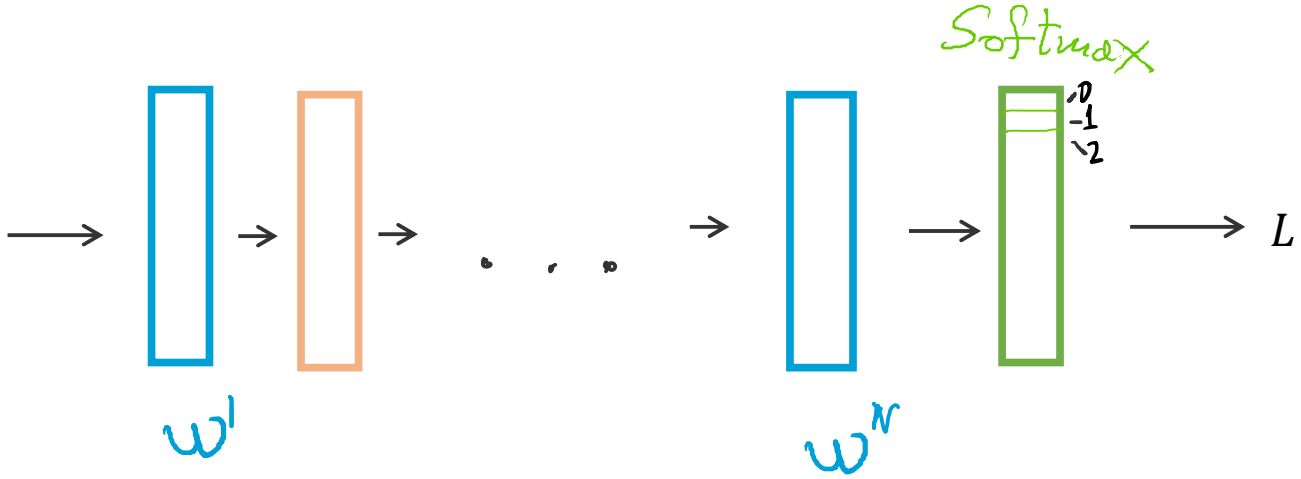
5

Нейронные сети — практика

Общая схема тренировки

Взять mini-batch

Прямой проход (forward pass) – посчитать loss



Обратный проход (backward pass) – посчитать градиент

Обновить параметры

$$\overrightarrow{w^1} = \overrightarrow{w^1} - \eta \overrightarrow{\nabla_{w^1} L}$$

...

$$\overrightarrow{w^n} = \overrightarrow{w^n} - \eta \overrightarrow{\nabla_{w^n} L}$$


PyTorch

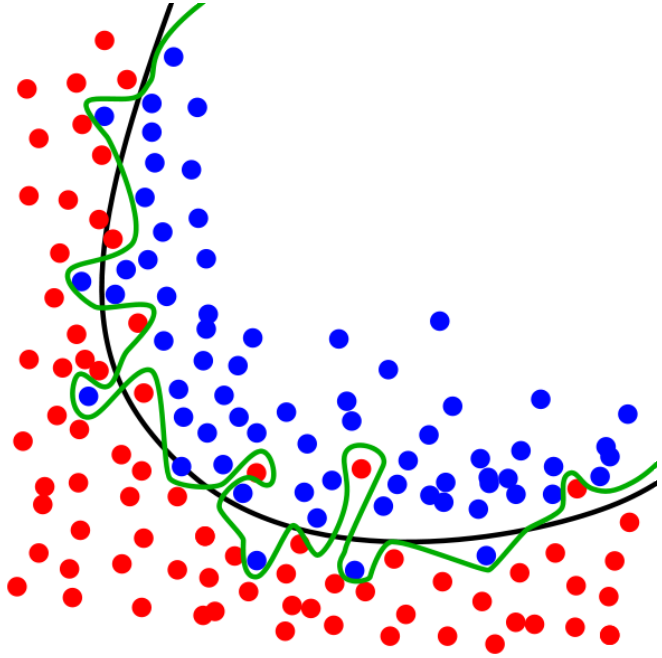
> "In theory, there is no difference between theory and practice. But, in practice, there is."

Jan L. A. van de Snepscheut as quoted by Doug Rosenberg and Matt Stephens (2007) *Use Case Driven Object Modeling with UMLTheory and Practice* p. xxvii; the quote is also cited without attribution in Doug Rosenberg and Kendall Scott (2001) *Applying Use Case Driven Object Modeling With UML*, p. 1
— This quote has also been attributed to Yogi Berra, to Chuck Reid, to William T. Harbaugh, to Manfred Eigen, to Alvin E. Roth *Wer hat das gesagt?* and to Karl Marx

