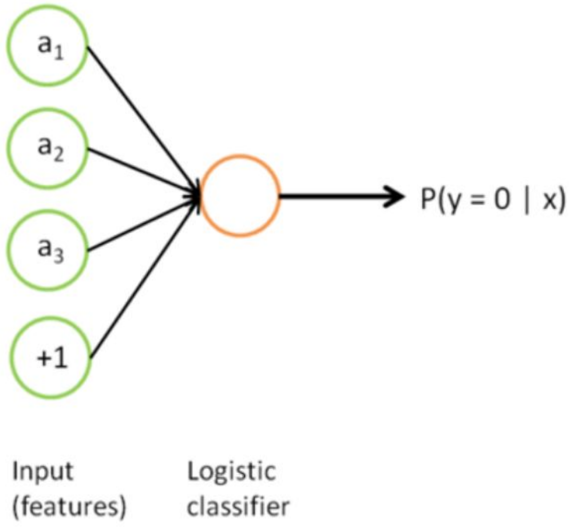
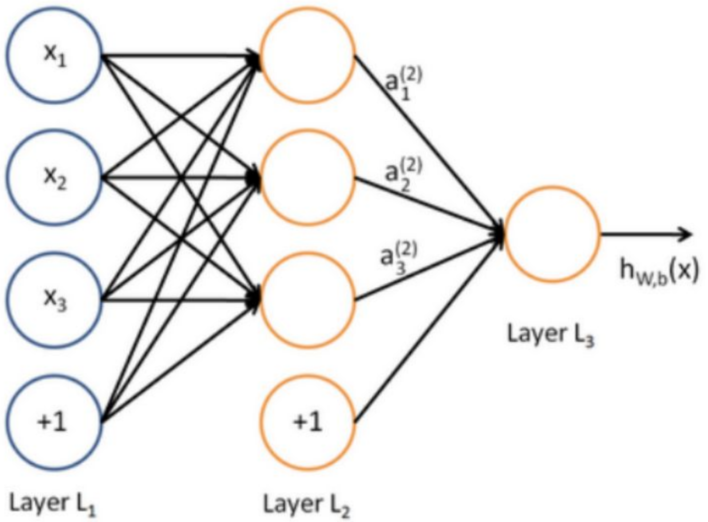
Нейронные сети

Занятие 5
Детали обучения глубоких сетей

Автоматический фич инжениринг



Logistic Regression



Neural Network

Текущий подход:

а) делаем **хороший inductive bias** (NO FREE LUNCH)

б) майним огромный преогромный датасет

в) делаем огромное число параметров (в современных моделях миллиарды)

г) тратим 100 тысяч долларов на обучение на 1000 гпу на неделю

е) ….

ж) PROFIT

Универсальный аппроксиматор

Можно показать, что полносвязная сеть с **одним** скрытым слоем может приблизить с любой точностью любую непрерывную на гиперкубе функцию

? Зачем нужно что-то еще?

? Глубже или шире?

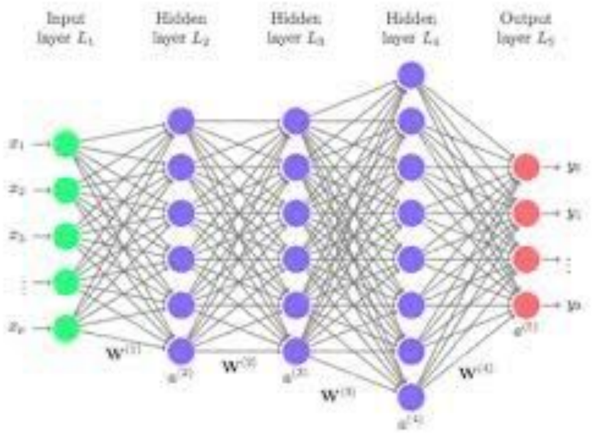
Глубинное обучение

т

Иерархическая структура извлечения признаков.

Каждый новый слой использует предыдущие признаки, чтобы делать новые.

По идее, глубоким сетям проще выучить сложные закономерности.



Программа занятия

1. Технические трюки
2. Борьба с переобучением

Часть 1. Технические трюки

Паралич в сети

input [841]	layer -5	layer -4	layer -3	layer -2	layer -1	output
neurons	100	100	100	100	100	26
$\|grad\|$	6.2e-8	2.2e-6	1.6e-5	1.1e-4	7e-4	0.015

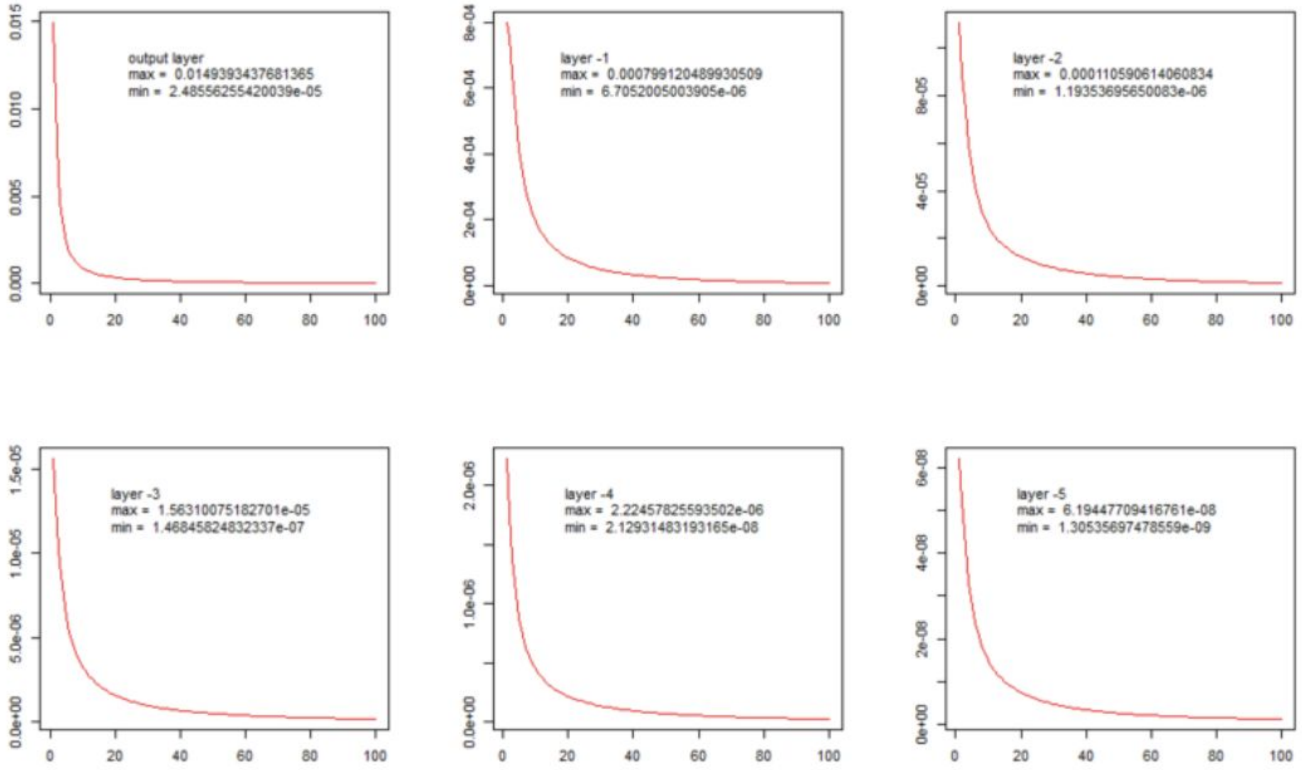


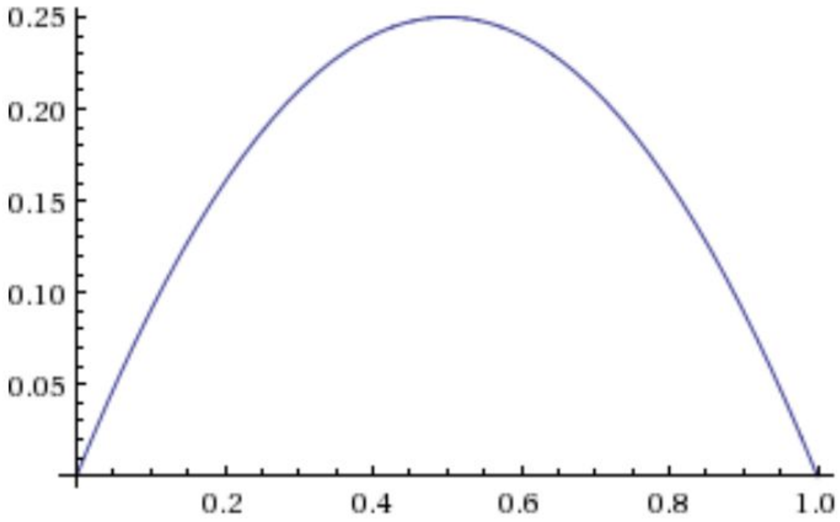
Figure: Средний модуль градиента в различных слоях

