

# Обучение здоровенных моделей с точки зрения методов оптимизации

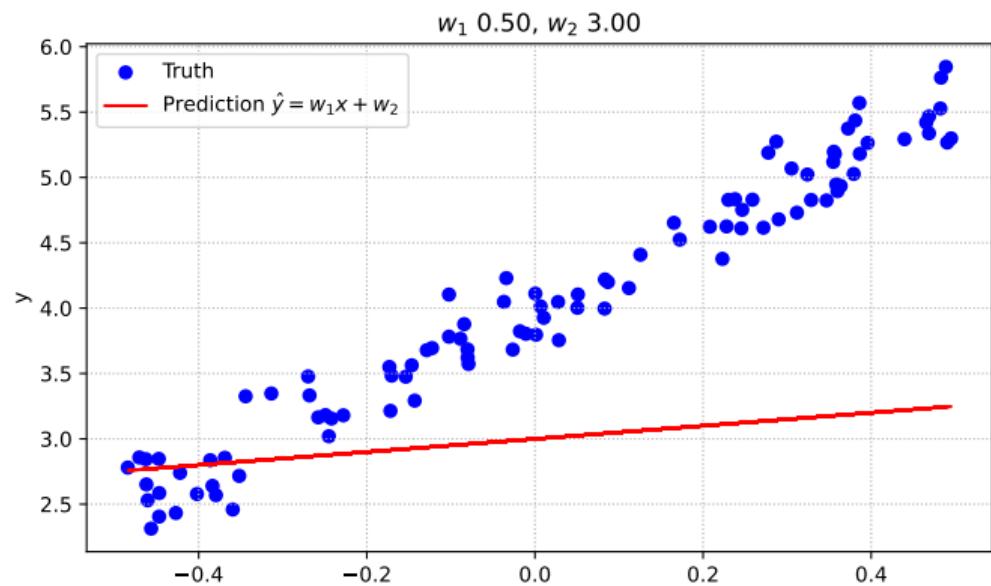
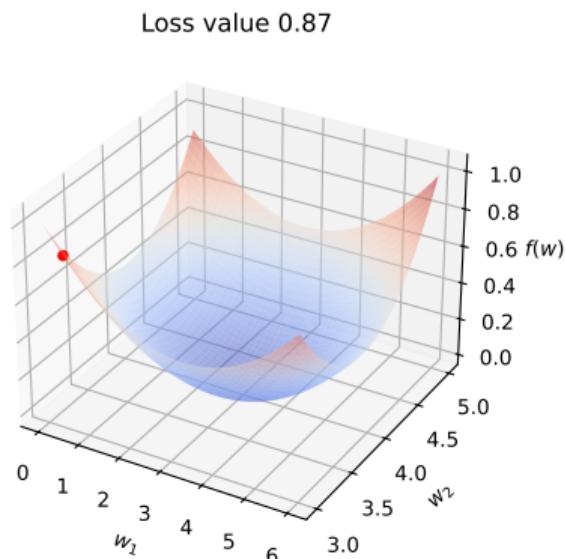
Даниил Меркулов

Международная олимпиада по искусственному интеллекту и анализу данных

30 ноября 2024

## Обучение как задача оптимизации

# Градиентный спуск для линейной регрессии



# Методы оптимизации

$$\min_{x \in \mathbb{R}^n} f(x)$$

## Методы нулевого порядка

- Метод Нелдера - Мида
- Эволюционные методы
- Генетические алгоритмы
- Безградиентные методы
- и другие...

# Методы оптимизации

$$\min_{x \in \mathbb{R}^n} f(x)$$

## Методы нулевого порядка

- Метод Нелдера - Мида
- Эволюционные методы
- Генетические алгоритмы
- Безградиентные методы
- и другие...

## Методы первого порядка

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

# Методы оптимизации

$$\min_{x \in \mathbb{R}^n} f(x)$$

## Методы нулевого порядка

- Метод Нелдера - Мида
- Эволюционные методы
- Генетические алгоритмы
- Безградиентные методы
- и другие...

## Методы первого порядка

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

## Методы второго порядка

$$x_{k+1} = x_k - \alpha_k [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$$

## Проклятие размерности методов нулевого порядка

$$\min_{x \in \mathbb{R}^n} f(x)$$

## Проклятие размерности методов нулевого порядка

$$\min_{x \in \mathbb{R}^n} f(x)$$

GD:  $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$

Zero order GD:  $x_{k+1} = x_k - \alpha_k G,$

где  $G$  - двухточечная или многоточечная оценка градиента по значениям функции

## Проклятие размерности методов нулевого порядка

$$\min_{x \in \mathbb{R}^n} f(x)$$

GD:  $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$

Zero order GD:  $x_{k+1} = x_k - \alpha_k G,$

где  $G$  - двухточечная или многоточечная оценка градиента по значениям функции

	$f(x)$ - гладкая	$f(x)$ - гладкая и выпуклая	$f(x)$ - гладкая и сильно выпуклая
GD	$\ \nabla f(x_k)\ ^2 \approx \mathcal{O}\left(\frac{1}{k}\right)$	$f(x_k) - f^* \approx \mathcal{O}\left(\frac{1}{k}\right)$	$\ x_k - x^*\ ^2 \approx \mathcal{O}\left(\left(1 - \frac{\mu}{L}\right)^k\right)$
Zero order GD	$\ \nabla f(x_k)\ ^2 \approx \mathcal{O}\left(\frac{n}{k}\right)$	$f(x_k) - f^* \approx \mathcal{O}\left(\frac{n}{k}\right)$	$\ x_k - x^*\ ^2 \approx \mathcal{O}\left(\left(1 - \frac{\mu}{nL}\right)^k\right)$

## Пример: задача многомерного шкалирования

Пусть у нас есть матрица попарных расстояний для  $N$   $d$ -мерных объектов  $D \in \mathbb{R}^{N \times N}$ . По этой матрице мы должны восстановить начальные координаты  $W_i \in \mathbb{R}^d$ ,  $i = 1, \dots, N$ .

## Пример: задача многомерного шкалирования

Пусть у нас есть матрица попарных расстояний для  $N$   $d$ -мерных объектов  $D \in \mathbb{R}^{N \times N}$ . По этой матрице мы должны восстановить начальные координаты  $W_i \in \mathbb{R}^d$ ,  $i = 1, \dots, N$ .

$$L(W) = \sum_{i,j=1}^N (\|W_i - W_j\|_2^2 - D_{i,j})^2 \rightarrow \min_{W \in \mathbb{R}^{N \times d}}$$

## Пример: задача многомерного шкалирования



## В машинном обучении задача оптимизации часто имеет особенный вид

$$\min_{x \in \mathbb{R}^n} f(x) = \min_{x \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^N f_i(x)$$

- $f_i(x)$  - значение функции потерь модели при весах  $x$  на  $i$ -ом объекте обучающей выборки

## В машинном обучении задача оптимизации часто имеет особенный вид

$$\min_{x \in \mathbb{R}^n} f(x) = \min_{x \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^N f_i(x)$$

- $f_i(x)$  - значение функции потерь модели при весах  $x$  на  $i$ -ом объекте обучающей выборки
- $n$  - число обучаемых параметров модели ( $175 \cdot 10^9$  для GPT-3.5,  $405 \cdot 10^9$  для Llama 3.2)

## В машинном обучении задача оптимизации часто имеет особенный вид

$$\min_{x \in \mathbb{R}^n} f(x) = \min_{x \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^N f_i(x)$$

- $f_i(x)$  - значение функции потерь модели при весах  $x$  на  $i$ -ом объекте обучающей выборки
- $n$  - число обучаемых параметров модели ( $175 \cdot 10^9$  для GPT-3.5,  $405 \cdot 10^9$  для Llama 3.2)
- $N$  - размер обучающей выборки (для ImageNet  $\approx 1.4 \cdot 10^7$ , для WikiText  $\approx 10^8$ , для FineWeb-Edu  $\approx 1.3 \cdot 10^{12}$ ).

## В машинном обучении задача оптимизации часто имеет особенный вид

$$\min_{x \in \mathbb{R}^n} f(x) = \min_{x \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^N f_i(x)$$

- $f_i(x)$  - значение функции потерь модели при весах  $x$  на  $i$ -ом объекте обучающей выборки
- $n$  - число обучаемых параметров модели ( $175 \cdot 10^9$  для GPT-3.5,  $405 \cdot 10^9$  для Llama 3.2)
- $N$  - размер обучающей выборки (для ImageNet  $\approx 1.4 \cdot 10^7$ , для WikiText  $\approx 10^8$ , для FineWeb-Edu  $\approx 1.3 \cdot 10^{12}$ ).

## В машинном обучении задача оптимизации часто имеет особенный вид

$$\min_{x \in \mathbb{R}^n} f(x) = \min_{x \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^N f_i(x)$$

- $f_i(x)$  - значение функции потерь модели при весах  $x$  на  $i$ -ом объекте обучающей выборки
- $n$  - число обучаемых параметров модели ( $175 \cdot 10^9$  для GPT-3.5,  $405 \cdot 10^9$  для Llama 3.2)
- $N$  - размер обучающей выборки (для ImageNet  $\approx 1.4 \cdot 10^7$ , для WikiText  $\approx 10^8$ , для FineWeb-Edu  $\approx 1.3 \cdot 10^{12}$ ).

$$\nabla f(x_k) = \frac{1}{N} \sum_{i=1}^N \nabla f_i(x)$$

## В машинном обучении задача оптимизации часто имеет особенный вид

$$\min_{x \in \mathbb{R}^n} f(x) = \min_{x \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^N f_i(x)$$

- $f_i(x)$  - значение функции потерь модели при весах  $x$  на  $i$ -ом объекте обучающей выборки
- $n$  - число обучаемых параметров модели ( $175 \cdot 10^9$  для GPT-3.5,  $405 \cdot 10^9$  для Llama 3.2)
- $N$  - размер обучающей выборки (для ImageNet  $\approx 1.4 \cdot 10^7$ , для WikiText  $\approx 10^8$ , для FineWeb-Edu  $\approx 1.3 \cdot 10^{12}$ ).

$$\nabla f(x_k) = \frac{1}{N} \sum_{i=1}^N \nabla f_i(x)$$

$$x_{k+1} = x_k - \frac{\alpha_k}{N} \sum_{i=1}^N \nabla f_i(x) \tag{GD}$$

Тяжело считать при больших  $N$ !

## В машинном обучении задача оптимизации часто имеет особенный вид

$$\min_{x \in \mathbb{R}^n} f(x) = \min_{x \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^N f_i(x)$$

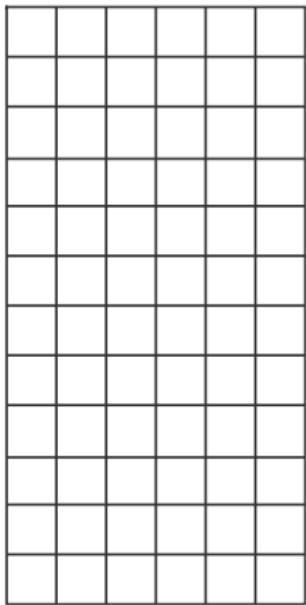
- $f_i(x)$  - значение функции потерь модели при весах  $x$  на  $i$ -ом объекте обучающей выборки
- $n$  - число обучаемых параметров модели ( $175 \cdot 10^9$  для GPT-3.5,  $405 \cdot 10^9$  для Llama 3.2)
- $N$  - размер обучающей выборки (для ImageNet  $\approx 1.4 \cdot 10^7$ , для WikiText  $\approx 10^8$ , для FineWeb-Edu  $\approx 1.3 \cdot 10^{12}$ ).

$$\nabla f(x_k) \approx \frac{1}{b} \sum_{i=1}^b \nabla f_{j_i}(x)$$

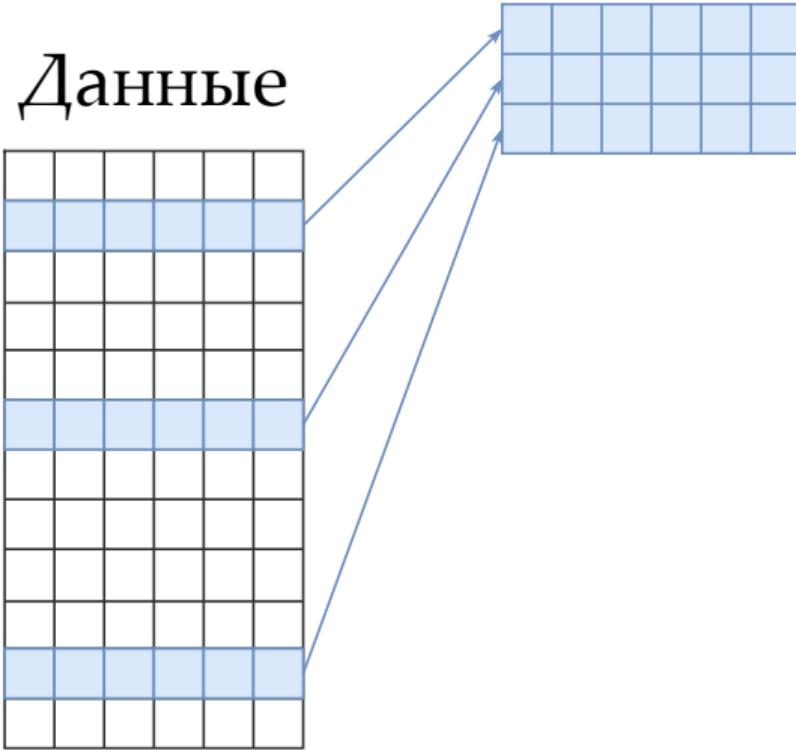
$$x_{k+1} = x_k - \frac{\alpha_k}{b} \sum_{i=1}^b \nabla f_{j_i}(x) \tag{SGD}$$

Можно считать при больших  $N$ !

