используют первые два)

Learning Rate Scheduling



У любого оптимизатора обязательным параметром выступает Learning Rate.

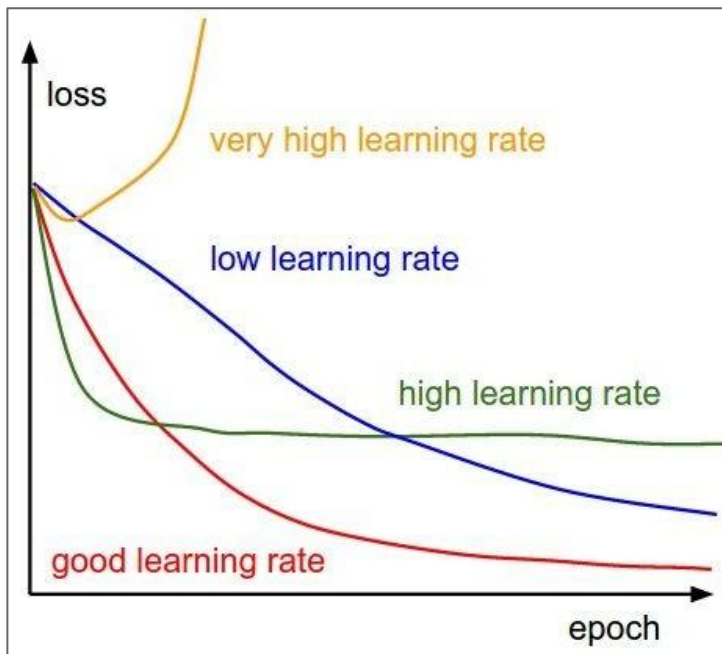


Какой из них лучше
всего подойдёт?

Learning Rate Scheduling



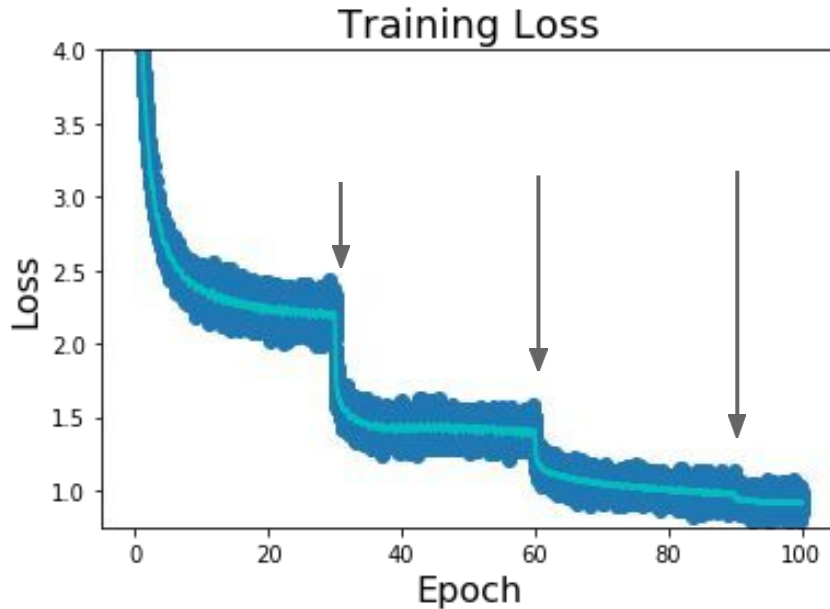
У любого оптимизатора обязательным параметром выступает Learning Rate.



Какой из них лучше
всего подойдёт?

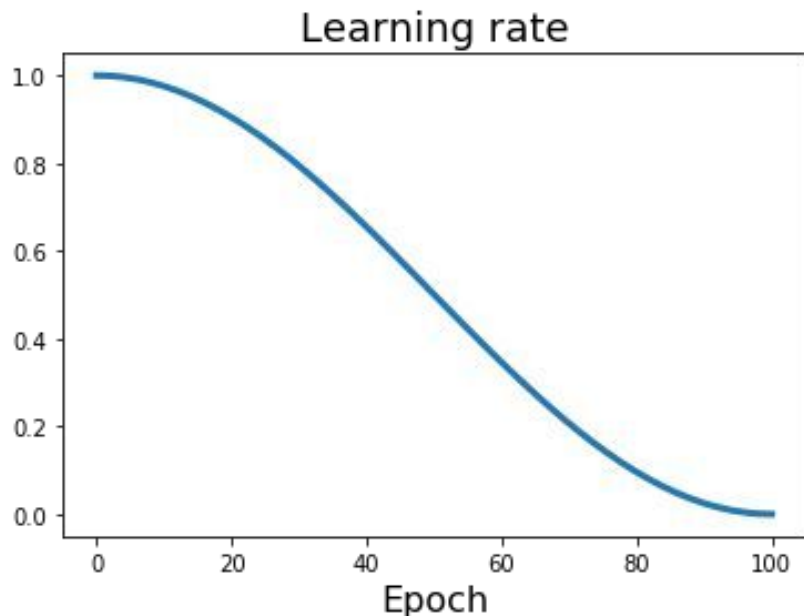
Все сразу! Начните с
большого шага, уменьшая
его со временем.