Производная сигмоида

Рассмотрим в качестве функции активации логистическую функцию:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$
$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z) \cdot (1 - \sigma(z))$$

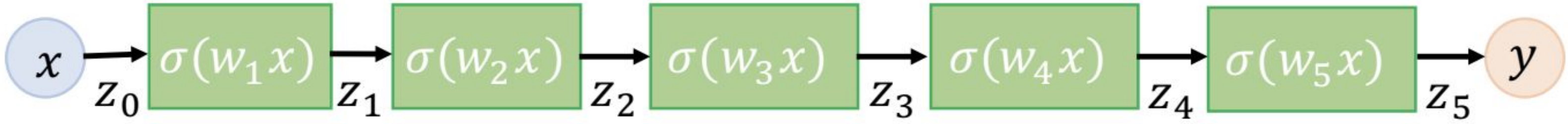
Построим график значений производной:



- максимум равен $\sigma_{\max} = \frac{1}{4}$

Пример подсчета градиентов

Рассмотрим простую сеть (один нейрон в каждом слое):



Прямой проход:

$x = z_0$

$z_k = \sigma(z_{k-1} w_k)$

$y = z_5$

Вычислим градиенты весов для $L(y,t) = \frac{1}{2}(y_j - t_j)^2$:

$$
\begin{aligned}
\frac{\partial L}{\partial z_4} &= \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_4} = \overbrace{(y-t)}^{\leq 2}\overbrace{\sigma'(w_5 z_4)}^{\leq \frac{1}{4}} w_5 \leq 2 \cdot \frac{1}{4} w_5 \\
\frac{\partial L}{\partial z_3} &= \frac{\partial L}{\partial z_4}\frac{\partial z_4}{\partial z_3} \leq 2 \cdot (\frac{1}{4})^2 w_4 w_5 \\
\frac{\partial L}{\partial x} &= \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial x} \leq 2 \cdot (\frac{1}{4})^5 w_1 w_2 w_3 w_4 w_5
\end{aligned}
$$

Резюме

– Градиенты или сильно растут, сеть расходится (exploding gradient problem)
– Или затухают, сеть медленно учится (vanishing gradient problem)
– Чем больше слоев, тем страшнее проблема
– Начнем с выбора функции активации

А можно без функций активации?

Градиент clipping

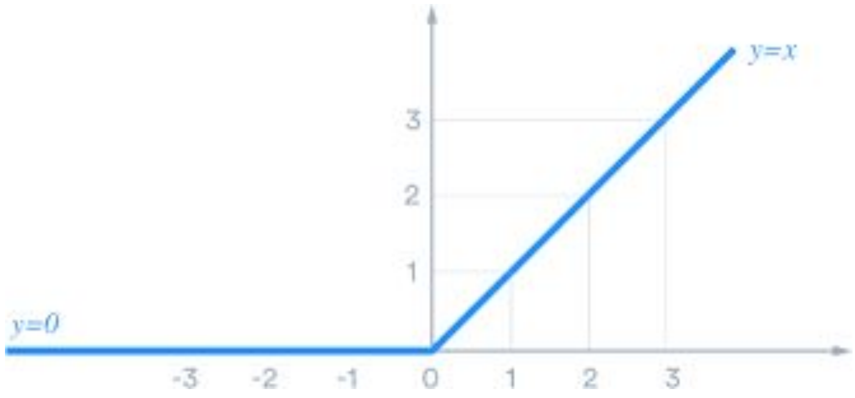
Самое простое рабочее решение – просто отнормировать градиент

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

Другие функции активации

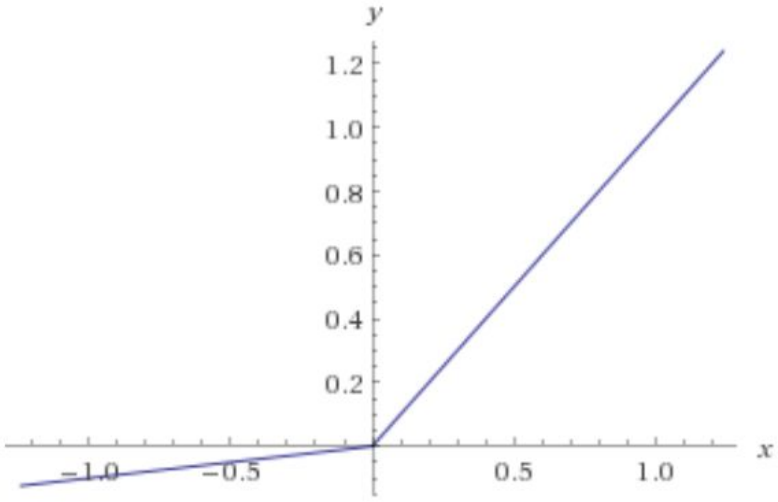
- $\text{ReLU}(x) = max(0, x)$
  - $\frac{d}{dx}\text{ReLU}(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$



Другие функции активации

Dead-relu проблема – если очень низкие b, градиент через нейрон не будет течь.

- $$\text{ELU}(x) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \le 0 \end{cases}, \text{ для } \alpha > 0$$
- $$\text{PReLU}(x) = \begin{cases} x, & x > 0 \\ ax, & x \le 0 \end{cases}$$



Инициализация весов

Задача обучения нейронной сети не является выпуклой, там может быть куча локальных минимумов.
Сделаете большие веса – все взорвется, сделаете маленькие – все затухнет.

? А можно просто нулем(константой) инициализировать?

Инициализация весов

В чем проблема с константой? Дисперсия нейронов нулевая!

Хотим, чтобы

а) дисперсия выходов была постоянной

б) дисперсия градиентов была постоянной

$o_i = \sum_{j=1}^{n_{\text{in}}} w_{ij} x_j$; $\quad w_{ij} \sim \mathcal{N}(0, \sigma^2)$, and $\mathbb{E}[x_j] = 0$ and $\mathbb{V}[x_j] = \gamma^2$,

$$\mathbb{E}[o_i] = \sum_{j=1}^{n_{\text{in}}} \mathbb{E}[w_{ij} x_j] = \sum_{j=1}^{n_{\text{in}}} \mathbb{E}[w_{ij}] \mathbb{E}[x_j] = 0$$

$$\mathbb{V}[o_i] = \mathbb{E}[o_i^2] - (\mathbb{E}[o_i])^2 = \sum_{j=1}^{n_{\text{in}}} \mathbb{E}[w_{ij}^2 x_j^2] - 0 = \sum_{j=1}^{n_{\text{in}}} \mathbb{E}[w_{ij}^2] \mathbb{E}[x_j^2] = n_{\text{in}} \sigma^2 \gamma^2$$

Источник: Probabilistic Machine Learning: An Introduction

Xavier (Glorot)

Хорошая инициализация:

$$\forall (i,j) \begin{cases} \mathbb{D}[z^i] = \mathbb{D}[z^j] \\ \mathbb{D}[\frac{\partial L}{\partial s^i}] = \mathbb{D}[\frac{\partial L}{\partial s^j}] \end{cases}$$

Это эквивалентно следующему:

$$\forall i \begin{cases} n_i \mathbb{D}[W^i] = 1 \\ n_{i+1} \mathbb{D}[W^i] = 1 \end{cases}$$

Компромисс: $\mathbb{D}[W^i] = \frac{2}{n_i + n_{i+1}}$

$$\boxed{W^i \sim U[-\frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}, \frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}]}$$

$\mathbb{D}[U(a,b)] = \frac{1}{12}(b-a)^2$

He

Рассмотрим ReLU в качестве активации:

- Функция не симметрична
- Не дифференцируема в нуле

$$\mathbb{D}[z^i] = \mathbb{D}[x](\prod_{k=0}^{i-1} \frac{1}{2} n_k \mathbb{D}[W^k]) \Rightarrow \mathbb{D}[W^k] = \frac{2}{n_k}$$

$$\mathbb{D}[\frac{\partial L}{\partial s^i}] = \mathbb{D}[\frac{\partial L}{\partial s^d}](\prod_{k=i}^{d} \frac{1}{2} n_{k+1} \mathbb{D}[W^k]) \Rightarrow \mathbb{D}[W^k] = \frac{2}{n_{k+1}}$$