### Данные

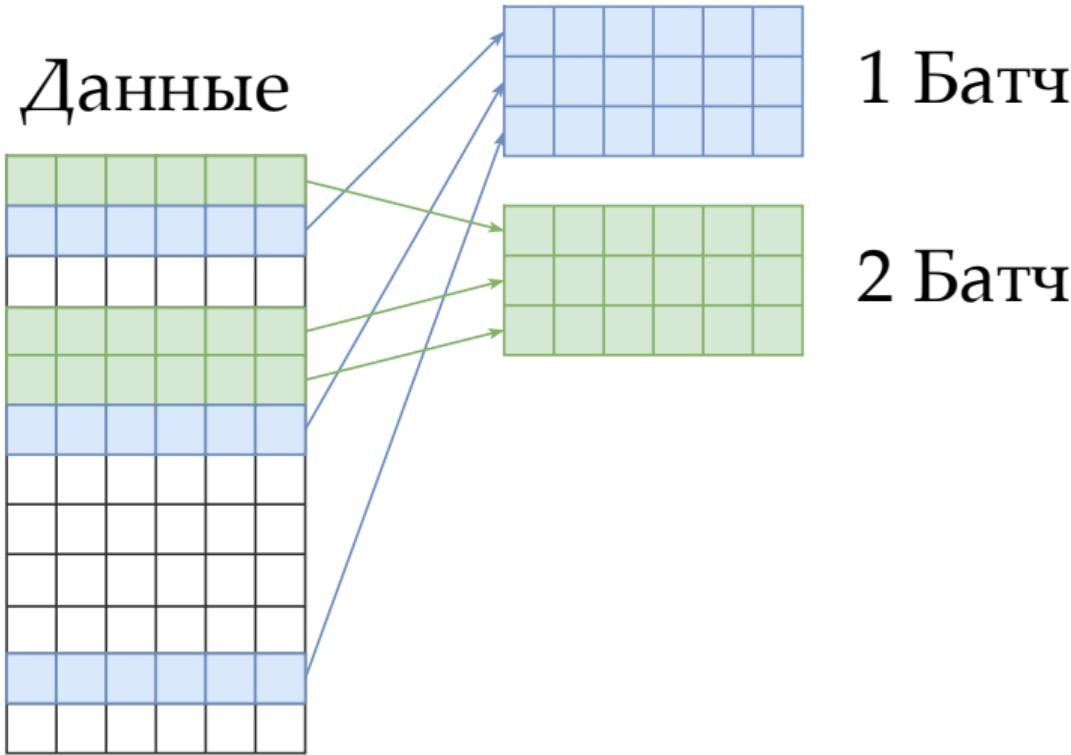


## Идея SGD и батчей

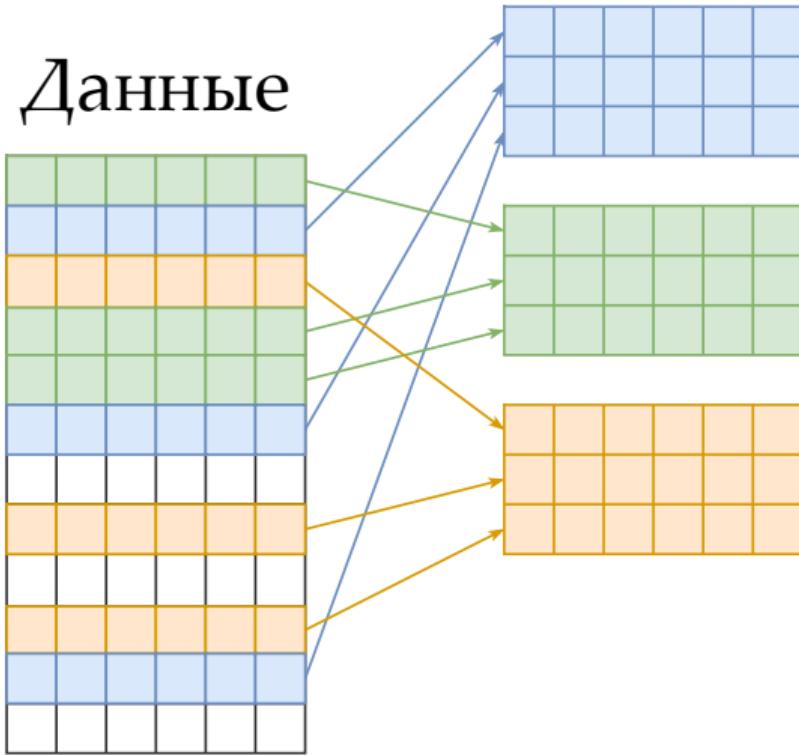


1 Батч

## Идея SGD и батчей



## Идея SGD и батчей

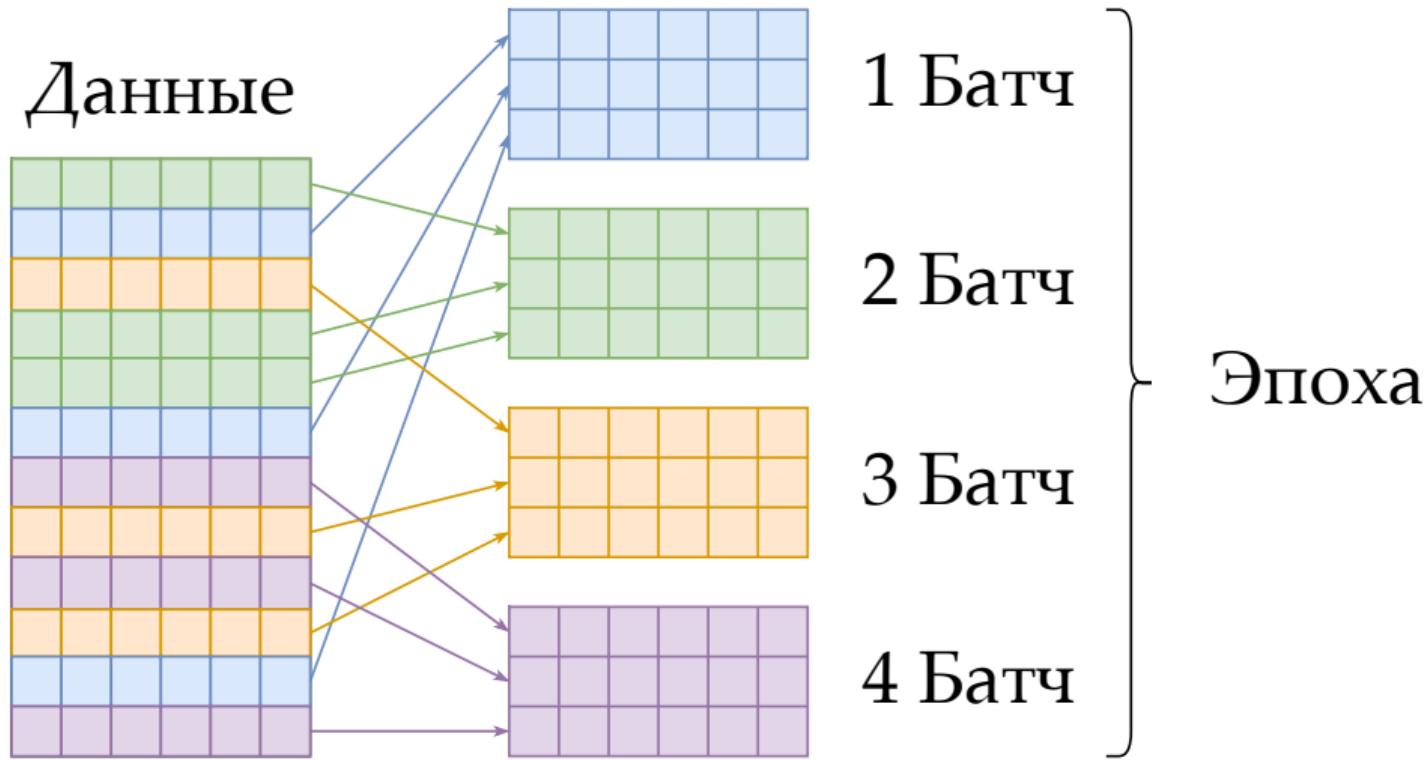


1 Батч

2 Батч

3 Батч

## Идея SGD и батчей



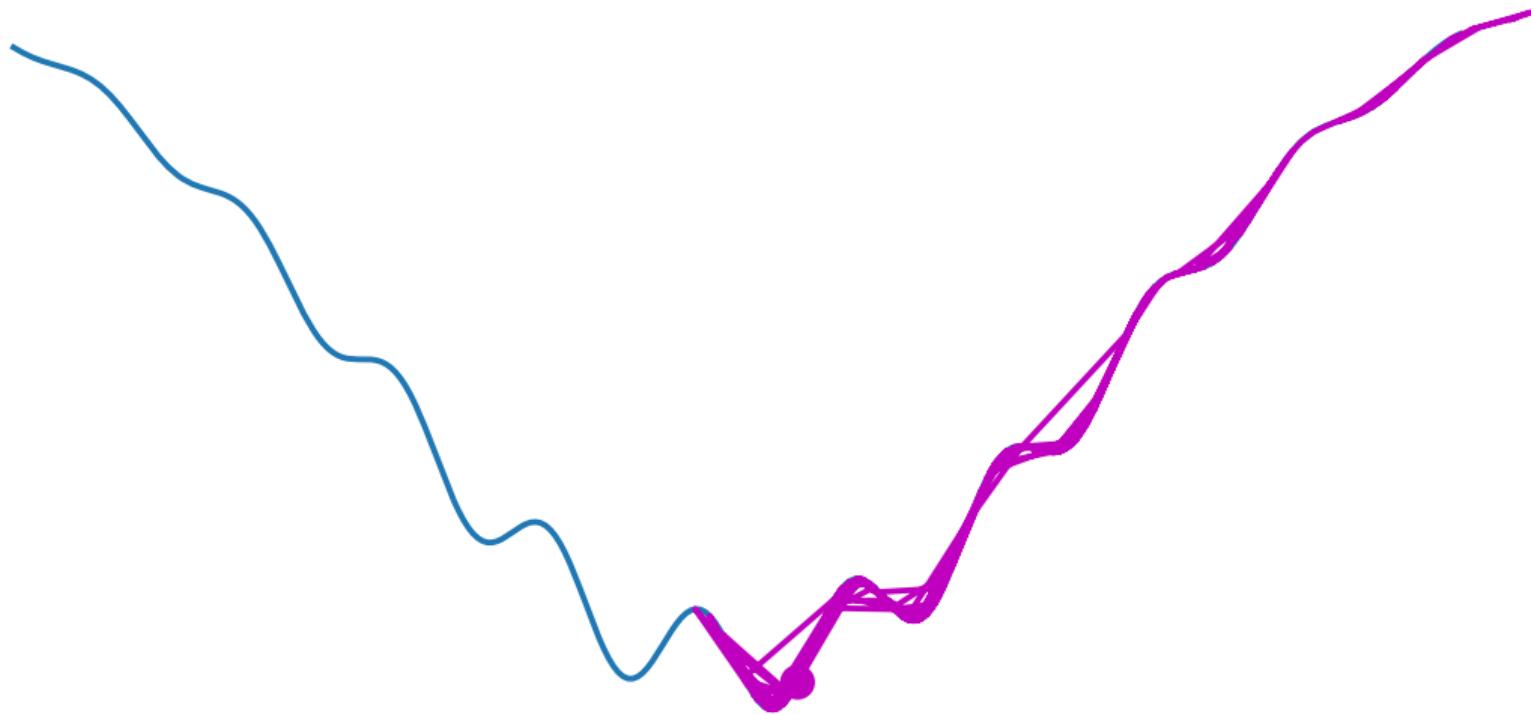
Градиентный спуск сходится к локальному минимуму



# Градиентный спуск сходится к локальному минимуму

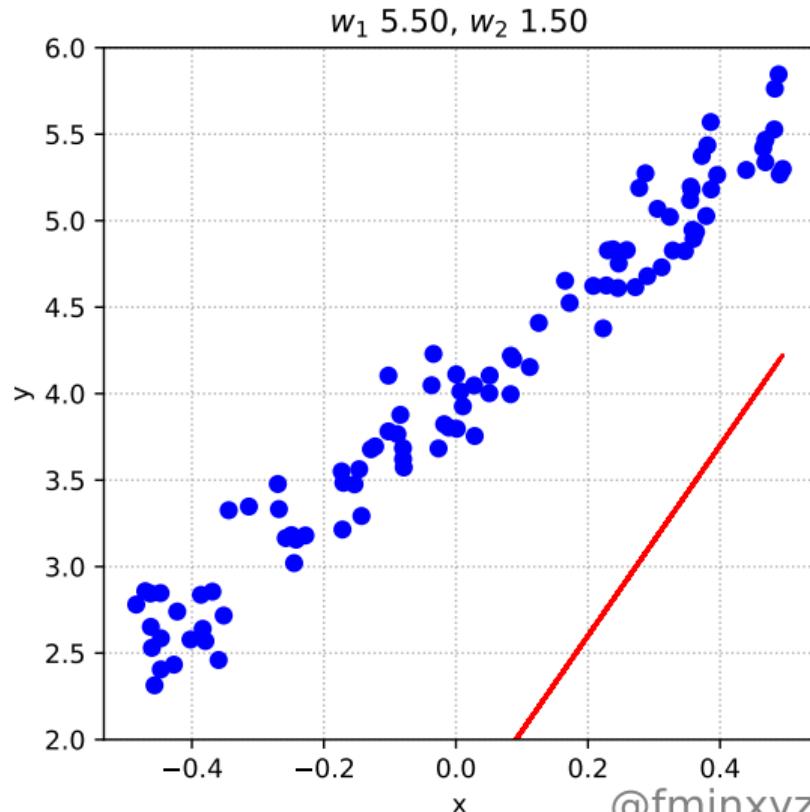
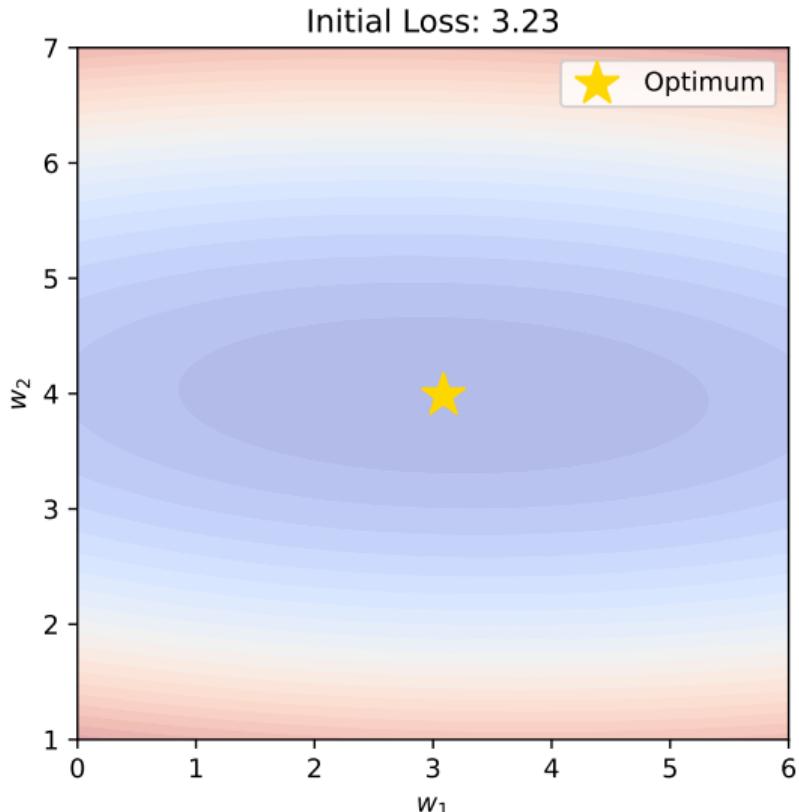