Source

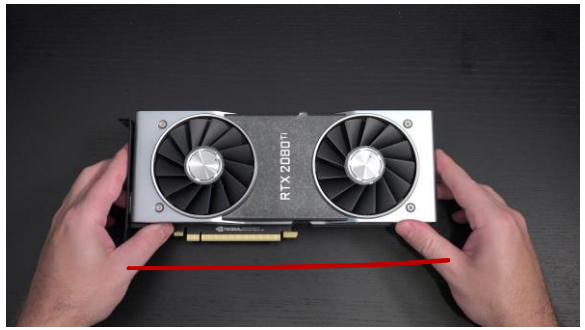
GPU

over/underfit

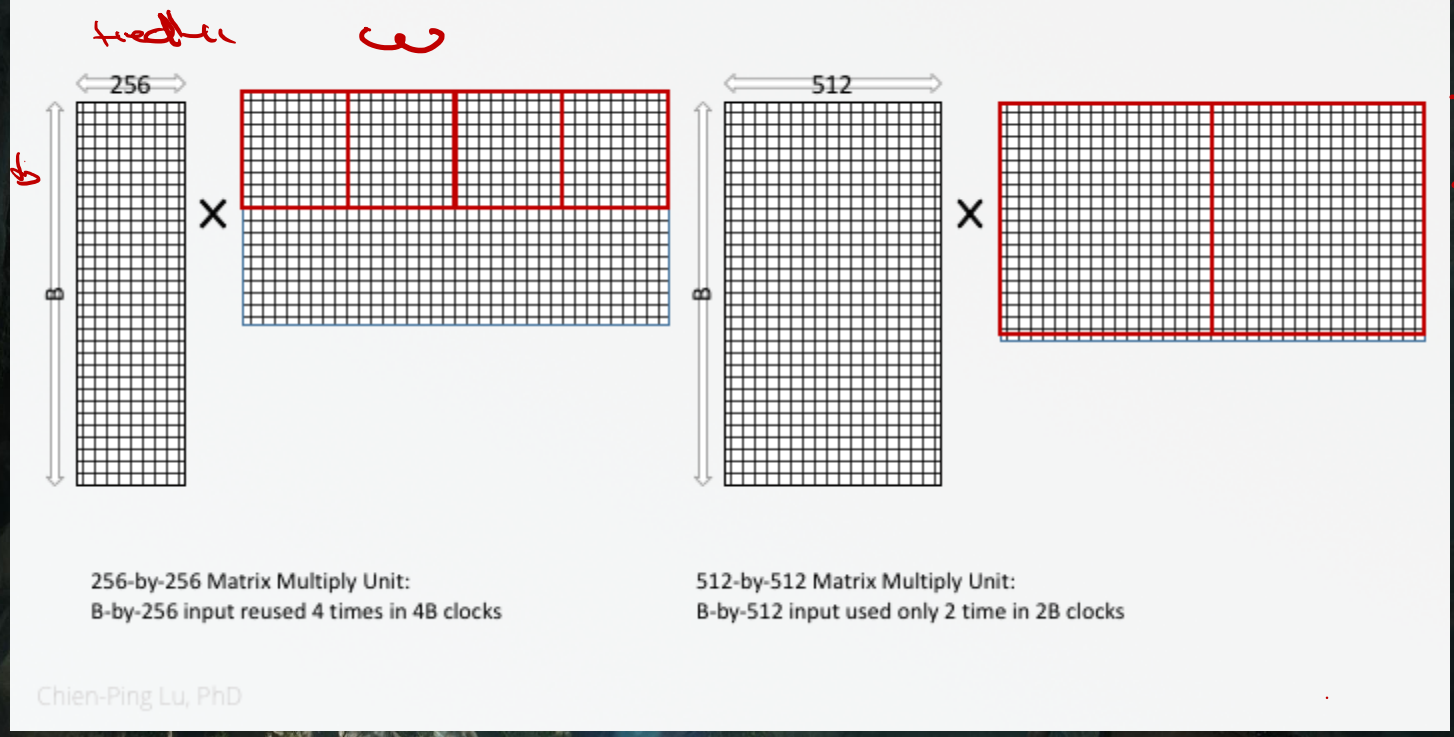


prod

GPU
Graphics Processing Unit



	Цена	Flops	Количество ядер	Скорость памяти
Intel Core i9-7980XE	$2000	1.8 TFlops	16	70 Gb/sec
Nvidia Geforce 2080 Ti	$1000	11.7 Tflops 107 Tensor Tflops	4352	616 Gb/sec



256-by-256 Matrix Multiply Unit:
B-by-256 input reused 4 times in 4B clocks

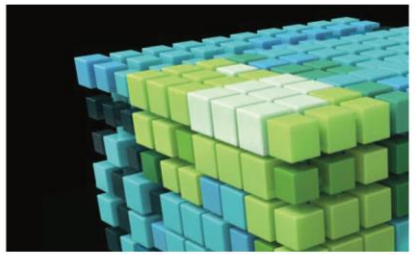
512-by-512 Matrix Multiply Unit:
B-by-512 input used only 2 time in 2B clocks

Как программировать на GPU


```cpp
// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C)
{
    int i = threadIdx.x;
    C[i] = A[i] + B[i];
}

int main()
{
    ...
    // Kernel invocation with N threads
    VecAdd<<<1, N>>>(A, B, C);
    ...
}
```

cuBLAS



Microsoft
CNTK
Caffe
Caffe2
dmlc
mxnet
theano
TensorFlow
K
torch
PYTORCH

THE

secret ingredient

IS

training

process

Machine Learning Flow

Ошибка на train

большая → underfitting
- Более мощную модель
- Больше ресурсов для тренировки
- Другой подход