Достаточно использовать только первое уравнение:

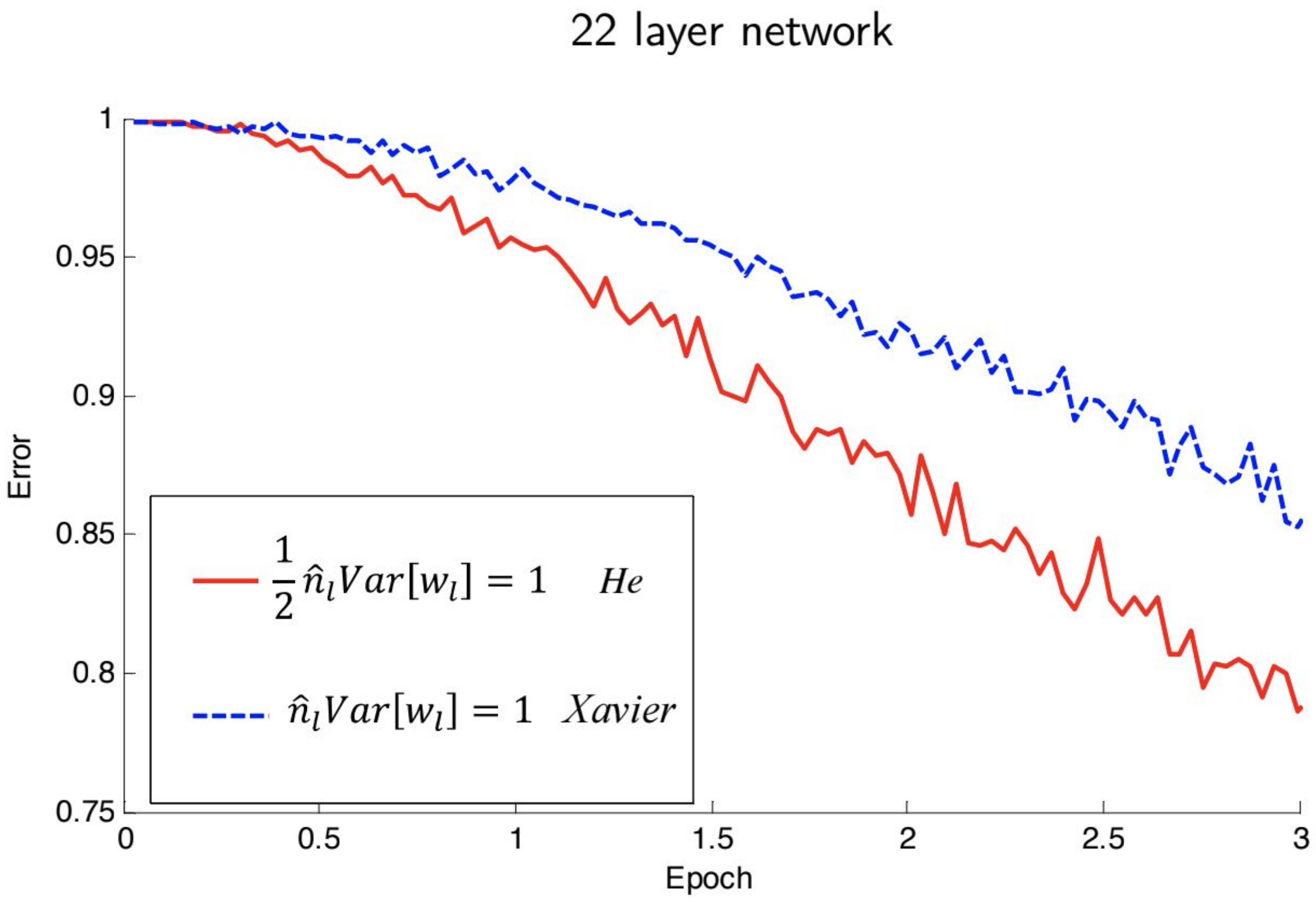
$$\mathbb{D}[\frac{\partial L}{\partial s^i}] = \mathbb{D}[\frac{\partial L}{\partial s^d}] \prod_{k=1}^{d} \frac{1}{2} n_{k+1} \mathbb{D}[W^k] = \frac{n_2}{n_d} \mathbb{D}[\frac{\partial L}{\partial s^d}]$$

n_2/n_d небольшое для сверточных сетей

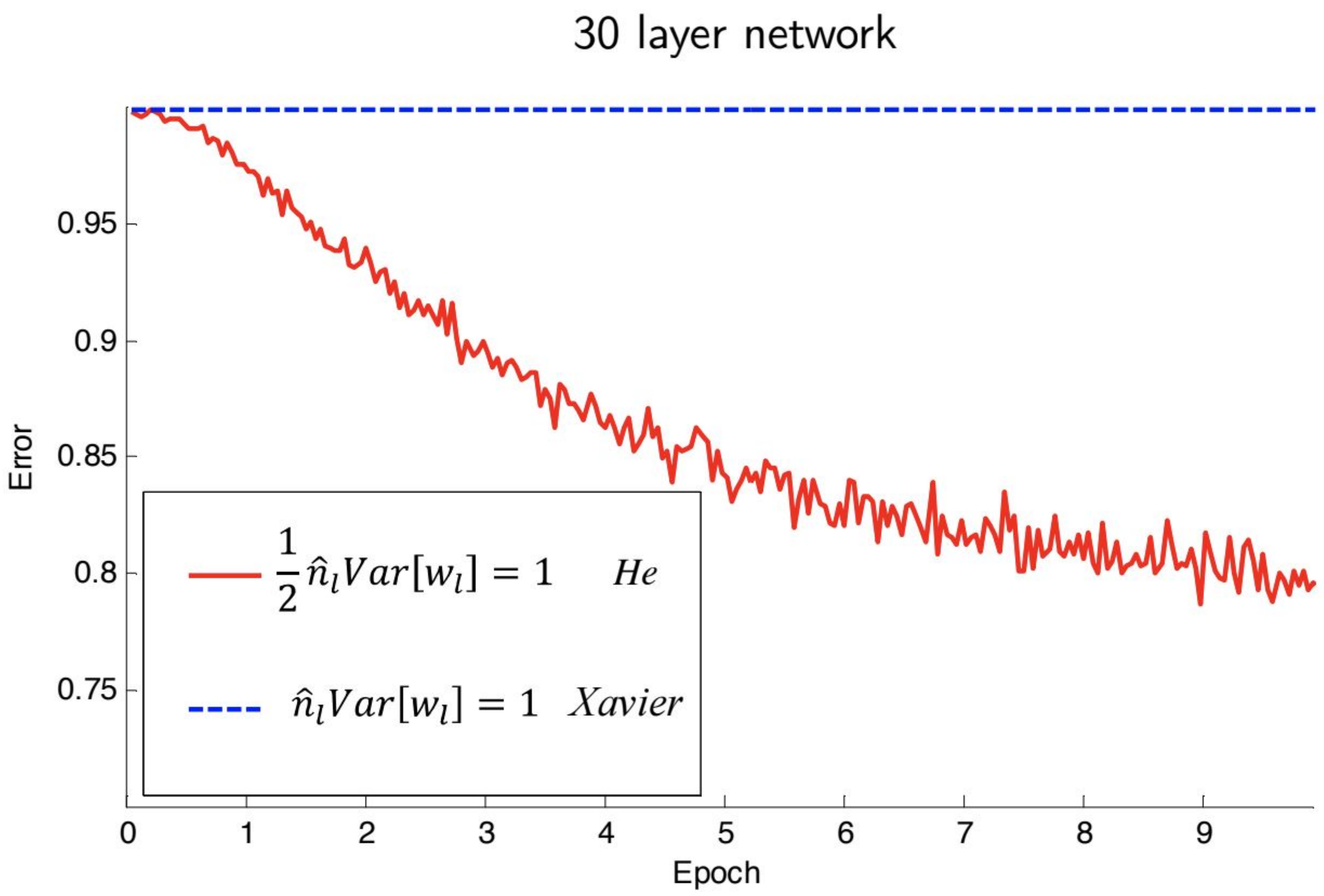
$$\boxed{\begin{array}{c} W^i \sim N(0, \frac{2}{n_i}) \\ \text{or} \\ W^i \sim N(0, \frac{2}{n_{i+1}}) \end{array}}$$

[1] https://arxiv.org/abs/1502.01852

Xavier против He для ReLU



Xavier против He для ReLU

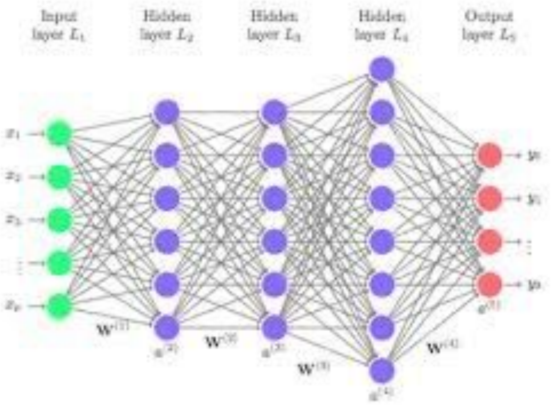


Батч нормализация

- Covariate shift: изменение распределения входов во время обучения
- Цель — уменьшить covariate shift скрытых слоев
- Нормализуем входы в каждый слой $\hat{x}^{(k)} = \frac{x^{(k)} - \mathbb{E}[x^{(k)}]}{\sqrt{\mathbb{D}[x^{(k)}]}}$
- Статистики $\mathbb{E}x$ и $\mathbb{D}x$ оценим для каждого мини-батча

? Почему этот метод плох для сетей с сигмоидами?