Стохастический градиентный спуск  
выпрыгивает из локальных минимумов



## SGD не сходится с постоянным шагом для выпуклой функции

Stochastic Gradient Descent. Batch = 1



@fminxyz

## Основные результаты сходимости SGD

- i** Пусть  $f$  -  $L$ -гладкая  $\mu$ -сильно выпуклая функция, а дисперсия стохастического градиента конечна ( $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ ). Тогда траектория стохастического градиентного спуска с постоянным шагом  $\alpha < \frac{1}{2\mu}$  будет гарантировать:

$$\mathbb{E}[f(x_{k+1}) - f^*] \leq (1 - 2\alpha\mu)^k [f(x_0) - f^*] + \frac{L\sigma^2\alpha}{4\mu}.$$

## Основные результаты сходимости SGD

- Пусть  $f$  -  $L$ -гладкая  $\mu$ -сильно выпуклая функция, а дисперсия стохастического градиента конечна ( $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ ). Тогда траектория стохастического градиентного спуска с постоянным шагом  $\alpha < \frac{1}{2\mu}$  будет гарантировать:

$$\mathbb{E}[f(x_{k+1}) - f^*] \leq (1 - 2\alpha\mu)^k [f(x_0) - f^*] + \frac{L\sigma^2\alpha}{4\mu}.$$

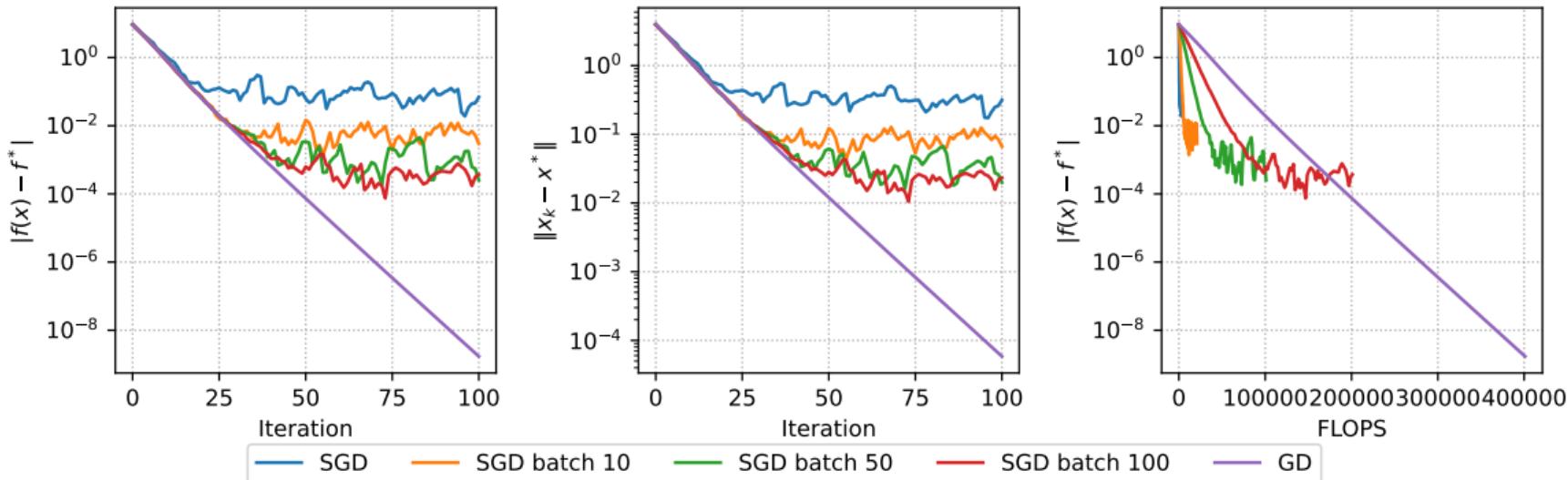
- Пусть  $f$  -  $L$ -гладкая  $\mu$ -сильно выпуклая функция, а дисперсия стохастического градиента конечна ( $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ ). Тогда стохастический градиентный шум с уменьшающимся шагом  $\alpha_k = \frac{2k+1}{2\mu(k+1)^2}$  будет сходиться сублинейно:

$$\mathbb{E}[f(x_{k+1}) - f^*] \leq \frac{L\sigma^2}{2\mu^2(k+1)}$$

## Сходимость в зависимости от размера батча

$$f(x) = \frac{\mu}{2} \|x\|_2^2 + \frac{1}{m} \sum_{i=1}^m \log(1 + \exp(-y_i \langle a_i, x \rangle)) \rightarrow \min_{x \in \mathbb{R}^n}$$

Strongly convex binary logistic regression. m=200, n=10, mu=1.



Эта задача оптимизации даже сложнее, чем кажется

## Улучшаем SGD - адаптивные методы (Adam) <sup>1 2</sup>

- Одна из самых цитируемых научных работ в мире

$$m_j^{(k)} = \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)}$$

$$v_j^{(k)} = \beta_2 v_j^{(k-1)} + (1 - \beta_2) (g_j^{(k)})^2$$

$$\hat{m}_j = \frac{m_j^{(k)}}{1 - \beta_1^k}, \quad \hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k}$$

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon}$$

## Улучшаем SGD - адаптивные методы (Adam) <sup>1 2</sup>

- Одна из самых цитируемых научных работ в мире
- В 2018-2019 годах вышли статьи, указывающие на ошибку в оригинальной статье

$$m_j^{(k)} = \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)}$$

$$v_j^{(k)} = \beta_2 v_j^{(k-1)} + (1 - \beta_2) (g_j^{(k)})^2$$

$$\hat{m}_j = \frac{m_j^{(k)}}{1 - \beta_1^k}, \quad \hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k}$$

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon}$$

## Улучшаем SGD - адаптивные методы (Adam) <sup>1 2</sup>

- Одна из самых цитируемых научных работ в мире
- В 2018-2019 годах вышли статьи, указывающие на ошибку в оригинальной статье
- Не сходится для некоторых простых задач (даже выпуклых)

$$m_j^{(k)} = \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)}$$

$$v_j^{(k)} = \beta_2 v_j^{(k-1)} + (1 - \beta_2) (g_j^{(k)})^2$$

$$\hat{m}_j = \frac{m_j^{(k)}}{1 - \beta_1^k}, \quad \hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k}$$

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon}$$

## Улучшаем SGD - адаптивные методы (Adam) <sup>1 2</sup>

- Одна из самых цитируемых научных работ в мире
- В 2018-2019 годах вышли статьи, указывающие на ошибку в оригинальной статье
- Не сходится для некоторых простых задач (даже выпуклых)
- Почему-то очень хорошо работает для некоторых сложных задач

$$m_j^{(k)} = \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)}$$

$$v_j^{(k)} = \beta_2 v_j^{(k-1)} + (1 - \beta_2) (g_j^{(k)})^2$$

$$\hat{m}_j = \frac{m_j^{(k)}}{1 - \beta_1^k}, \quad \hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k}$$

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon}$$

# Улучшаем SGD - адаптивные методы (Adam) <sup>1 2</sup>

- Одна из самых цитируемых научных работ в мире
- В 2018-2019 годах вышли статьи, указывающие на ошибку в оригинальной статье
- Не сходится для некоторых простых задач (даже выпуклых)
- Почему-то очень хорошо работает для некоторых сложных задач
- Гораздо лучше работает для языковых моделей, чем для задач компьютерного зрения - почему?

$$m_j^{(k)} = \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)}$$

$$v_j^{(k)} = \beta_2 v_j^{(k-1)} + (1 - \beta_2)(g_j^{(k)})^2$$

$$\hat{m}_j = \frac{m_j^{(k)}}{1 - \beta_1^k}, \quad \hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k}$$

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon}$$

<sup>1</sup>Adam: A Method for Stochastic Optimization

<sup>2</sup>On the Convergence of Adam and Beyond

## Adam работает хуже для CV, чем для LLM? <sup>3</sup>

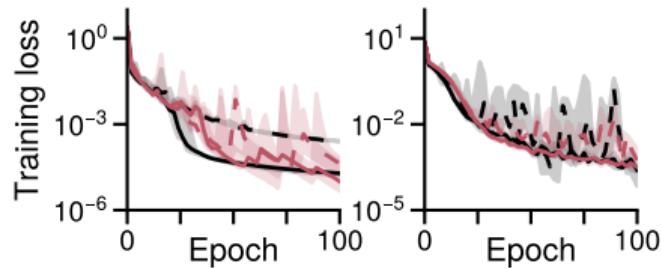


Рис. 1: CNNs on MNIST and CIFAR10

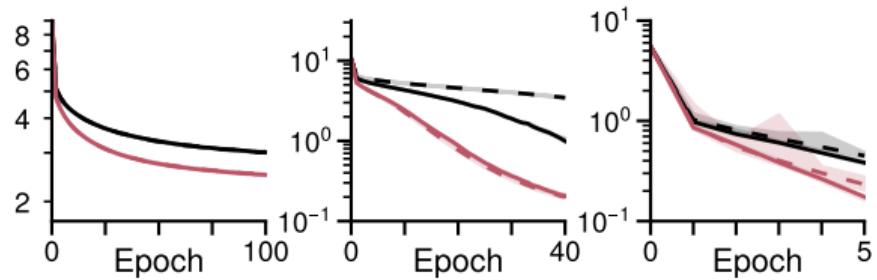
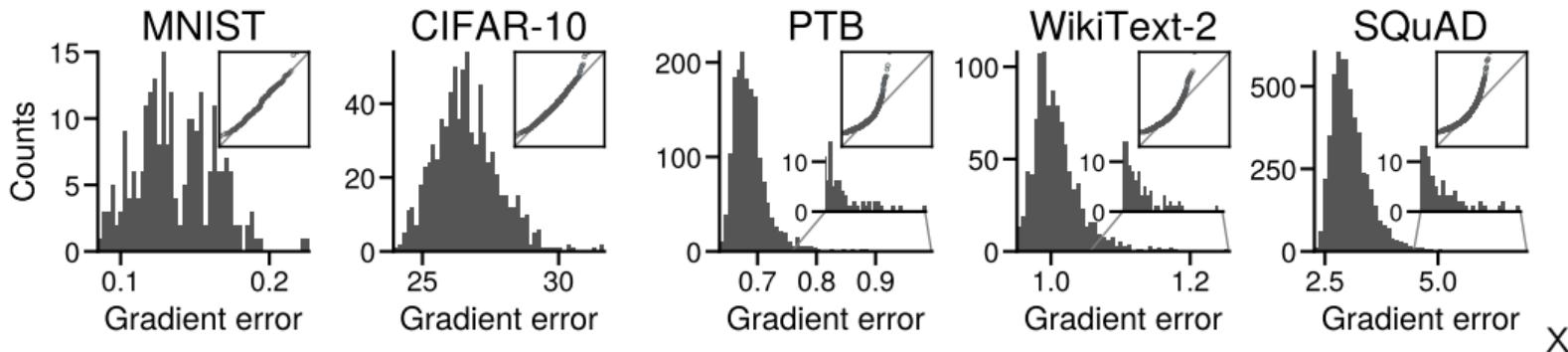


Рис. 2: Transformers on PTB, WikiText2, and SQuAD

<sup>3</sup>Linear attention is (maybe) all you need (to understand transformer optimization)

# Почему Adam работает хуже для CV, чем для LLM? <sup>4</sup>

Потому что шум градиентов в языковых моделях имеет тяжелые хвосты?



<sup>4</sup>Linear attention is (maybe) all you need (to understand transformer optimization)

## Почему Adam работает хуже для CV, чем для LLM? <sup>5</sup>

Нет! Метки имеют тяжелые хвосты!