маленькая ↓

Ошибка на val

большая → overfitting
- Больше данных
- Больше регуляризации
- Другой подход

маленькая ↓

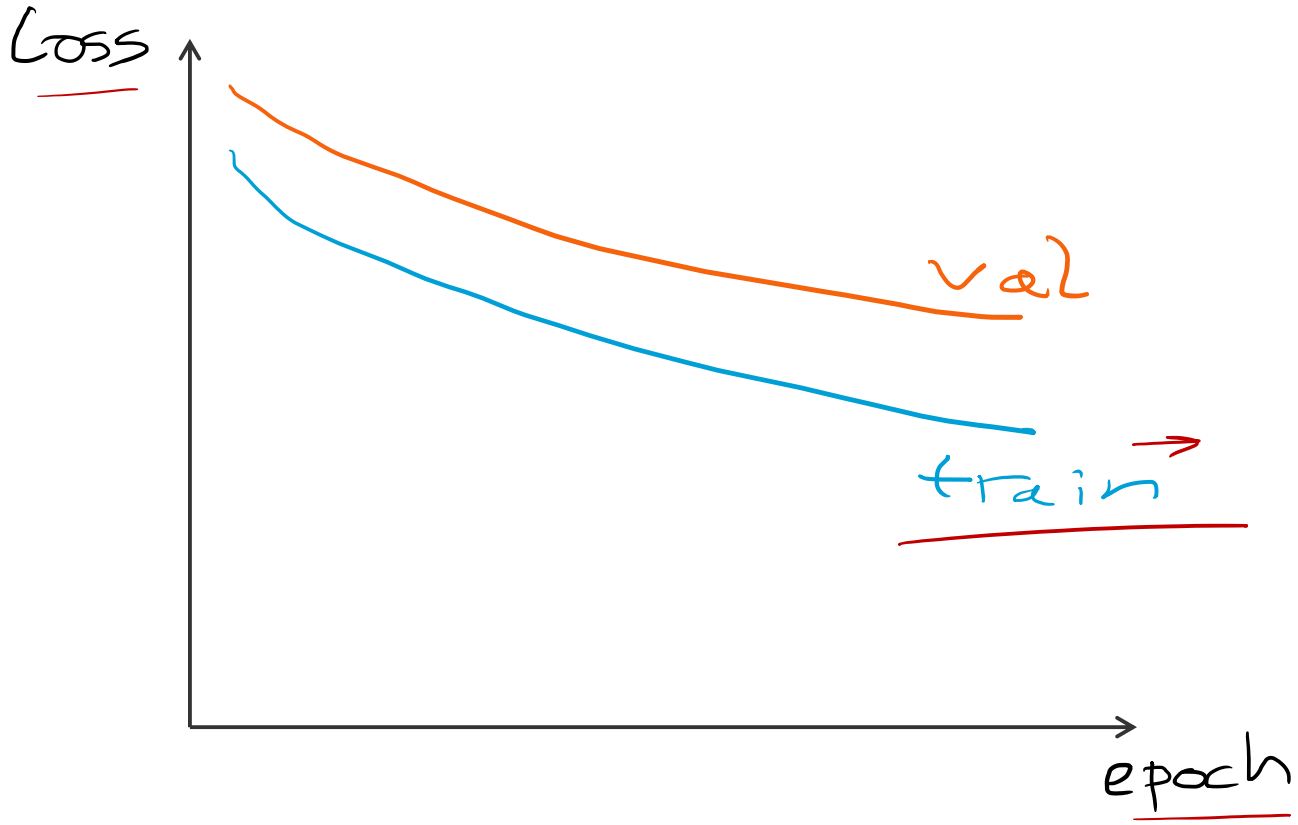
Ошибка на test

большая
- Отличаются train и test
- Больше данных, таких как test

маленькая ←

Смотрим на графики тренировки

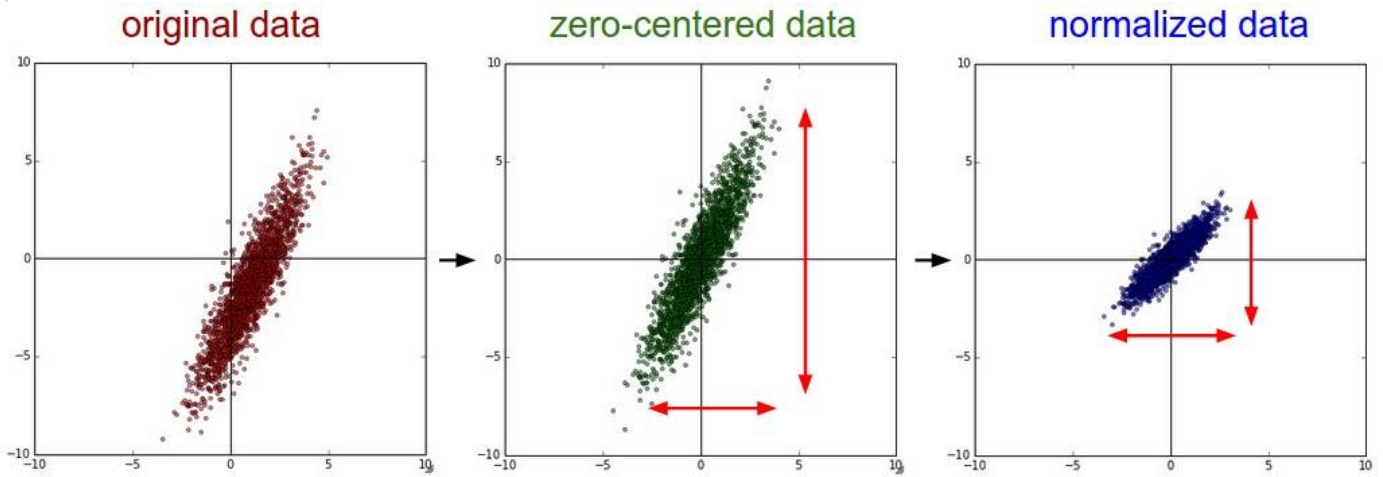
Train loss и val loss уменьшаются медленно
Находятся близко друг к другу



underfitting

То, о чем мы говорили раньше

Preprocessing



$$x' = \frac{x - \bar{x}}{\sigma}$$

$$\sigma = \frac{\sqrt{(x - \bar{x})^2}}{N - 1}$$

Adam

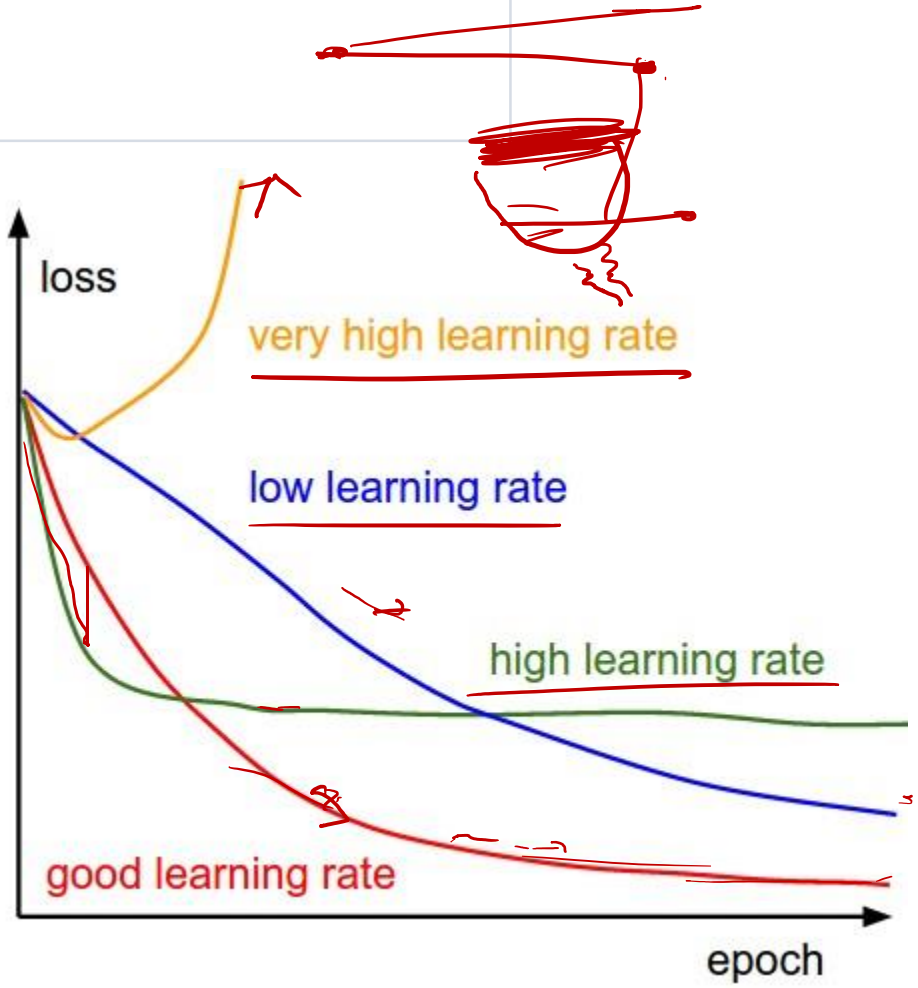
```
velocity = beta1 * velocity+ (1-beta1) * gradient
accumulated = beta2 * accumulated + (1-beta2) * gradient ** 2

adaptive_learning_rate = learning_rate / sqrt(accumulated)

w = w - adaptive_learning_rate * velocity
```