- Сигмоиды становятся почти линейными $\Rightarrow$ линейная модель :(
- Доп. параметры: $y^{(k)} = \gamma^{(k)} \hat{x}^{(k)} + \beta^{(k)}$



Батч нормализация

Входы: Значения x в мини-батче $\mathcal{B} = \{x_i\}_{i=1}^{m}$;
Параметры: γ, β
Выход: $\{y_i = \mathsf{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// среднее мини-батча}$$

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

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

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathsf{BN}_{\gamma,\beta}(x_i) \qquad \text{// растяжение и сдвиг}$$

? А что делать на тесте?

? Можно ли это рассматривать как борьба с переобучением?

Батч нормализация

Во время предсказания батч-нормализация является линейным слоем:

$$\hat{x} = \frac{x - \mathbb{E}[x]}{\sqrt{\mathbb{D}[x] + \epsilon}}$$

$$y = \gamma \cdot \hat{x} + \beta$$

$$y = \frac{\gamma}{\sqrt{\mathbb{D}[x] + \epsilon}} \cdot x + (\beta - \frac{\gamma \mathbb{E}[x]}{\sqrt{\mathbb{D}[x] + \epsilon}})$$

$\mathbb{E}[x]$ и $\mathbb{D}[x]$ вычисляются по всему обучающему множеству.
На практике статистики вычисляются во время обучения экспоненциальным средним: $E_{i+1} = (1 - \alpha)E_i + \alpha E_{\mathcal{B}}$

`nn.BatchNorm1d nn.BatchNorm2d`

во время обучения model.train(), во время инференса model.eval()

Батч нормализация

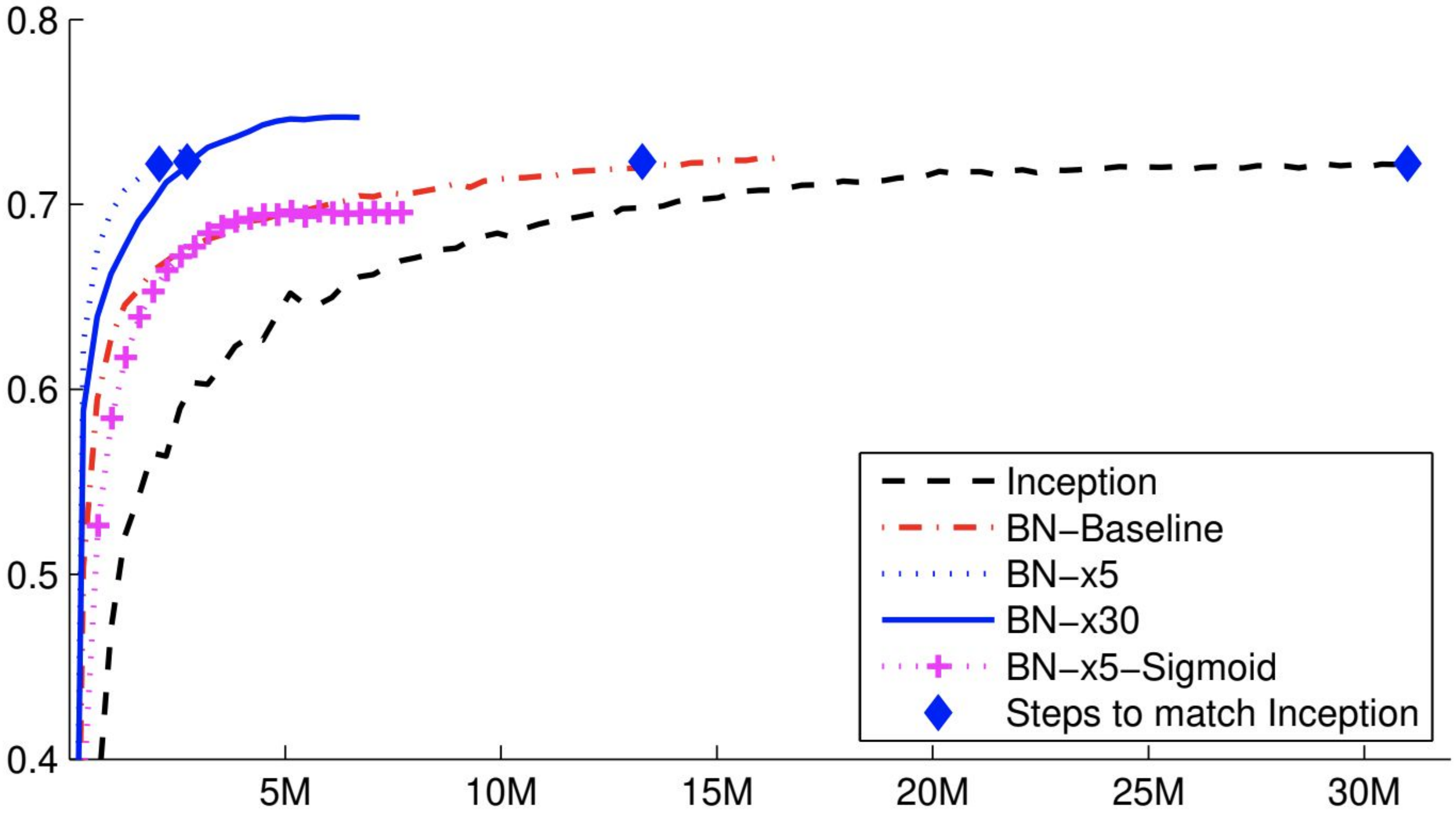
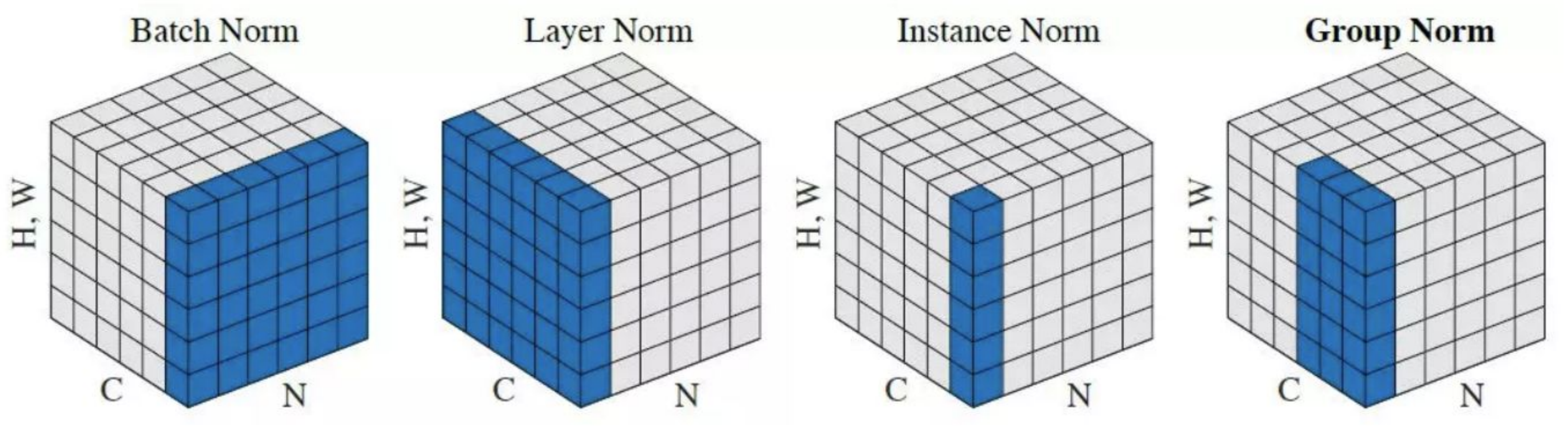