Learning Rate Decay



Step: Уменьшайте LR каждые N шагов
Например, умножайте LR на 0.1 каждые 30 эпох.

Learning Rate Decay



Step: Уменьшайте LR каждые N шагов
Например, умножайте LR на 0.1 каждые 30 эпох.

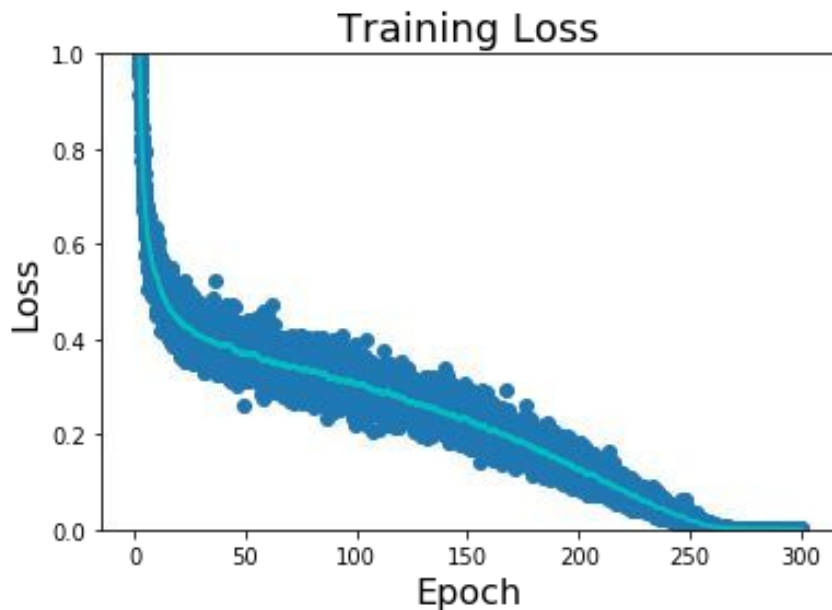
Cosine:
$$\alpha_t = \frac{1}{2} \alpha_0 (1 + \cos(t\pi/T))$$

α_0 : Initial learning rate

α_t : Learning rate at epoch t

T : Total number of epochs

Learning Rate Decay



Step: Уменьшайте LR каждые N шагов
Например, умножайте LR на 0.1 каждые 30 эпох.

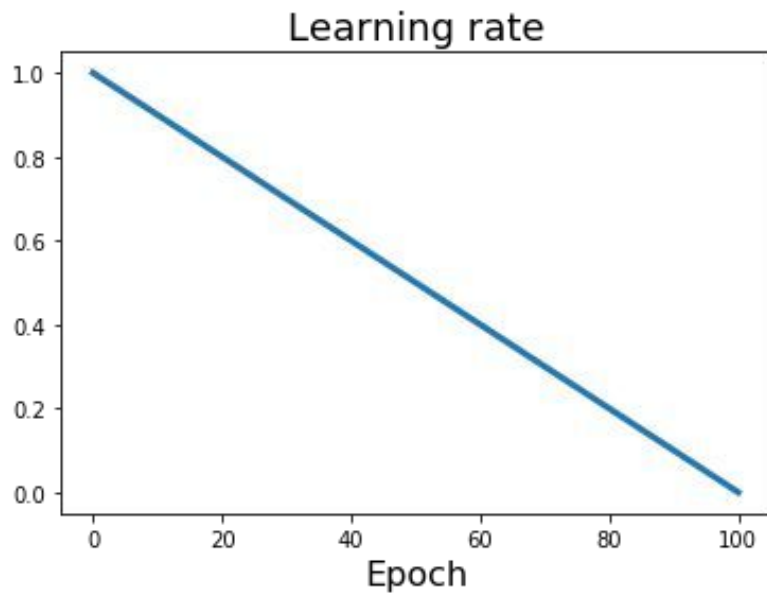
Cosine:
$$\alpha_t = \frac{1}{2} \alpha_0 (1 + \cos(t\pi/T))$$

α_0 : Initial learning rate

α_t : Learning rate at epoch t

T : Total number of epochs

Learning Rate Decay



Step: Уменьшайте LR каждые N шагов
Например, умножайте LR на 0.1 каждые 30 эпох.

