

курс «Глубокое обучение»

Борьба с переобучением в нейронных сетях

Александр Дьяконов

17 февраля 2020 года

Борьба с переобучением

Очень много параметров \Rightarrow переобучение

- **Нормировки (Normalization of Data)**
- **Инициализация весов**
- **Верификация – ранний останов (Early Stopping)**
- **Настройка темпа обучения (Learning Rate)**
- **Мини-батчи (Mini-Batches) / Batch-обучение**
- **Продвинутая оптимизация**
- **Регуляризация + Weight Decay**
- **Max-norm-регуляризация**
- **Dropout**
- **Увеличение выборки + Расширение выборки (Data Augmentation)**
- **Обрезка градиентов (Gradient clipping)**

Борьба с переобучением

- **Доучивание уже настроенных нейросетей (Pre-training)**
- **Unsupervised Learning**
- **Техника зануления весов (разреживания НС)**
- **Использование специальных архитектур под задачу**
(например, использующих локальность – свёрточных НС)
- **зашумление (inject noise)**

Нормировки (Normalization of Data)

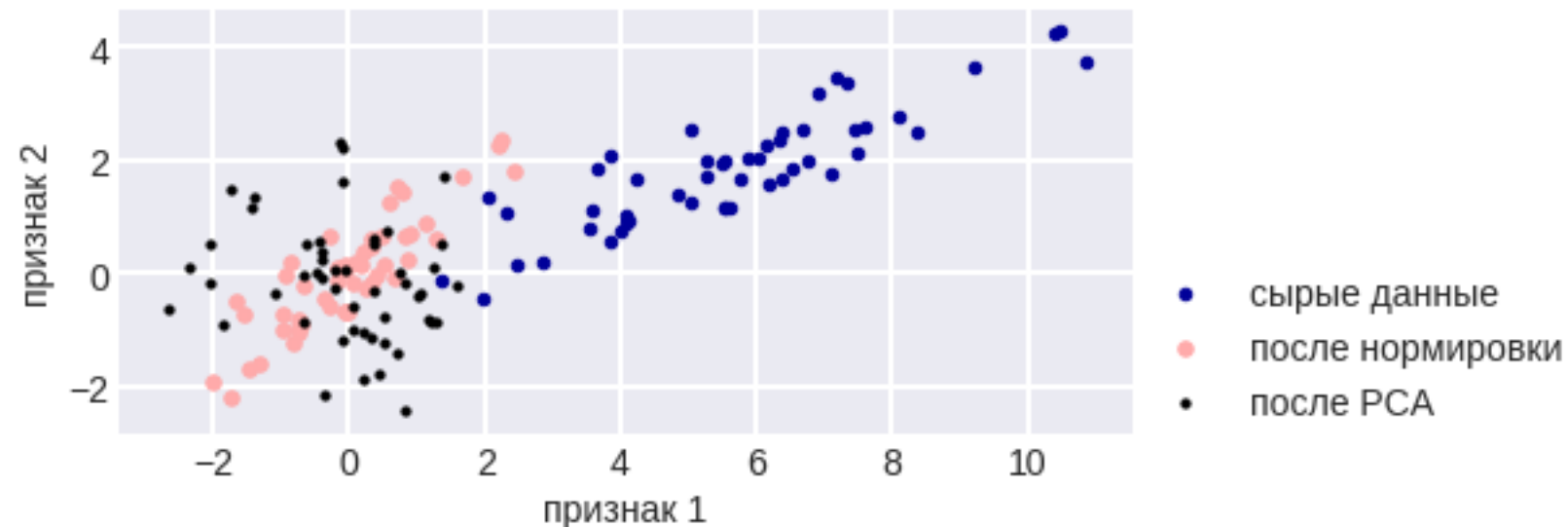
Признак $X = (X_1, \dots, X_m)$

$$\mu = \frac{1}{m} \sum_{i=1}^m X_i \text{ среднее признака}$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^m (X_i - \mu)^2 \text{ дисперсия признака}$$

$$X = \frac{X - \mu}{\sqrt{\sigma^2}} \text{ нормировка}$$

иногда м.б. PCA



Инициализация весов

- **нарушение симметричности**
(чтобы нейроны были разные)
- **недопустить «насыщенности» нейрона**
(почти всегда близок к нулю или 1)
- **ключевая идея – входы на все слои должны иметь одинаковую дисперсию**
(для избегания «насыщения» нейронов)

смещения := 0 (зависит от того, где смещение; если на выходе...)

Xavier initialization

$$w_{ij}^{(k)} \sim U \left[-\sqrt{\frac{6}{n_{\text{in}}^{(k)} + n_{\text{out}}^{(k)}}}, +\sqrt{\frac{6}{n_{\text{in}}^{(k)} + n_{\text{out}}^{(k)}}} \right]$$

[Glorot & Bengio, 2010]

Распределение, чтобы $Dw_{ij}^{(k)} = \frac{2}{n_{\text{in}}^{(k)} + n_{\text{out}}^{(k)}}$

Формула выведена в предположении, что нет нелинейностей...

$$z^{(k+1)} = f(W^{(k)} z^{(k)}) \equiv W^{(k)} z^{(k)}$$

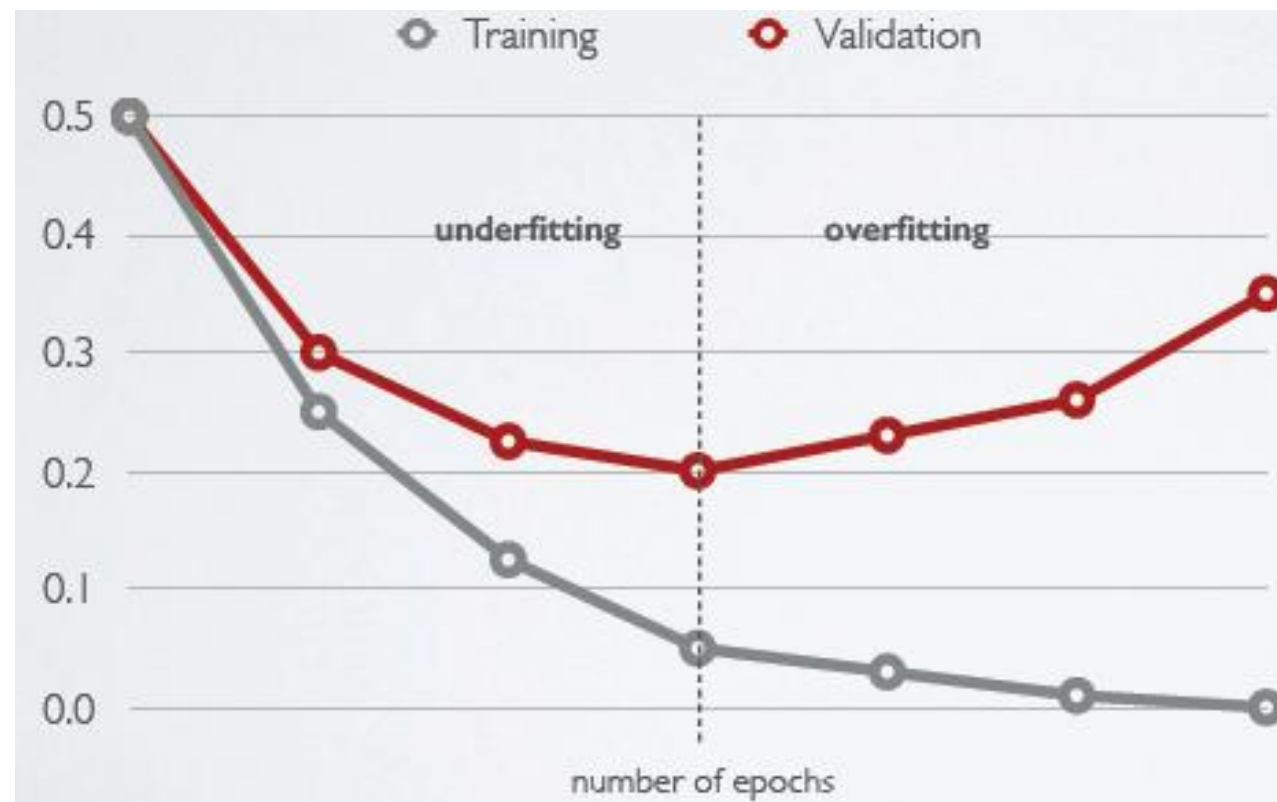
МОЖНО ПОСЧИТАТЬ СМ.

https://www.youtube.com/watch?v=PjS2y8LBMLc&list=PLrCZzMib1e9oOGNLh6_d65HyfdqIJwTQP&index=5

Плохо для ReLu – [He et al., 2015]

Верификация – ранний останов (Early Stopping)

Смотрим ошибку на отложенной выборке!
Выбираем итерацию, на которой наименьшая ошибка.



Настройка темпа обучения (Learning Rate)

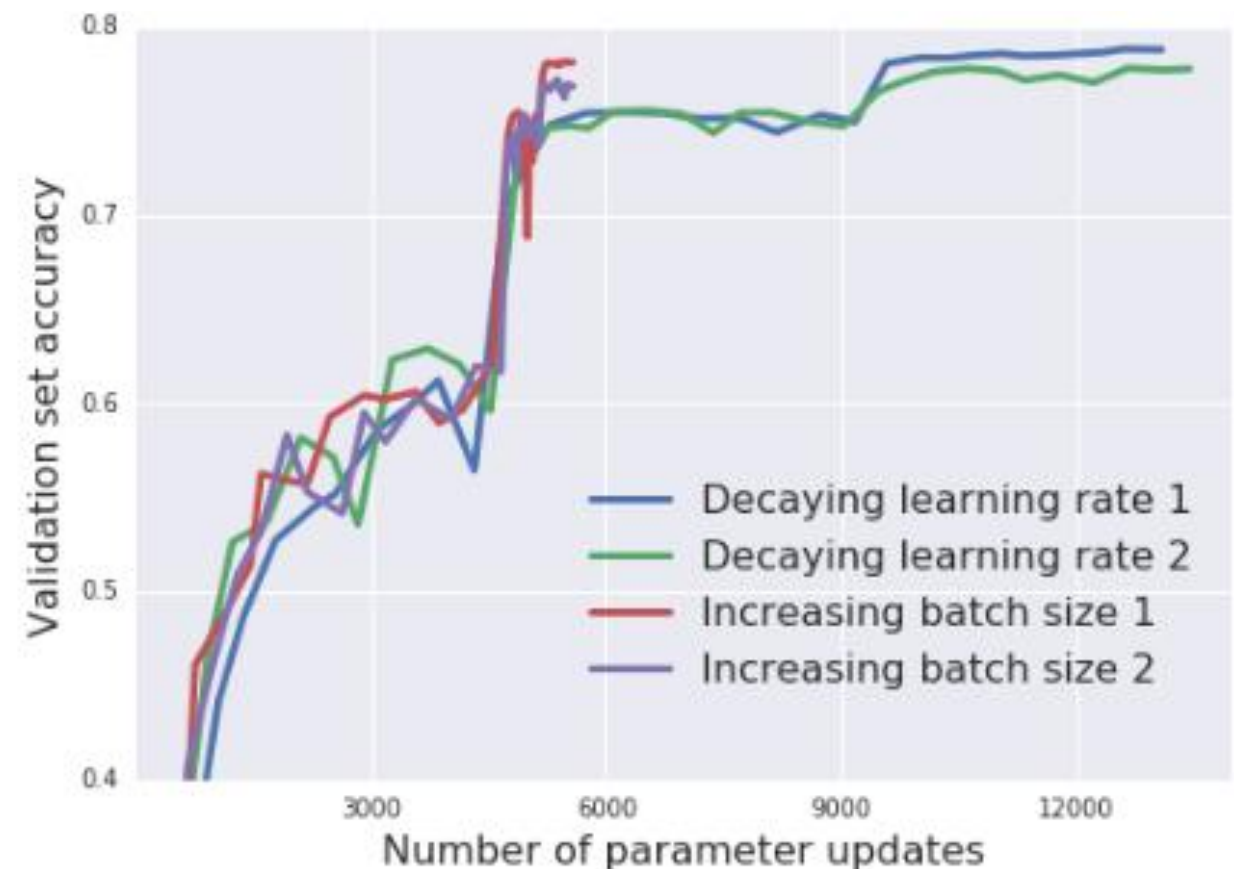
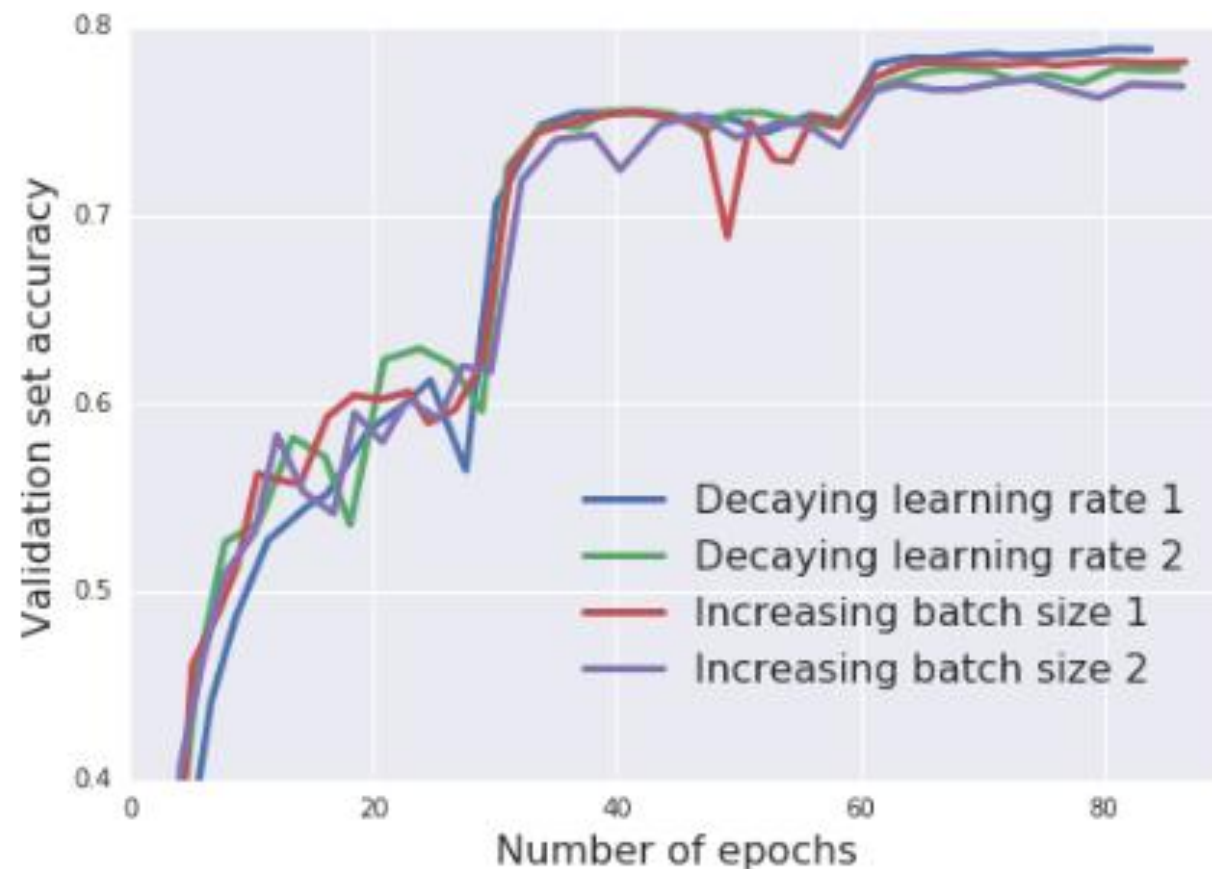
Мини-батчи (mini-batches) / Batch-обучение

$$w^{(t+1)} = w^{(t)} - \frac{\eta}{|I|} \sum_{i \in I} \nabla [l(a(x_i | w^{(t)}), y_i) + \lambda R(w^{(t)})]$$

- Градиенты не такие случайные (оцениваем по подвыборке)
- Можно вычислять быстрее (на современных архитектурах)
Можно делать максимальный батч, который влезает в память
- Можно делать нормировку по батчу
- Немного противоречит теории ?!

м.б. специально организовывать батчи
(должны содержать представителей всех классов)

Мини-батчи (mini-batches) / Batch-обучение



**увеличение размера батча – тот же эффект,
что и уменьшение темпа обучения**

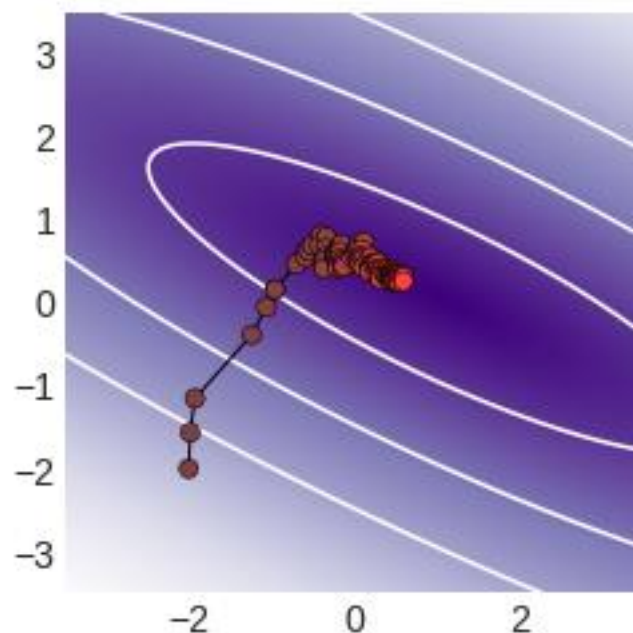
Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, Quoc V. Le «Don't Decay the Learning Rate, Increase the Batch Size»

<https://arxiv.org/abs/1711.00489>

Продвинутая оптимизация

Обучение: стохастический градиент

$$w^{(t+1)} = w^{(t)} - \eta \nabla L^{(t)}(w^{(t)})$$



Эпоха – проход по всей обучающей выборке

Надо случайно перемешивать данные перед каждой эпохой

Продвинутая оптимизация

стохастический градиент с моментом (momentum)

$$m^{(t+1)} = \rho m^{(t)} + \nabla L^{(t)}(w^{(t)})$$

$$w^{(t+1)} = w^{(t)} - \eta m^{(t+1)}$$

добавление инерции

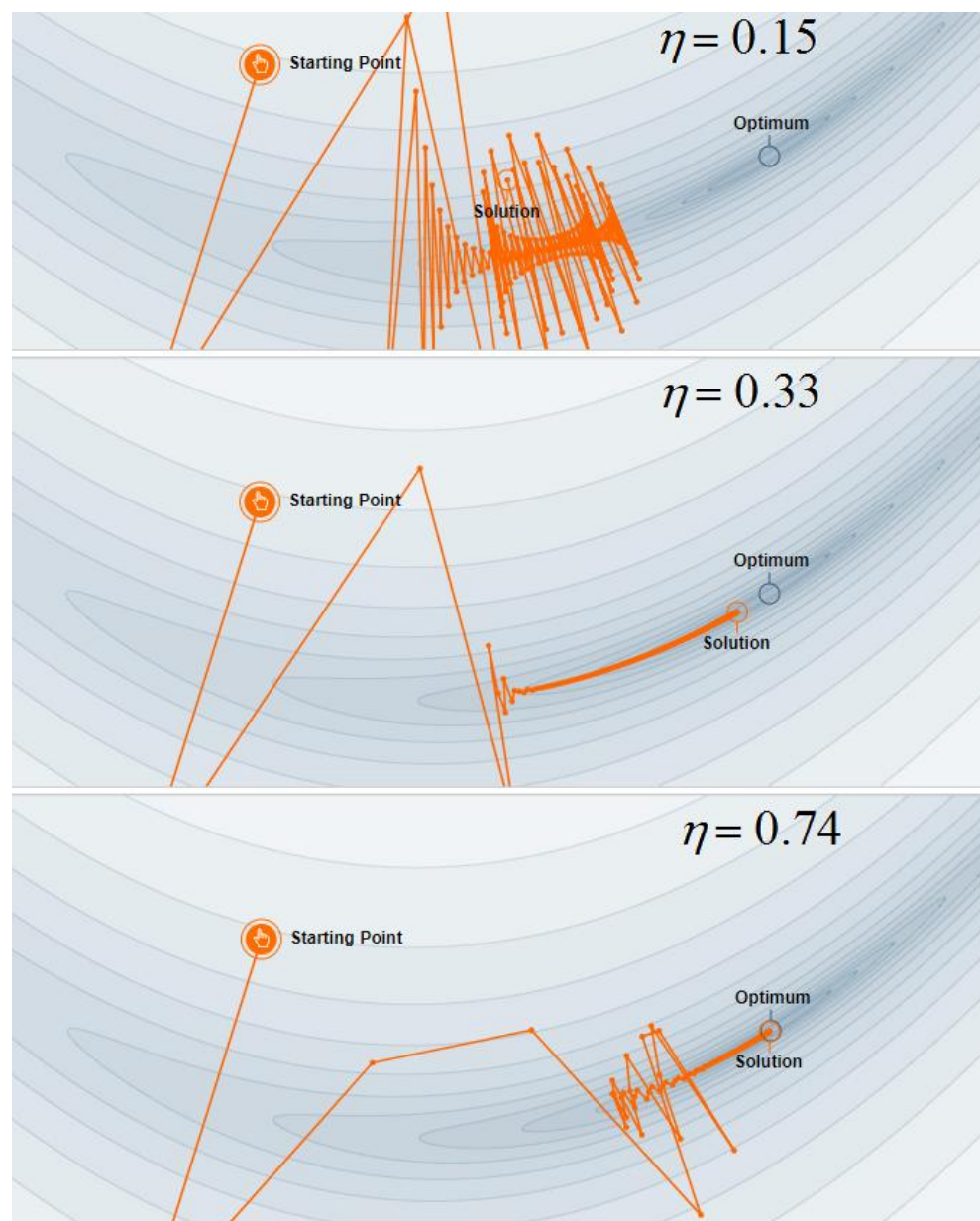
метод Нестерова

$$m^{(t+1)} = \rho m^{(t)} + \nabla L^{(t)}(w^{(t)} - \eta m^{(t)})$$

$$w^{(t+1)} = w^{(t)} - \eta m^{(t+1)}$$

- **градиент случаен (зависит от батча)**
добавляем градиенту инертность
- **уровни функции могут быть сильно вытянуты**
adam
- **все параметры разные**
скорость обучения – для каждого параметра