Figure: Обучение Inception с и без батч-нормализации.[6]

[6] x30 — увеличение темпа обучения в 30 раз

Леер нормализация

Проблема: еще один параметр, размер батча



$$\mu_i = \frac{1}{|\mathcal{S}_i|} \sum_{k \in \mathcal{S}_i} z_k, \quad \sigma_i = \sqrt{\frac{1}{|\mathcal{S}_i|} \sum_{k \in \mathcal{S}_i} (z_k - \mu_i)^2 + \epsilon}$$

$$i = (i_N, i_H, i_W, i_C).$$

Источник: Probabilistic Machine Learning: An Introduction

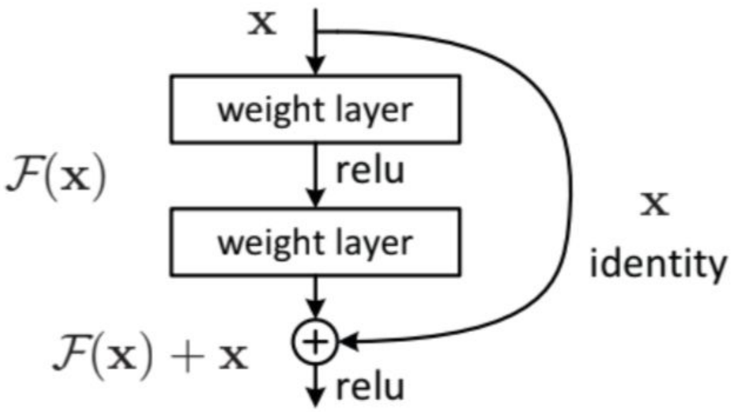
`torch.nn.LayerNorm`

Residual connection

Эффективное решение, чтобы бороться с vanishing gradients

$$\mathcal{F}'_l(\boldsymbol{x}) = \mathcal{F}_l(\boldsymbol{x}) + \boldsymbol{x} \qquad \boldsymbol{z}_L = \boldsymbol{z}_l + \sum_{i=l}^{L-1} \mathcal{F}_i(\boldsymbol{z}_i; \boldsymbol{\theta}_i).$$

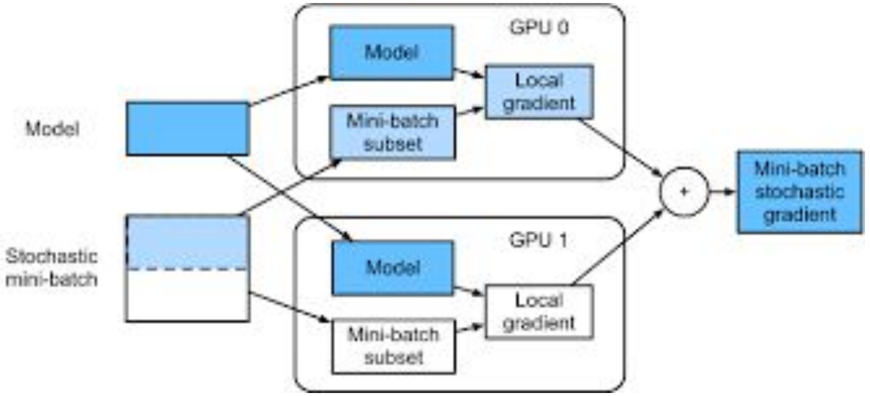
$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_l} &= \frac{\partial \boldsymbol{z}_l}{\partial \boldsymbol{\theta}_l} \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_l} \\
&= \frac{\partial \boldsymbol{z}_l}{\partial \boldsymbol{\theta}_l} \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_L} \frac{\partial \boldsymbol{z}_L}{\partial \boldsymbol{z}_l} \\
&= \frac{\partial \boldsymbol{z}_l}{\partial \boldsymbol{\theta}_l} \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_L} \left(1 + \sum_{i=l}^{L-1} \frac{\partial \mathcal{F}_i(\boldsymbol{z}_i; \boldsymbol{\theta}_i)}{\partial \boldsymbol{z}_l}\right) \\
&= \frac{\partial \boldsymbol{z}_l}{\partial \boldsymbol{\theta}_l} \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_L} + \text{otherterms}
\end{aligned}$$



Источник: Probabilistic Machine Learning: An Introduction

Параллельное обучение

Глубокие сети учить надо долго, чтобы ускориться нужно несколько GPU
Есть параллелизм по модели и по данным
https://pytorch.org/tutorials/beginner/blitz/data_parallel_tutorial.html

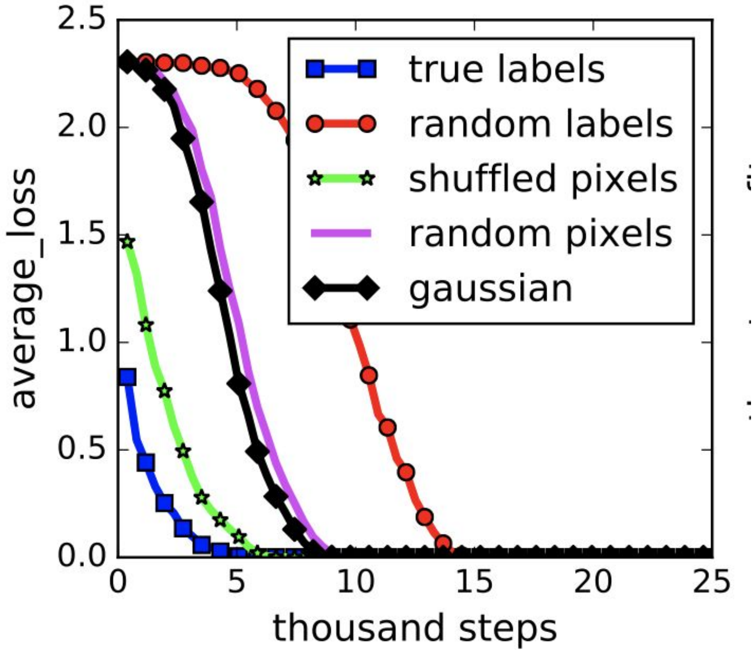


Источник: Probabilistic Machine Learning: An Introduction

Часть 2. Борьба с переобучением

Насколько оверпараметризованы современные сети

Очень сильно https://arxiv.org/pdf/1611.03530.pdf



(a) learning curves

?

Почему это должно шокировать?

Регуляризация

Метод максимума правдоподобия "хорошо работает", только если число объектов много больше числа параметров. А если вы 3 раза подбросили монетку?

Спасает формула Байеса!

Хотим делать не ММП, а MAP, чтобы уменьшить переобучение.

$\theta_{MAP} = \text{argmax} L(\theta)p(\theta) = \text{argmax} \log L(\theta) + \log p(\theta)$

Пусть каждая компонента $p(\theta)$ распределена нормально с нулевым средним и все компоненты независимы и имеют одинаковую дисперсию σ^2:

$$p(\theta) = \prod_{j=1}^{D} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\theta_j^2}{2\sigma^2}\right)$$

$$\theta_{MAP} = \text{argmax} \sum_{i=1}^{N} y_i \log p(y|x,\theta) + (1-y_i)\log(1-p(y|x,\theta)) - \sum_{j=1}^{D} C\frac{\theta_j^2}{\sigma^2}$$

$$\theta_{MAP} = \text{argmax} \sum_{i=1}^{N} y_i \log p(y|x,\theta) + (1-y_i)\log(1-p(y|x,\theta)) - \frac{C}{\sigma^2} \cdot ||\theta||^2$$

Берем с минусом, получаем loglocc с l2 регуляризацией!