Adam: A Method for Stochastic Optimization'14

Скорость обучения
Learning rate

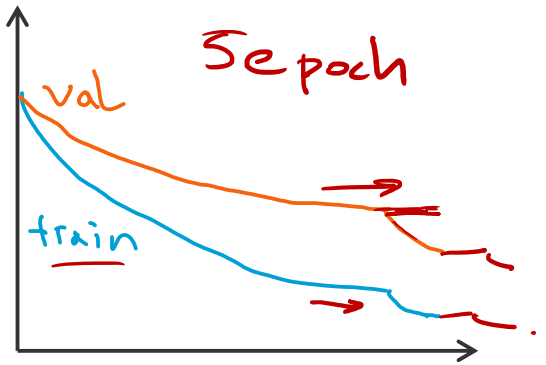


Annealing (decay)

$\gamma = 1$, $\lambda = 0.99$ $\gamma = 30$ $\lambda = 0.5$ 0.1

- Каждые X эпох умножать на λ
- Умножать на λ на плато

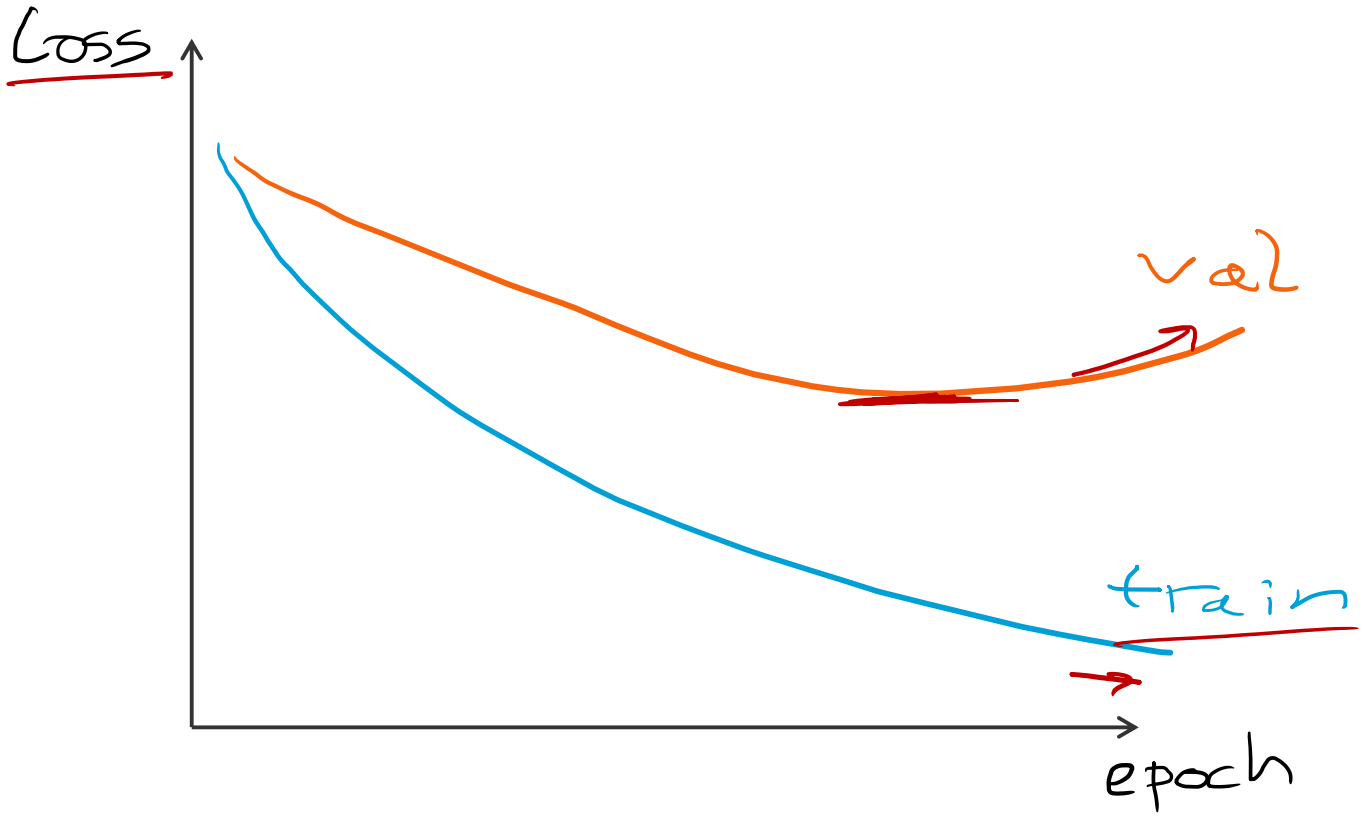
0.1

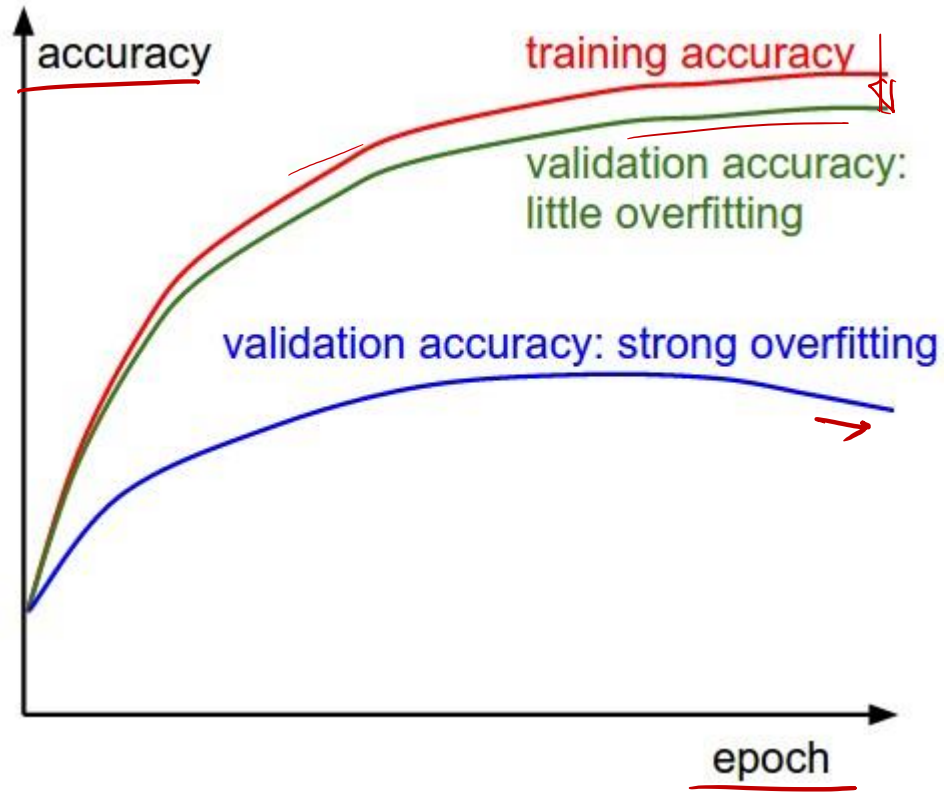



```
torch.optim.lr_scheduler.
```

http://cs231n.github.io/neural-networks-3/

overfitting

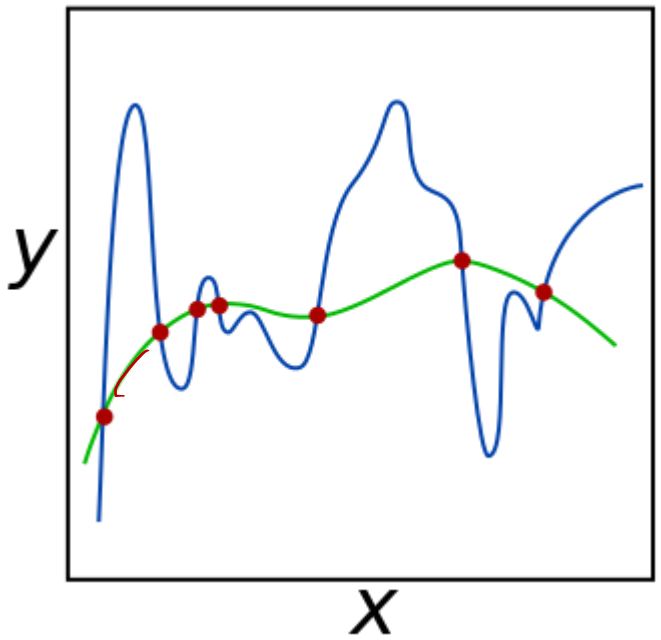




http://cs231n.github.io/neural-networks-3/

Регуляризация
Regularization