Cosine: $\alpha_t = \frac{1}{2}\alpha_0 (1 + \cos(t\pi/T))$

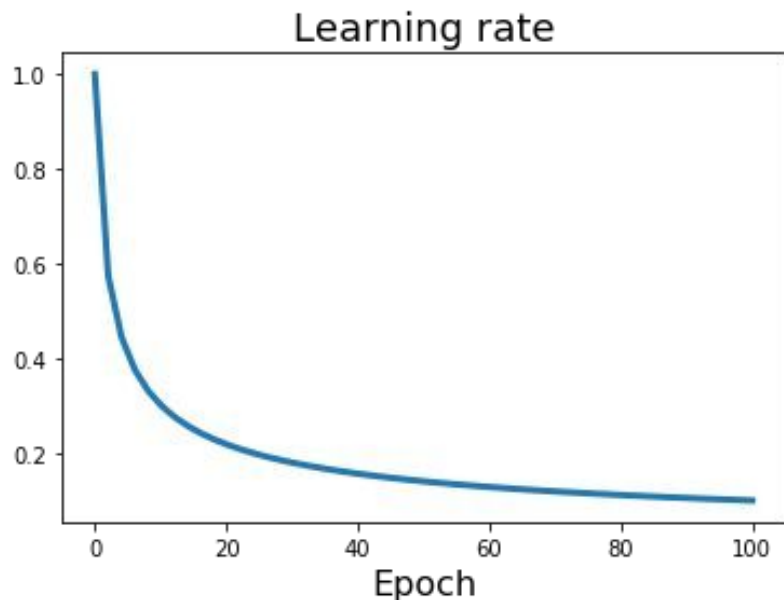
Linear: $\alpha_t = \alpha_0(1 - t/T)$

α_0 : Initial learning rate

α_t : Learning rate at epoch t

T : Total number of epochs

Learning Rate Decay



Step: Уменьшайте LR каждые N шагов
Например, умножайте LR на 0.1 каждые 30 эпох.

Cosine: $\alpha_t = \frac{1}{2}\alpha_0 (1 + \cos(t\pi/T))$

Linear: $\alpha_t = \alpha_0(1 - t/T)$

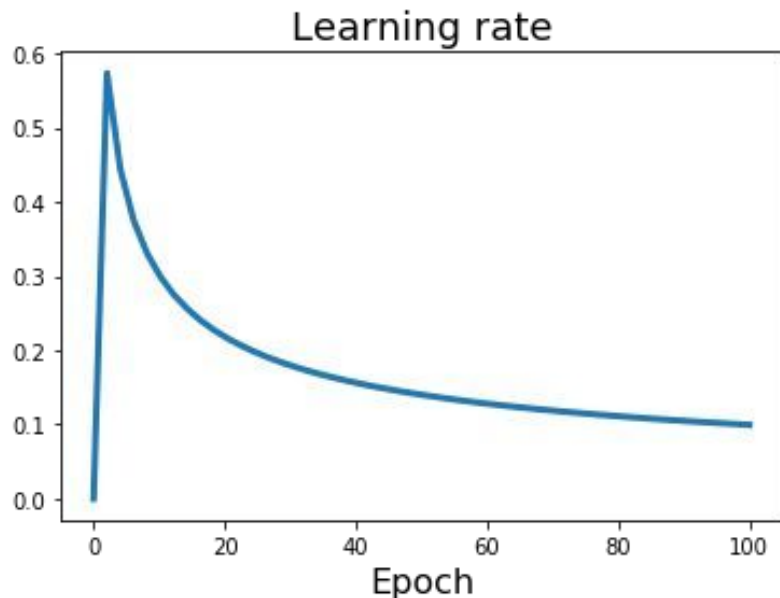
Inverse square: $\alpha_t = \alpha_0/\sqrt{t}$

α_0 : Initial learning rate

α_t : Learning rate at epoch t

T : Total number of epochs

Learning Rate Decay: Linear Warmup

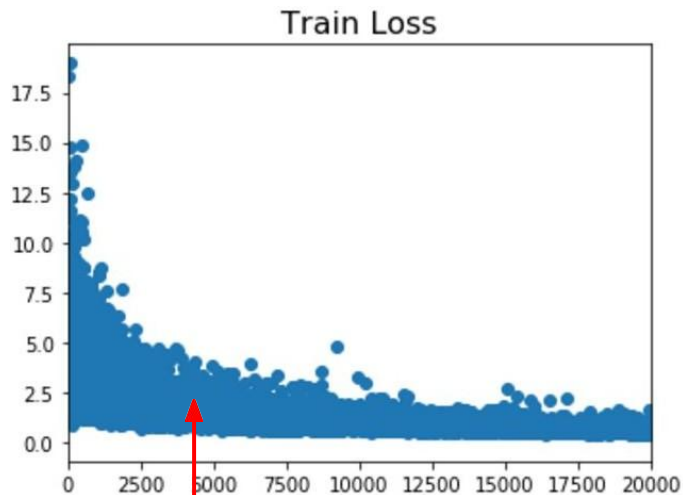


Высокое начальное значение LR может спровоцировать “взрыв градиентов”. Чтобы этого избежать, начинайте с маленького значения LR, постепенно повышая его в течении первых эпох обучения, а затем начинайте уменьшать по любой из схем, представленных ранее.