Иллюстрация СГ с моментом



Добавление инерции может помочь, когда

- **линии уровня вытянуты...**
- **для проскакивания седловых точек**

<https://distill.pub/2017/momentum/>

Продвинутая оптимизация – Адаптивная

Adagrad [Duchi и др., 2011]

$$v_i^{(t+1)} = v_i^{(t)} + (\nabla_i L^{(t)}(w^{(t)}))^2$$
$$w_i^{(t+1)} = w_i^{(t)} - \frac{\eta}{\sqrt{v_i^{(t+1)} + \varepsilon}} \nabla_i L^{(t)}(w^{(t)})$$

RMSprop = Adagrad + MA [Hinton, 2012]

$$v_i^{(t+1)} = \beta v_i^{(t)} + (1 - \beta)(\nabla_i L^{(t)}(w^{(t)}))^2$$
$$w_i^{(t+1)} = w_i^{(t)} - \frac{\eta}{\sqrt{v_i^{(t+1)} + \varepsilon}} \nabla_i L^{(t)}(w^{(t)})$$

Root Mean Squared Propagation (RMSprop)

Продвинутая оптимизация – Адаптивная

Adam = RMSprop + momentum

$$m_i^{(t+1)} = \alpha m_i^{(t)} + (1 - \alpha) \nabla_i L^{(t)}(w^{(t)})$$

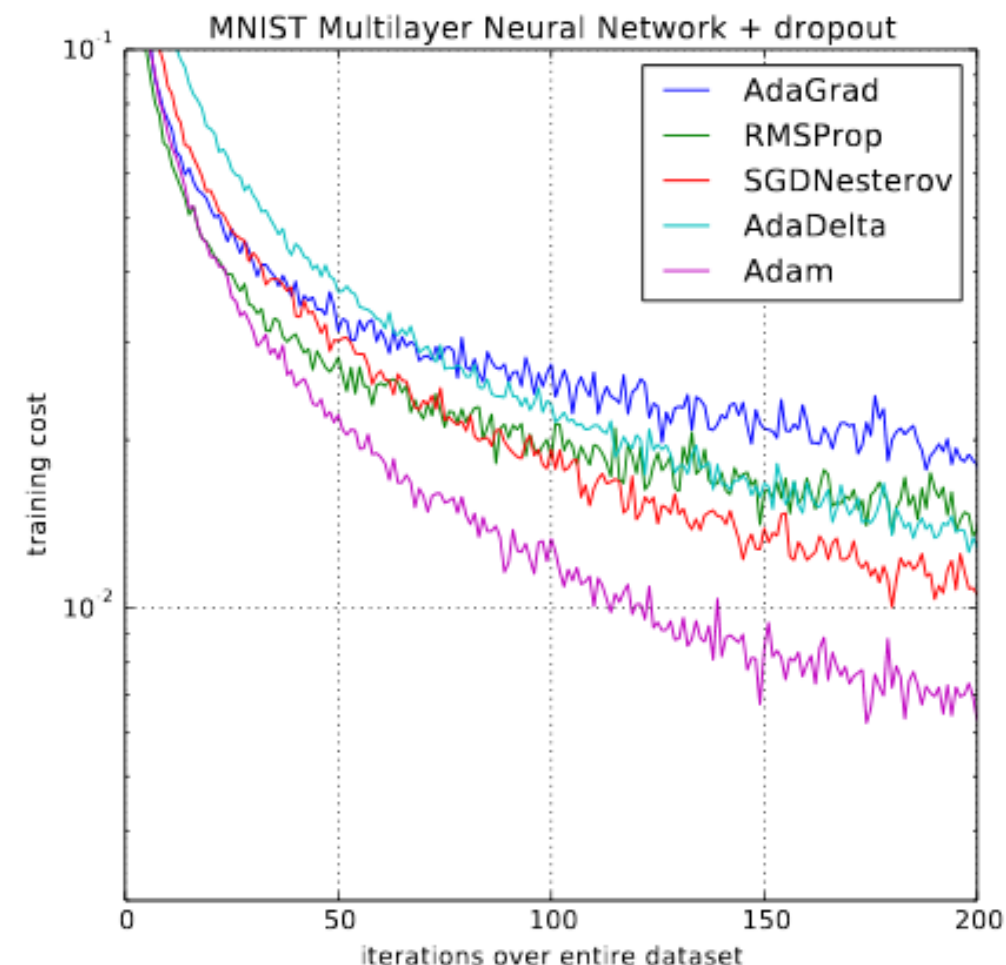
$$v_i^{(t+1)} = \beta v_i^{(t)} + (1 - \beta) (\nabla_i L^{(t)}(w^{(t)}))^2$$

корректировка смещения:

$$\hat{m}_i^{(t+1)} = m_i^{(t+1)} / (1 - \alpha^t)$$

$$\hat{v}_i^{(t+1)} = v_i^{(t+1)} / (1 - \beta^t)$$

$$w_i^{(t+1)} = w_i^{(t)} - \frac{\eta}{\sqrt{\hat{v}_i^{(t+1)} + \varepsilon}} \hat{m}_i^{(t)}$$



Adam = «Adaptive Moment Estimation»

18k ссылок, неверное доказательство сходимости

[Kingma, Ba, 2014 <https://arxiv.org/abs/1412.6980>]

Продвинутая оптимизация

AdaDelta (усл. Adagrad)

$$v_i^{(t+1)} = v_i^{(t)} + (\nabla_i L^{(t)}(w^{(t)}))^2$$

$$\Delta w_i^{(t+1)} = -\frac{\sqrt{\eta_i^{(t)} + \varepsilon}}{\sqrt{v_i^{(t+1)} + \varepsilon}} \nabla_i L^{(t)}(w^{(t)})$$

$$w_i^{(t+1)} = w_i^{(t)} + \Delta w_i^{(t+1)}$$

$$\eta_i^{(t+1)} = \gamma \eta_i^{(t)} + (1 - \gamma)(\Delta w_i^{(t+1)})^2$$

Подбор гиперпараметров очень важен!
Подбор метода оптимизации очень важен!

Зашумление

Например, добавление к градиенту шума (вываливаемся из седловых точек)

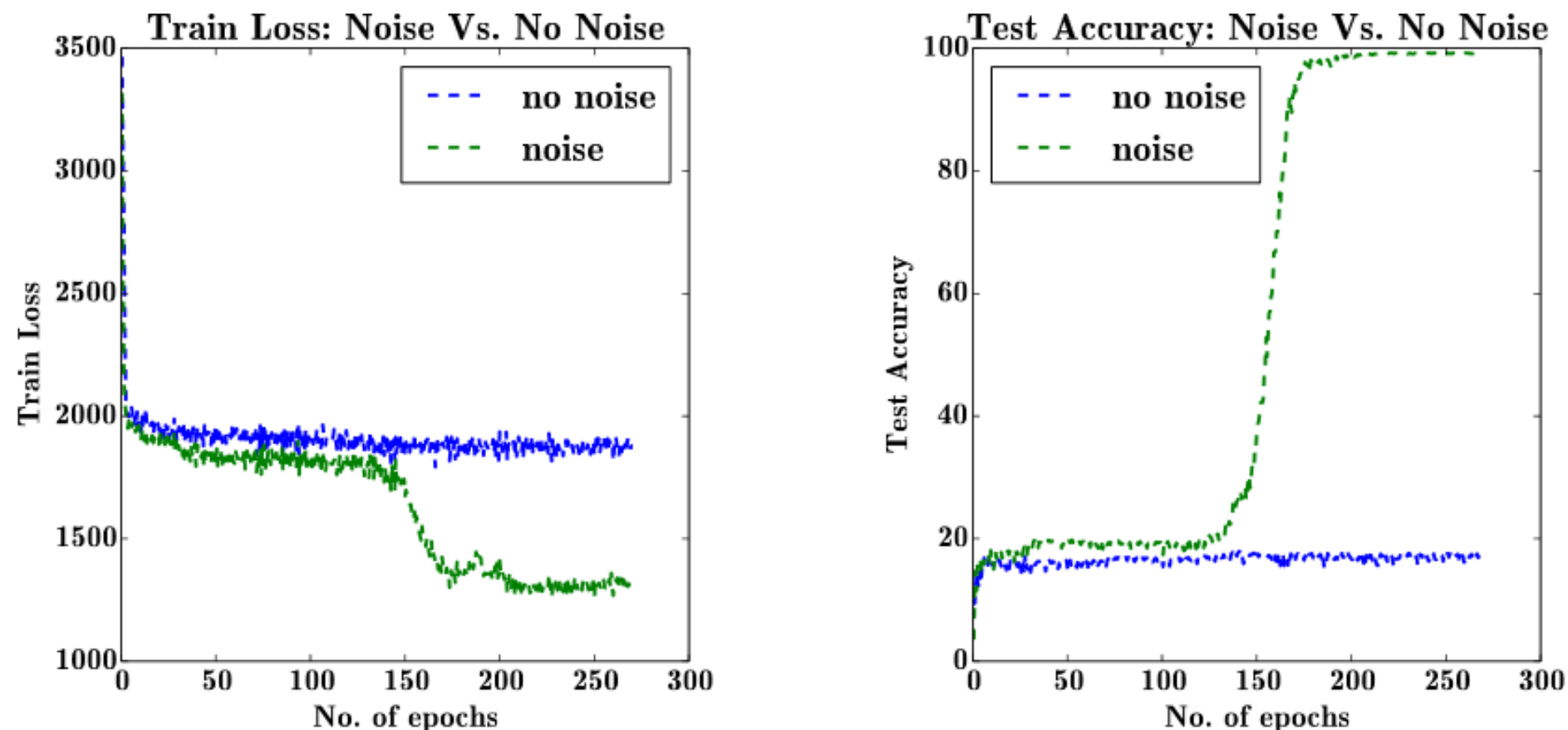


Figure 2: Noise vs. No Noise in our experiment with tables containing 5 columns. The models trained with noise generalizes almost always better.

Neelakantan et al. «Adding gradient noise improves learning for very deep networks» // <https://arxiv.org/abs/1511.06807>

Anandkumar and Ge «Efficient approaches for escaping higher order saddle points in non-convex optimization» <https://arxiv.org/abs/1602.05908>

Регуляризация + Weight Decay

Не применяется к весам к константным входам (смещениям)

L2, L1 – регуляризация

Уменьшение весов (вид регуляризации)

$$w^{(t+1)} = (1 - \lambda)w^{(t)} - \eta \nabla L^{(t)}(w^{(t)})$$

На самом деле это L2-регуляризация:

$$\begin{aligned} \nabla(L(w) + \lambda \|w\|^2) &= \nabla L(w) + \lambda w \\ w^{(t+1)} &= w^{(t)} - \eta(\nabla L(w^{(t)}) + \lambda w^{(t)}) = (1 - \lambda\eta)w^{(t)} - \eta \nabla L(w^{(t)}) \end{aligned}$$

```
from keras.regularizers import l2 # L2-regularisation
l2_lambda = 0.0001
conv_1 = Convolution2D(conv_depth, kernel_size, kernel_size,
                        border_mode='same', W_regularizer=l2(l2_lambda),
                        activation='relu')(inp)
```

Max-norm-регуляризация

Для каждого нейрона ограничиваем норму весов:

$$\|w_{\text{neuron}}\| \leq c$$

если превысила – проекция

Results from MNIST (handwritten digit recognition)

Method	Unit Type	Architecture	Error %
Standard Neural Net (Simard et al., 2003)	Logistic	2 layers, 800 units	1.60
SVM Gaussian kernel	NA	NA	1.40
Dropout NN	Logistic	3 layers, 1024 units	1.35
Dropout NN	ReLU	3 layers, 1024 units	1.25
Dropout NN + max-norm constraint	ReLU	3 layers, 1024 units	1.06
Dropout NN + max-norm constraint	ReLU	3 layers, 2048 units	1.04
Dropout NN + max-norm constraint	ReLU	2 layers, 4096 units	1.01
Dropout NN + max-norm constraint	ReLU	2 layers, 8192 units	0.95
Dropout NN + max-norm constraint (Goodfellow et al., 2013)	Maxout	2 layers, (5 × 240) units	0.94

<http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf>

Оптимизаторы

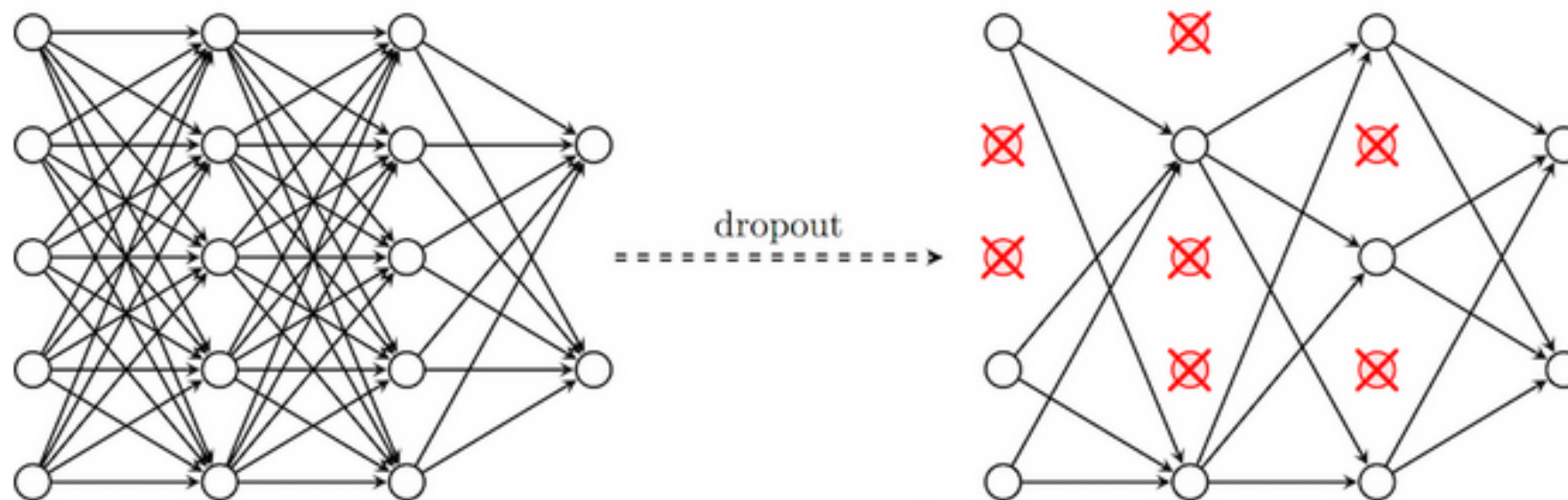
«An overview of gradient descent optimization algorithms»

<http://runder.io/optimizing-gradient-descent/>

Rectified Adam (RAdam, 2019)

<https://medium.com/@lessw/new-state-of-the-art-ai-optimizer-rectified-adam-radam-5d854730807b>

Dropout

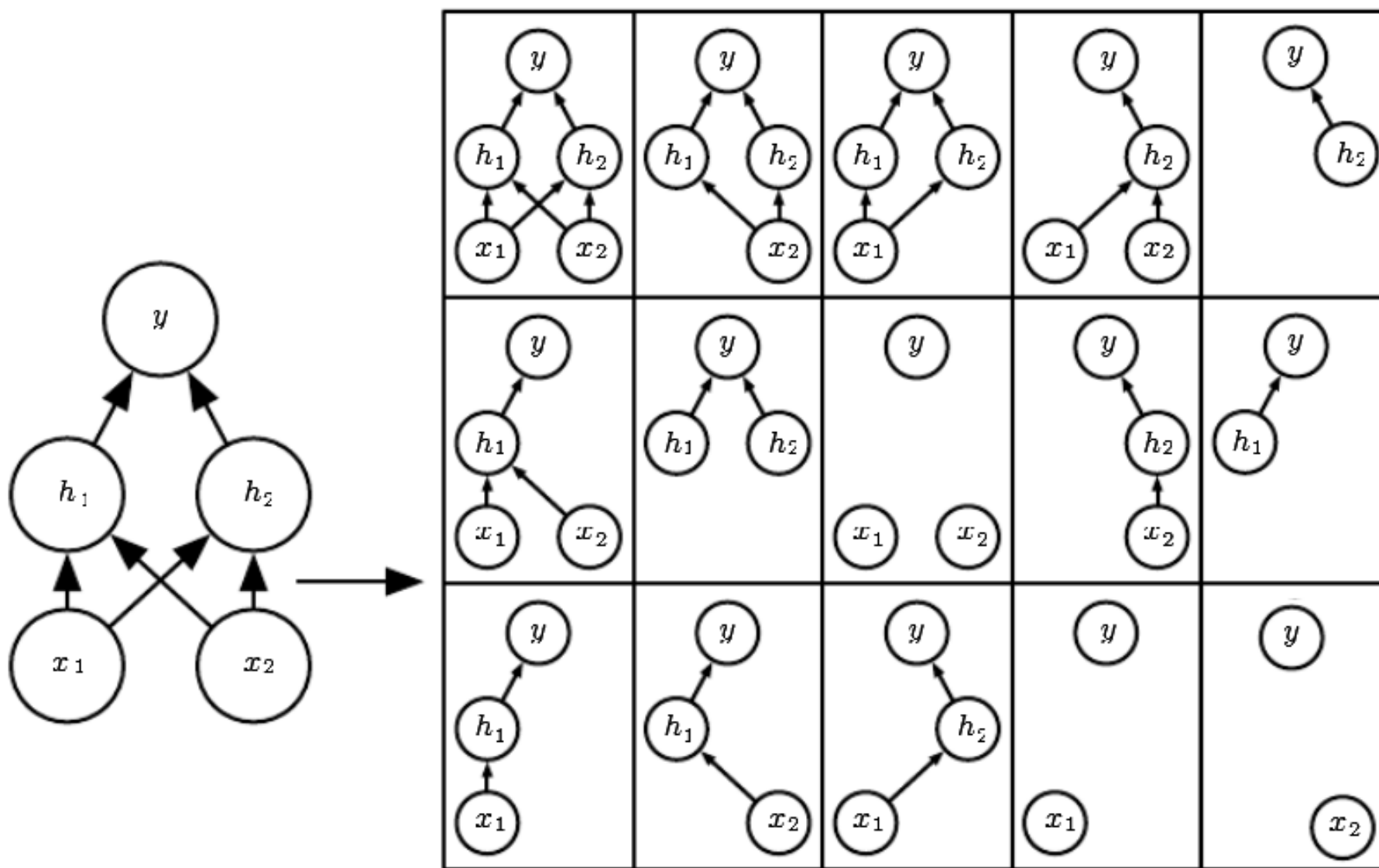


случайное обнуление активаций
~ выбрасывание нейронов из сети
не выбрасываем из последнего слоя

Обычно в полносвязных слоях!