Tensor board



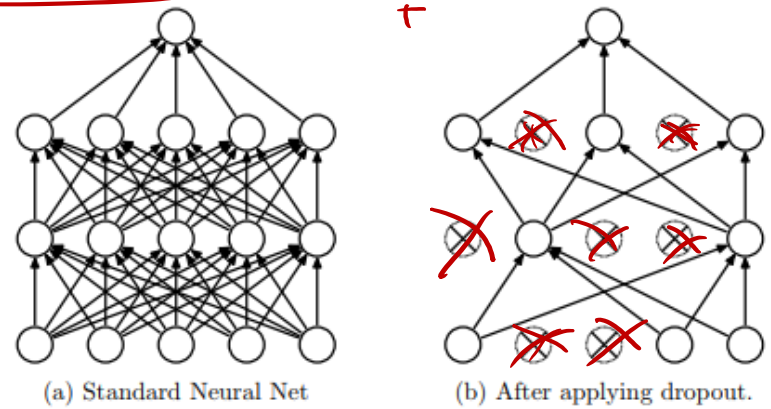
Wikipedia

L2 regularization (или weight decay)

$$+\lambda\|w\|_2^2$$

tail → cat
+
ears →
+

Dropout



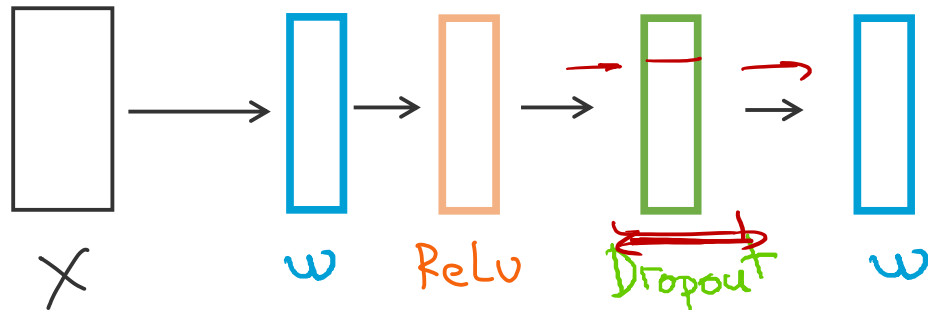
(a) Standard Neural Net

(b) After applying dropout.

0.3

0,5

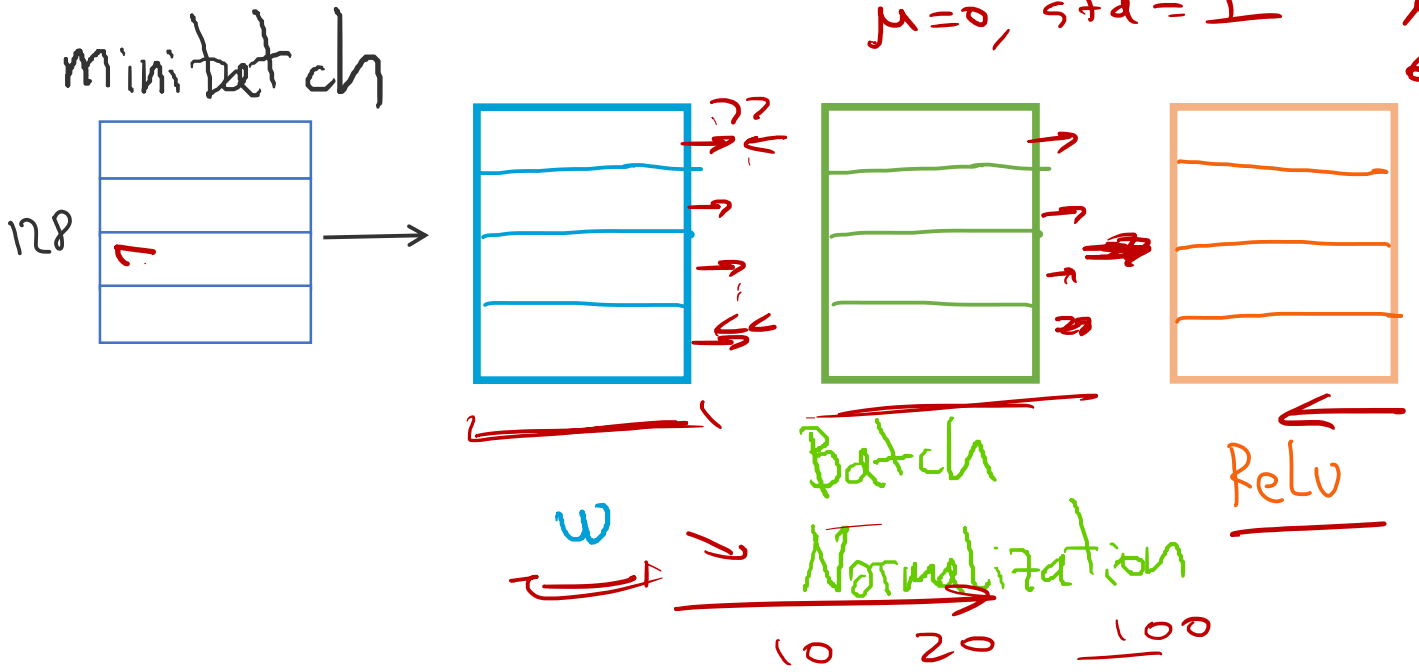
0.7



Batch Normalization

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

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}$$



- Ускоряет и стабилизирует тренировку
- Регуляризирует
- Не так важна инициализация
- Счастье какое-то