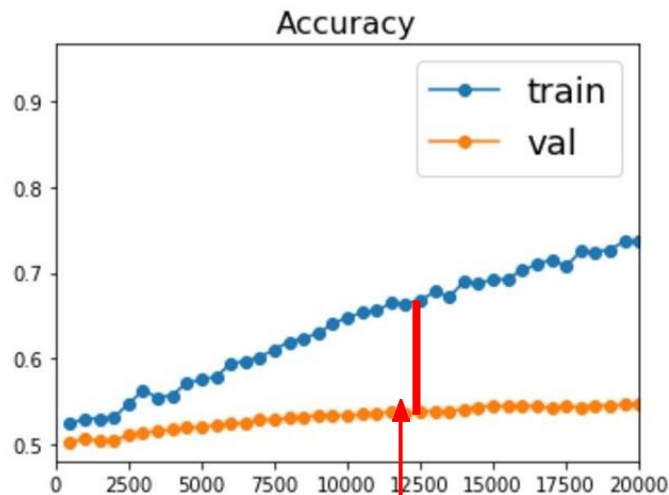
Если вы увеличиваете батчсайд в N раз, так же увеличьте LR в N раз.

Улучшение
точности на
тестах

Разница в Train/Test

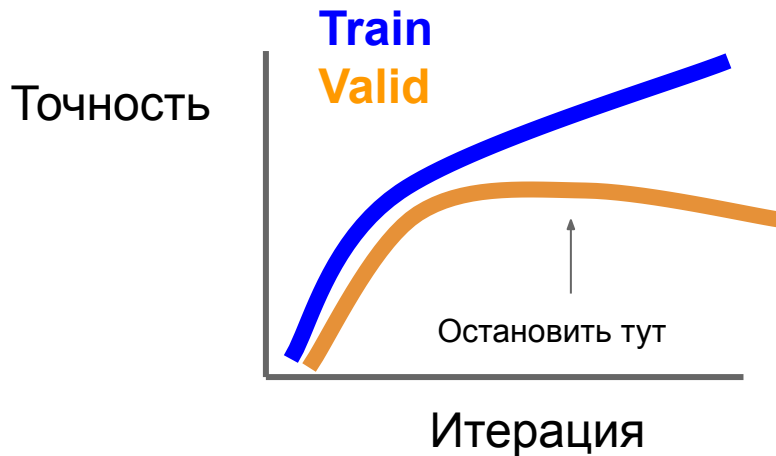
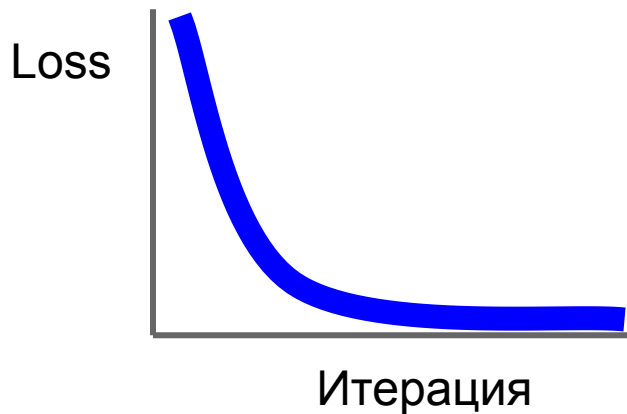


Лучший алгоритм оптимизации
позволяет улучшить метрики на
обучающем датасете



Но нам так же важно, что наш
алгоритм хорошо работал и на
новых данных

Ранняя остановка



Останавливайте обучение если видите, что точность на тесте перестала расти или начала падать. Так же сохраняйте чекпоинт модели с лучшей метрикой на валидационном датасете.

Регуляризация в функции ошибки



$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \boxed{\lambda R(W)}$$

Часто используемые:

L2 регуляризация

$$R(W) = \sum_k \sum_l W_{k,l}^2 \quad (\text{Weight decay})$$

L1 регуляризация

$$R(W) = \sum_k \sum_l |W_{k,l}|$$

Elastic net (L1 + L2)