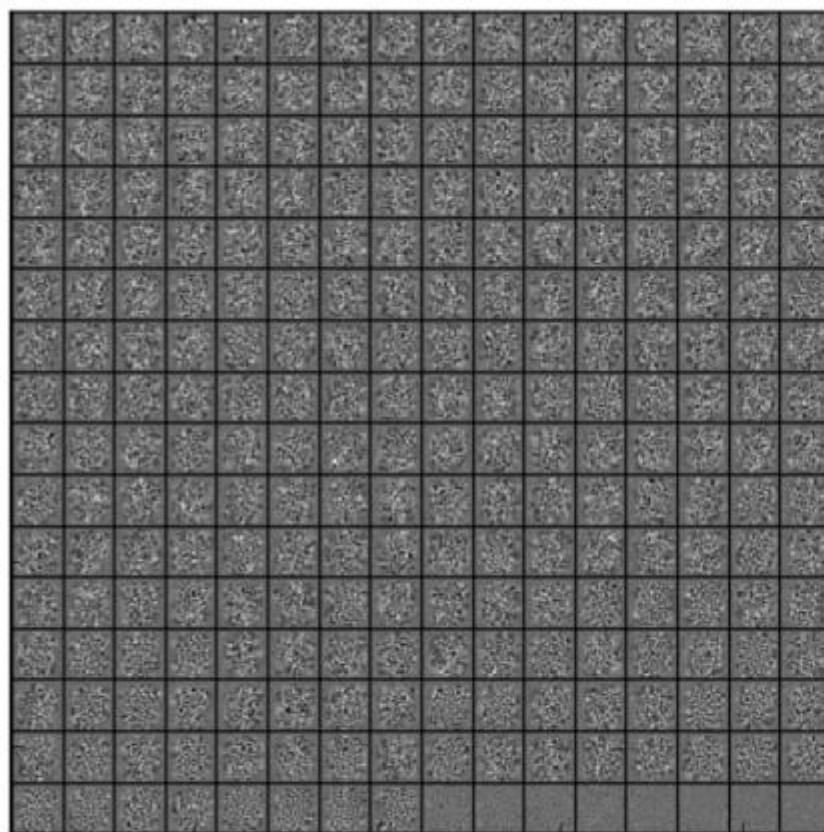
Отключают в режиме теста – выход умножается на вероятность выбрасывания.

Dropout

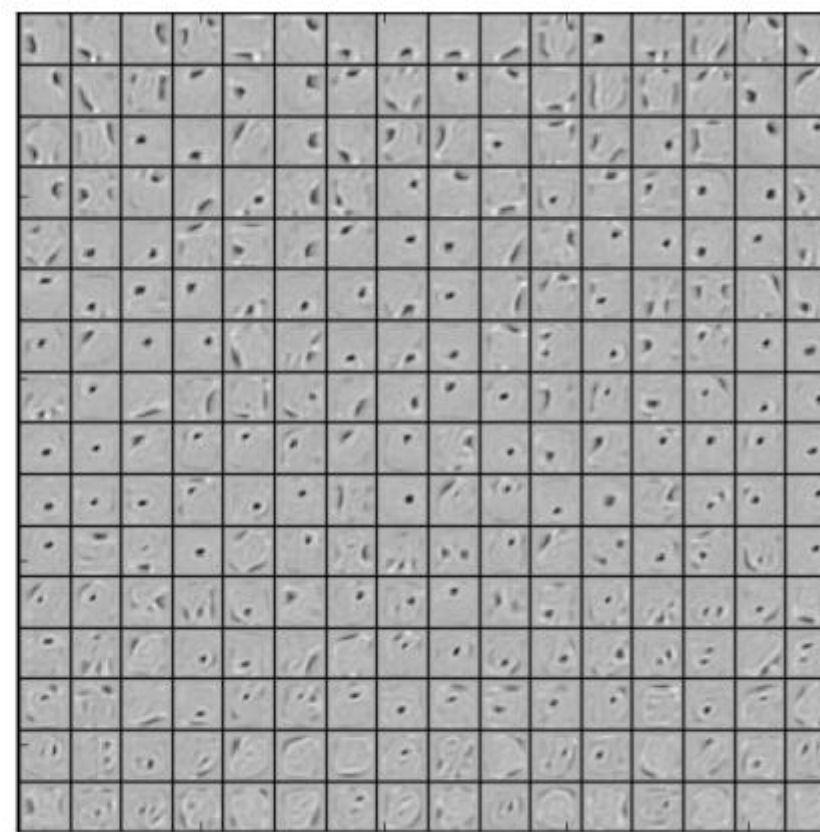


В отличие от бэгинга все модели делят параметры (не независимы) [DLbook]

Dropout



(a) Without dropout



(b) Dropout with $p = 0.5$.

MNIST: автокодировщик с 1 скрытым слоем и 256 линейными нейронами
[Srivastava, 2014]

Inverted Dropout

Во время обучения выход нейрона умножается на

$$\frac{1}{1-p}$$

p – вероятность выбрасывания

(в обычном варианте – умножаем на p во время теста)

Во время тестирования ничего не делаем...

SpatialDropout

выбрасываем каналы, а не нейроны

[Tompson et al. (2015)]

но нельзя реализовать как отдельный слой;)

DropConnect

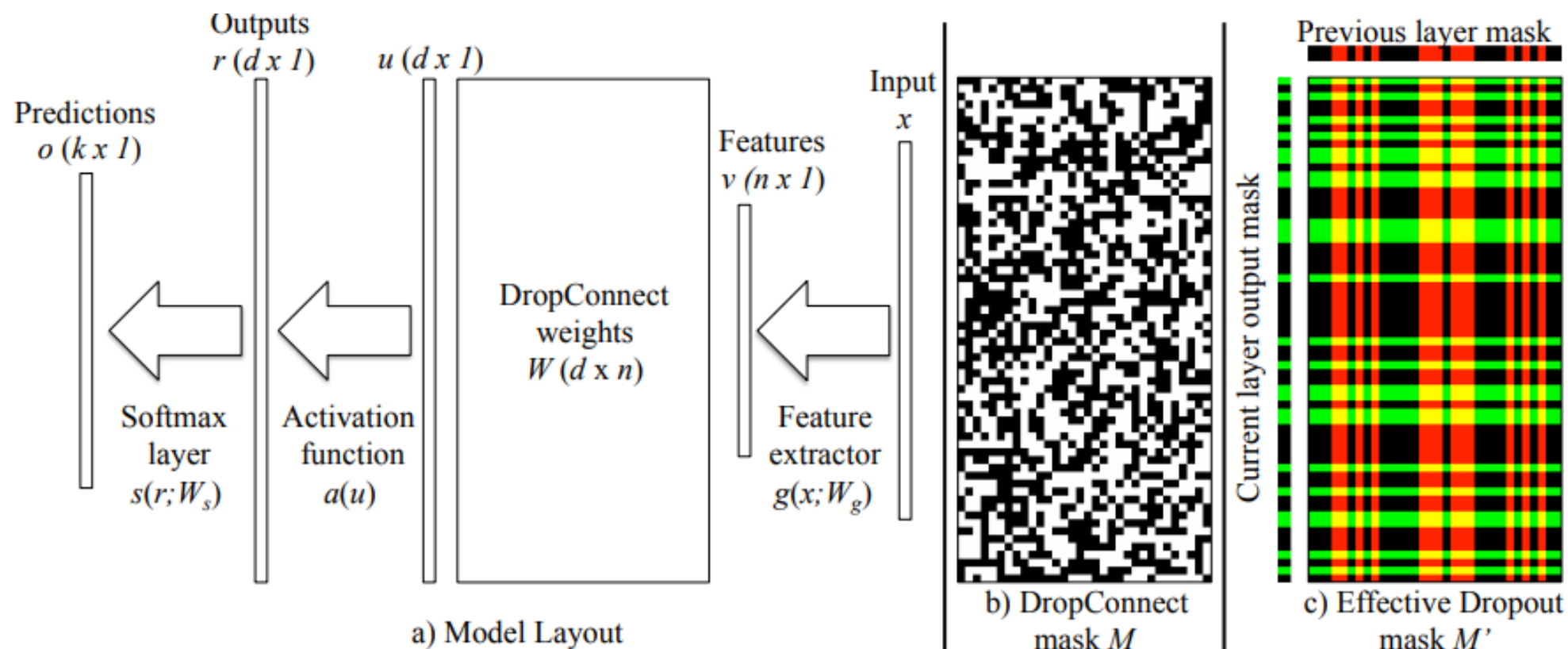


Figure 1. (a): An example model layout for a single DropConnect layer. After running feature extractor $g()$ on input x , a random instantiation of the mask M (e.g. (b)), masks out the weight matrix W . The masked weights are multiplied with this feature vector to produce u which is the input to an activation function a and a softmax layer s . For comparison, (c) shows an effective weight mask for elements that Dropout uses when applied to the previous layer's output (red columns) and this layer's output (green rows). Note the lack of structure in (b) compared to (c).

Зануление отдельных весов, а не нейронов

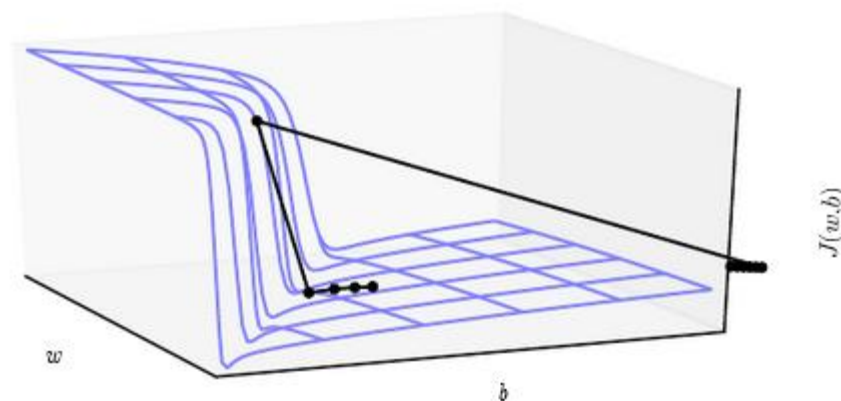
<https://cs.nyu.edu/~wanli/dropc/>

Обрезка градиентов (Gradient clipping)

$$g = \frac{\partial L}{\partial w}$$

$$g^{\text{new}} = \frac{\min(\theta, \|g\|)}{\|g\|} g$$

Работает пока дисперсия градиента маленькая...



Обрезка помогает, когда попадаем на утёс...

Pascanu, Mikolov, Bengio для RNN

Батч-нормализация (Batch normalization)

– метод адаптивной перепараметризации

Проблема:

**градиент – как изменять параметры,
при условии, что вся остальная сеть **не меняется****

**трудно предсказать, насколько изменится какое-то значение
(оно зависит от всех предыдущих в суперпозиции)**

**Covariate shift – изменение распределений входов во время обучения
Надо уменьшить это изменение в скрытых слоях!**

Ioffe and Szegedy, 2015 <https://arxiv.org/abs/1502.03167>

Батч-нормализация (Batch normalization)

мини-батч $\{x_i\}_{i=1}^m$

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i \quad \text{среднее по мини-батчу}$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \quad \text{дисперсия по мини-батчу}$$

$$x_i^{\text{new}} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \quad \text{нормировка}$$

$$y_i = \gamma x_i^{\text{new}} + \beta \quad \text{растяжение и сдвиг}$$

Надо определить параметры γ и β

Зачем центрировать, а потом смещать?

Так эффективнее обучать: смещение является параметром в чистом виде

При обучении

среднее и дисперсия – по мини-батчу

При тесте

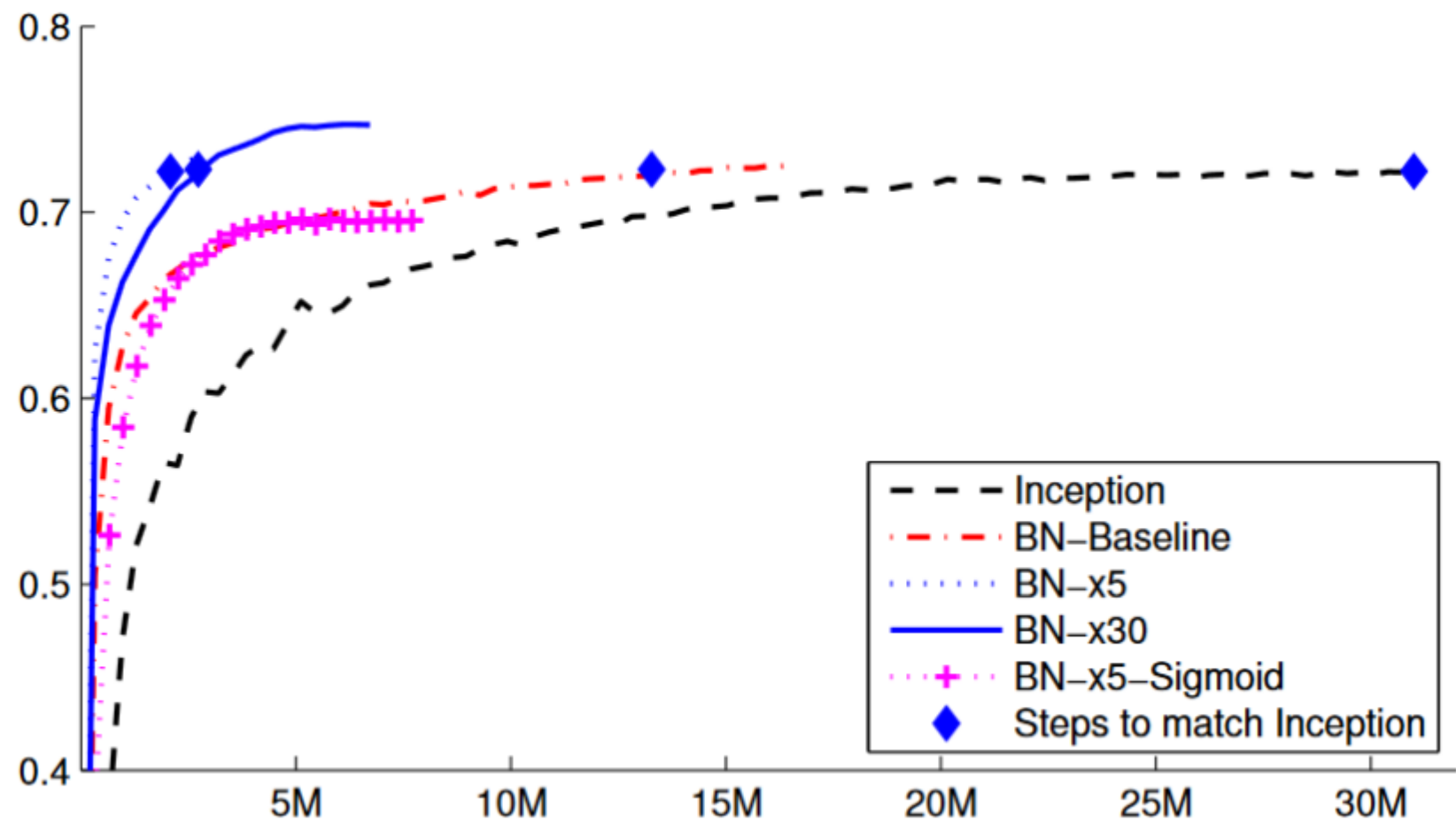
среднее и дисперсия –

- **по обучению**
- **усреднение значений, что были во время обучения**
- **экспоненциальное среднее $=*$**

нормализация перед входом в каждый слой (иногда до активации, иногда после)

- **можно увеличить скорость обучения**
 - **можно убрать dropout**
 - **можно уменьшить регуляризацию**
- **можно использовать более глубокие сети**

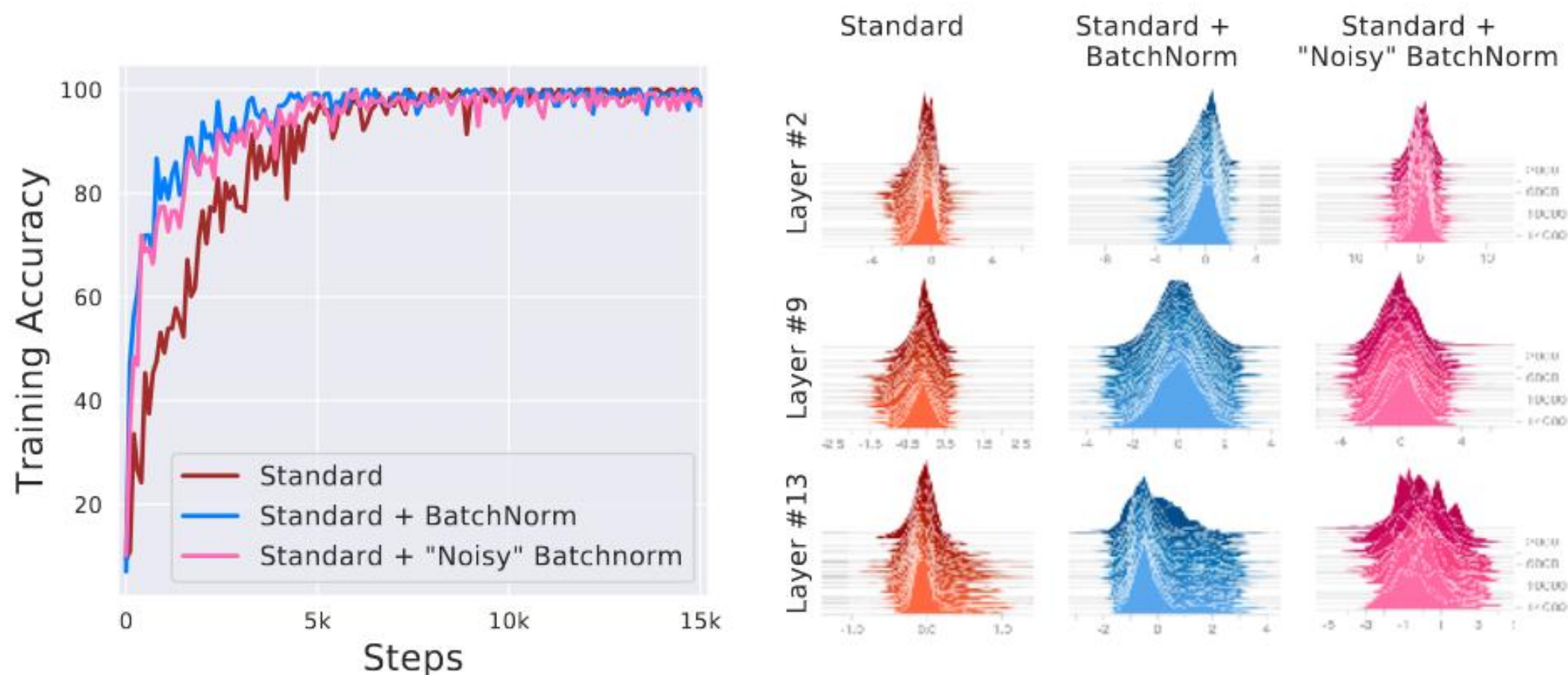
Батч-нормализация (Batch normalization)



Обучение Inception с / без батч-нормализацией

Батч-нормализация (Batch normalization)

– обоснование эффективности – открытая проблема



дело не в изменении распределения, а в сглаживании функции

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, Aleksander Madry

How Does Batch Normalization Help Optimization? 2018 // <https://arxiv.org/abs/1805.11604>

Батч-нормализация (Batch normalization)

```
from keras.layers.normalization import BatchNormalization

inp_norm = BatchNormalization(axis=1)(inp) # применить к inp
# conv_1 = Convolution2D(...)(inp_norm)
conv_1 = BatchNormalization(axis=1)(conv_1) # применить к conv_1
```

Batch Normalization (BN) – если размер батча маленький, то всё плохо...
мало статистики

Хотя есть мнение, что маленький батч сильнее регуляризует!