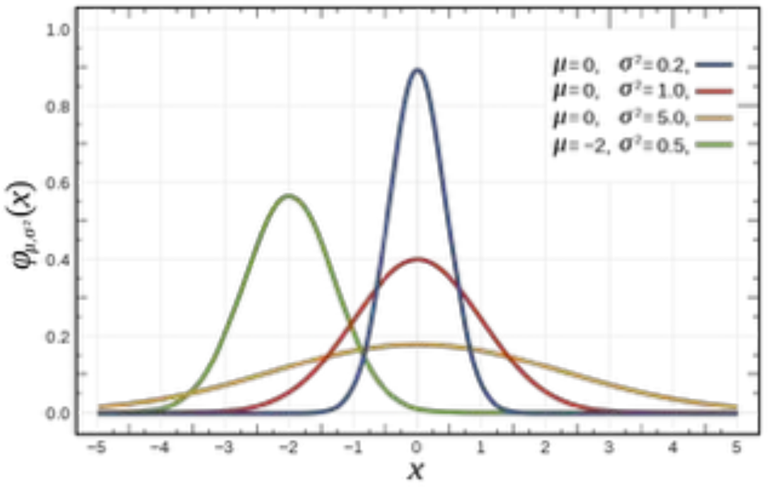
Какой смысл у дисперсии σ^2?

$$p(\theta|Y) = \frac{p(Y|\theta)p(\theta)}{p(Y)}$$

$$\theta_{MAP} = \text{argmax} p(\theta|Y) = \text{argmax} p(Y|\theta)p(\theta)$$

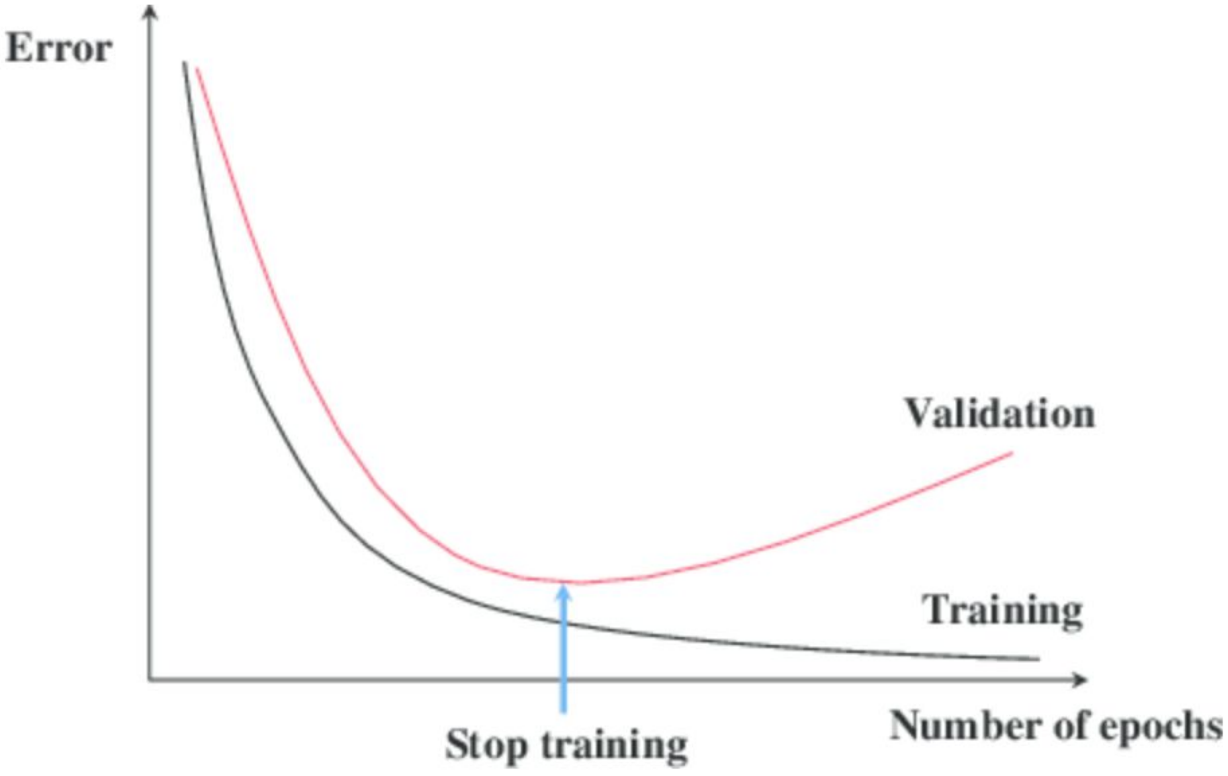
Регуляризация

Мы хотим, чтобы наши веса были маленькими, потому что большие веса — источник переобучения. Чем меньше сигма, тем уже нормальное распределение, то есть тем сильнее мы регуляризуем `torch.optim.Adam(params, lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0,`



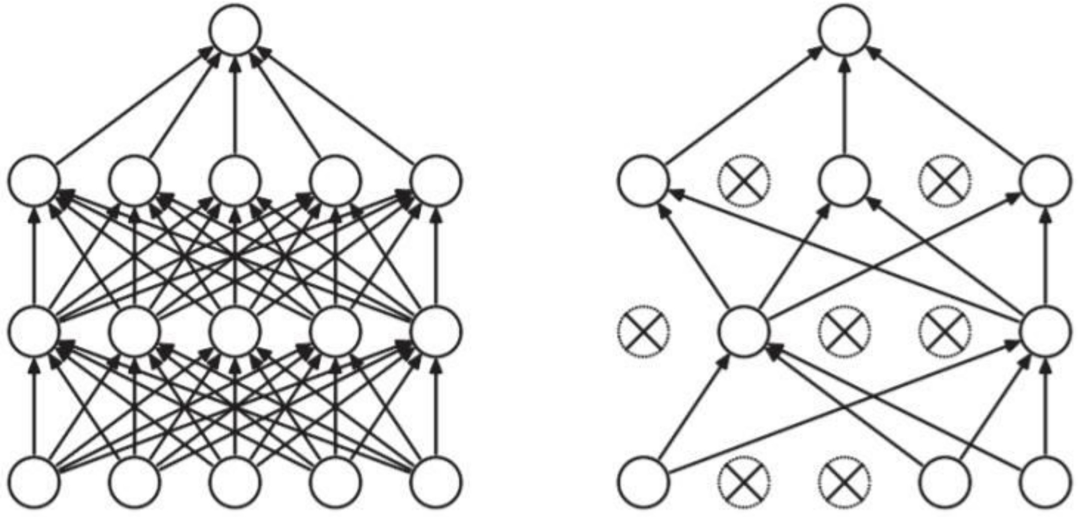
Early stopping

Со временем модель учиться уже не обобщать зависимости, а просто учится запоминать обучение



Dropout

С вероятностью p занулим выход нейрона (например, p = 0.3)
Борьба с соадаптацией – нейроны больше не могут рассчитывать на наличие соседей
Биология: не все гены родителей будут присутствовать у потомков
Усреднение большого 2^n моделей



(a) Standard Neural Net (b) After applying dropout.

?

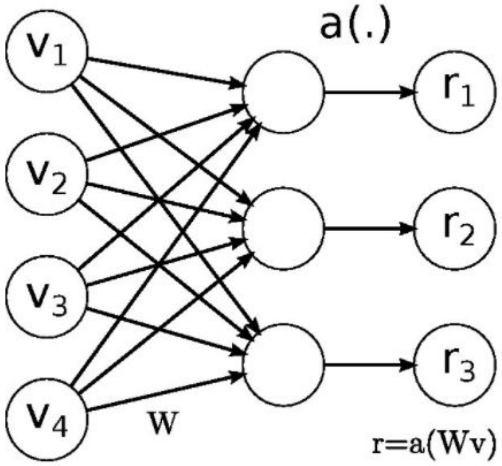
А во время инференса?

Dropout

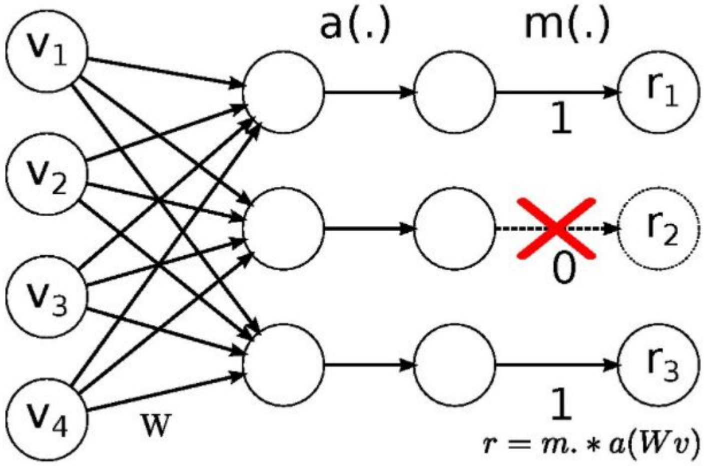
В тесте домножаем выход каждый нейрона не вероятность быть не выкинутым. Надо сделать model.eval()