В компьютерном зрении датасеты часто сбалансированы: 1000 котиков, 1000 песелей и т.д.  
В языковых датасетах почти всегда не так: слово the встречается часто, слово tie на порядки реже

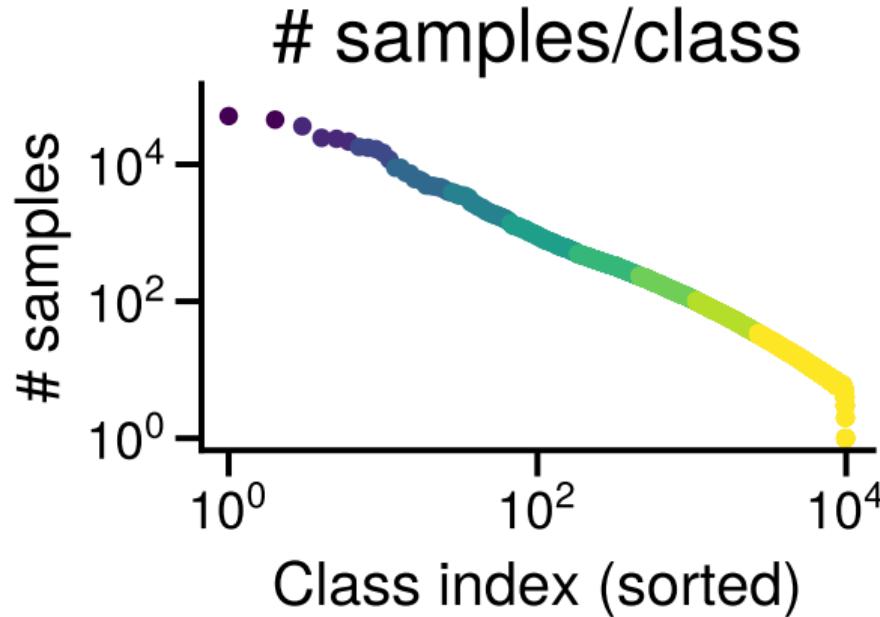
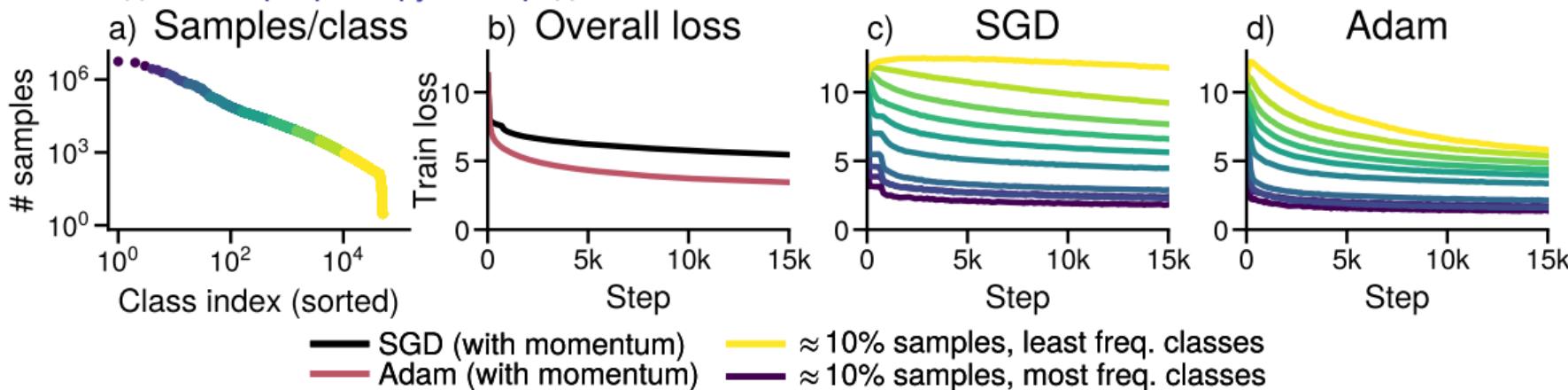


Рис. 3: Распределение частоты токенов в PTB

<sup>5</sup>Heavy-Tailed Class Imbalance and Why Adam Outperforms Gradient Descent on Language Models

# Почему Adam работает хуже для CV, чем для LLM? <sup>6</sup>

SGD медленно прогрессирует на редких классах

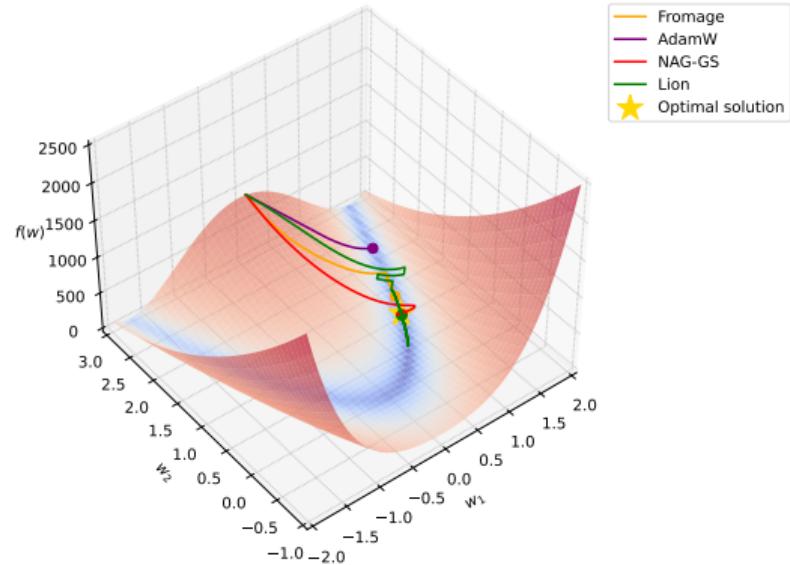


SGD не добивается прогресса на низкочастотных классах, в то время как Adam добивается. Обучение GPT-2 S на WikiText-103. (a) Распределение классов, отсортированных по частоте встречаемости, разбитых на группы, соответствующие  $\approx 10\%$  данных. (b) Значение функции потерь при обучении. (c, d) Значение функции потерь при обучении для каждой группы при использовании SGD и Adam.

<sup>6</sup>Heavy-Tailed Class Imbalance and Why Adam Outperforms Gradient Descent on Language Models

- Требует хранения одного дополнительного вектора, вместо двух, как в Adam.

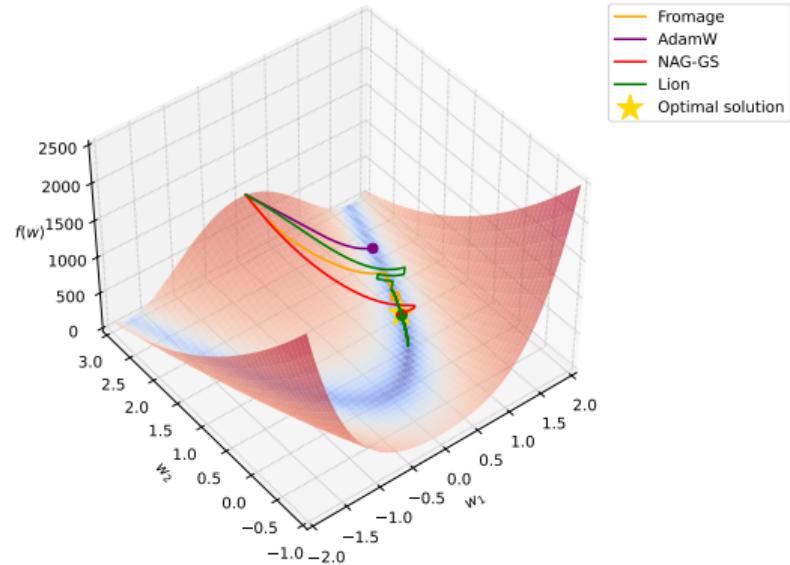
Rosenbrock Function.  
Adaptive stochastic gradient algorithms.  
Learning rate 0.003



# NAG-GS<sup>7</sup>

- Требует хранения одного дополнительного вектора, вместо двух, как в Adam.
- Качество в ряде задач сопоставимо с AdamW

Rosenbrock Function.  
Adaptive stochastic gradient algorithms.  
Learning rate 0.003



<sup>7</sup>NAG-GS: Semi-Implicit, Accelerated and Robust Stochastic Optimizer

## Визуализация с помощью проекции на прямую

- Обозначим начальную точку как  $w_0$ , представляющую собой веса нейронной сети при инициализации. Веса, полученные после обучения, обозначим как  $\hat{w}$ .

$$L(\alpha) = L(w_0 + \alpha w_1), \text{ where } \alpha \in [-b, b].$$

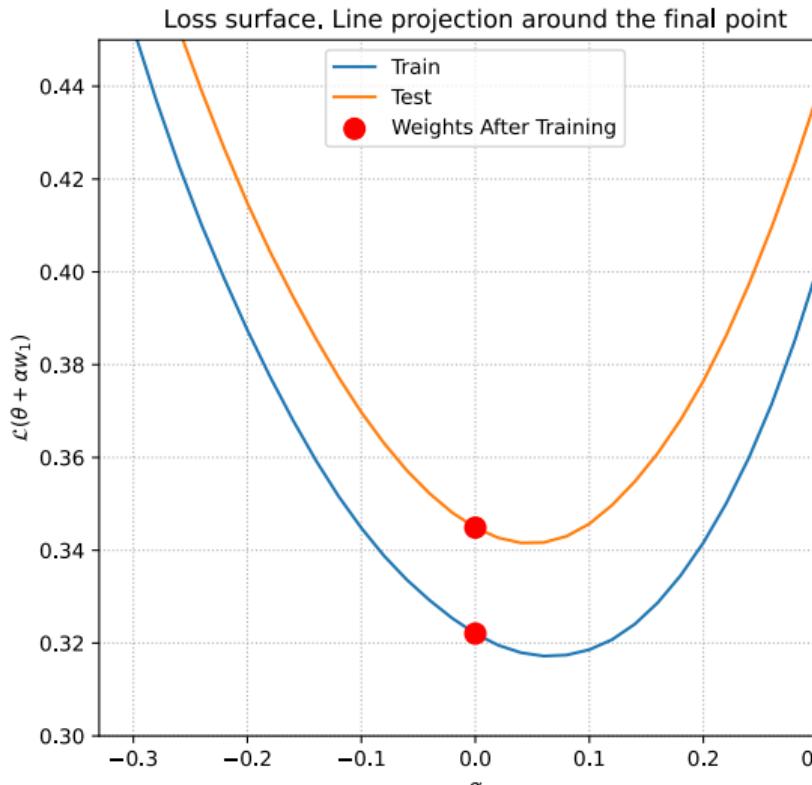
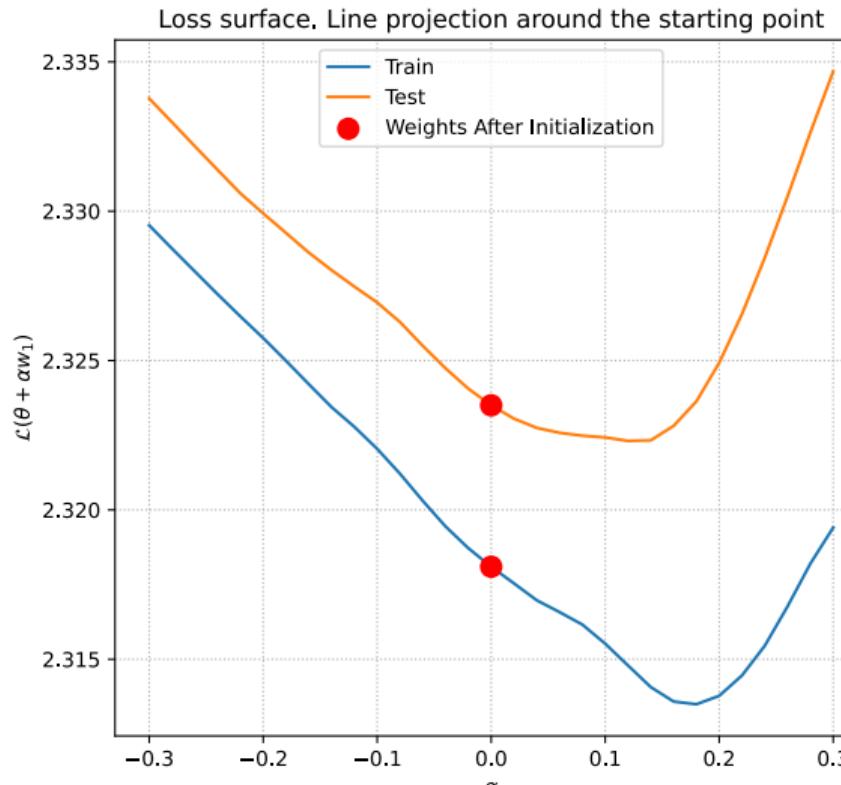
## Визуализация с помощью проекции на прямую

- Обозначим начальную точку как  $w_0$ , представляющую собой веса нейронной сети при инициализации. Веса, полученные после обучения, обозначим как  $\hat{w}$ .
- Генерируем случайный вектор такой же размерности и нормы  $w_1 \in \mathbb{R}^p$ , затем вычисляем значение функции потерь вдоль этого вектора:

$$L(\alpha) = L(w_0 + \alpha w_1), \text{ where } \alpha \in [-b, b].$$

# Проекция функции потерь нейронной сети на прямую

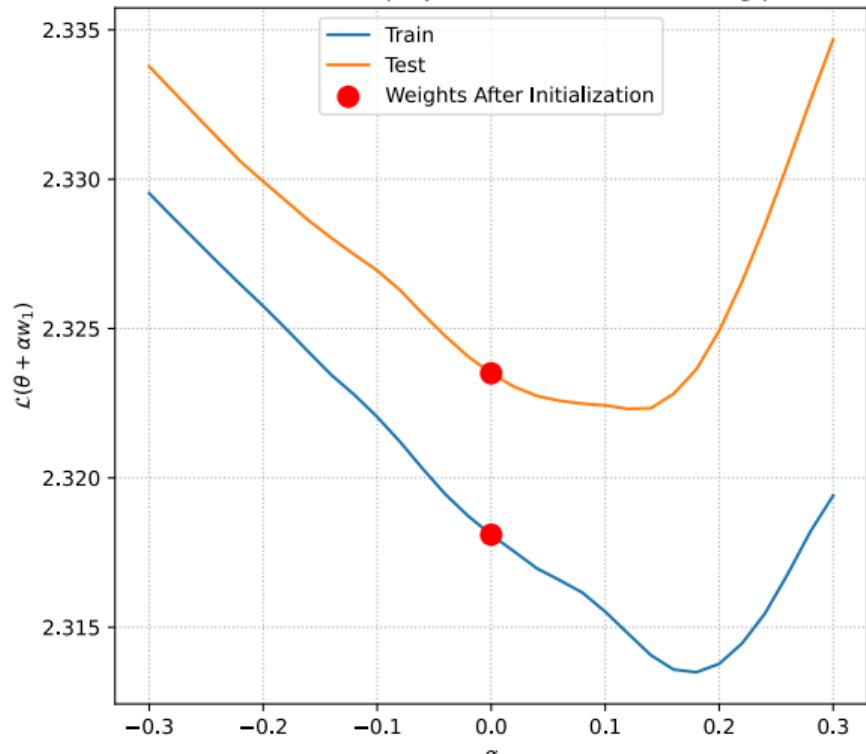
No Dropout



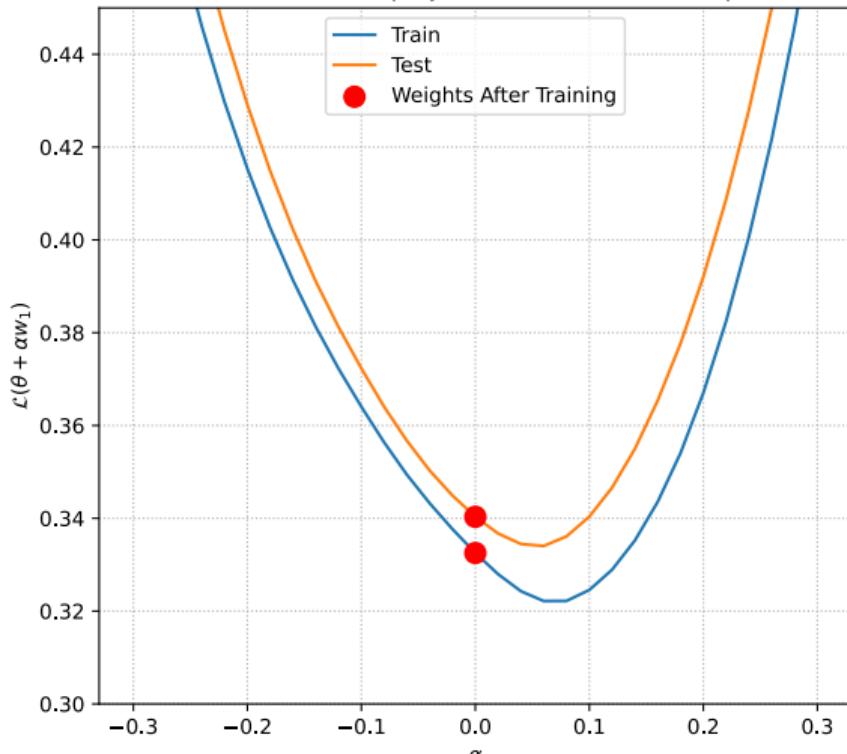
# Проекция функции потерь нейронной сети на прямую