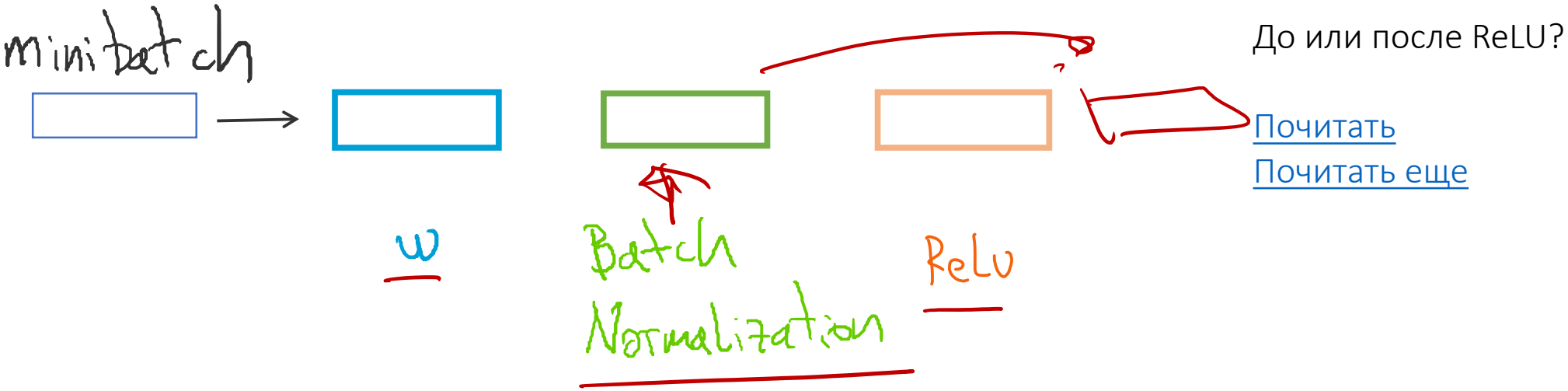
[Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift'15]

Batch Normalization – test time

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

Используем значения среднего и дисперсии, накопленные во время тренировки

$$y^{(k)} = \gamma^{(k)}\widehat{x}^{(k)} + \beta^{(k)}$$



До или после ReLU?

Почитать

Почитать еще

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift'15

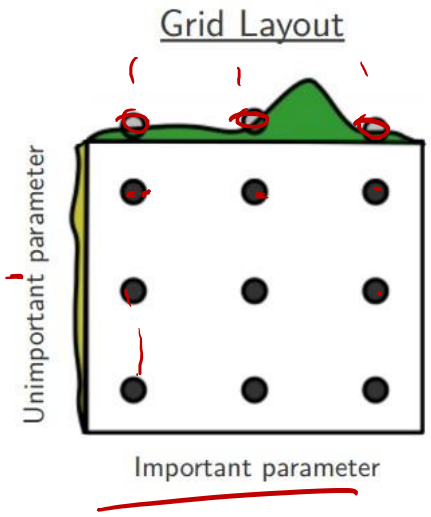
Поиск гиперпараметров
Hyperparameter search

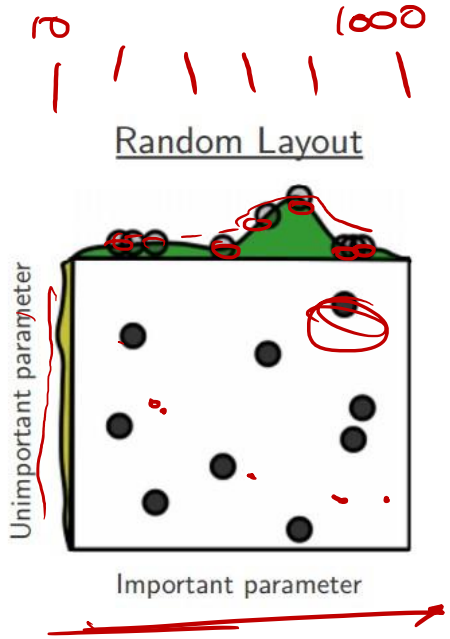
- Начальный learning rate
- Коэфф. annealing
- Коэфф. L2 регуляризации
- Размер батча
- ....

cross validation

Поиск гиперпараметров
Hyperparameter search

1 fold





В любом случае

- Coarse to fine
- Log space

```
lr = random(10, 10000)
lr = 10**random(1,4)
```

← плохо

↑ хорошо

[Random Search for Hyper-Parameter Optimization'11]