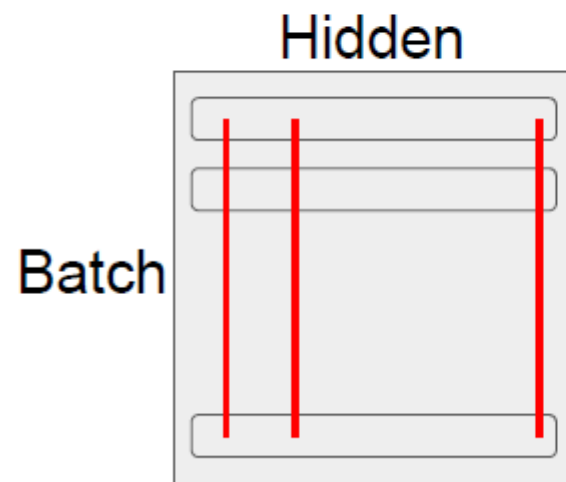
дальше ещё варианты нормализации

Ещё... нормализации

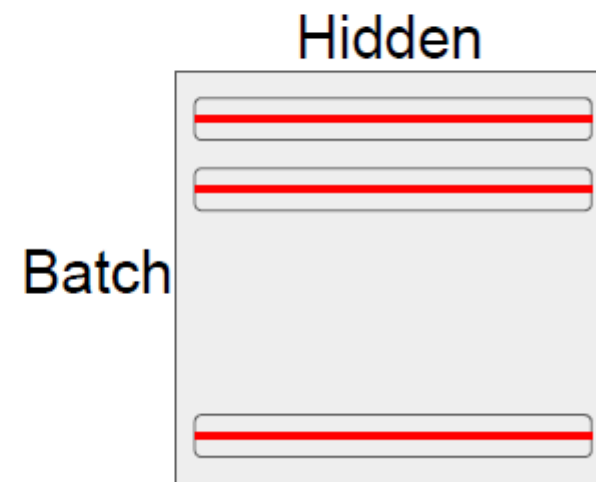
Layer Normalization

Ba, Kiros, and Hinton, «Layer Normalization», arXiv 2016

Batch Normalization



Layer Normalization



Ещё... нормализации

Instance Normalization

Ulyanov et al, Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesis, CVPR 2017

Wu and He, "Group Normalization", arXiv 2018

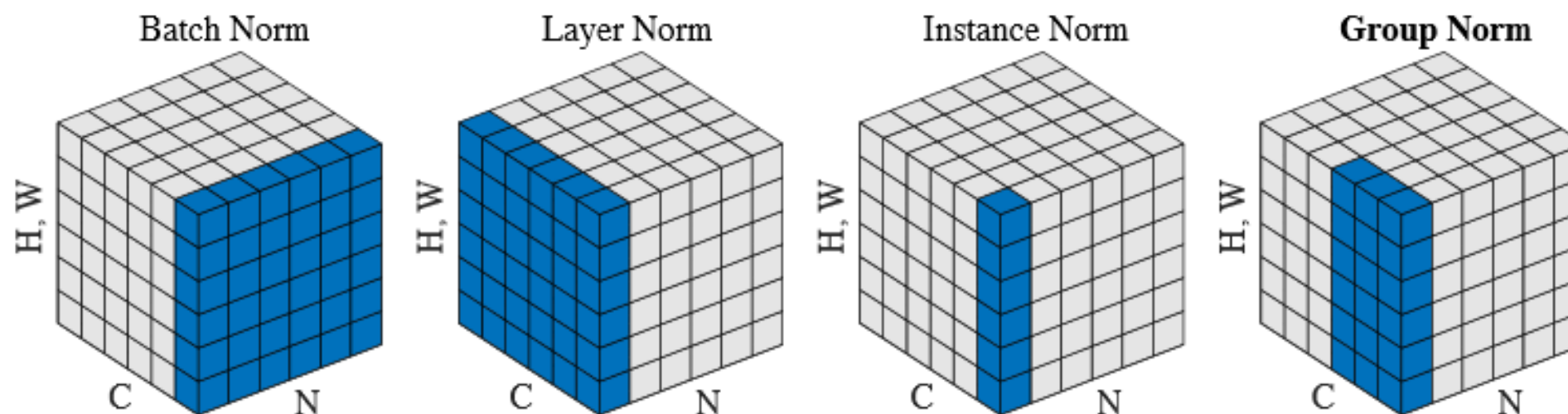
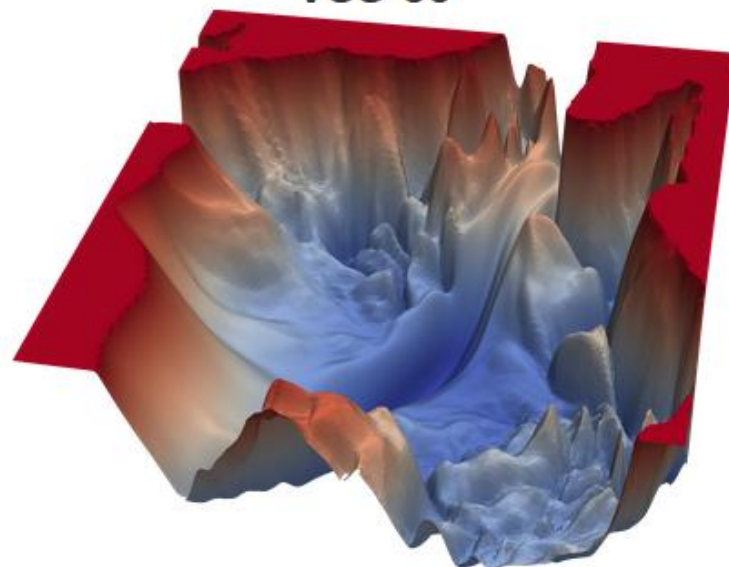


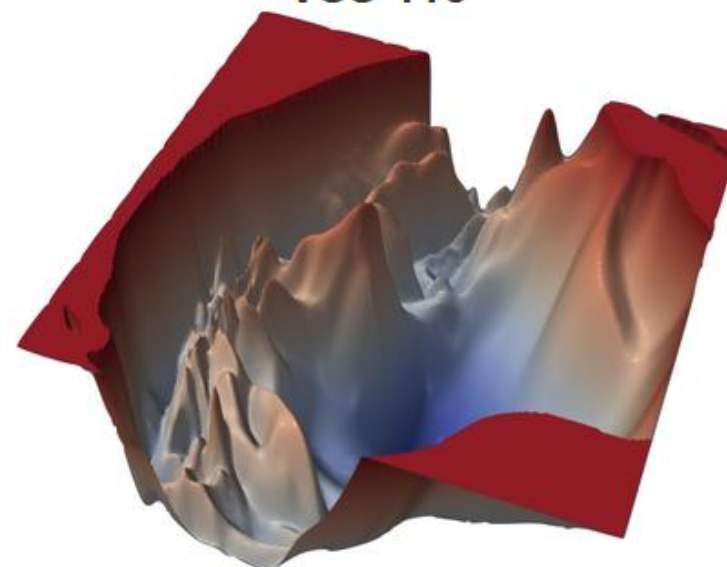
Figure 2. **Normalization methods.** Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

Использование специальных архитектур – **потом подробнее**

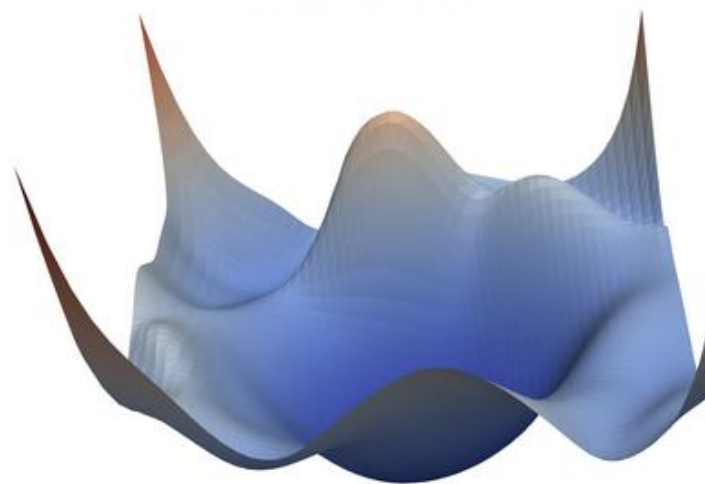
VGG-56



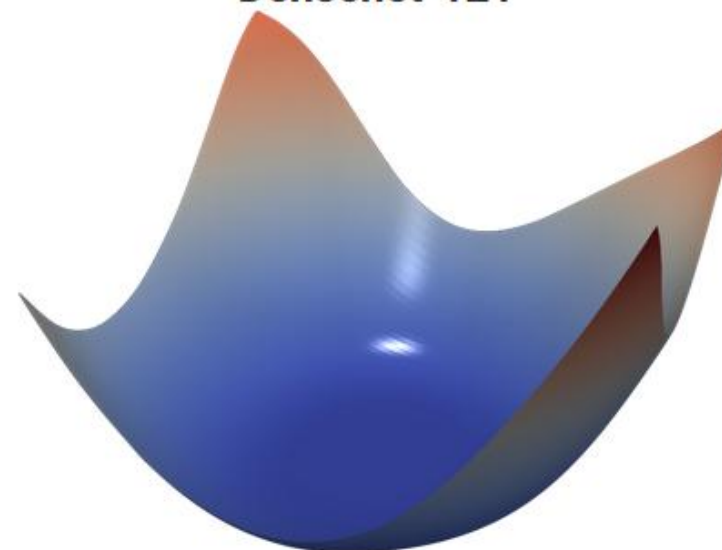
VGG-110



Resnet-56



Densenet-121



<https://www.cs.umd.edu/~tomg/projects/landscapes/>

Расширение обучающего множества (Data Augmentation)

Аугментация – построение дополнительных данных из исходных

Изображения

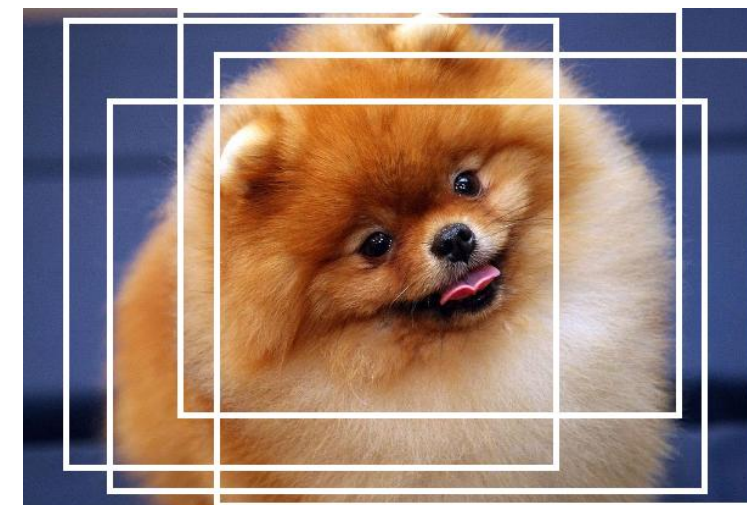
- симметрии (flip)
- вырезки (crop)
- изменение масштаба (rescaling)
- случайные модификации (+шум)
 - повороты (rotation)
 - сдвиги (shift)
- изменение яркости, контраста, палитры
 - эффекты линзы
- перерисовка изображения (ex GAN)

Звук

- +фоновый шум
- тональность

Текст

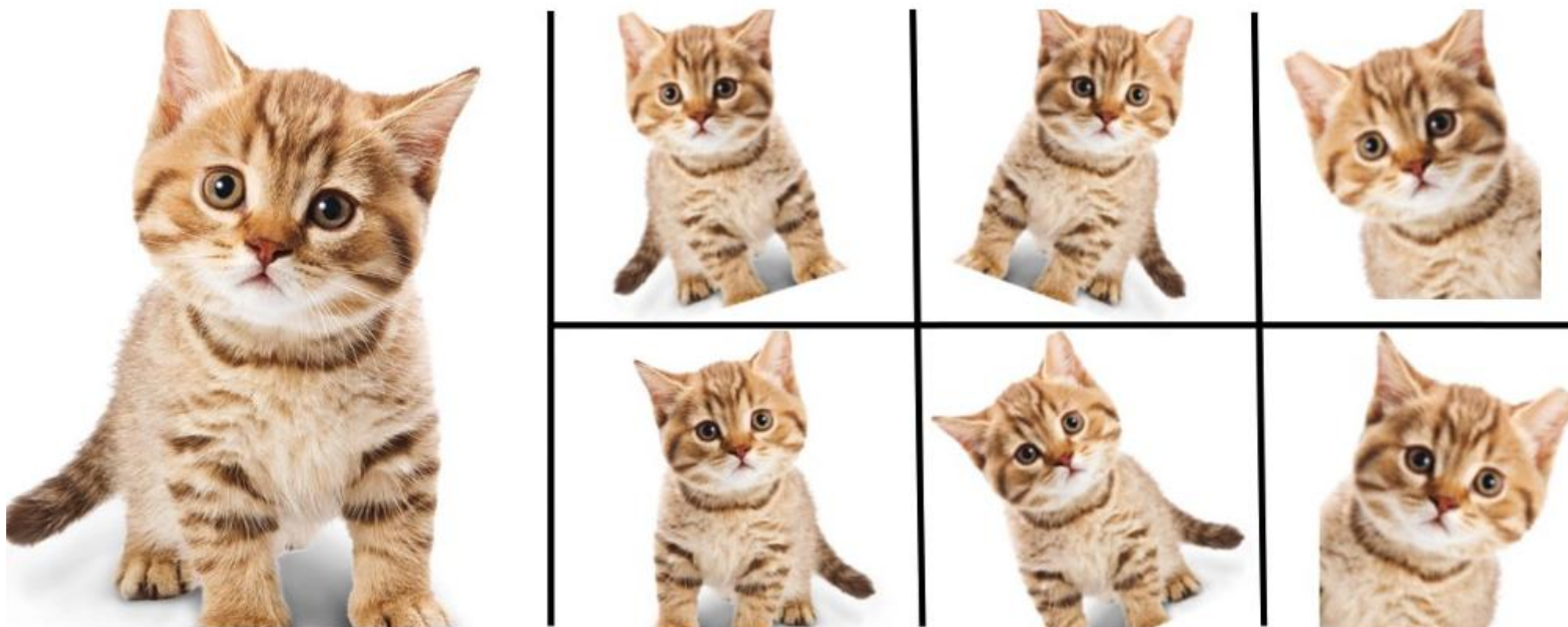
- замена синонимов
- перестановки, удаления слов



Тонкость: преобразования могут переводить объект в другой класс, например повороты «6» и «9».

- смесь с другим текстом
- преобразования (ex: переводчик и обратно)

Расширение обучающего множества (Data Augmentation)



- простая
- агрессивная
- креативная (симуляция, GANы и т.п.)

<https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced>

<https://github.com/aleju/imgaug>

Расширение обучающего множества (Data Augmentation)

```
from keras.preprocessing.image import ImageDataGenerator
# после model.compile(...)

datagen = ImageDataGenerator(
    width_shift_range=0.1, # случайные сдвиги
    height_shift_range=0.1) # случайные сдвиги
datagen.fit(X_train)

# обучение на батчах, которые генерирует datagen.flow()

model.fit_generator(datagen.flow(X_train, Y_train,
                                batch_size=batch_size),
                    samples_per_epoch=X_train.shape[0],
                    nb_epoch=num_epochs,
                    validation_data=(X_val, Y_val),
                    verbose=1)
```

может быть online- и offline- аугментации
м.б. test time- аугментация (TTA)

Аугментация: Mixup

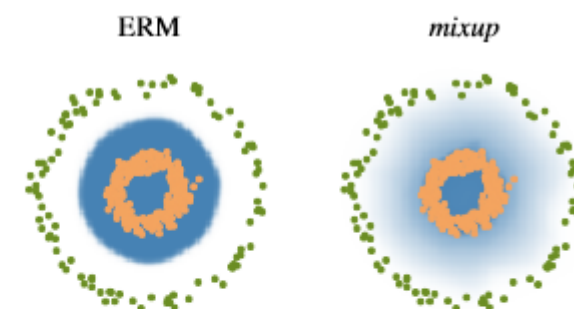
Contribution Motivated by these issues, we introduce a simple and data-agnostic data augmentation routine, termed *mixup* (Section 2). In a nutshell, *mixup* constructs virtual training examples

$$\begin{aligned}\tilde{x} &= \lambda x_i + (1 - \lambda)x_j, & \text{where } x_i, x_j \text{ are raw input vectors} \\ \tilde{y} &= \lambda y_i + (1 - \lambda)y_j, & \text{where } y_i, y_j \text{ are one-hot label encodings}\end{aligned}$$

(x_i, y_i) and (x_j, y_j) are two examples drawn at random from our training data, and $\lambda \in [0, 1]$.

```
# y1, y2 should be one-hot vectors
for (x1, y1), (x2, y2) in zip(loader1, loader2):
    lam = numpy.random.beta(alpha, alpha)
    x = Variable(lam * x1 + (1. - lam) * x2)
    y = Variable(lam * y1 + (1. - lam) * y2)
    optimizer.zero_grad()
    loss(net(x), y).backward()
    optimizer.step()
```

(a) One epoch of *mixup* training in PyTorch.



(b) Effect of *mixup* ($\alpha = 1$) on a toy problem. Green: Class 0. Orange: Class 1. Blue shading indicates $p(y = 1|x)$.

Figure 1: Illustration of *mixup*, which converges to ERM as $\alpha \rightarrow 0$.