Dropout 0.2

Loss surface, Line projection around the starting point



Loss surface, Line projection around the final point

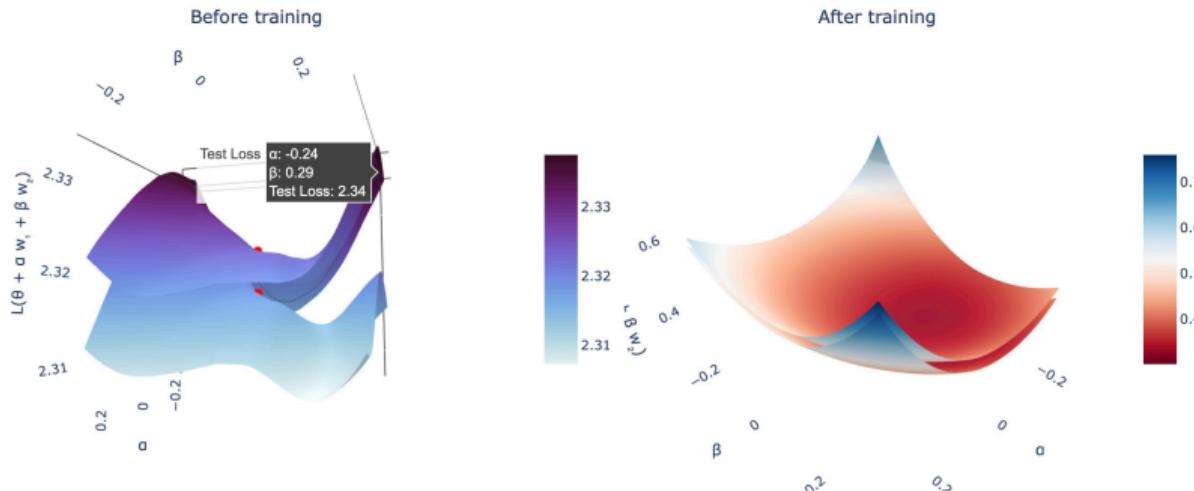


# Проекция функции потерь нейронной сети на плоскость

- Мы можем расширить эту идею и построить проекцию поверхности потерь на плоскость, которая задается 2 случайными векторами.

$$L(\alpha, \beta) = L(w_0 + \alpha w_1 + \beta w_2), \text{ where } \alpha, \beta \in [-b, b]^2.$$

No Dropout. Plane projection of loss surface.

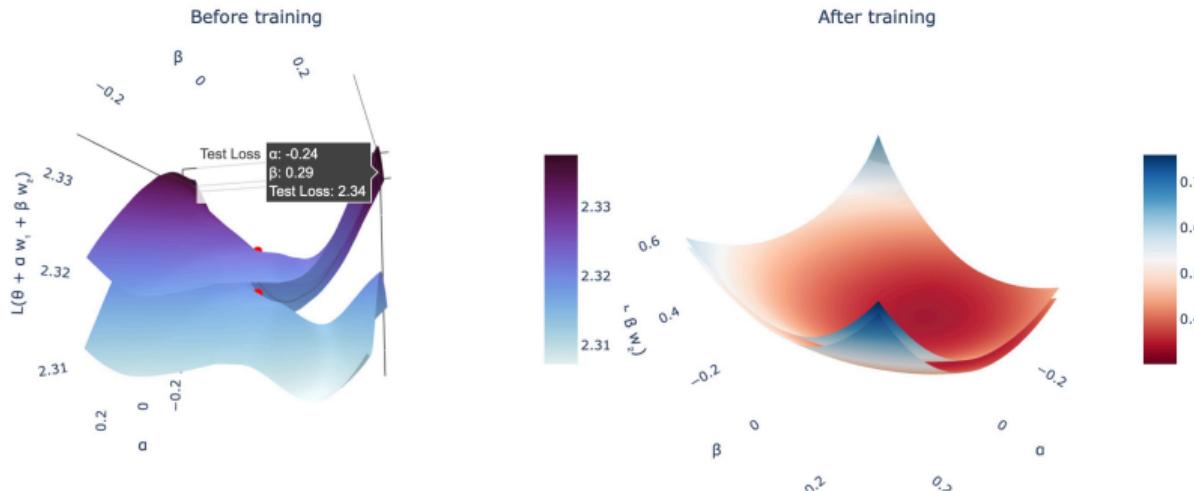


## Проекция функции потерь нейронной сети на плоскость

- Мы можем расширить эту идею и построить проекцию поверхности потерь на плоскость, которая задается 2 случайными векторами.
- Два случайных гауссовых вектора в пространстве большой размерности с высокой вероятностью ортогональны.

$$L(\alpha, \beta) = L(w_0 + \alpha w_1 + \beta w_2), \text{ where } \alpha, \beta \in [-b, b]^2.$$

No Dropout. Plane projection of loss surface.



Может ли быть полезно изучение таких проекций? <sup>8</sup>

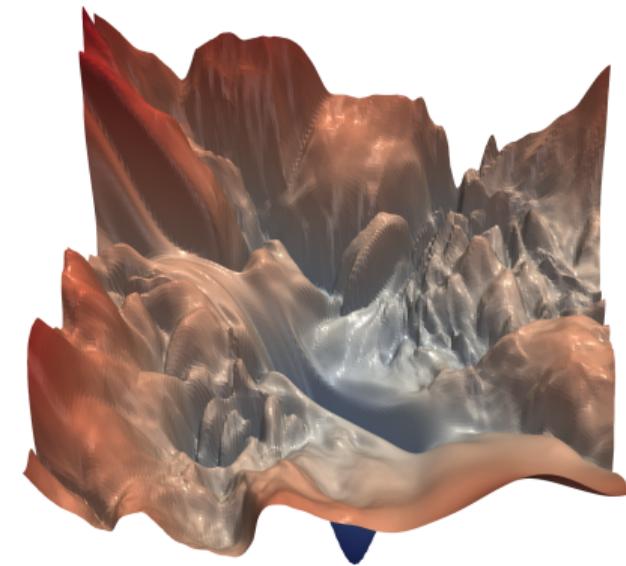


Рис. 7: The loss surface of ResNet-56 without  
skip connections

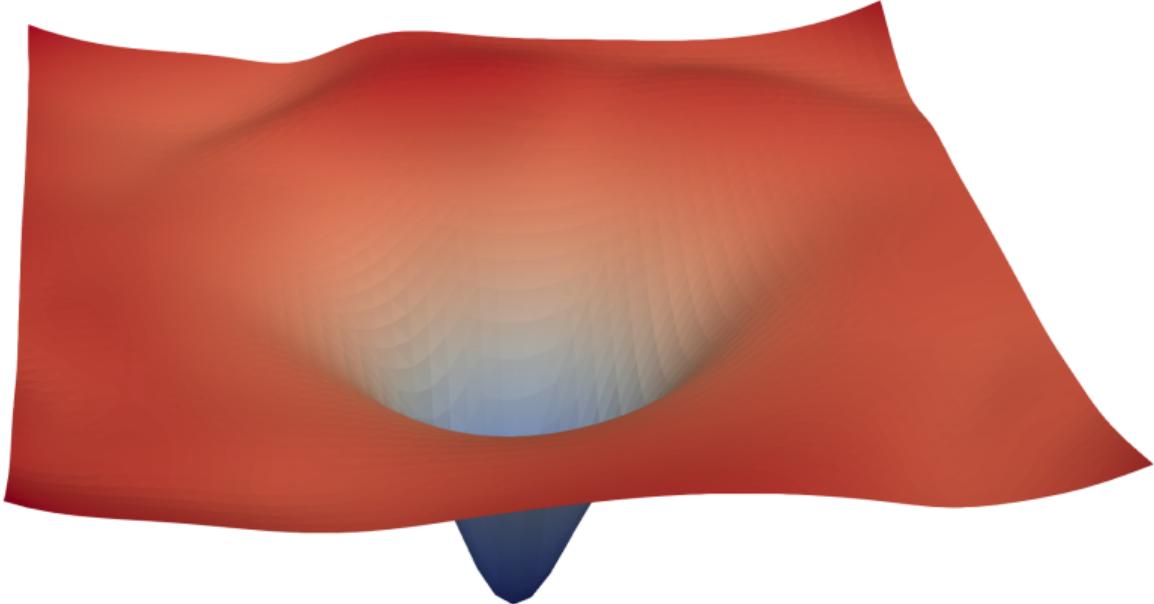


Рис. 8: The loss surface of ResNet-56 with skip connections

<sup>8</sup>Visualizing the Loss Landscape of Neural Nets, Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein

# Может ли быть полезно изучение таких проекций, если серьезно? <sup>9</sup>

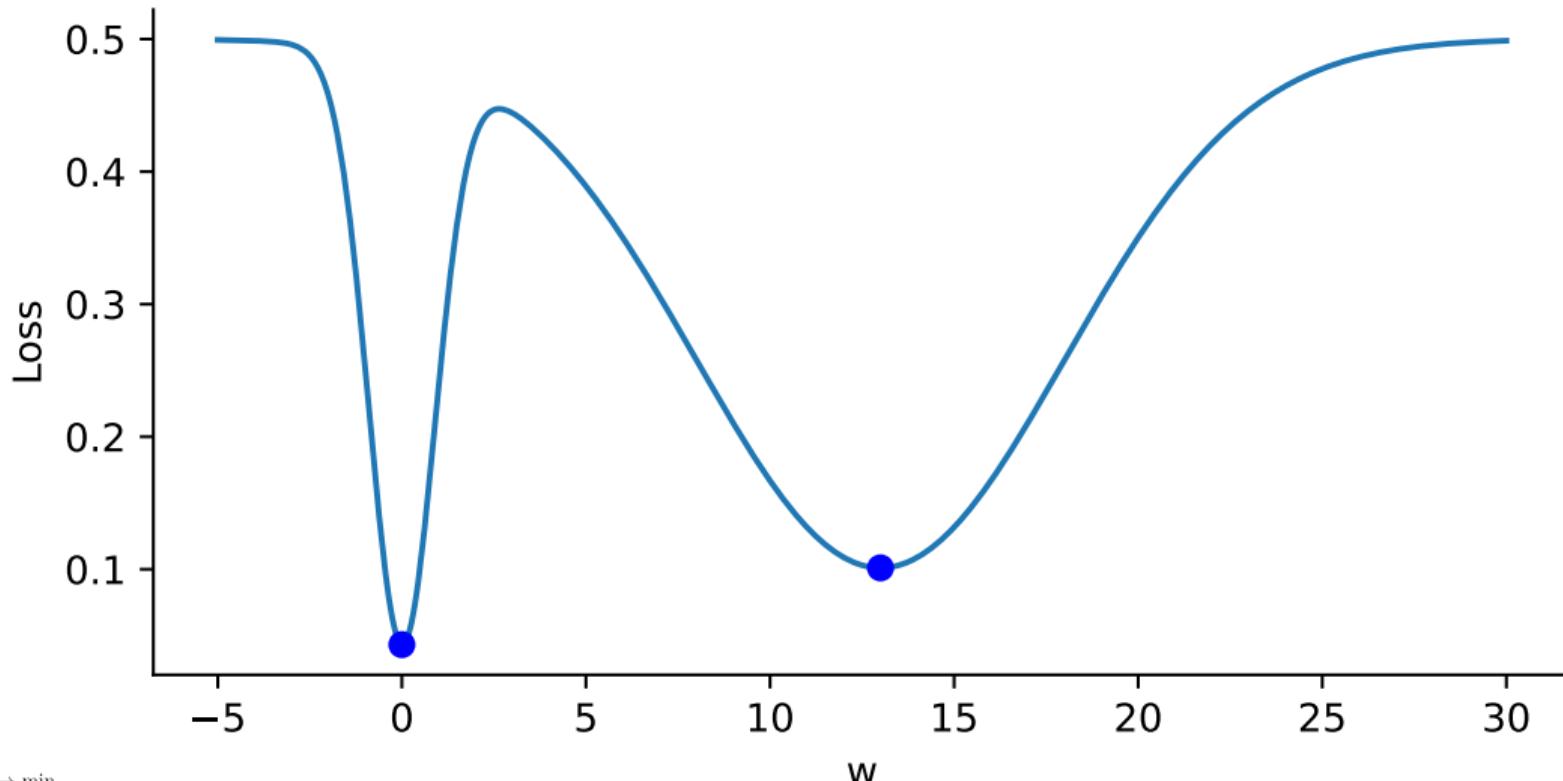


Рис. 9: Examples of a loss landscape of a typical CNN model on FashionMNIST and CIFAR10 datasets found with MPO. Loss values are color-coded according to a logarithmic scale

<sup>9</sup>Loss Landscape Sightseeing with Multi-Point Optimization, Ivan Skorokhodov, Mikhail Burtsev