Не стоит выкидывать нейроны последнего слоя

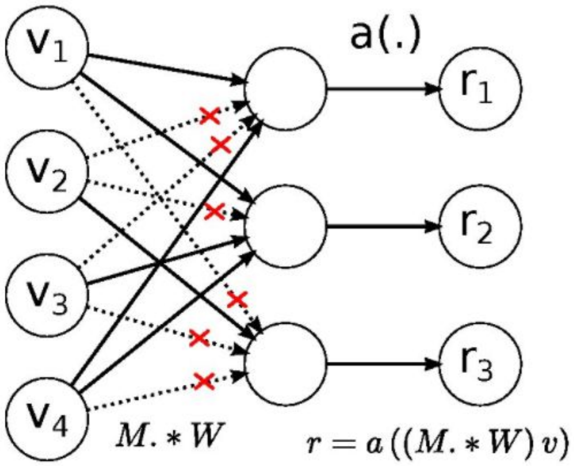
Dropconnect



No-Drop Network



DropOut Network



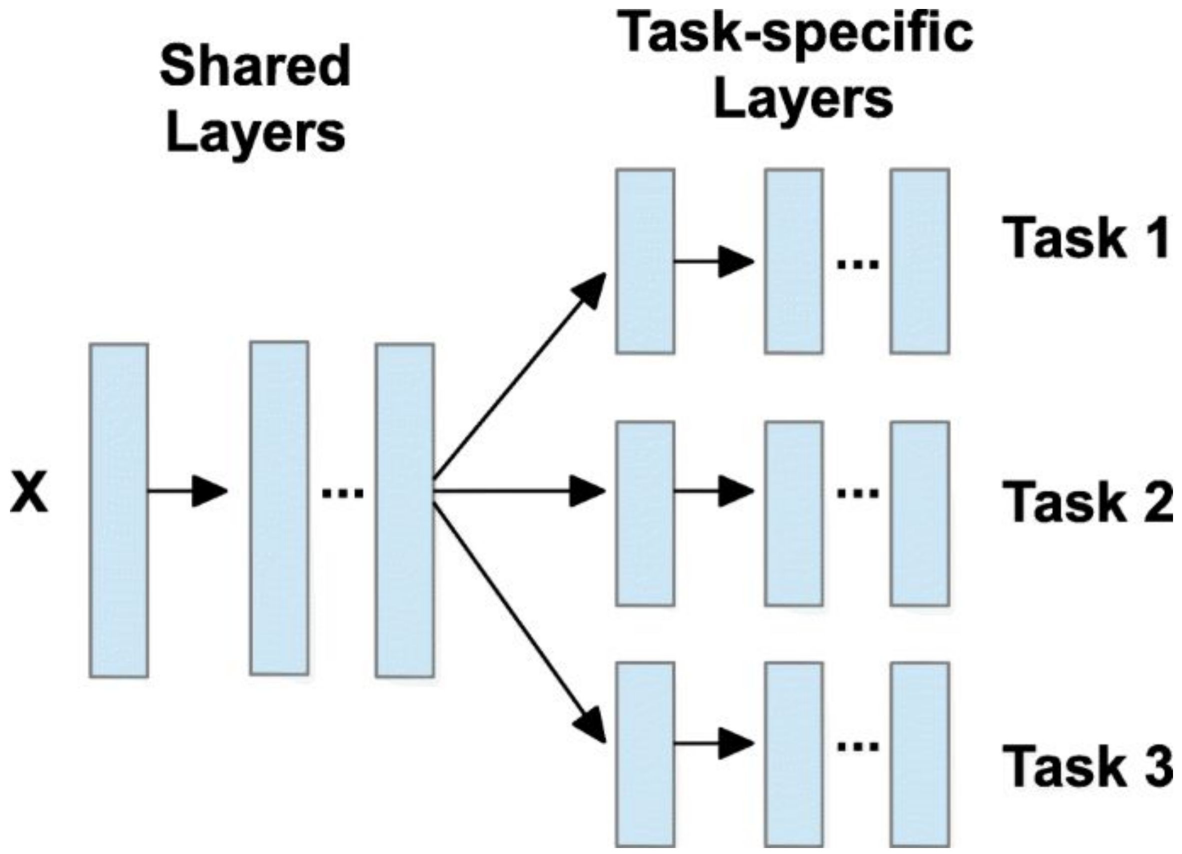
DropConnect Network

Аугментация

Небольшие преобразования входных данных, которые не меняют таргет.
– Для картинок сдвиги, повороты, кроп и тд
– Для текстов: синонимы, перевод в обе стороны
– Для аудио: добавления шумов, музыки, ускорение аудио и тд

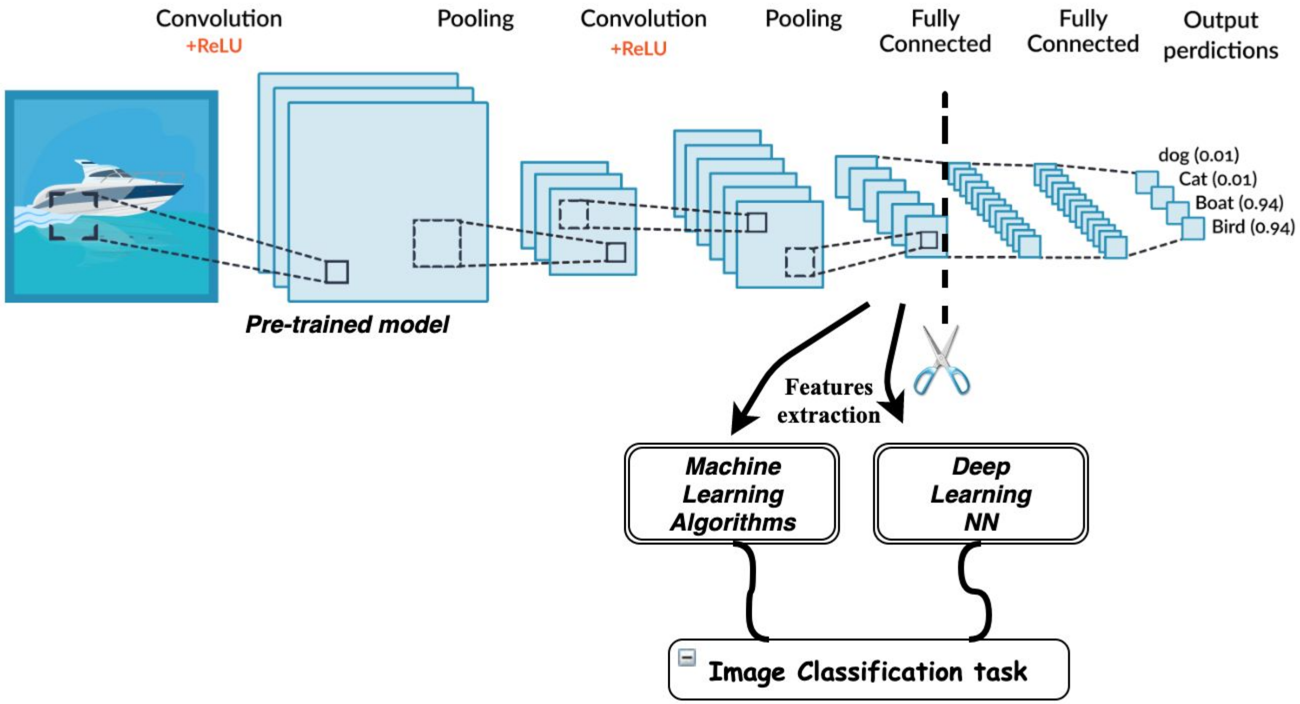
Multi task learning

Если данных мало, можно учиться на связанных задачах, чтобы модели было сложнее переобучиться