<https://github.com/facebookresearch/mixup-cifar10>

<https://arxiv.org/abs/1710.09412>





Аугментация: Mixup

Model	Method	Epochs	Top-1 Error	Top-5 Error
ResNet-50	ERM (Goyal et al., 2017)	90	23.5	-
	<i>mixup</i> $\alpha = 0.2$	90	23.3	6.6
ResNet-101	ERM (Goyal et al., 2017)	90	22.1	-
	<i>mixup</i> $\alpha = 0.2$	90	21.5	5.6
ResNeXt-101 32*4d	ERM (Xie et al., 2016)	100	21.2	-
	ERM	90	21.2	5.6
	<i>mixup</i> $\alpha = 0.4$	90	20.7	5.3
ResNeXt-101 64*4d	ERM (Xie et al., 2016)	100	20.4	5.3
	<i>mixup</i> $\alpha = 0.4$	90	19.8	4.9
ResNet-50	ERM	200	23.6	7.0
	<i>mixup</i> $\alpha = 0.2$	200	22.1	6.1
ResNet-101	ERM	200	22.0	6.1
	<i>mixup</i> $\alpha = 0.2$	200	20.8	5.4
ResNeXt-101 32*4d	ERM	200	21.3	5.9
	<i>mixup</i> $\alpha = 0.4$	200	20.1	5.0

Table 1: Validation errors for ERM and *mixup* on the development set of ImageNet-2012.**SOTA:****CIFAR-10****CIFAR-100****ImageNet-2012**

- learning from corrupt labels
- facing adversarial examples
- generalization on speech / tabular data
- to stabilize the training of GANs

Аугментация: Cutout, CutMix

	ResNet-50	Mixup	Cutout	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	78.4 (+2.1)
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	47.3 (+1.0)
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	76.7 (+1.1)

Sangdoo Yun et al «CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features» // <https://arxiv.org/pdf/1905.04899.pdf>

Аугментация: Cutout, CutMix

Let $x \in \mathbb{R}^{W \times H \times C}$ and y denote a training image and its label, respectively. The goal of CutMix is to generate a new training sample (\tilde{x}, \tilde{y}) by combining two training samples (x_A, y_A) and (x_B, y_B) . The generated training sample (\tilde{x}, \tilde{y}) is used to train the model with its original loss function. We define the combining operation as

$$\begin{aligned}\tilde{x} &= \mathbf{M} \odot x_A + (\mathbf{1} - \mathbf{M}) \odot x_B \\ \tilde{y} &= \lambda y_A + (1 - \lambda) y_B,\end{aligned}\tag{1}$$

where $\mathbf{M} \in \{0, 1\}^{W \times H}$ denotes a binary mask indicating where to drop out and fill in from two images, $\mathbf{1}$ is a binary mask filled with ones, and \odot is element-wise multiplication. Like Mixup [48], the combination ratio λ between two data points is sampled from the beta distribution $\text{Beta}(\alpha, \alpha)$. In our all experiments, we set α to 1, that is λ is sampled from the uniform distribution $(0, 1)$. Note that the major difference is that CutMix replaces an image region with a patch from another training image and generates more locally natural image than Mixup does.

```
lam = np.random.beta(args.beta, args.beta)
rand_index = torch.randperm(input.size()[0]).cuda()
target_a = target
target_b = target[rand_index]
bbx1, bby1, bbx2, bby2 = rand_bbox(input.size(), lam)
input[:, :, bbx1:bbx2, bby1:bby2] = input[rand_index, :, bbx1:bbx2,
bby1:bby2]
# adjust lambda to exactly match pixel ratio
lam = 1 - ((bbx2 - bbx1) * (bby2 - bby1) / (input.size()[-1] *
input.size()[-2]))
output = model(input)
loss = criterion(output, target_a) * lam + criterion(output,
target_b) * (1. - lam)
```

есть несоответствие между кодом и статьёй

Аугментация: GridMask

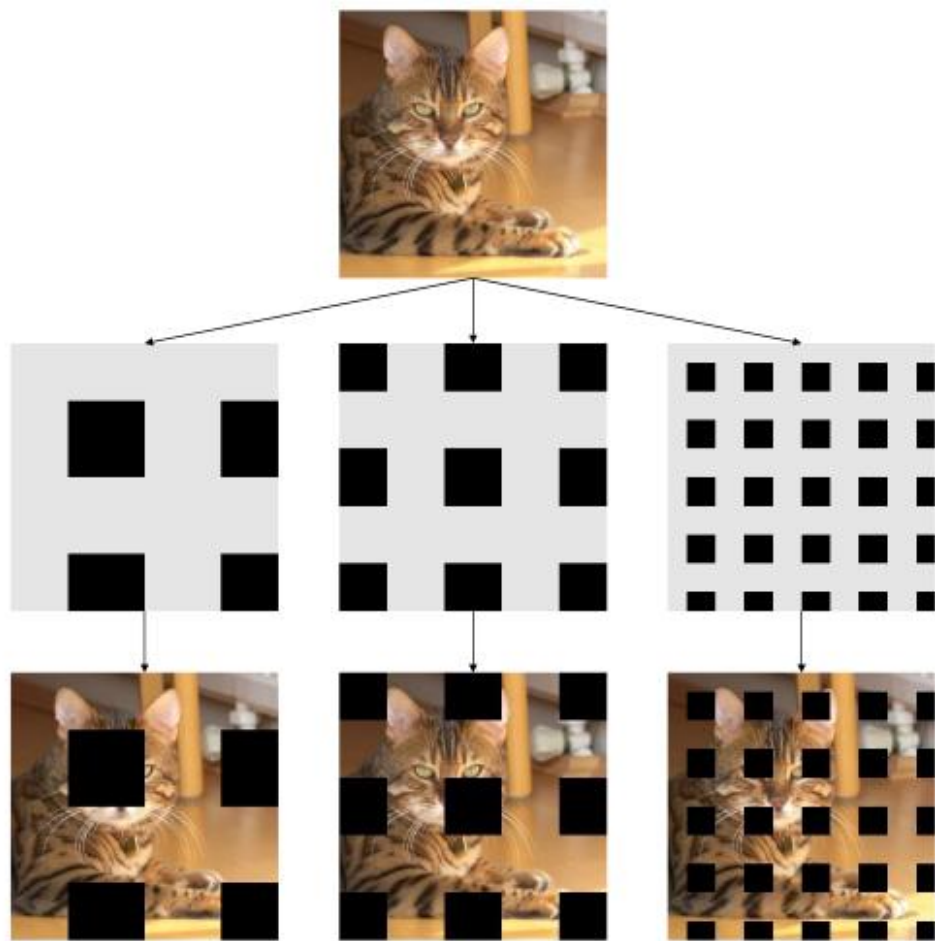


Figure 3. This image shows examples of GridMask. First, we produce a mask according to the given parameters $(r, d, \delta_x, \delta_y)$. Then we multiply it with the input image. The result is shown in the last row. In the mask, gray value is 1, representing the reserved regions; black value is 0, for regions to be deleted.

Model	Accuracy (%)
ResNet18 [10]	95.28
+ Randomerasing [29]	95.32
+ Cutout [4]	96.25
+ HaS [17]	96.10
+ AutoAugment [3]	96.07
+ GridMask (Ours)	96.54
+ AutoAugment & Cutout [3]	96.51
+ AutoAugment & GridMask (Ours)	96.64
WideResNet-28-10 [24]	96.13
+ Radnomerasing [29]*	96.92
+ Cutout [4]	97.04
+ HaS [17]	96.94
+ AutoAugment [3]	97.01
+ GridMask (Ours)	97.24
+ AutoAugment & Cutout [3]	97.39
+ AutoAugment & GridMask (Ours)	97.48
ShakeShake-26-32 [5]	96.42
+ Randomerasing [29]	96.46
+ Cutout [4]	96.96
+ Has [17]	96.89
+ Autoaugment [3]	96.96
+ GridMask (Ours)	97.20
+ AutoAugment & Cutout [3]	97.36
+ AutoAugment & GridMask (Ours)	97.42

Table 3. Results on CIFAR10 are summarized in this table. We achieve the best accuracy on different models. * means results reported in the original paper.

Р Chen «GridMask Data Augmentation» <https://arxiv.org/pdf/2001.04086.pdf>

Ансамбль нейросетей

- несколько независимых моделей (как всегда +2%)
- усреднение ~ одной НС на разных эпохах обучения
(аналог усреднения Поляка)

Loshchilov and Hutter, “SGDR: Stochastic gradient descent with restarts”, arXiv 2016

Huang et al, “Snapshot ensembles: train 1, get M for free”, ICLR 2017

```
from keras.layers import merge # for merging predictions in an ensemble
# ...
ens_models = 3 # we will train three separate models on the data
# ...
inp_norm = BatchNormalization(axis=1)(inp) # Apply BN to the input (N.B. need to rename here)

outs = [] # the list of ensemble outputs
for i in range(ens_models):
    # conv_1 = Convolution2D(...)(inp_norm)
    # ...
    outs.append(Dense(num_classes, init='glorot_uniform', W_regularizer=l2(12_lambda),
activation='softmax')(drop)) # Output softmax layer

out = merge(outs, mode='ave') # average the predictions to obtain the final output
```

Диагностика проблем с НС

1. Численно проверить градиенты (с помощью конечных разностей)

$$\frac{\partial f(x)}{\partial x} \approx \frac{f(x + \varepsilon) - f(x - \varepsilon)}{2\varepsilon}$$

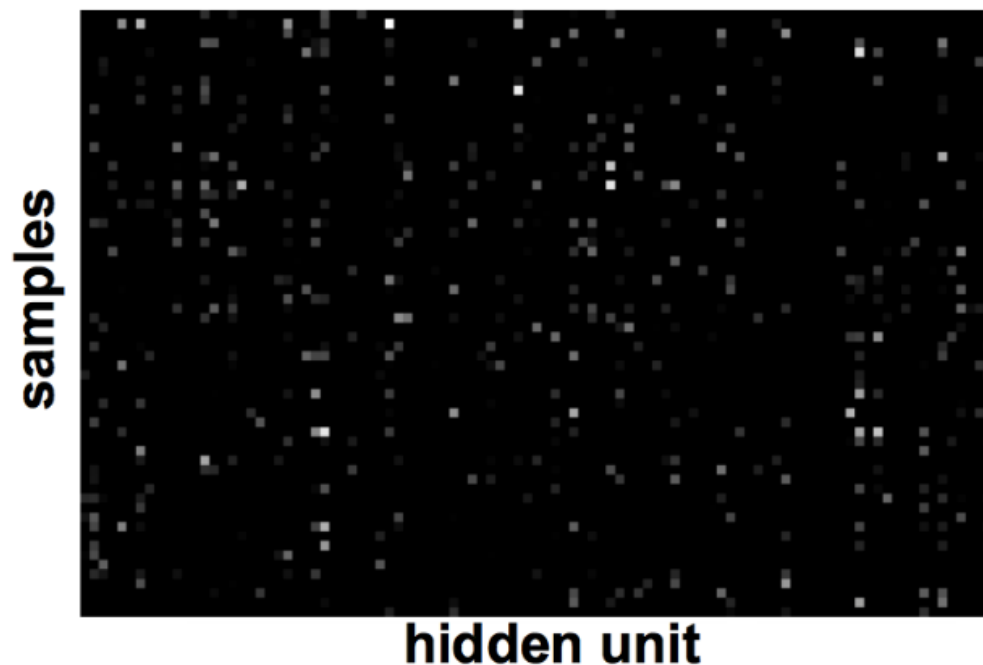
проверка реализации обратного и прямого распространения

Диагностика проблем с НС

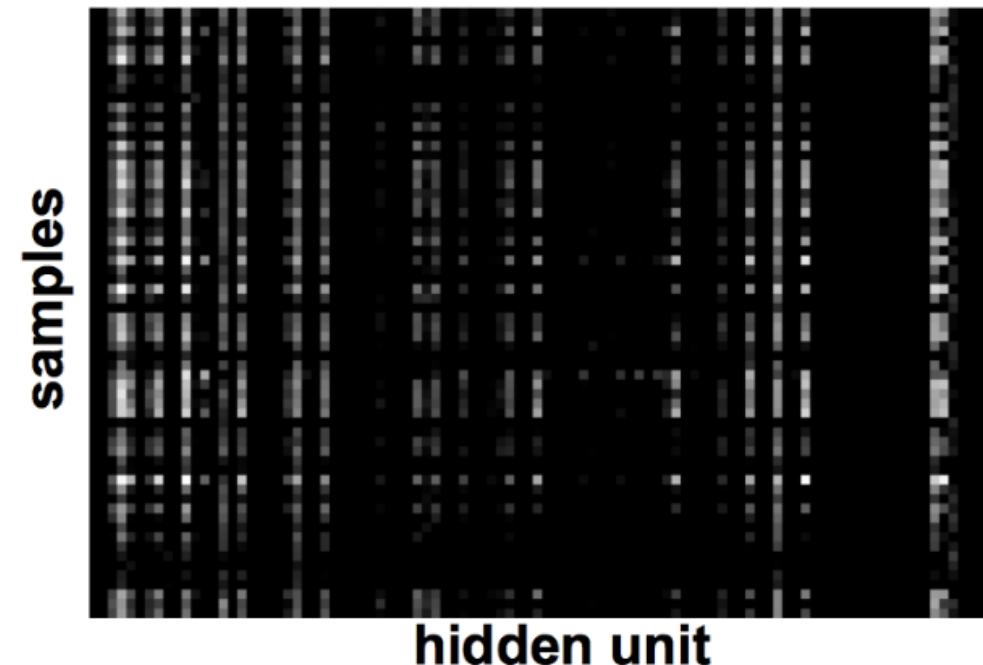
2. Визуализация

Признаки (в общем смысле) должны быть некоррелированными и с большой дисперсией

Хорошо:

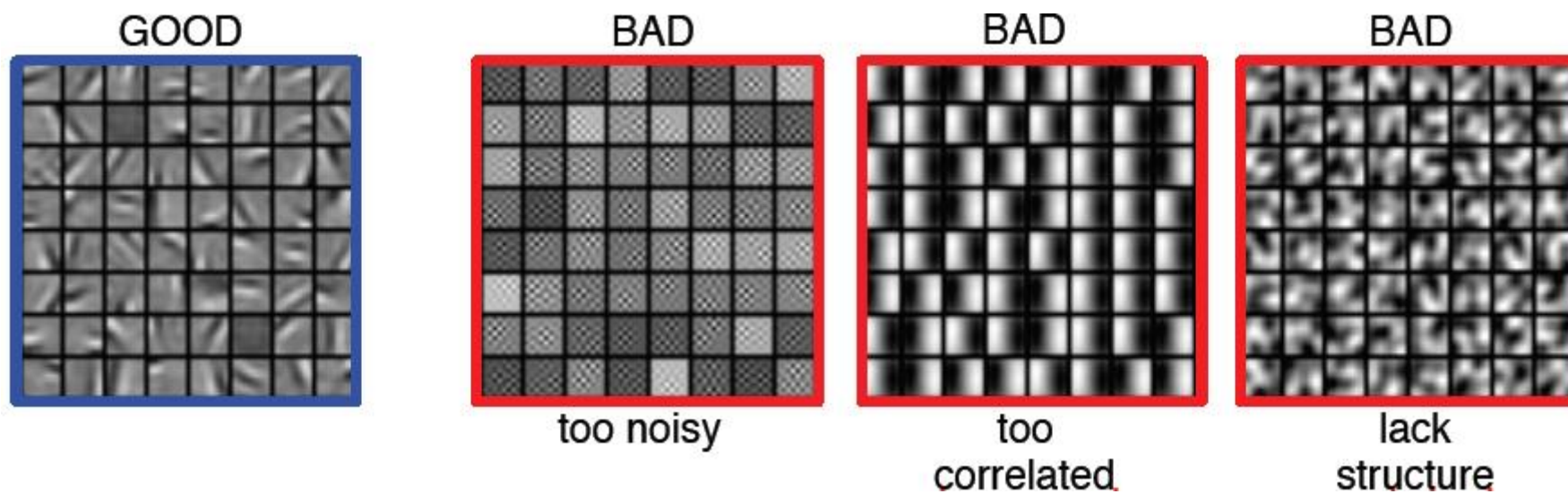


Плохо:



Marc'Aurelio Ranzato, CVPR 2014

Хорошие фильтры имеют структуру и некоррелированные



**3. Убедиться, что сеть работает на небольшом куске данных
(~ 100 – 500 объектов)**

**4. Насыщены ли нейроны ещё до обучения?
Нормировка!**

**5. Как ведёт себя ошибка обучения
Настроить темп обучения!**

6. Насколько меняются веса за итерацию (~ 0.1%)

Недообучение

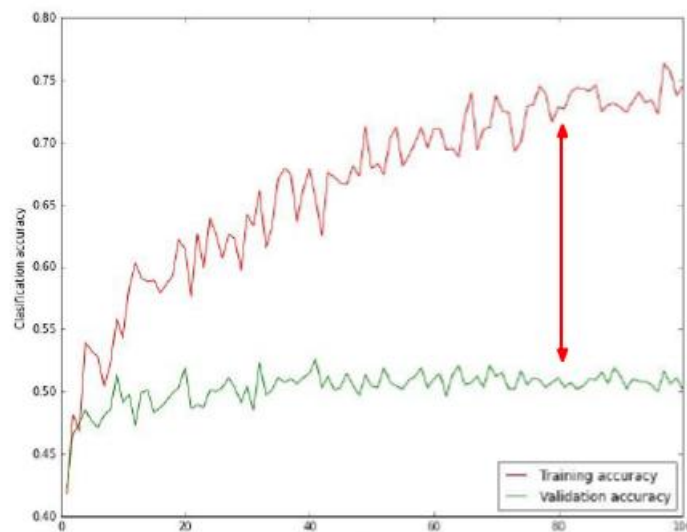
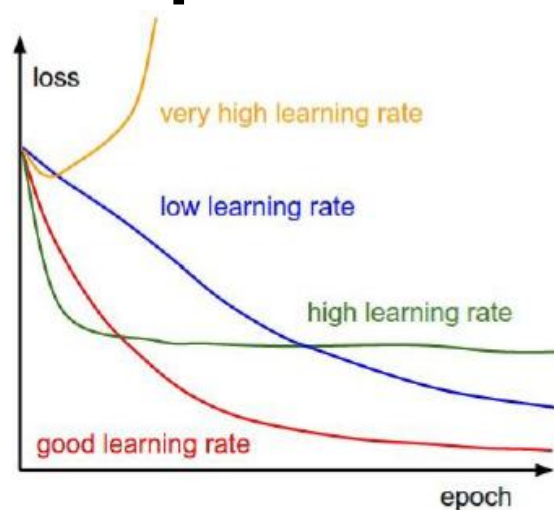
Переобучение

**Другие методы оптимизации
GPU**

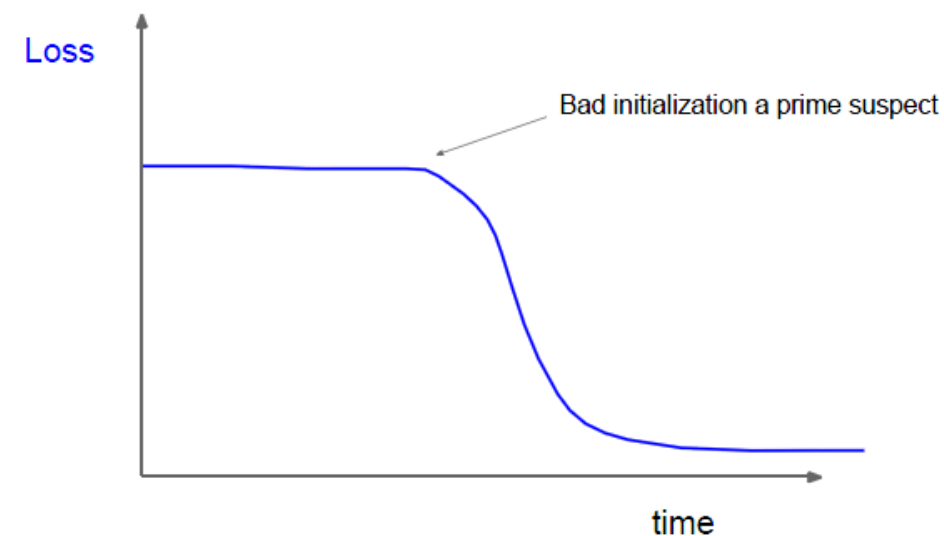
**Регуляризация
Обучение без учителя
(более сложная задача \Rightarrow меньше
переобучения)
Dropout**

Кривые ошибок

Настройка темпа



Плохая инициализация



**Большой зазор \Rightarrow переобучение \Rightarrow
усилить регуляризацию**

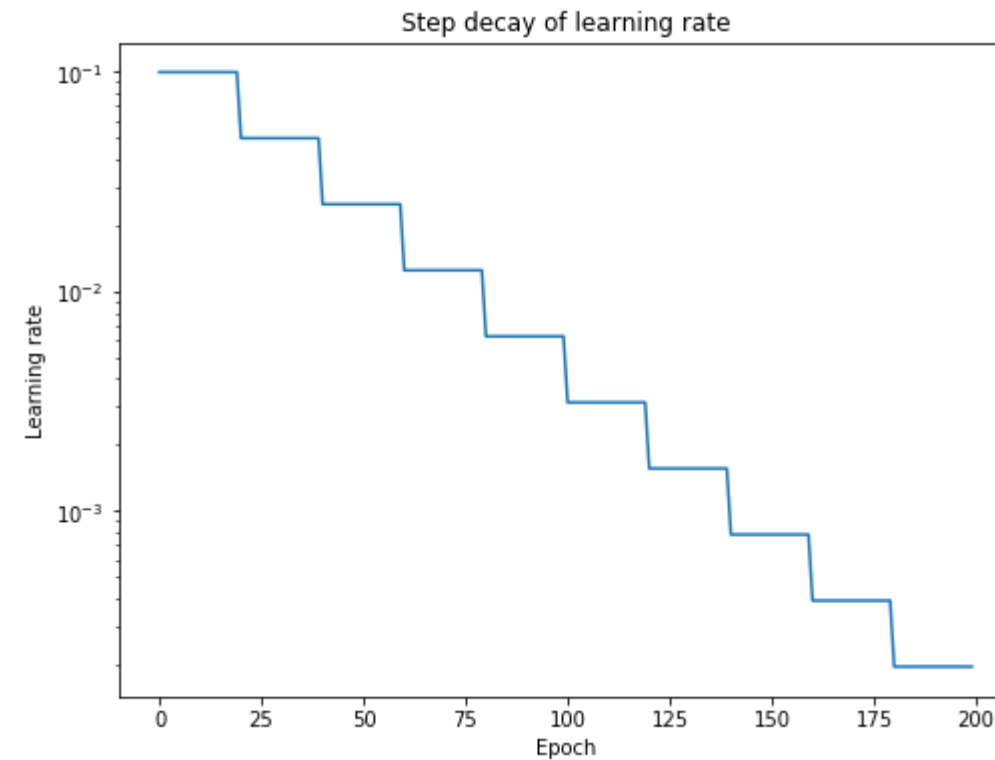
Маленький \Rightarrow усложнить модель (?)