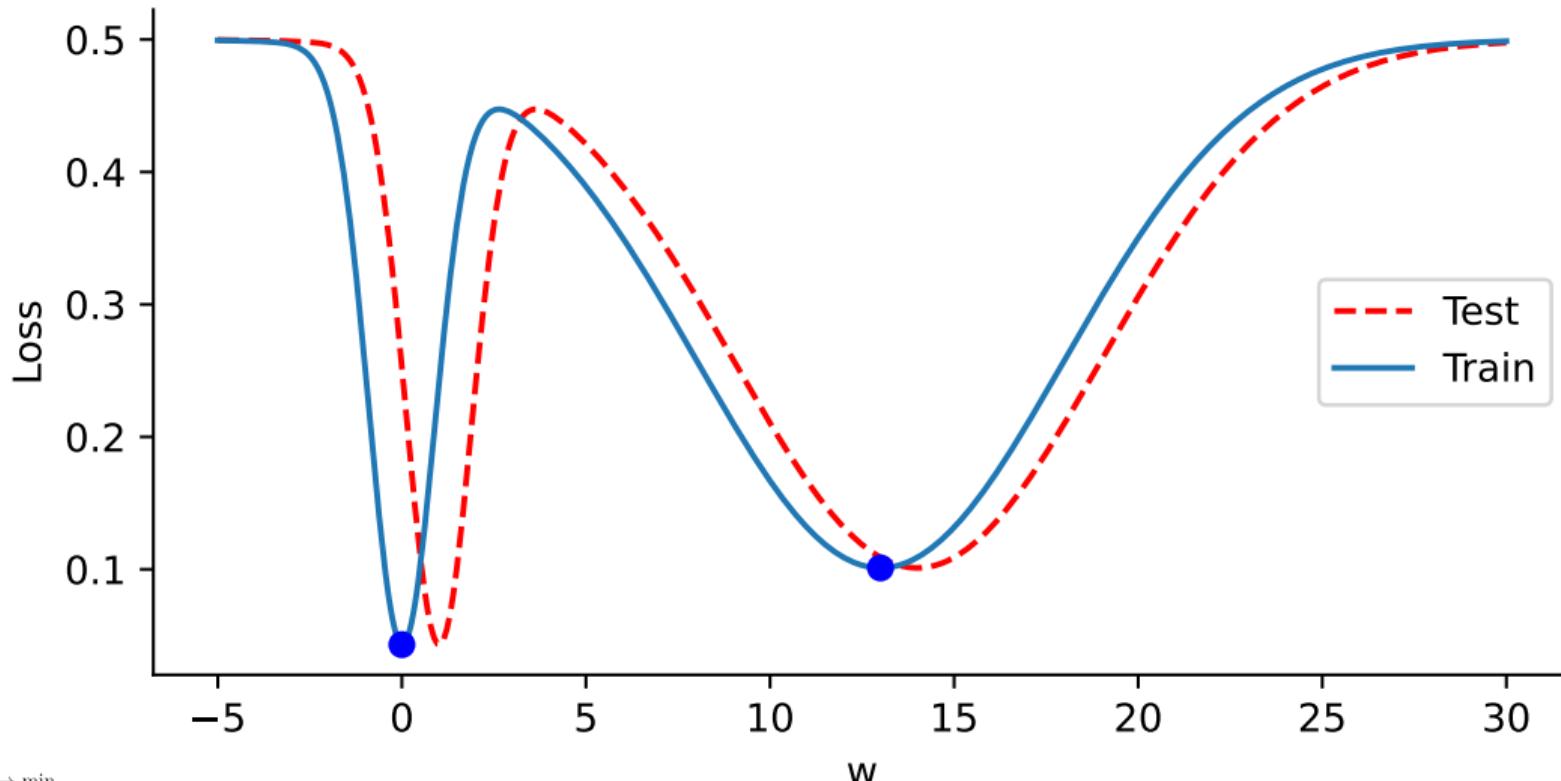
## Ширина локальных минимумов

Узкие и широкие локальные минимумы



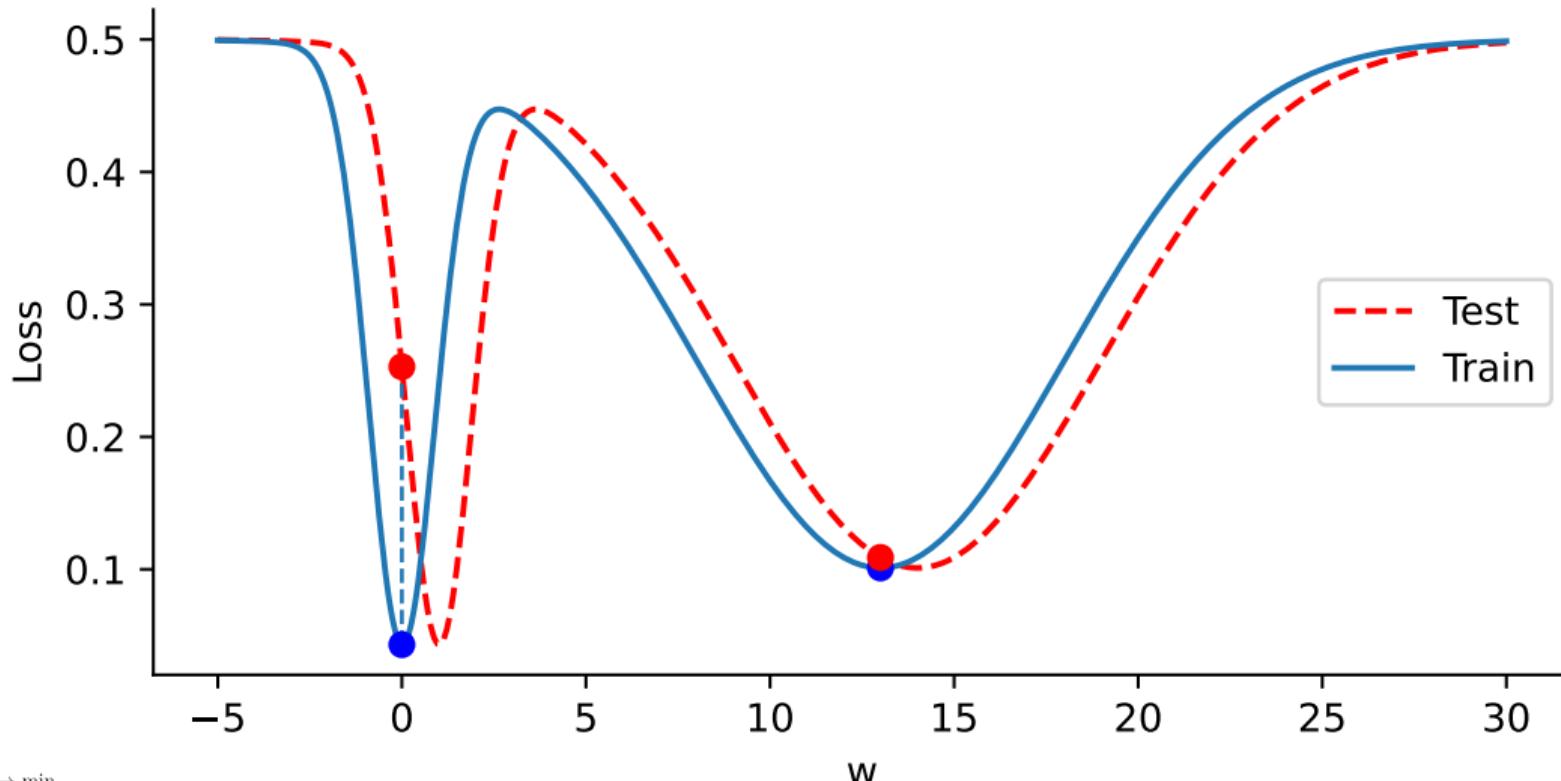
## Ширина локальных минимумов

### Узкие и широкие локальные минимумы

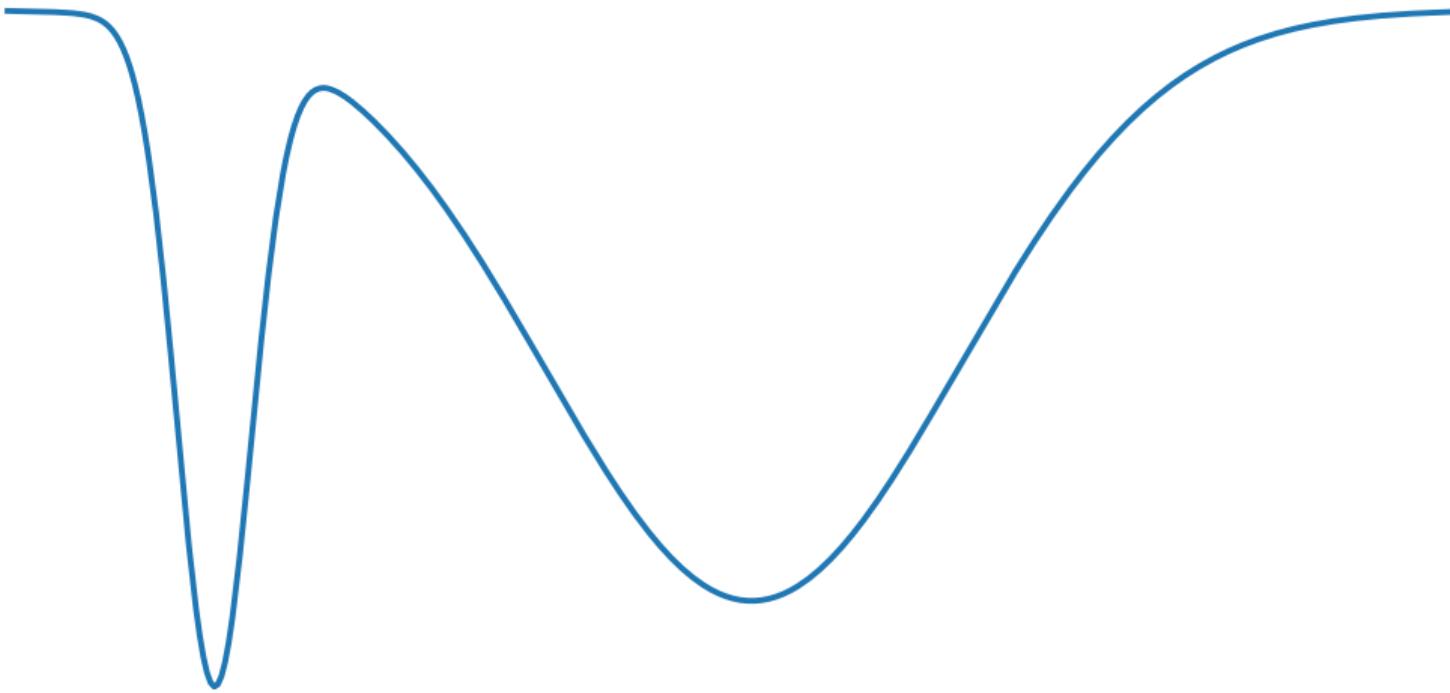


## Ширина локальных минимумов

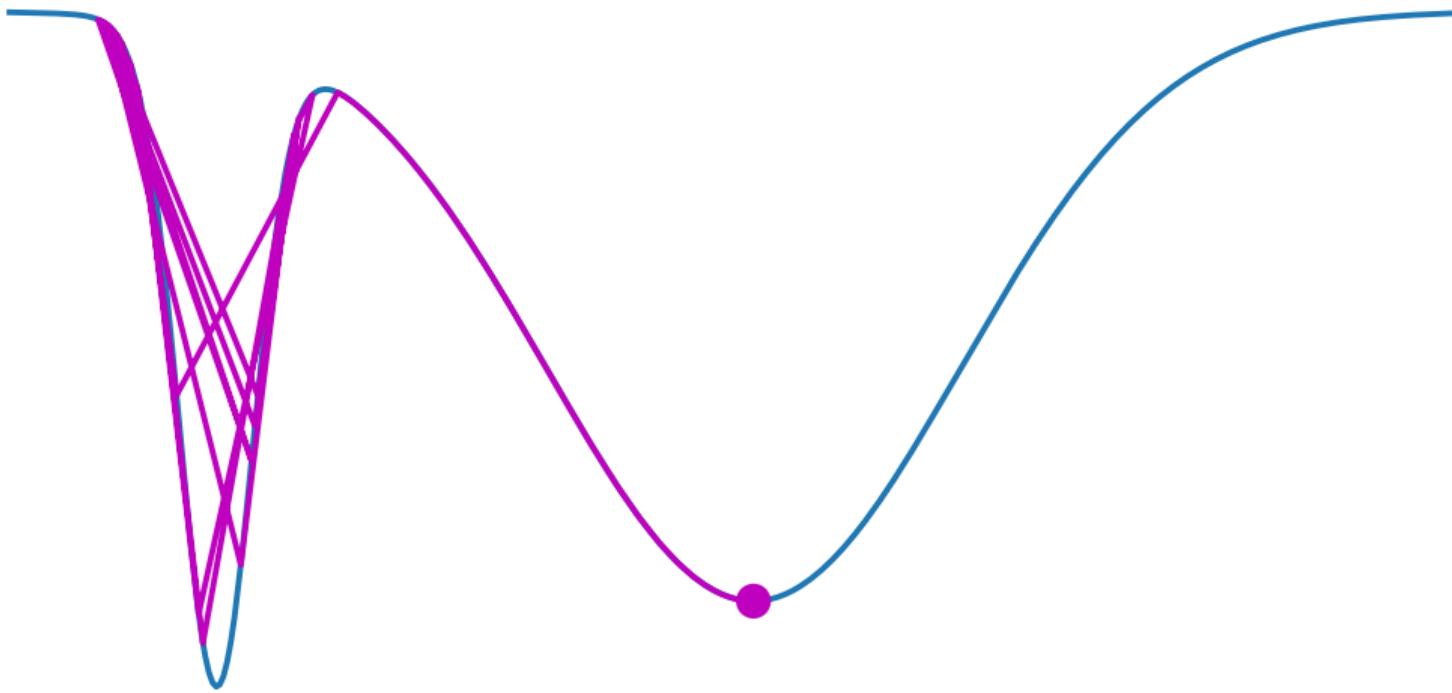
### Узкие и широкие локальные минимумы



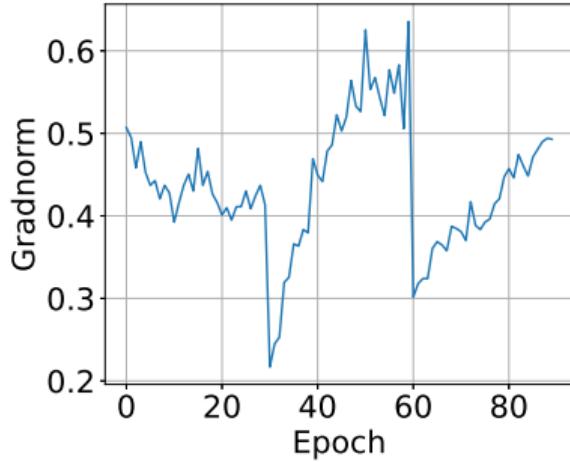
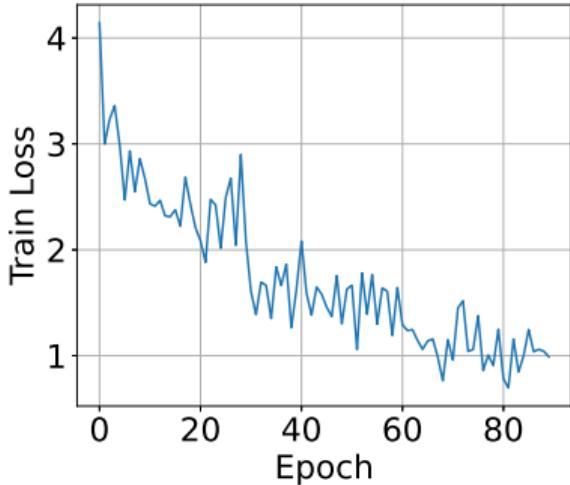
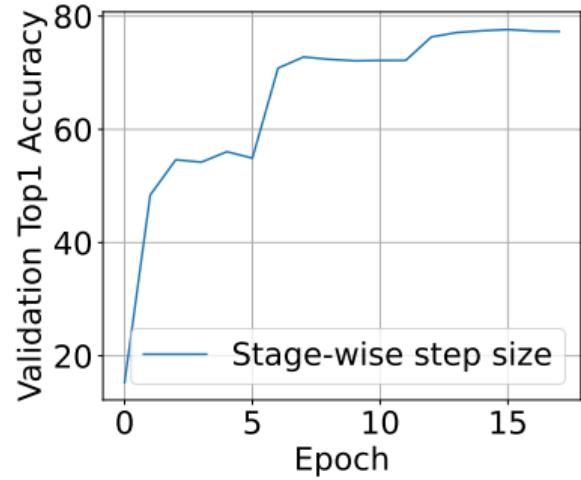
Градиентный спуск с маленьким шагом  
сходится в узкий локальный минимум