$$R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$$

Регуляризация



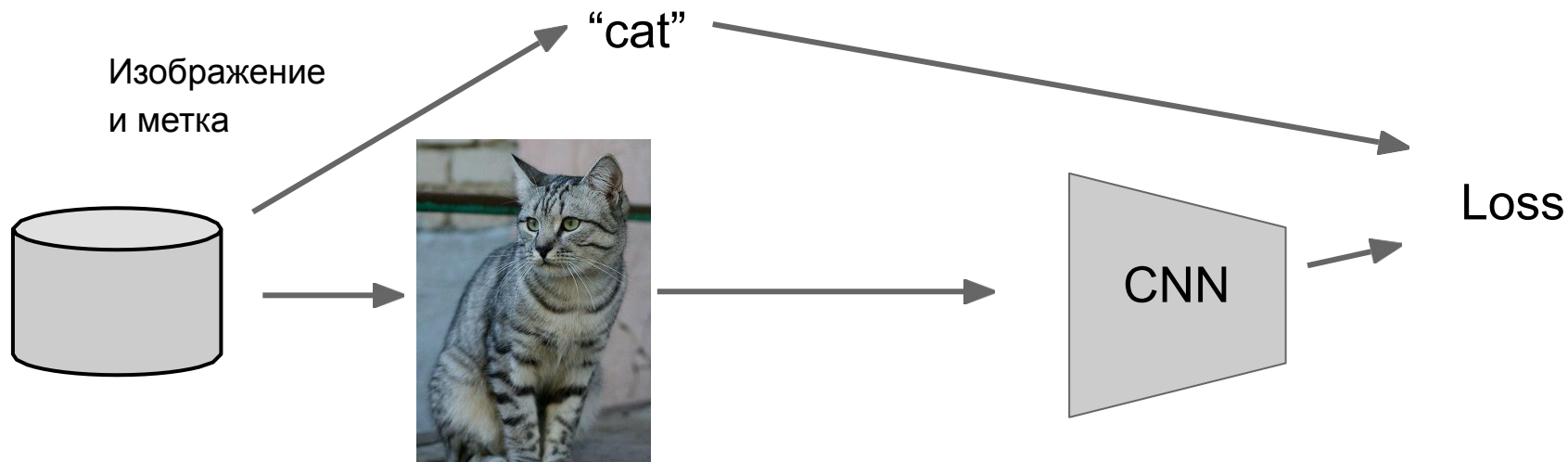
Тренировка: Добавляйте случайный шум в данные

$$y = f_W(x, z)$$

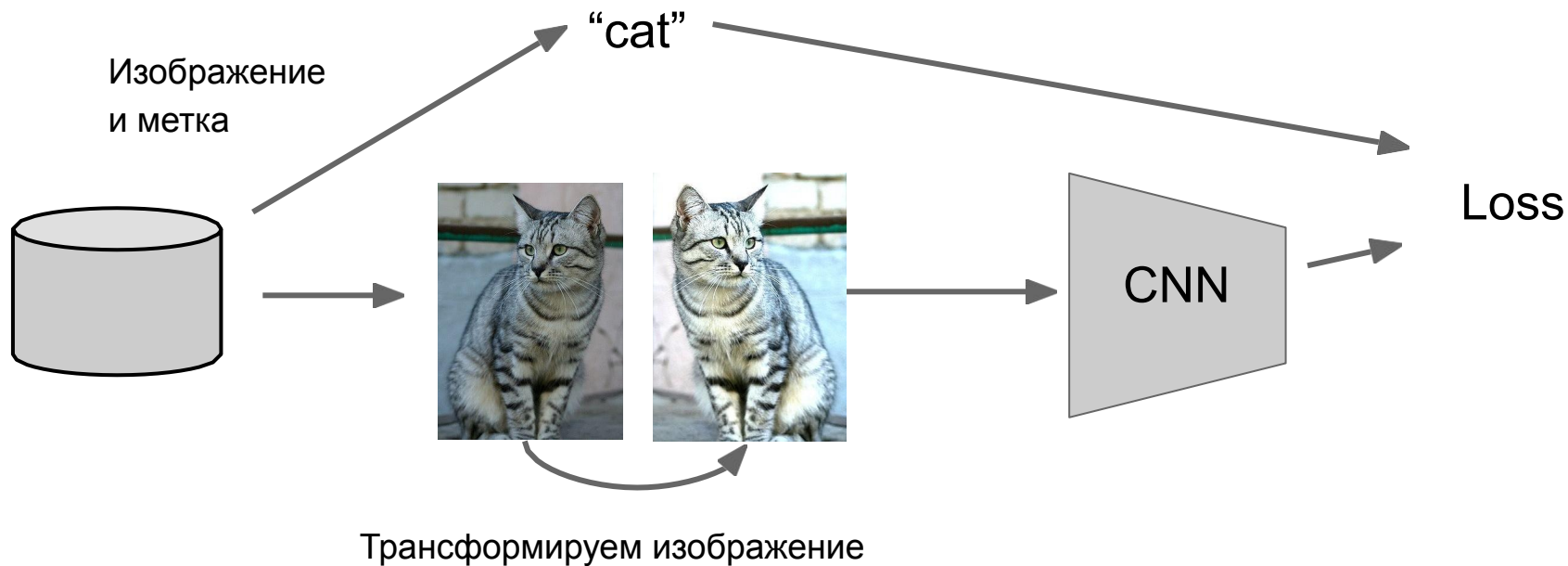
Тест: Усредняйте зашумлённые предсказания