Настройка темпа обучения

Всегда смотрите на кривые ошибок во время обучения!

Learning Rate Annealing

уменьшать через каждые N эпох



<https://www.jeremyjordan.me/nn-learning-rate/>

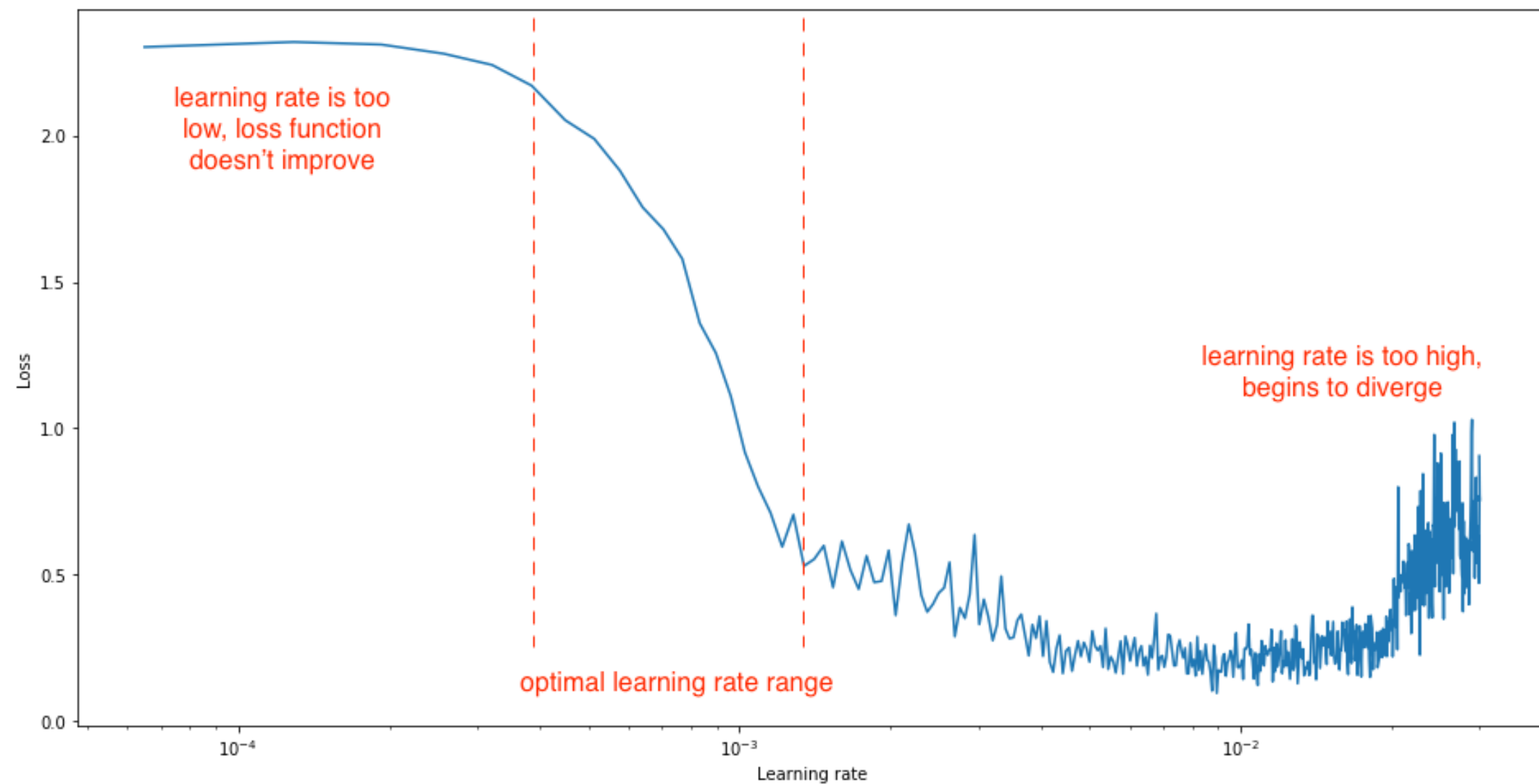
Настройка темпа обучения

Learning Rate Schedule

уменьшать после выхода на плато
(качество не улучшается в течение N эпох)
или какого-то другого специфического условия

```
torch.optim.lr_scheduler
```

Настройка темпа обучения



<https://www.jeremyjordan.me/nn-learning-rate/>

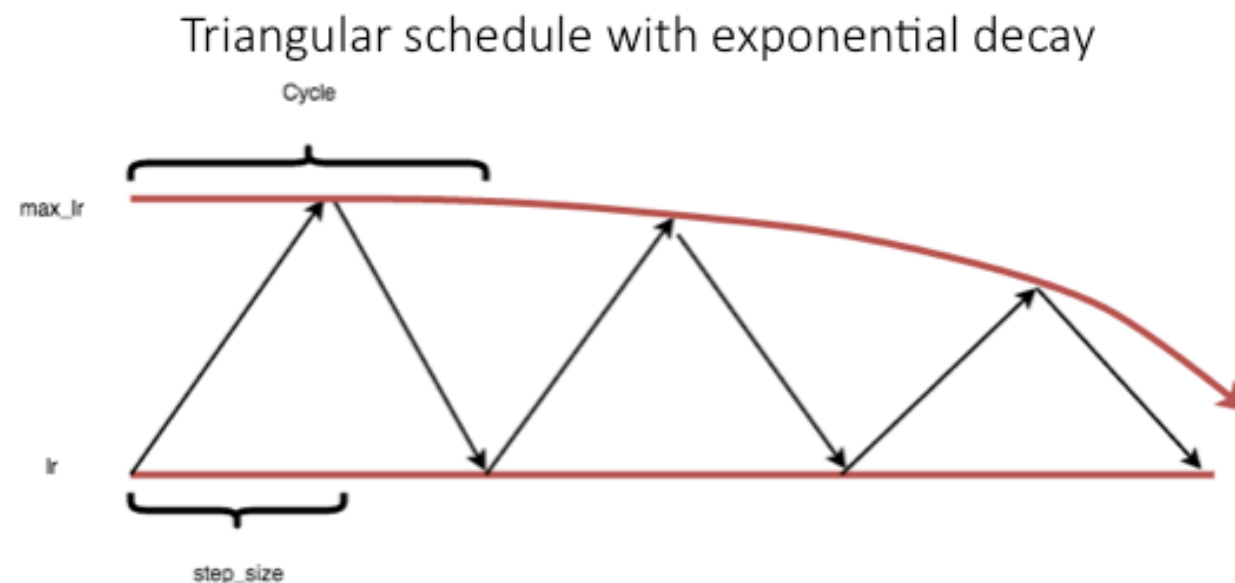
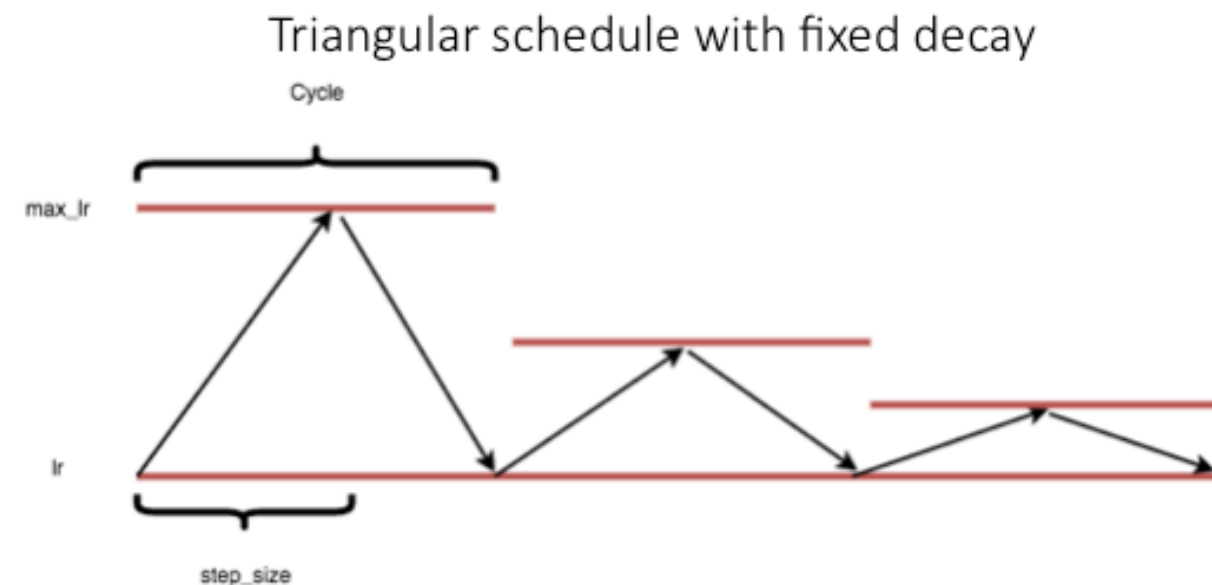
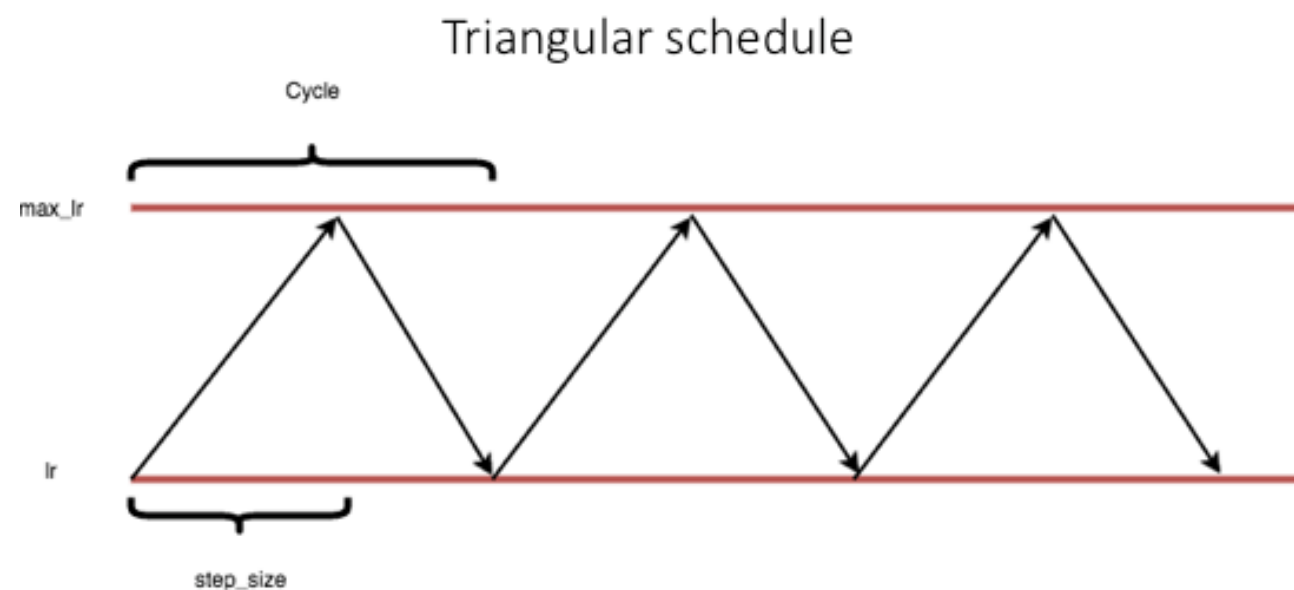
Настройка темпа обучения

«Cyclical Learning Rates for Training Neural Networks»

<https://arxiv.org/abs/1506.01186>

– как на предыдущем рисунке – увеличивать темп после каждого батча, по графику ошибки найти оптимальный темп

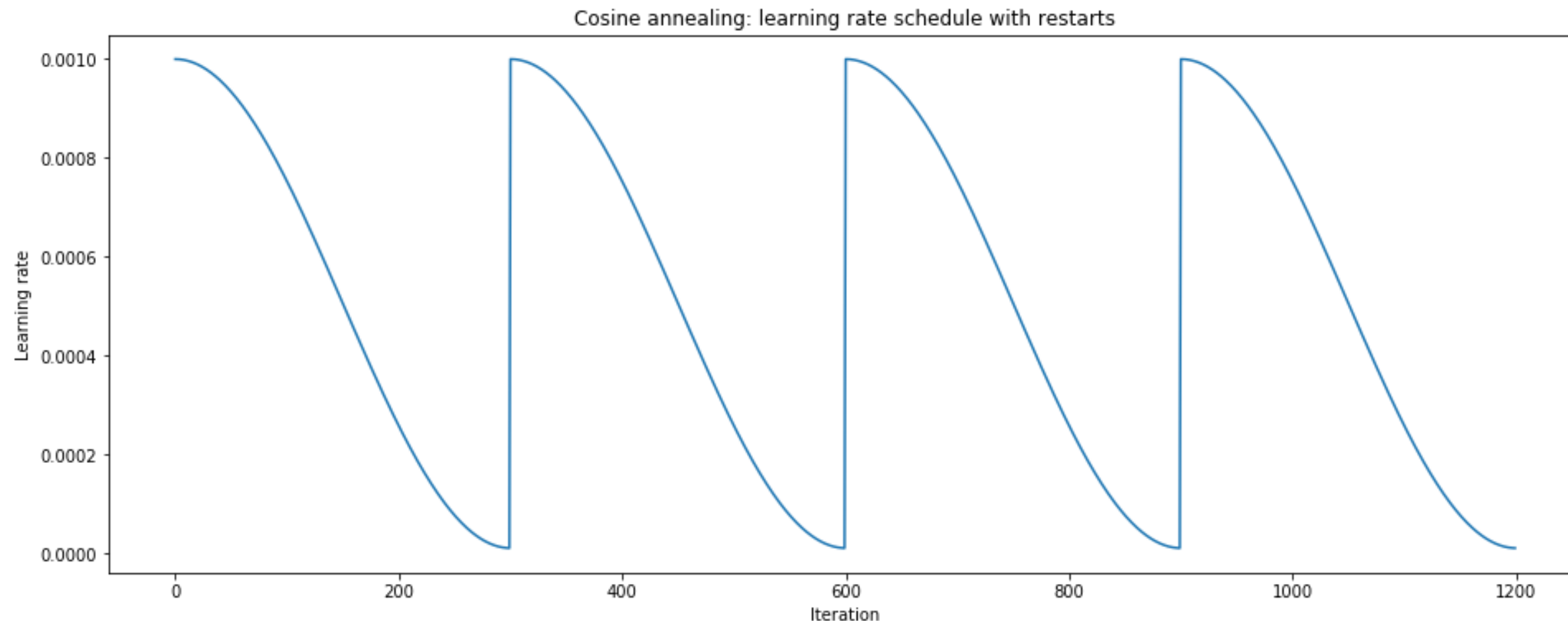
Настройка темпа обучения



большой темп тоже нужен для быстрого покидания седловых точек

Настройка темпа обучения

Stochastic Gradient Descent with Warm Restarts (SGDR)



и можно строить ансамбль моделей в точках затухания!