Градиентный спуск с большим шагом  
избегает узкого локального минимума



## Модели не сходятся к стационарным точкам, но это не страшно<sup>10</sup>



<sup>10</sup>NN Weights Do Not Converge to Stationary Points

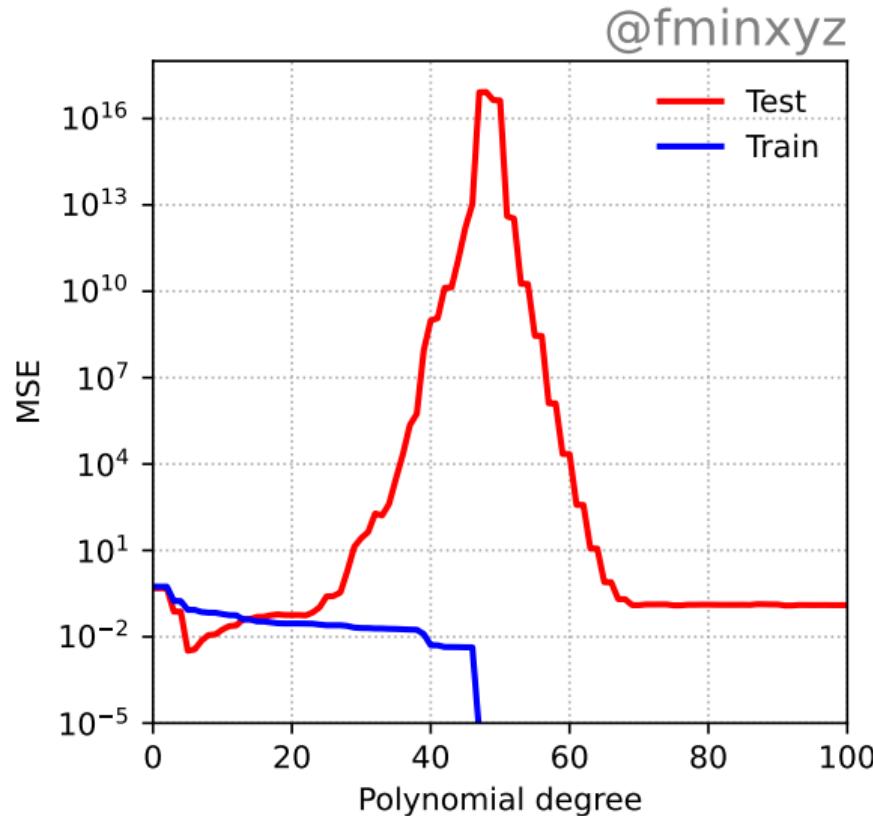
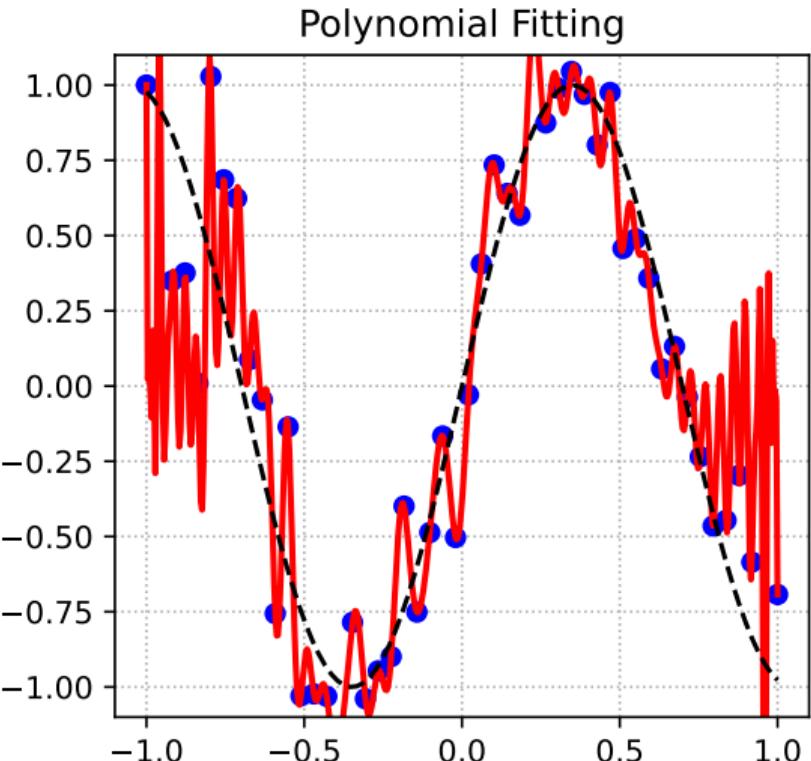
$f \rightarrow \min_{x,y,z}$  Эта задача оптимизации даже сложнее, чем кажется



Рис. 10: Training transformer with 2 layers, width 128, and 4 attention heads, with a total of about  $4 \cdot 10^5$  non-embedding parameters. Reproduction of experiments (~ half an hour) is available [here](#)

<sup>11</sup>Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets, Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, Vedant Misra

## Double Descent <sup>12</sup>



<sup>12</sup>Reconciling modern machine learning practice and the bias-variance trade-off, Mikhail Belkin, Daniel Hsu, Siyuan Ma, Soumik Mandal  
 $f \rightarrow \min_{x,y,z} f$  Эта задача оптимизации даже сложнее, чем кажется

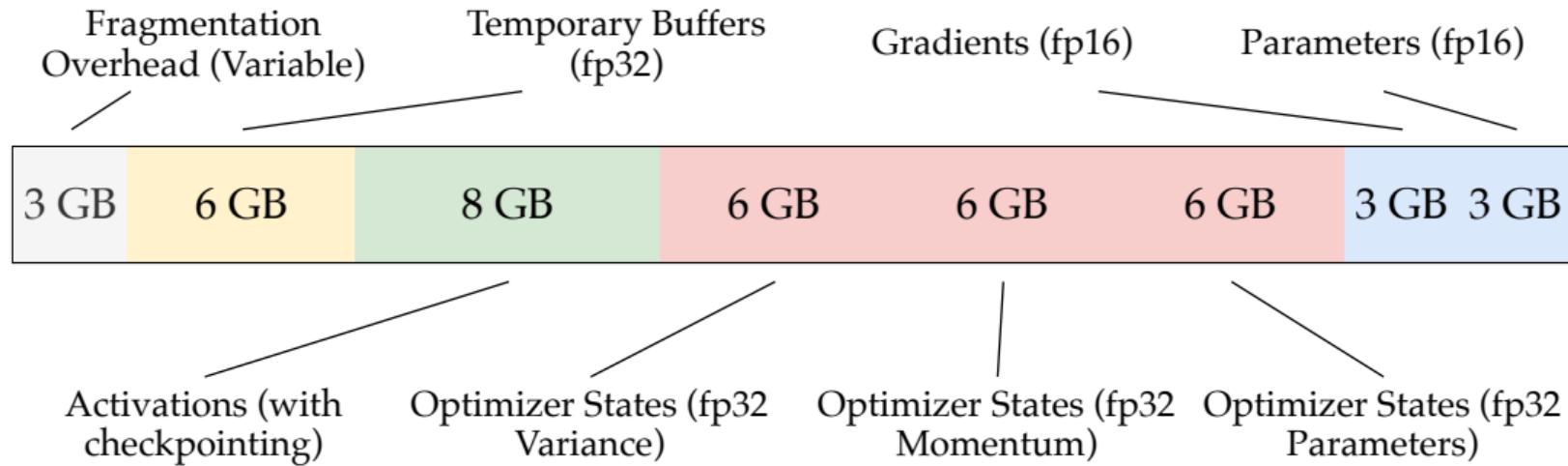
## Обучение больших моделей

## Потребление памяти при обучении GPT-2<sup>13</sup>



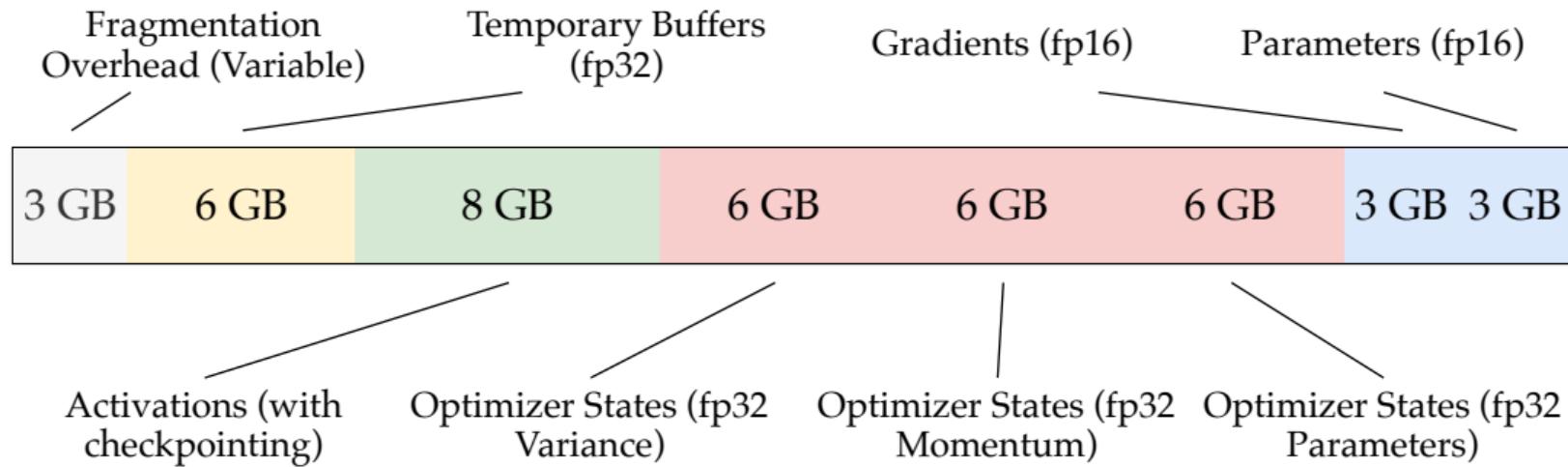
- Размер модели 1.5 В. Веса модели в fp16 занимают всего 3 GB, однако, для наивного обучения не хватит GPU даже на 32 GB

## Потребление памяти при обучении GPT-2<sup>13</sup>



- Размер модели 1.5 В. Веса модели в fp16 занимают всего 3 GB, однако, для наивного обучения не хватит GPU даже на 32 GB
- Для использования Adam в режиме mixed precision необходимо хранить 3 (!) копии модели в fp32.