$$y = f(x) = E_z[f(x, z)] = \int p(z)f(x, z)dz$$

Аугментация данных





















Аугментация данных

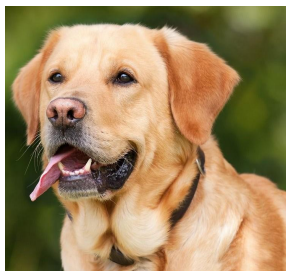


Автоматическая аугментация данных



	Original	Sub-policy 1	Sub-policy 2	Sub-policy 3	Sub-policy 4	Sub-policy 5
Batch 1						
Batch 2						
Batch 3						
		ShearX, 0.9, 7 Invert, 0.2, 3	ShearY, 0.7, 6 Solarize, 0.4, 8	ShearX, 0.9, 4 AutoContrast, 0.8, 3	Invert, 0.9, 3 Equalize, 0.6, 3	ShearY, 0.8, 5 AutoContrast, 0.7, 3

Регуляризация: Микс



Случайно смешиваем
пиксели изображений,
например, в соотношении
40% cat, 60% dog

CNN

Целевая метка:

cat: 0.4

dog: 0.6

Выбор гиперпараметров

(без сотен видеокарт)

Выбор гиперпараметров



Шаг 1: Проверяйте значение ошибки в начале

Отключите `weight decay`. Убедитесь, что ошибка не улетает в бесконечность

Выбор гиперпараметров



Шаг 1: Проверяйте значение ошибки в начале

Шаг 2: Переобучитесь на нескольких примерах

Обучите до 100% тренировочной точности на небольшом наборе тренировочных данных (~5-10 батчей);

Подберите архитектуру, оптимизатор, LR.

Ошибка не падает? Низкий LR, Плохая инициализация

Ошибка уходит в Inf или NaN? Плохая инициализация

Выбор гиперпараметров



Шаг 1: Проверяйте значение ошибки в начале

Шаг 2: Переобучитесь на нескольких примерах

Шаг 3: Найдите LR, для которого ошибка падает

Используйте все тренировочные данные, добавьте небольшой weight decay, найдите LR, с которым ошибка стабильно падает в течении `that` примерно ~ 100 итераций

Можно попробовать такие значения: $1e-1$, $1e-2$, $1e-3$, $1e-4$

Выбор гиперпараметров



Шаг 1: Проверяйте значение ошибки в начале

Шаг 2: Переобучитесь на нескольких примерах

Шаг 3: Найдите LR, для которого ошибка падает

Шаг 4: Обучайте модель в течении примерно 1-5 эпох

Выбор гиперпараметров



Шаг 1: Проверяйте значение ошибки в начале

Шаг 2: Переобучитесь на нескольких примерах

Шаг 3: Найдите LR, для которого ошибка падает

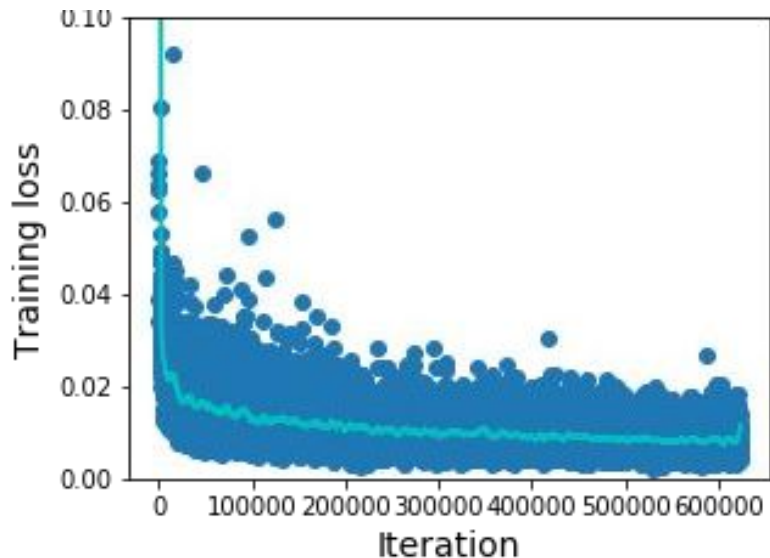
Шаг 4: Обучайте модель в течении примерно 1-5 эпох