<https://www.jeremyjordan.me/nn-learning-rate/>

Настройка темпа обучения

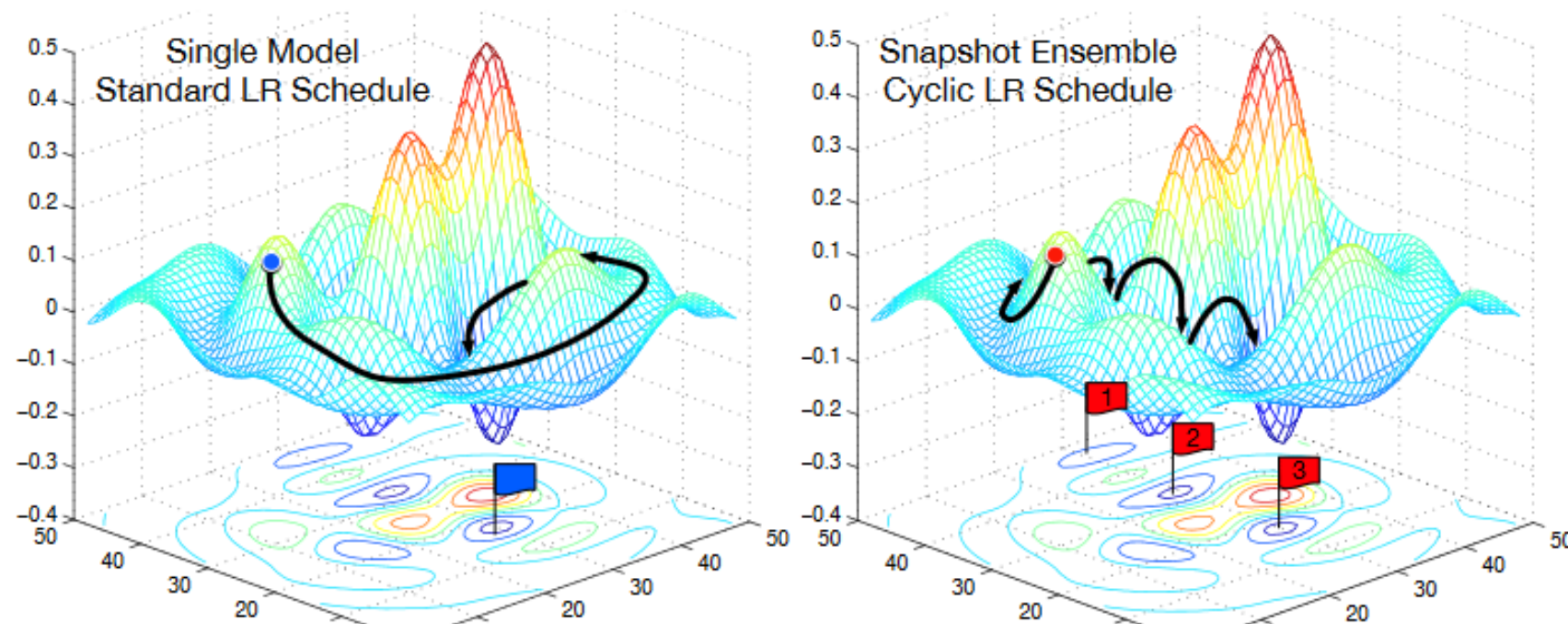


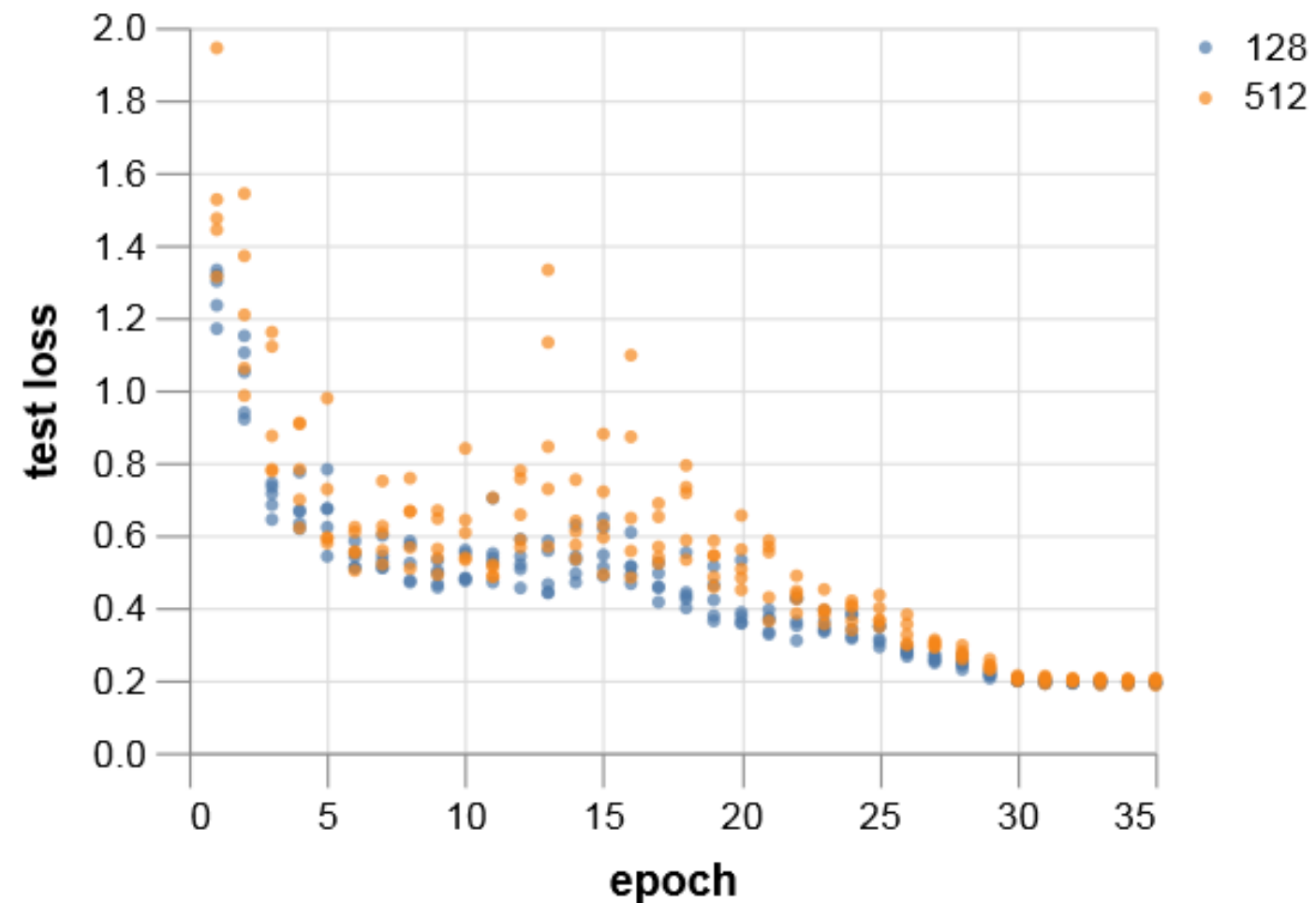
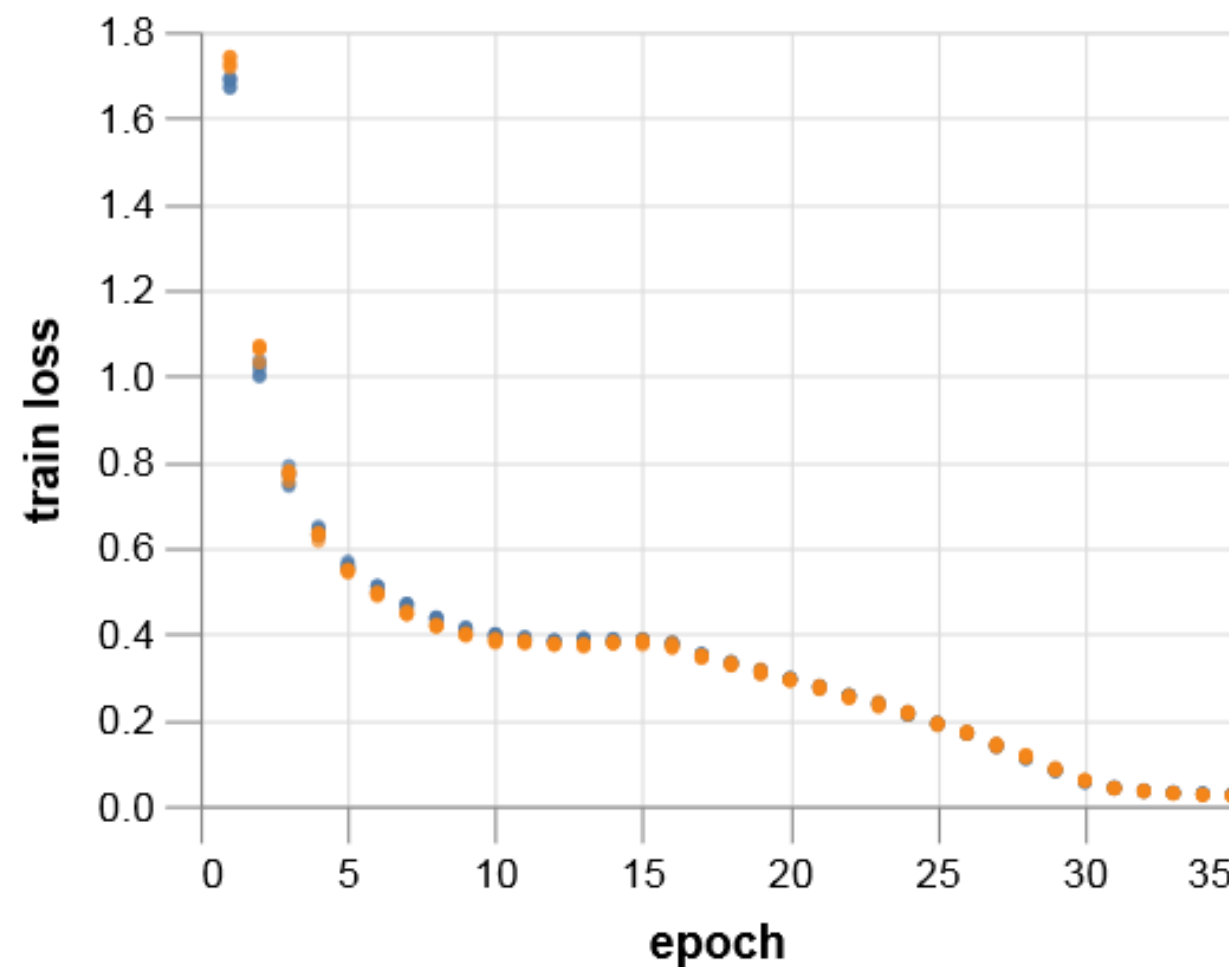
Figure 1: **Left:** Illustration of SGD optimization with a typical learning rate schedule. The model converges to a minimum at the end of training. **Right:** Illustration of Snapshot Ensembling. The model undergoes several learning rate annealing cycles, converging to and escaping from multiple local minima. We take a snapshot at each minimum for test-time ensembling.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, Kilian Q. Weinberger

«Snapshot Ensembles: Train 1, get M for free»

<https://arxiv.org/pdf/1704.00109.pdf>

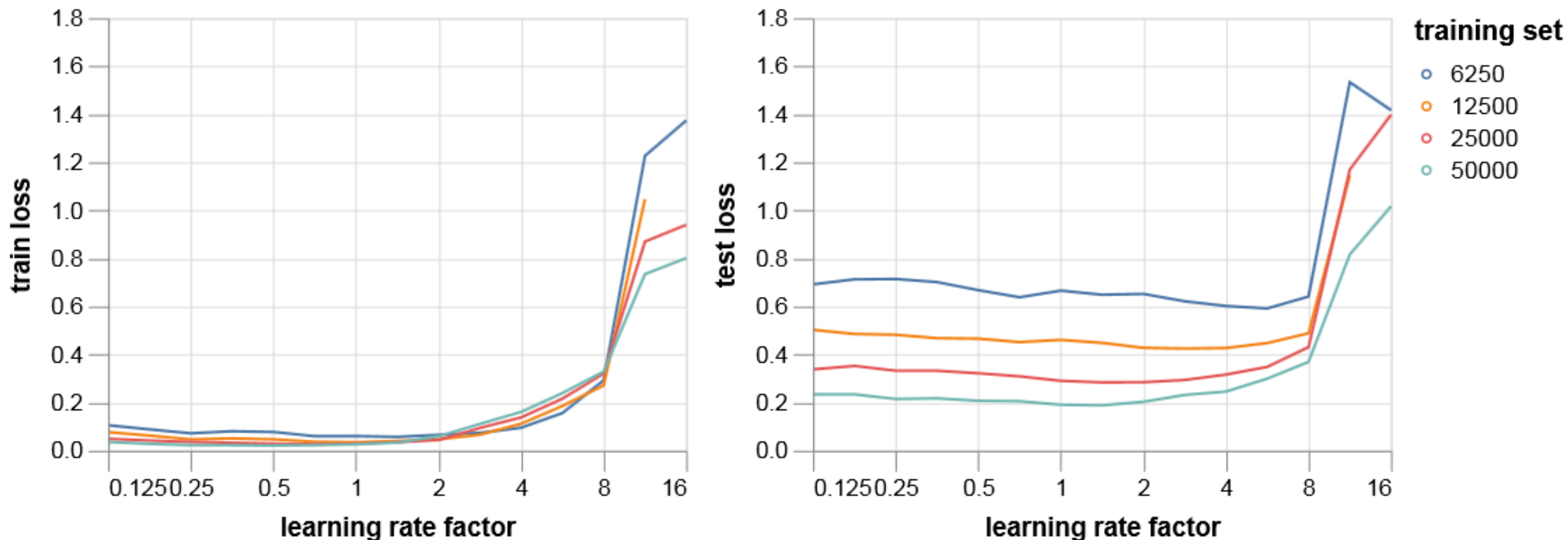
Обучение нейронных сетей



зависимость от размера батча

<https://myrtle.ai/learn/how-to-train-your-resnet-2-mini-batches/>

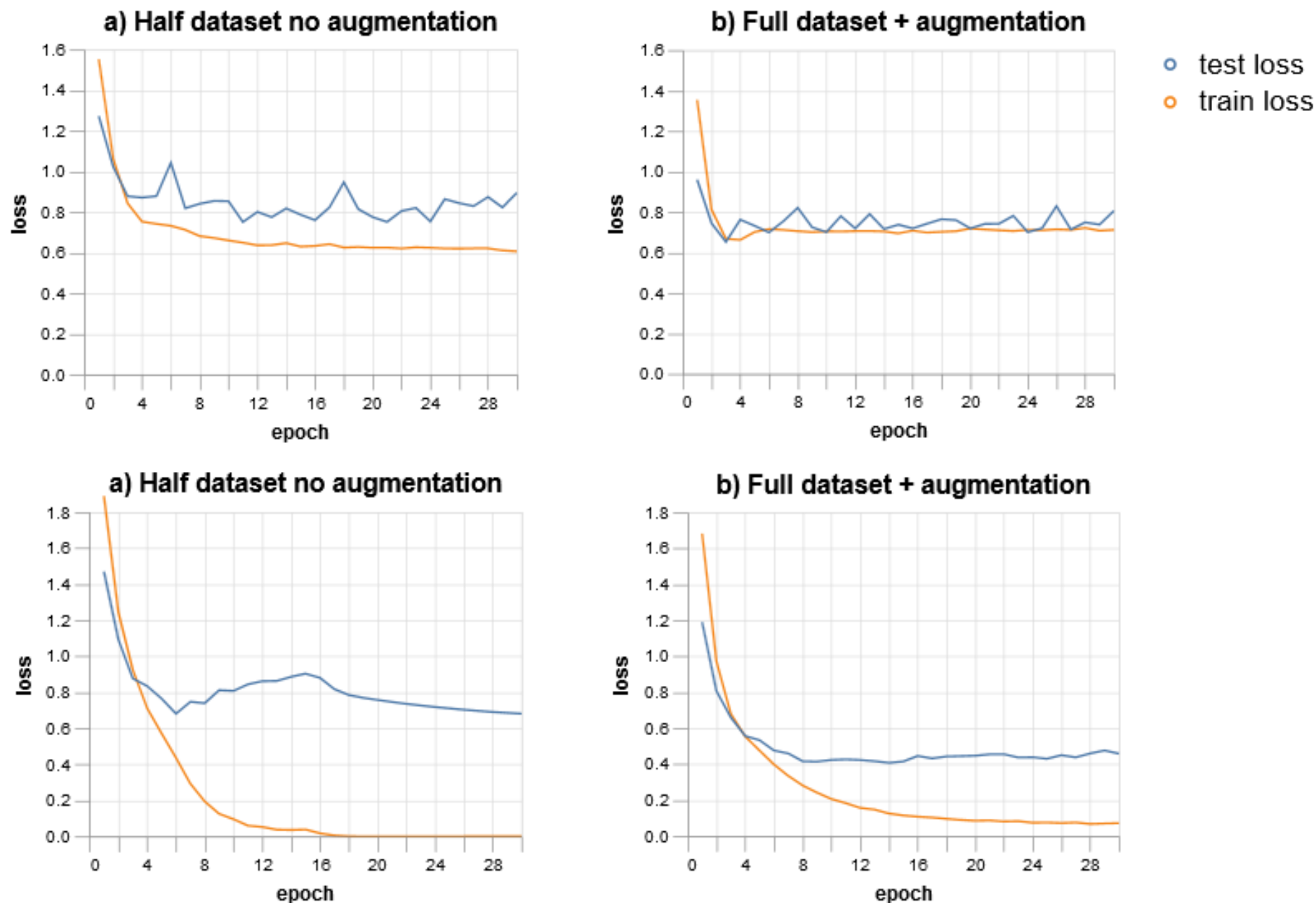
Обучение нейронных сетей



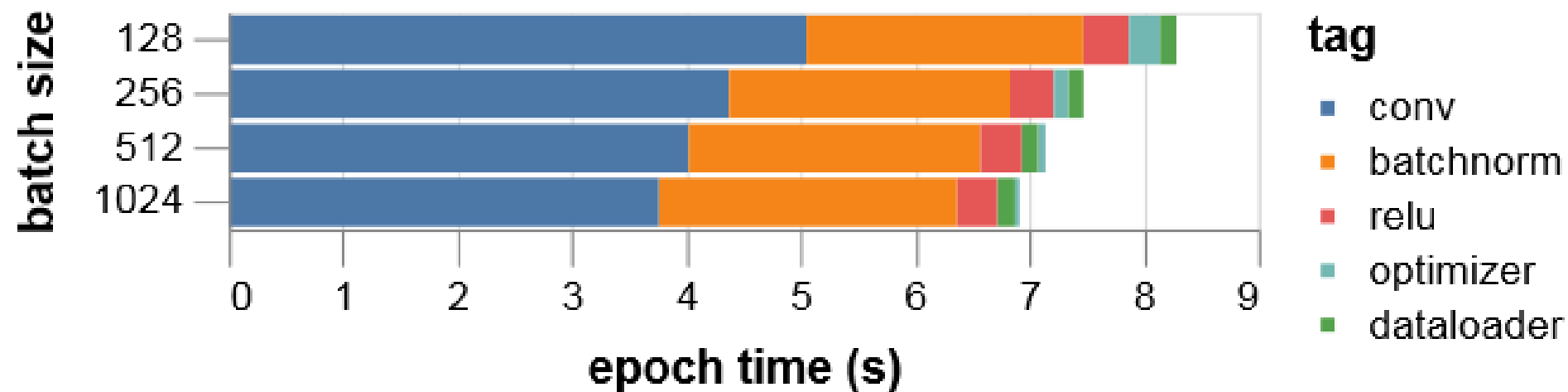
зависимость от темпа обучения при разном размере обучения

Обучение нейронных сетей

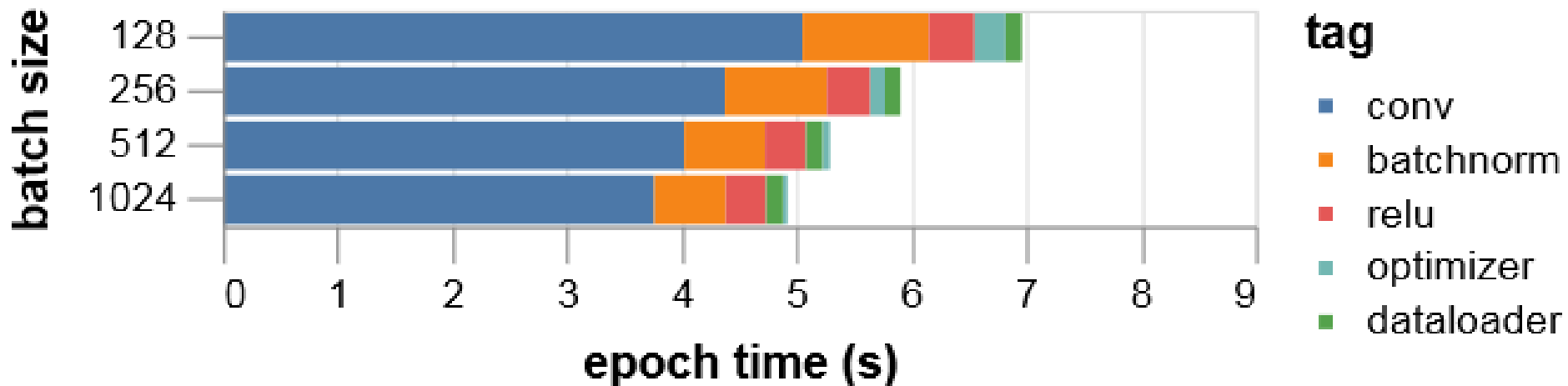
maximum learning rate 4× higher than the original training setup:



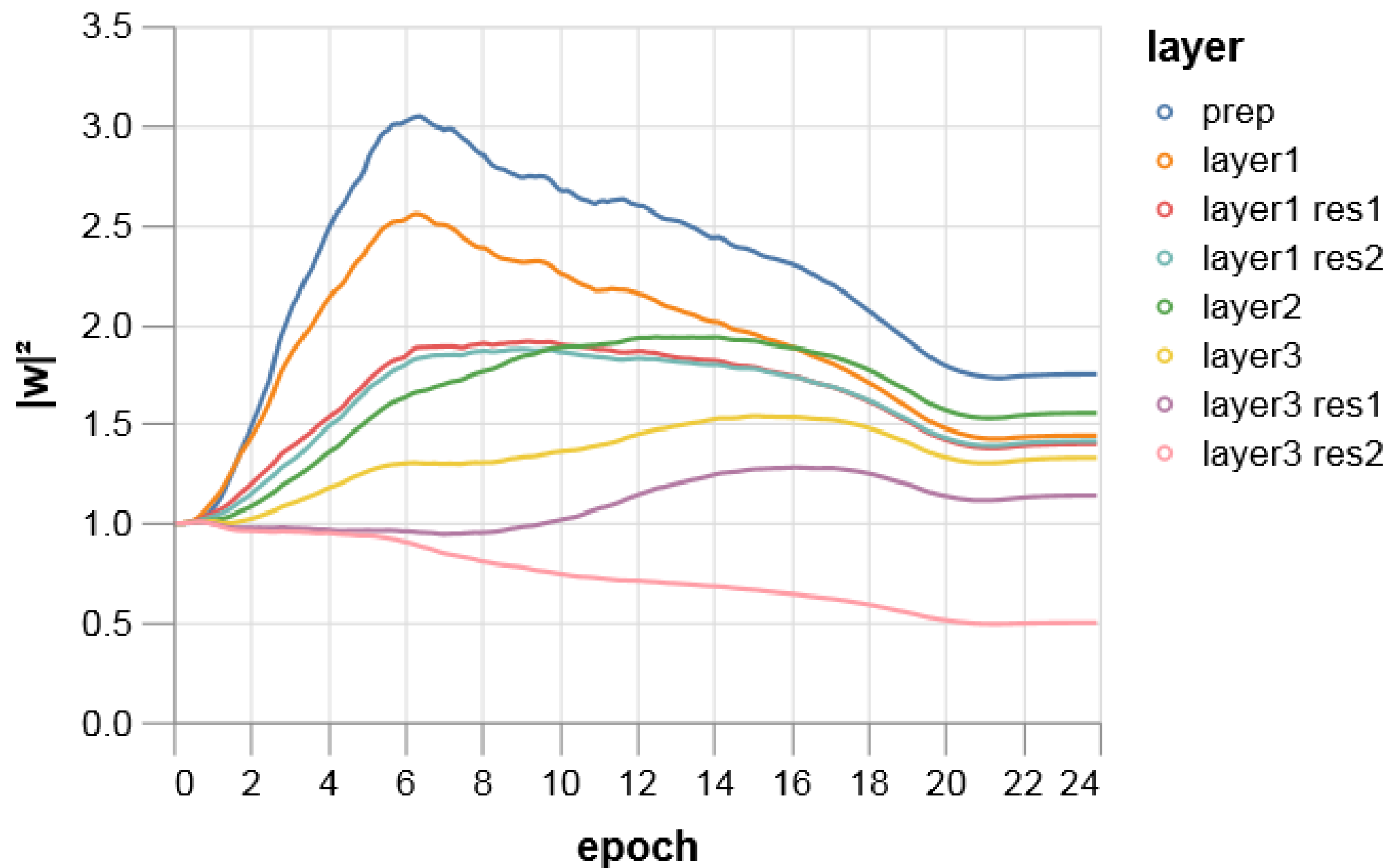
Обучение нейронных сетей



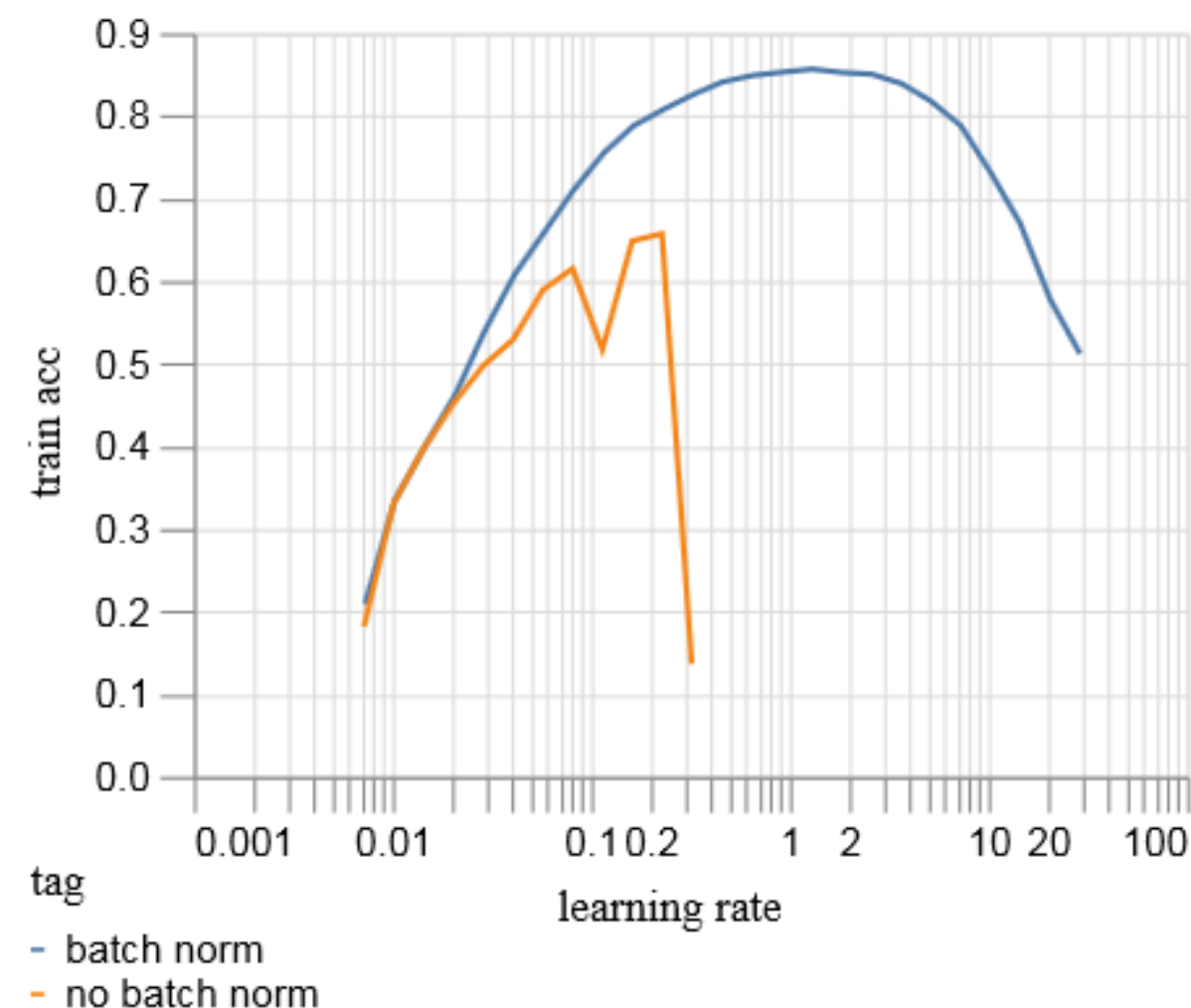
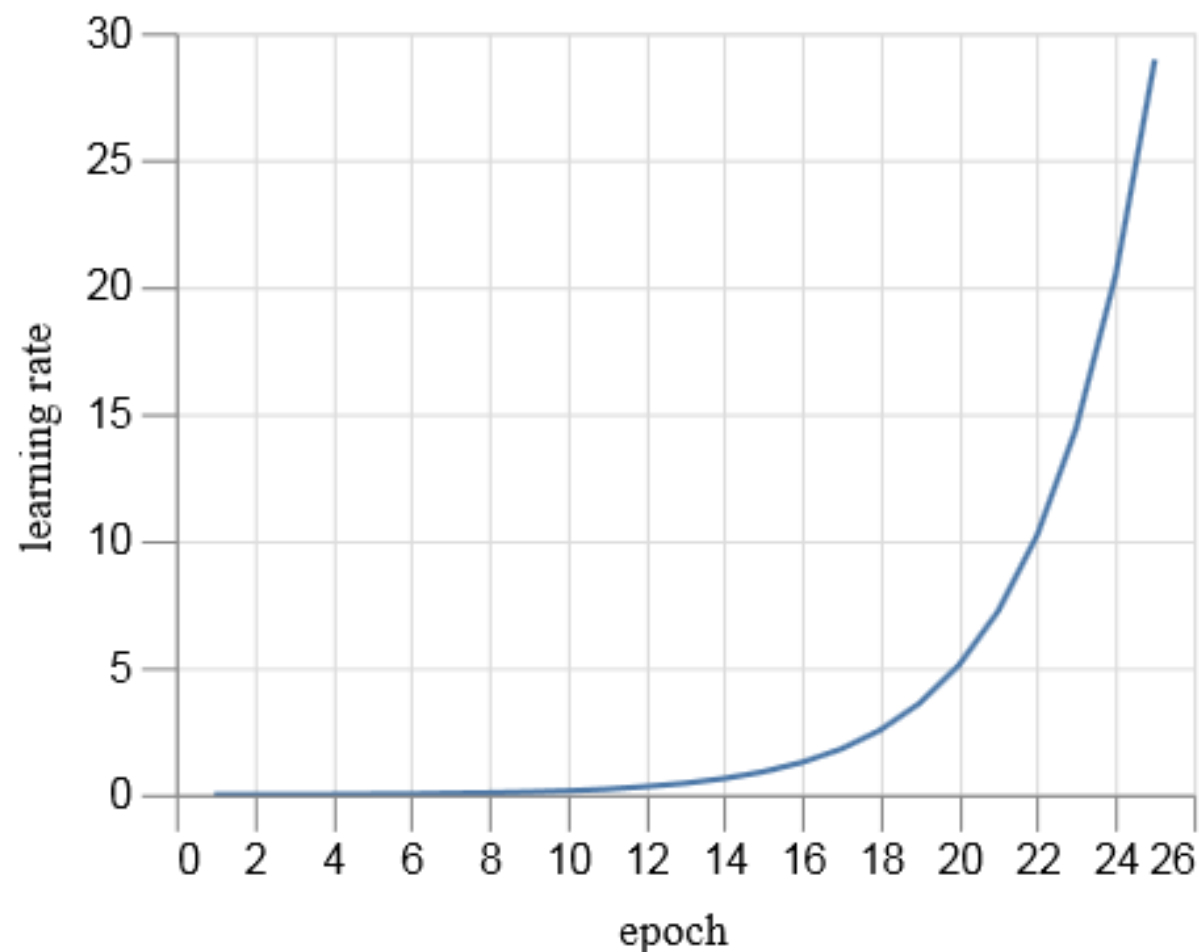
convert batch norm weights back to single precision:
(небольшой хак с реализацией)



Обучение нейронных сетей



Обучение нейронных сетей



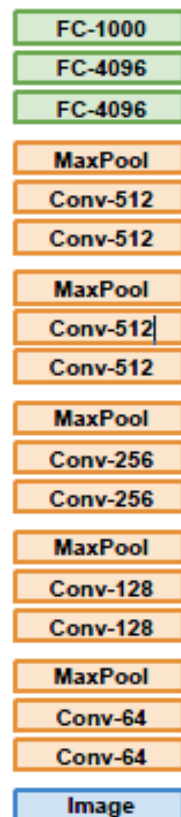
Эксперимент, в котором увеличивали темп обучения (слева – как)

<https://myrtle.ai/learn/how-to-train-your-resnet-7-batch-norm/>

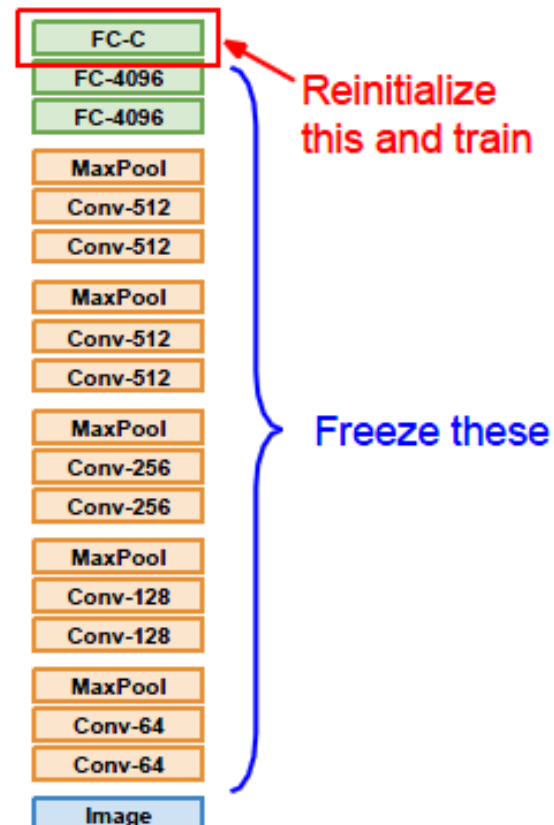
Transfer Learning

**Чтобы решать задачи нужны данные...
если данных мало, берём предобученную НС**

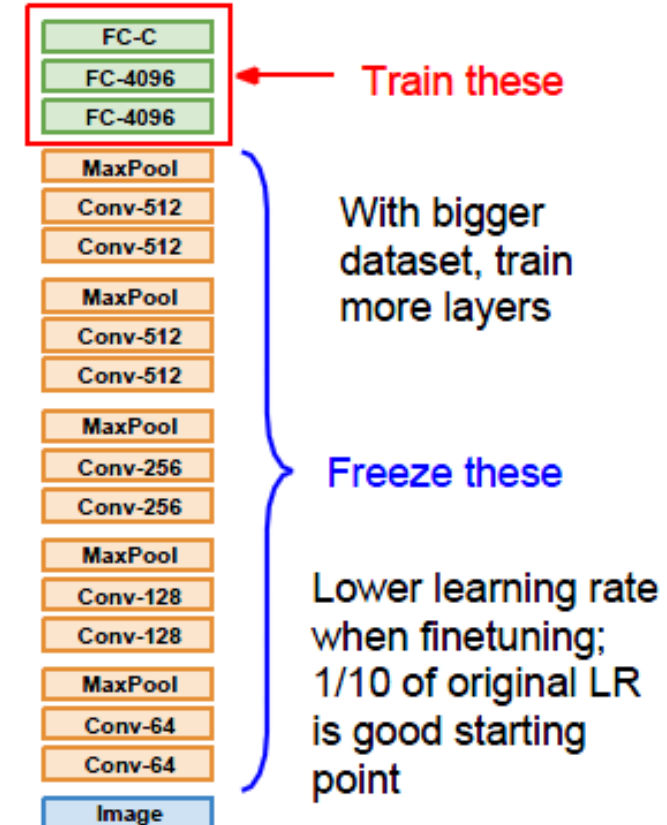
1. Train on Imagenet



2. Small Dataset (C classes)



3. Bigger dataset



Можно не морозить слои, а обучать с меньшим темпом [cs231]

Transfer Learning

Можно решать и другие задачи – для этого перестраивается голова сети
(как бы получаем признаки обученной сетью)

Всю обученную сеть можно:

- **полностью переучивать (можно с маленькими темпами на первых слоях)**
- **переучивать с какого-то слоя**
- **оставить как есть**

Узкие глубокие сети учат с помощью уже обученных неглубоких широких

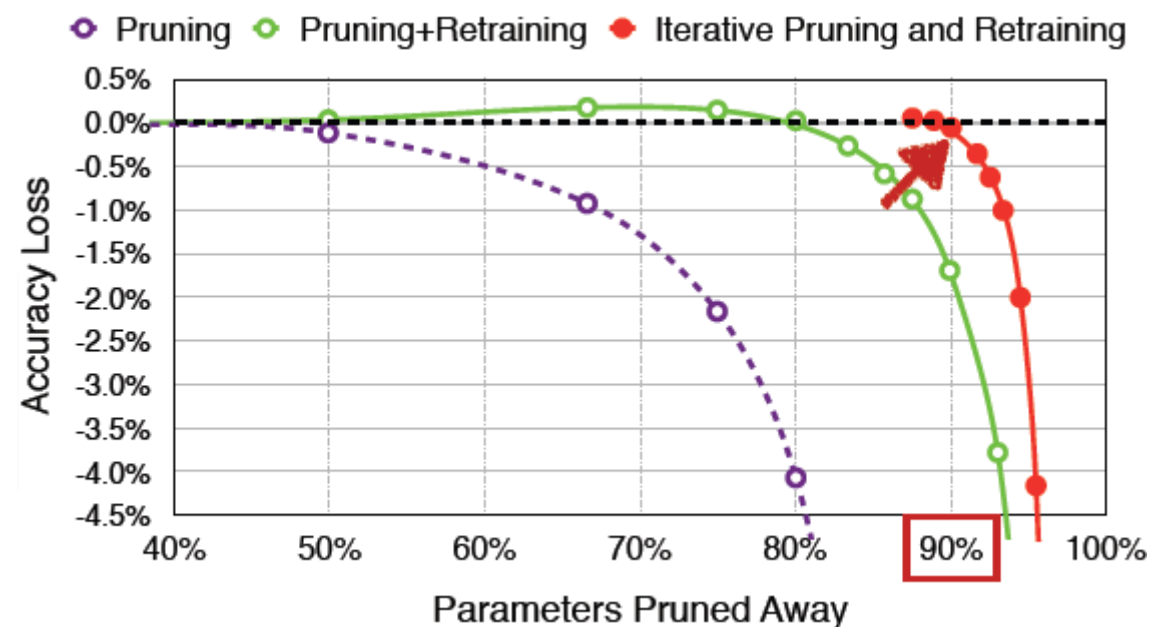
Есть предтренированные НС:

Caffe: <https://github.com/BVLC/caffe/wiki/Model-Zoo>

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

PyTorch: <https://github.com/pytorch/vision>

Упрощение НС (Pruning)



[Han et al. NIPS'15]



- **Original** : a man is riding a surfboard on a wave
- **Pruned 90%**: a man in a wetsuit is riding a wave on a beach



- **Original** : a soccer player in red is running in the field
- **Pruned 95%**: a man in a red shirt and black and white black shirt is running through a field

Оптимизация гиперпараметров

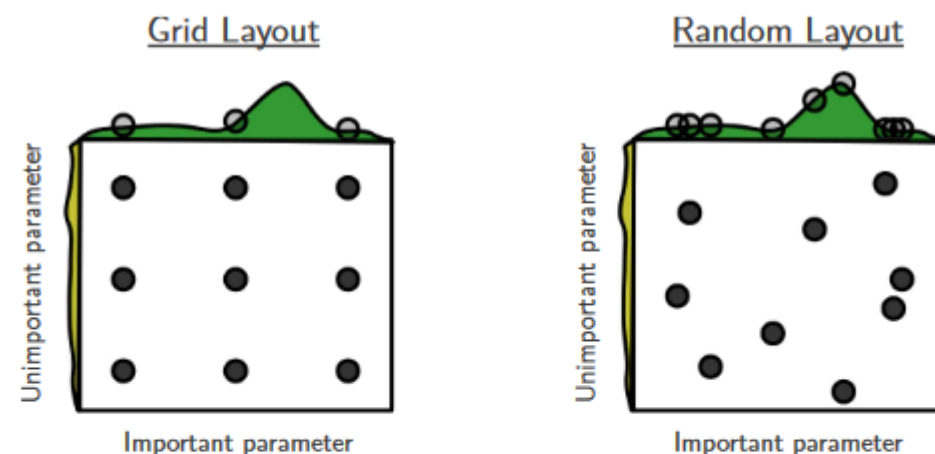
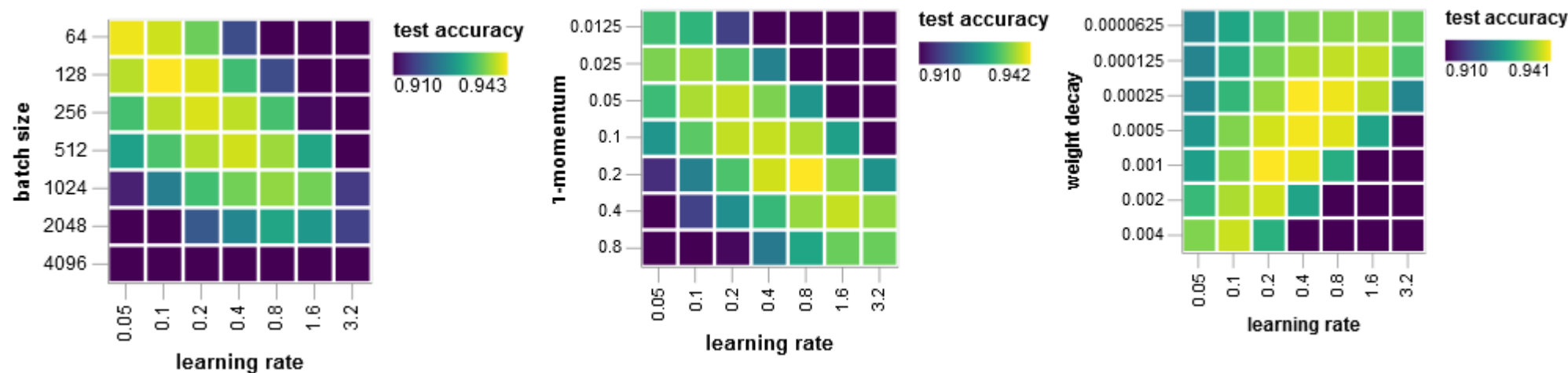


Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g . This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

можно вести более интенсивный поиск в окрестности уже найденного решения

есть другие методы оптимизации!

Оптимизация параметров



можно подойти к поиску оптимальных гиперпараметров как к отдельной задаче оптимизации

покоординатный спуск в пространстве $(\frac{\lambda\alpha}{1-\rho}, \rho, \alpha)$

λ - the maximal learning rate N - batch size ρ - momentum

<https://myrtle.ai/learn/how-to-train-your-resnet-5-hyperparameters/>

Практические советы

- начинайте с простых архитектур и методов (оптимизации)
- начинайте с небольшого набора данных (для начальных экспериментов, попробуйте переобучиться на батче)
- выбирайте правильную архитектуру (классификация изображений – CNN, последовательности – LSTM/GRU и т.п.)
- Добавление параметров \Rightarrow усложнение сети (больше времени на обучение, риск переобучения)
- Используйте средства борьбы с переобучением (см. выше)
- Если данных слишком много средства могут не понадобиться. Если можно – собирайте данные!
- Используйте уже натренированные модели (не геройствуйте – берите проверенное)
- Learning rate часто самый важный параметр – лучше уменьшать (рекомендуют Adam)
- Есть методы настройки параметров лучше, чем structured (grid) search
- Визуализируйте!
- Смотрите на несколько метрик (обязательно на интерпретируемые)
- 1% Rule (???) – веса должны меняться на 1% от своих значений

Практические советы

- **Правильная инициализация**

ех: при дисбалансе в последнем слое надо так, чтобы выдавалась малая вер-ть

- **если что-то не получается можно понизить размерность**

по-умному (РСА и т.п.) или просто уменьшить картинки

- **если что-то не получается можно упростить / усложнить сеть**

- **меняйте сеть поэтапно (не вносите более одного изменения)**

- **осторожней со смешиванием техник (например, dropout и BN)**

Li et al. «Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift» // <https://arxiv.org/abs/1801.05134>

- **тренируйте дольше**

«One time I accidentally left a model training during the winter break and when I got back in January it was SOTA»

- **не доверяйте встроенным программам понижения темпа**

Практические советы

Советы по настройке сетей

<https://karpathy.github.io/2019/04/25/recipe/>