Transfer learning

Если данных мало, чтобы учить модель с нуля, то можно доучить другую модель



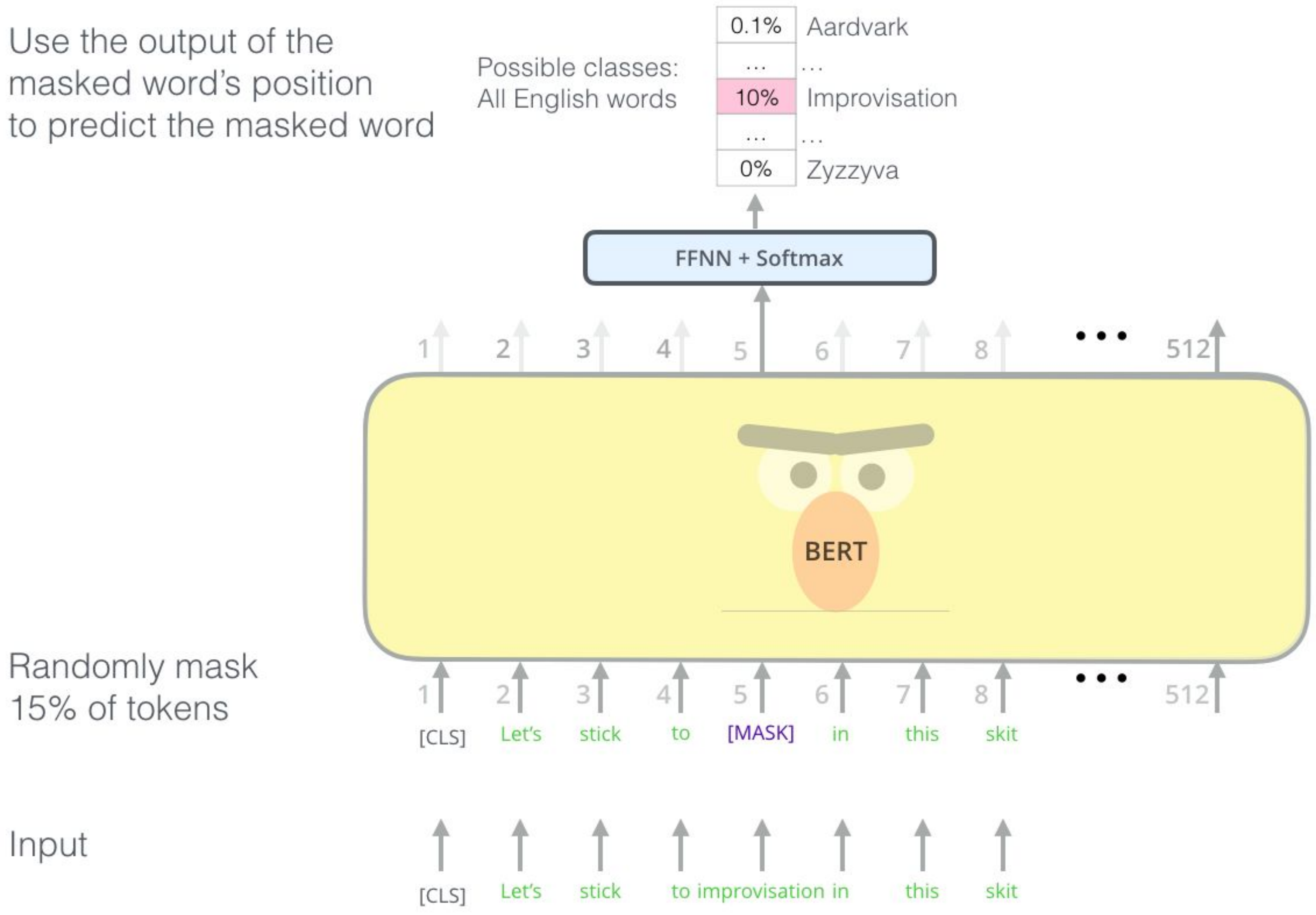

```
for param in model.parameters():
    param.requires_grad = False
```

Semi supervised learing

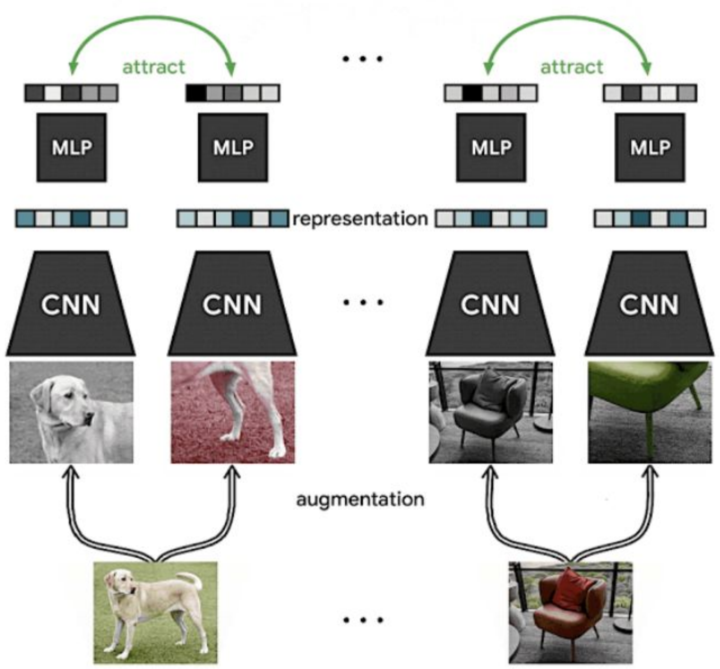
Трансфер лернинг это предобучение нейронной сети на другом сете с другими метками. А если у нас нет меток?
Self-supervised learning – супер горячая тема DL, позволяет учить просто ЖИРНЮЩИЕ модели без меток.
Self-supervised learning – предобучение модели на задаче без меток.

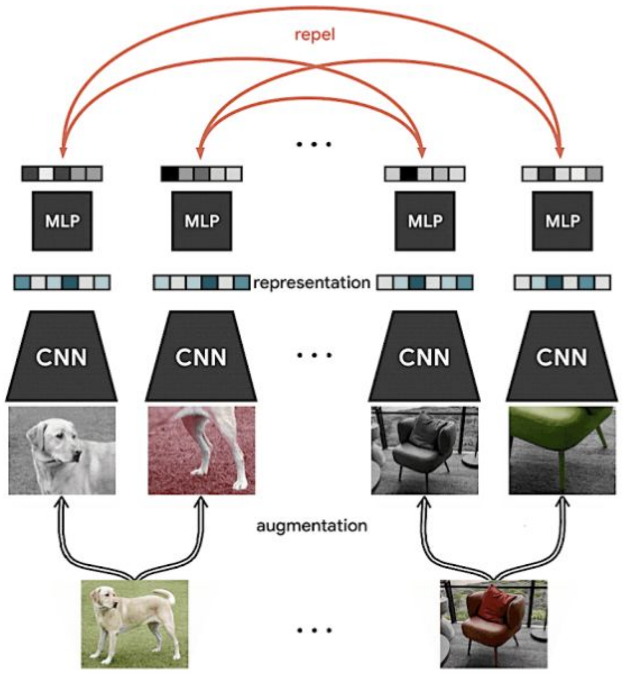
<здесь место для воды>

Fill in the blank



Contrastive learning





SIMCLR

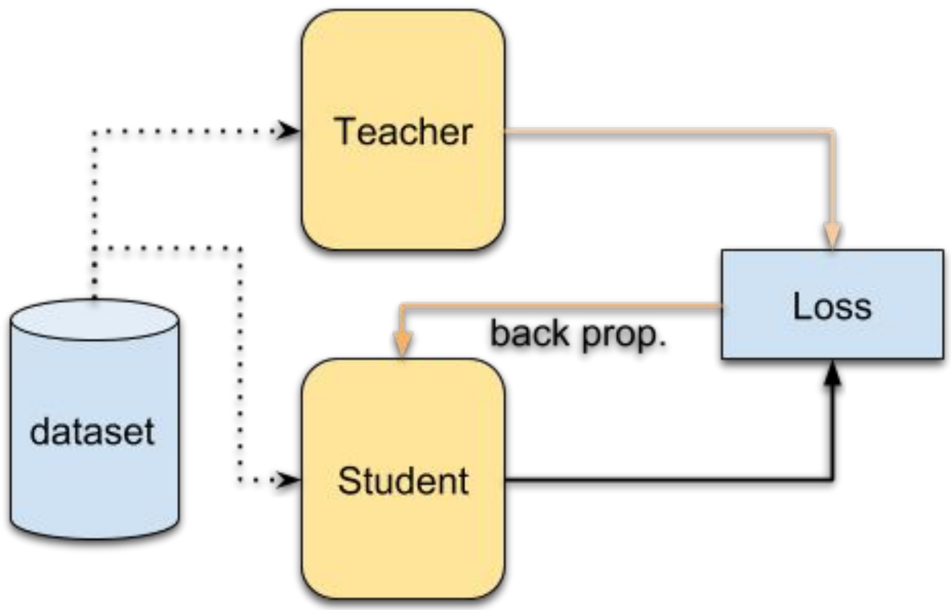
https://ai.googleblog.com/2020/04/advancing-self-supervised-and-semi.html

Очень важный слайд

Сейчас это все супер популярно в текстах/звуке/картинках.
Эти подходы выбивают SOTA (state of the art) качество в задачах с небольшой выборкой.
Если у вас 1ккк картинок с правильными таргетами, вам скорее всего ничего этого не нужно уже.

Knowledge distillation

Если кто-то обучил глубокую модель за вас



Итоговое резюме

Сегодня мы:

- Поняли, что нам нужны ГЛУБОЧАЙШИЕ сети, чтобы учить сложные иерархические закономерности в данных;
- Увидели некоторые проблемы, которые возникают во время обучения;
- Посмотрели способы решения этих проблем;
- Узнали, как можно бороться с переобучением в нейросетях.

Вопросы?