Шаг 5: Смотрите на кривые ошибки

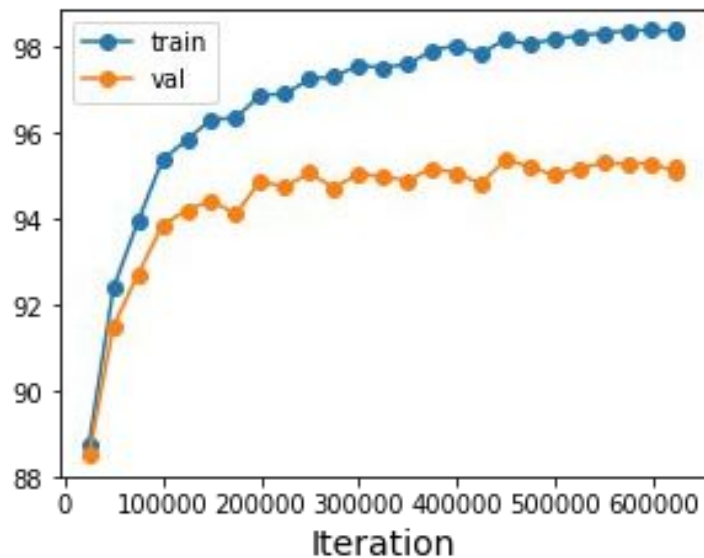
Кривые ошибки



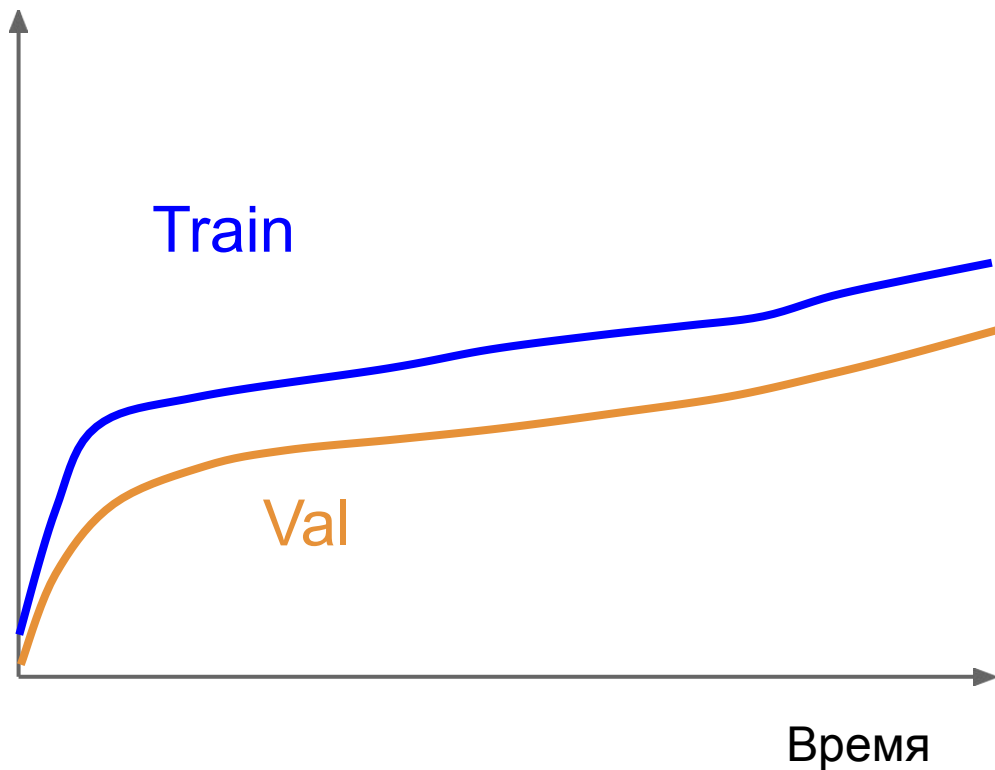
Тренировочная ошибка



Точность на Train / Val

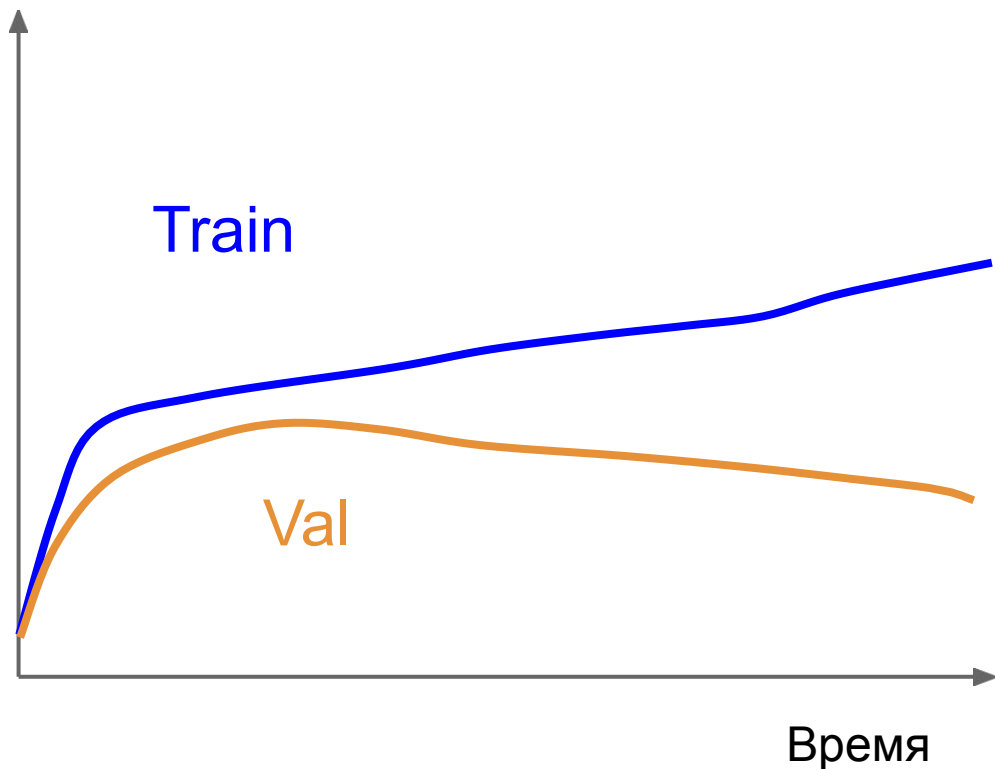


Кривые ошибки



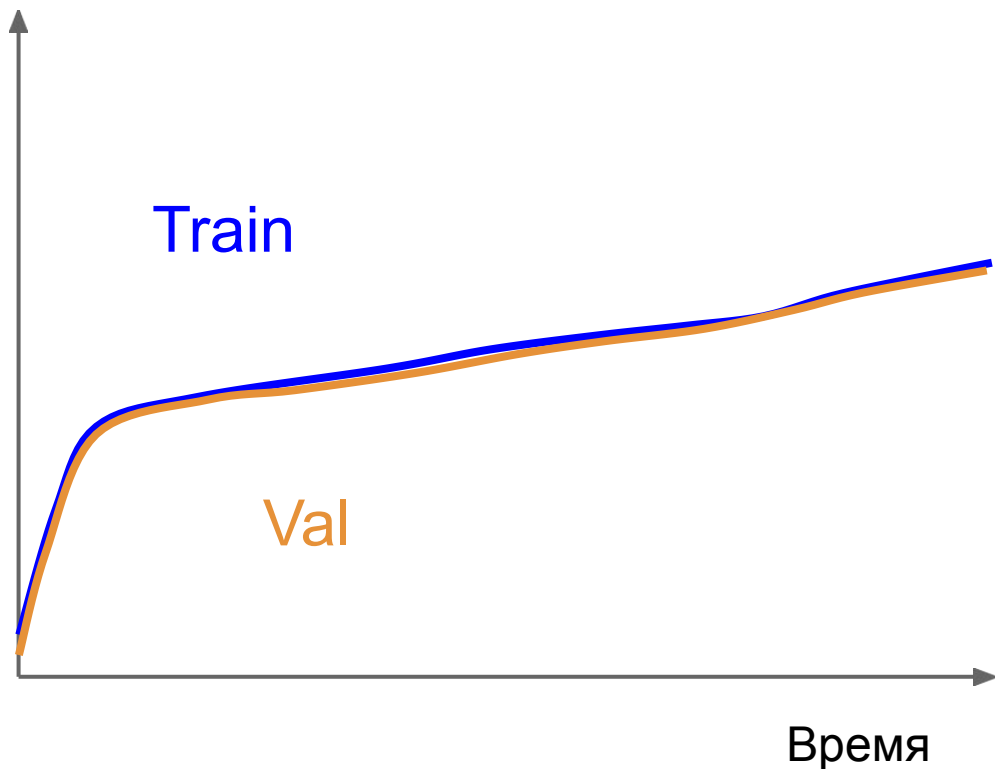
Всё хорошо,
продолжайте обучение

Кривые ошибки



Разрыв между обучением и валидацией растёт. Модель переобучилась. Добавьте регуляризацию, аугментации или новые данные

Кривые ошибки



Кривые почти совпадают.
Модель недоучилась.
Учите дольше или
используйте более
сложную модель

Выбор гиперпараметров



Шаг 1: Проверяйте значение ошибки в начале

Шаг 2: Переобучитесь на нескольких примерах

Шаг 3: Найдите LR, для которого ошибка падает

Шаг 4: Обучайте модель в течении примерно 1-5 эпох

Шаг 5: Смотрите на кривые ошибки

Шаг 6: Сделайте изменения параметрах. Учите дальше

Выбор гиперпараметров



Шаг 1: Проверяйте значение ошибки в начале

Шаг 2: Переобучитесь на нескольких примерах

Шаг 3: Найдите LR, для которого ошибка падает

Шаг 4: Обучайте модель в течении примерно 1-5 эпох

Шаг 5: Смотрите на кривые ошибки

Шаг 6: Сделайте изменения параметрах. Учите дальше

Шаг 7: Возвращайтесь к шагу 4 пока не получите желаемые метрики (если ваши желания совпадают с возможностями).