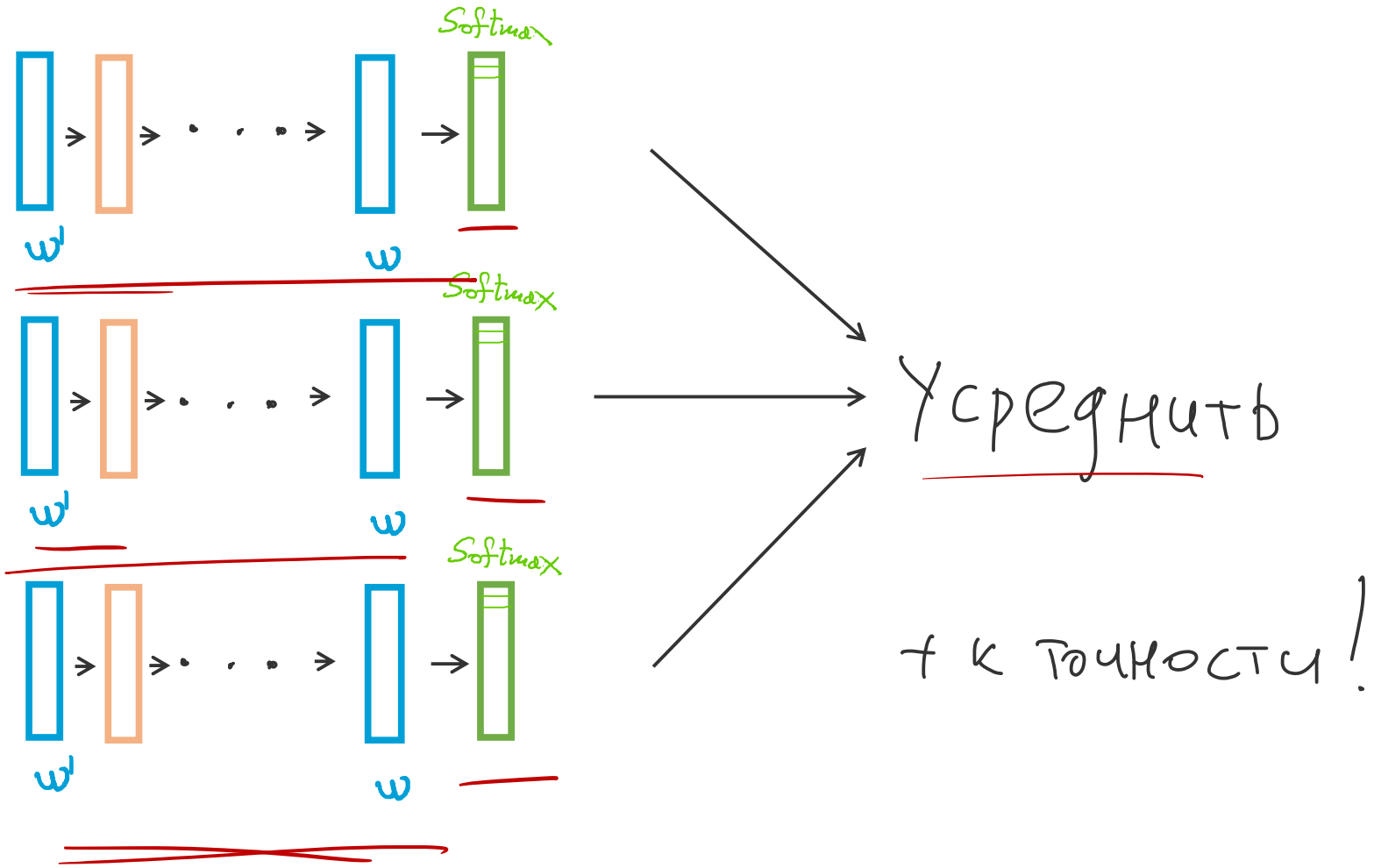
Короче!
Начинать с:

- Вычитаем среднее из данных
- Оптимизатор - Adam
- Batch Normalization – добро
- Learning rate annealing на плато
- Перебираем гиперпараметры
  - Важнее всего – learning rate
- **Смотрим на графики тренировки!**

Ансабль моделей
Model ensemble

Откуда брать разнообразие?

- Разные модели и подходы
- Cross-validation folds
- Креатив!



LR Range test
+ Cyclical Learning Rate

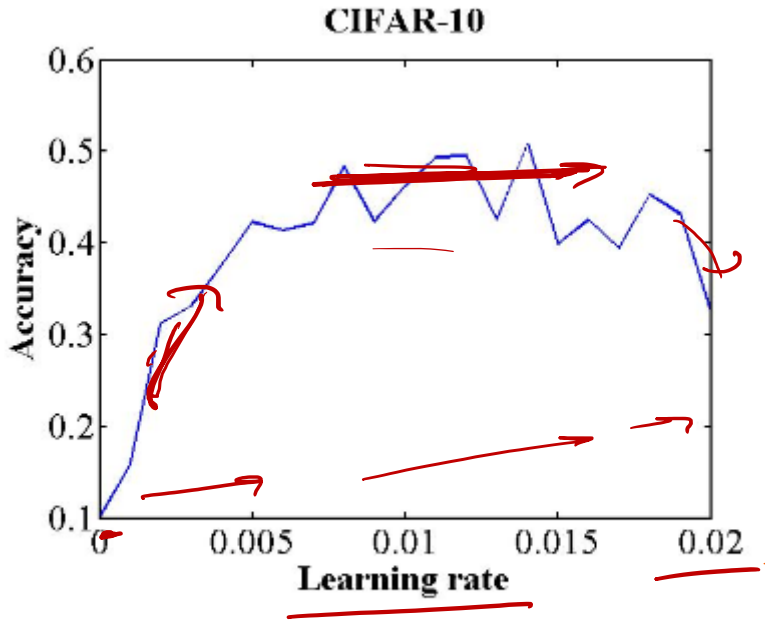


Figure 3. Classification accuracy as a function of increasing learning rate for 8 epochs (LR range test).

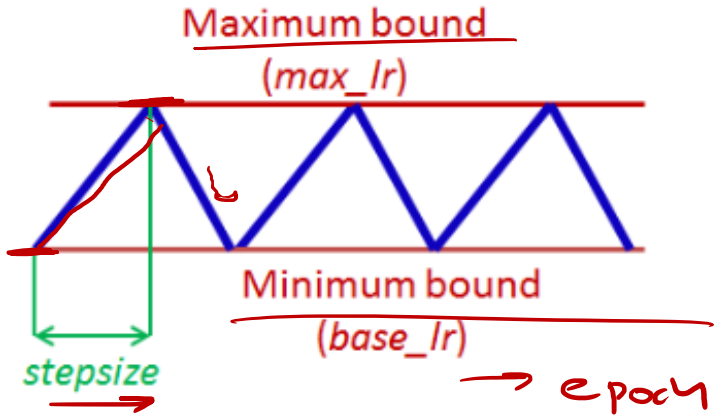
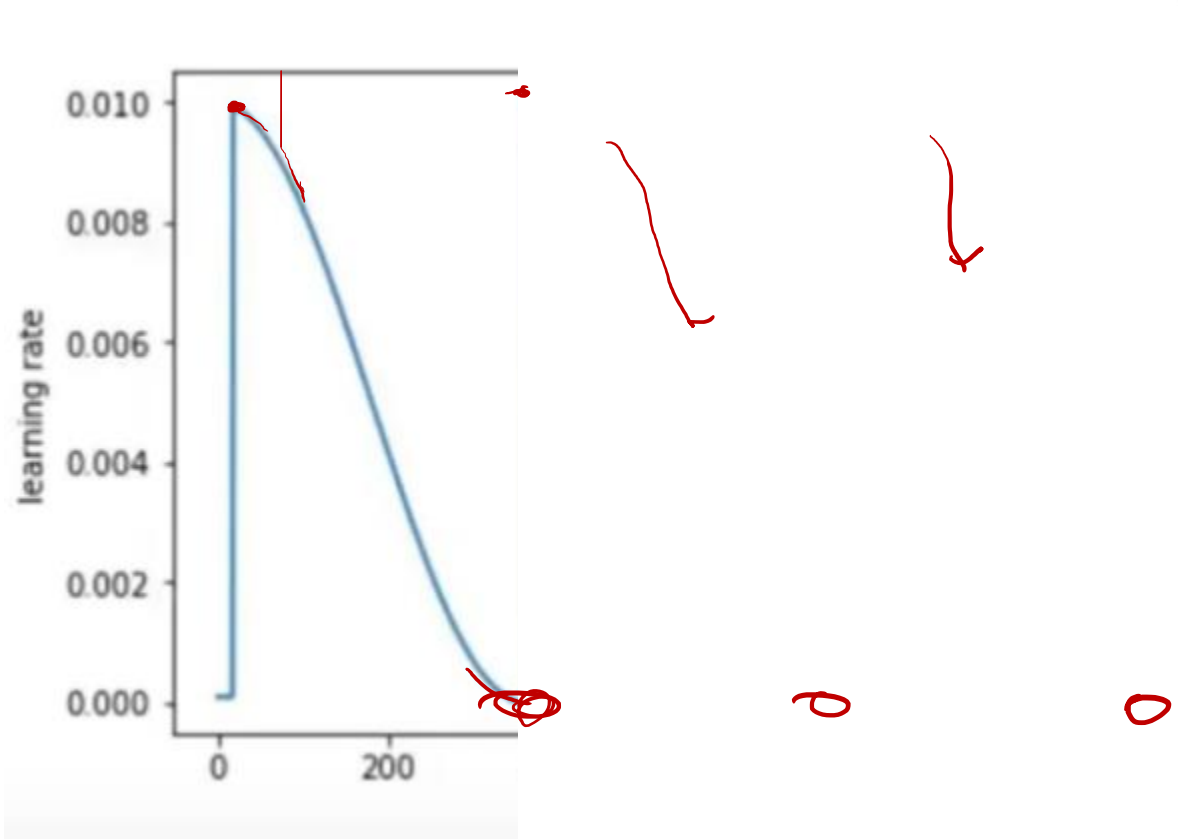


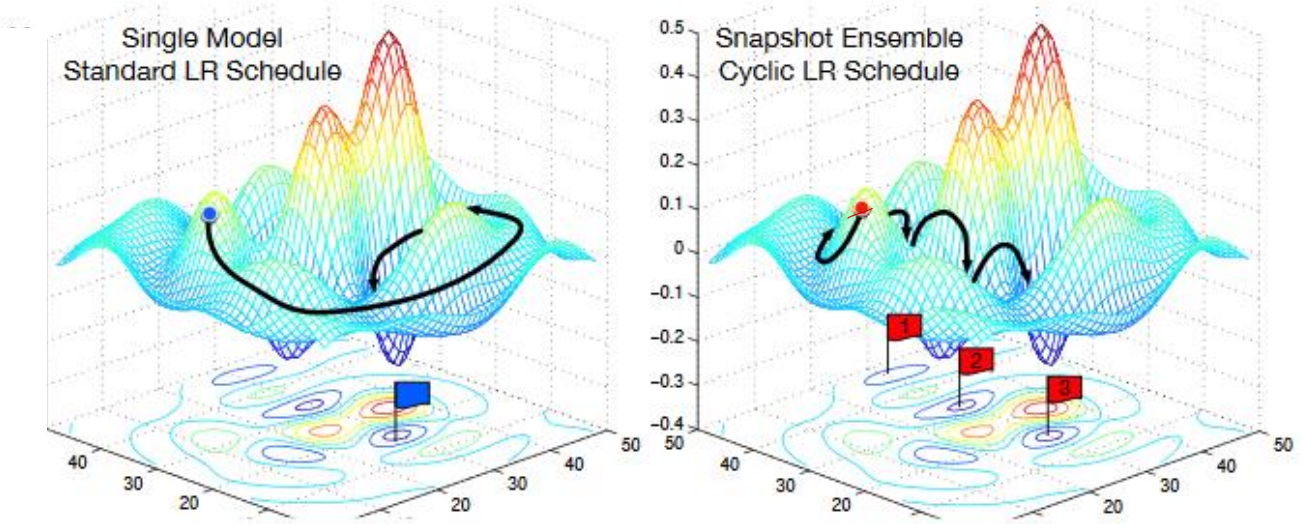
Figure 2. Triangular learning rate policy. The blue lines represent learning rate values changing between bounds. The input parameter *stepsize* is the number of iterations in half a cycle.

[Cyclical Learning Rates for Training Neural Networks'15]

Cosine learning rate

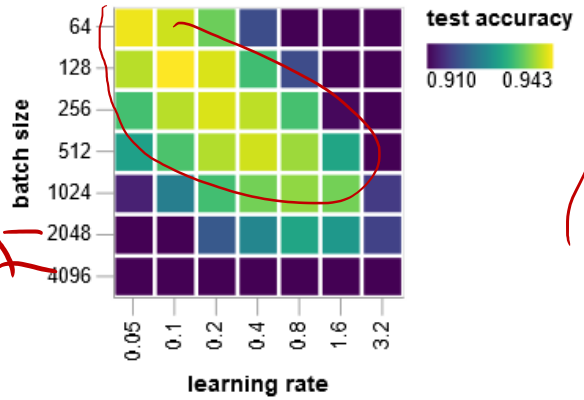
+ Stochastic Gradient Descent with Restarts (SGDR)'16
+ Snapshot Ensembles'17





Medium Post

Инженерный подход к перебору гиперпараметров



α = learning rate

ρ = momentum

λ = lr

$\left(\frac{\lambda\alpha}{1-\rho}, \rho, \alpha\right)$

train run	λ	ρ	α	$\frac{\lambda\alpha}{1-\rho}$	test acc
0	0.001	0.5	.01	.00002	30.6%
10	**0.256**	0.5	.01	.00512	93.0%
13	**0.128**	**0.75**	.01	.00512	93.2%
17	**1.024**	0.75	**.00125**	.00512	93.9%
20	**0.512**	0.75	.00125	.00256	94.1%
22	**0.256**	**0.875**	.00125	.00256	94.2%

[How to Train Your Resnet 5]


STAND BACK
WE'RE TRYING THIS IN PRODUCTION

Research vs Deployment

Microsoft CNTK

Caffe

Caffe2

dmlc mxnet

theano

TensorFlow

K

torch

PYTORCH

Deployment

- Интегрируется не только с Python
- Высокие требования по стабильности и надежности
- Инструменты для запуска модели на сервере
  - Например, Tensorflow Serving
- Оптимизировано под железо, где запускается модель
  - Например, телефоны

Что делать, если тренировал в одном фреймворке, а в прод надо другой?

конвертор