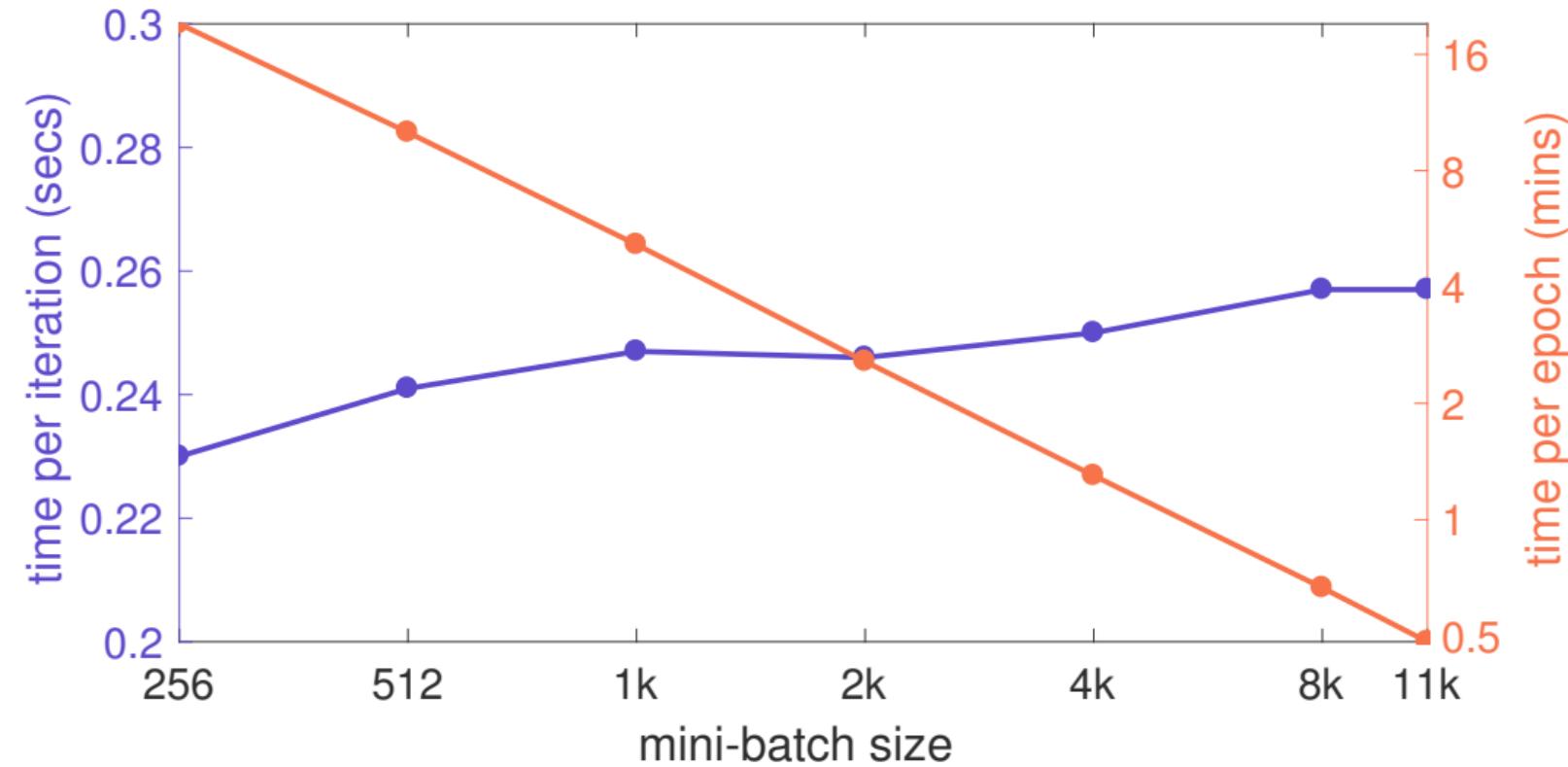
## Потребление памяти при обучении GPT-2<sup>13</sup>



- Размер модели 1.5 В. Веса модели в fp16 занимают всего 3 GB, однако, для наивного обучения не хватит GPU даже на 32 GB
- Для использования Adam в режиме mixed precision необходимо хранить 3 (!) копии модели в fp32.
- Активации в наивном режиме могут занимать гораздо больше памяти: для длины последовательности 1K и размера батча 32 нужно 60 GB для хранения всех промежуточных активаций. Чекпоинтинг активаций позволяет сократить потребление до 8 GB за счёт их перевычисления (33% computational overhead)

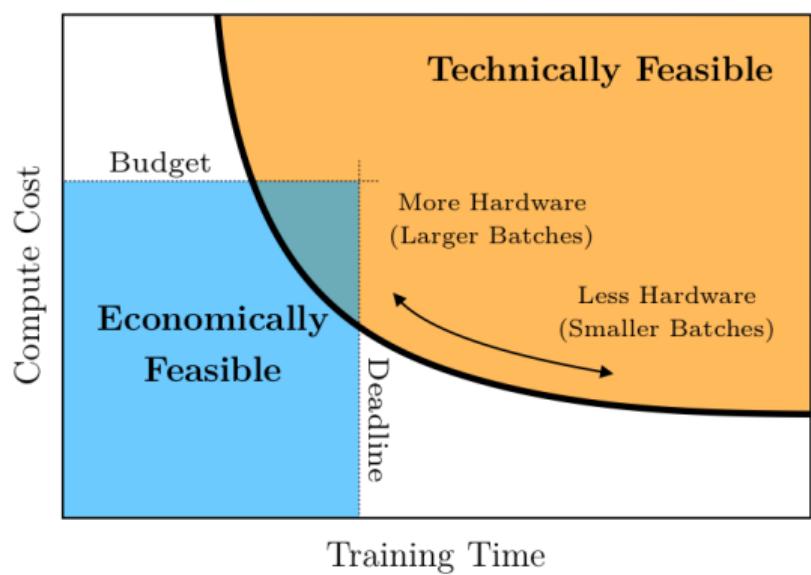
<sup>13</sup>ZeRO: Memory Optimizations Toward Training Trillion Parameter Models

## Large batch training <sup>14</sup>

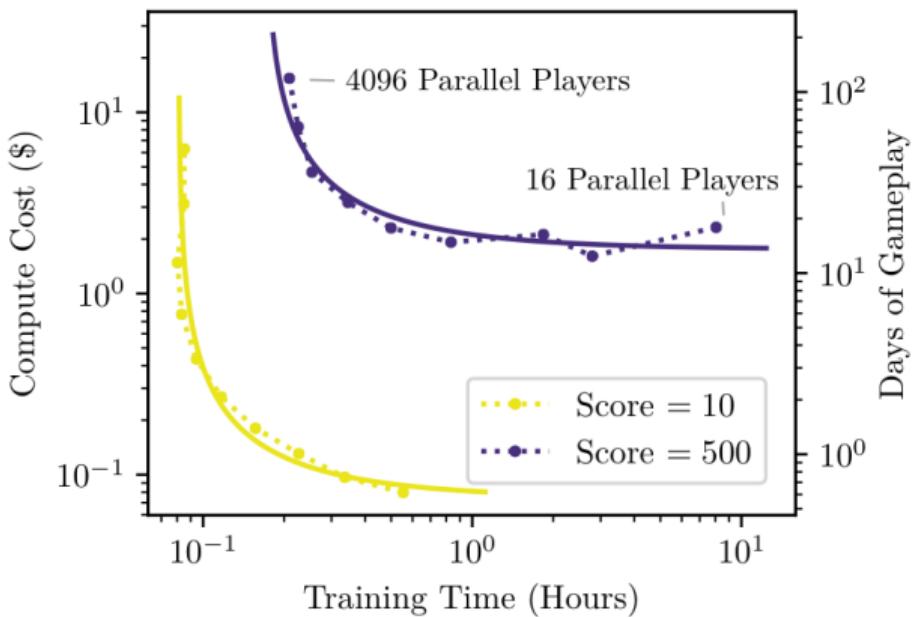


<sup>14</sup>Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour

## Large batch training<sup>15</sup>

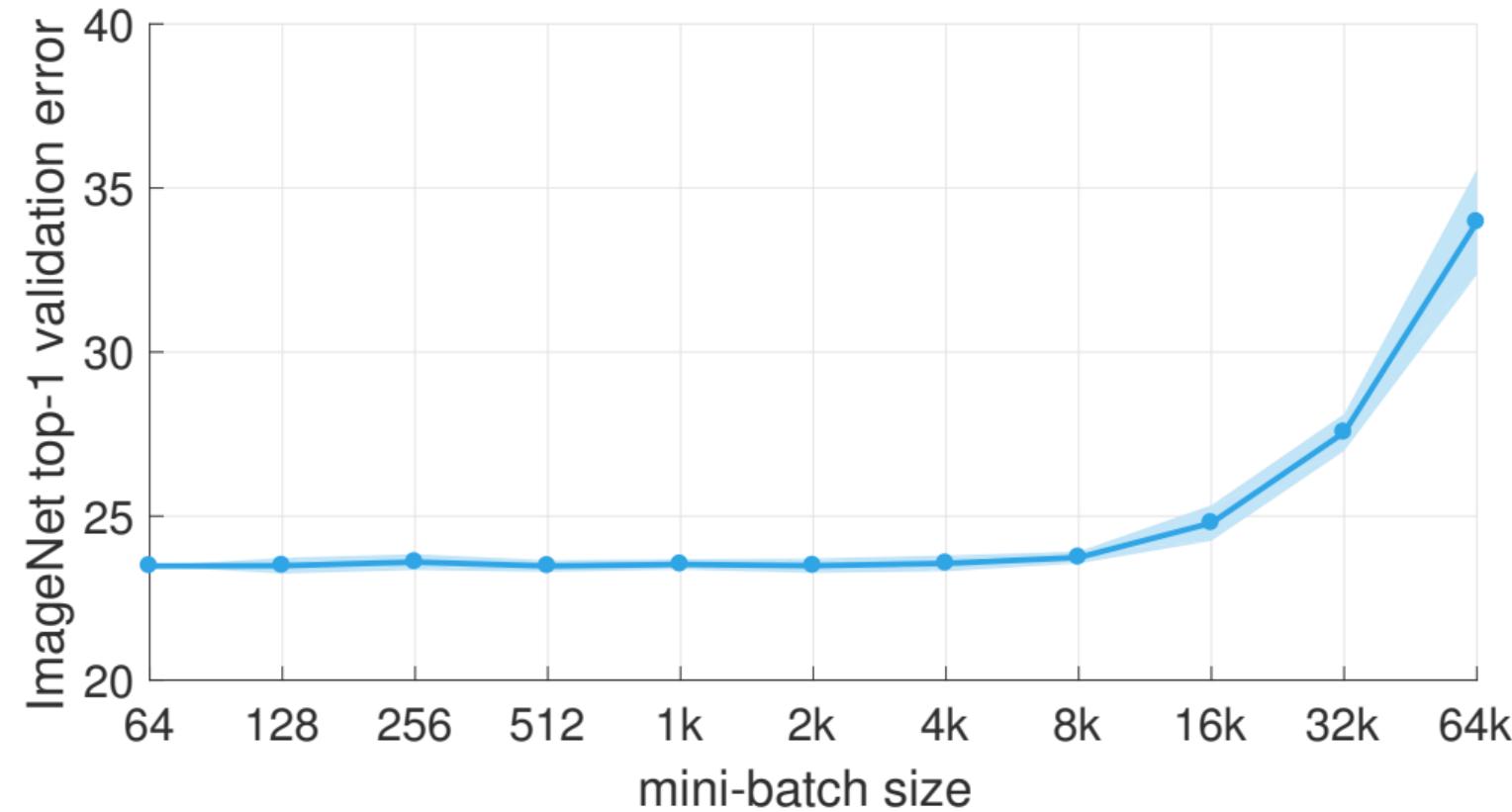


Atari Breakout - Pareto Fronts



<sup>15</sup>An Empirical Model of Large-Batch Training

## Large batch training <sup>16</sup>



<sup>16</sup>Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour

## Linear and square root scaling rules

When training with large batches, the learning rate must be adjusted to maintain convergence speed and stability. The **linear scaling rule**<sup>17</sup> suggests multiplying the learning rate by the same factor as the increase in batch size:

$$\alpha_{\text{new}} = \alpha_{\text{base}} \cdot \frac{\text{Batch Size}_{\text{new}}}{\text{Batch Size}_{\text{base}}}$$

The **square root scaling rule**<sup>18</sup> proposes scaling the learning rate with the square root of the batch size increase:

$$\alpha_{\text{new}} = \alpha_{\text{base}} \cdot \sqrt{\frac{\text{Batch Size}_{\text{new}}}{\text{Batch Size}_{\text{base}}}}$$

Authors claimed, that it suits for adaptive optimizers like Adam, RMSProp and etc. while linear scaling rule serves well for SGD.

<sup>17</sup> Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour

<sup>18</sup> Learning Rates as a Function of Batch Size: A Random Matrix Theory Approach to Neural Network Training

## Gradual warmup<sup>19</sup>

Gradual warmup helps to avoid instability when starting with large learning rates by slowly increasing the learning rate from a small value to the target value over a few epochs. This is defined as:

$$\alpha_t = \alpha_{\max} \cdot \frac{t}{T_w}$$

where  $t$  is the current iteration and  $T_w$  is the warmup duration in iterations. In the original paper, authors used first 5 epochs for gradual warmup.

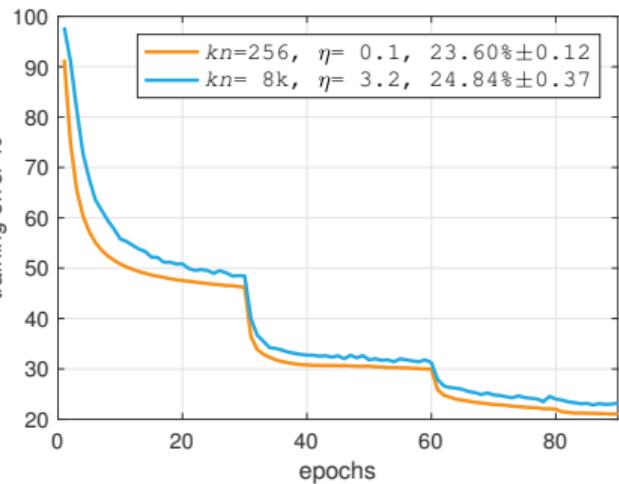


Рис. 11: no warmup

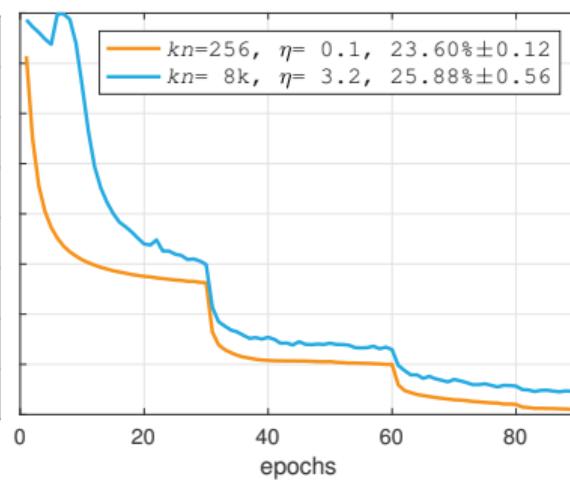


Рис. 12: constant warmup

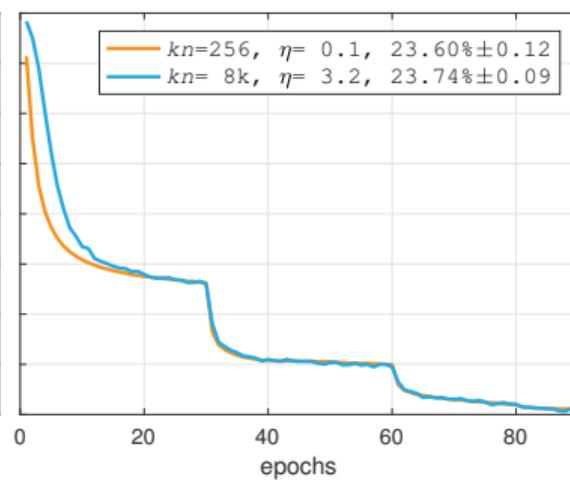
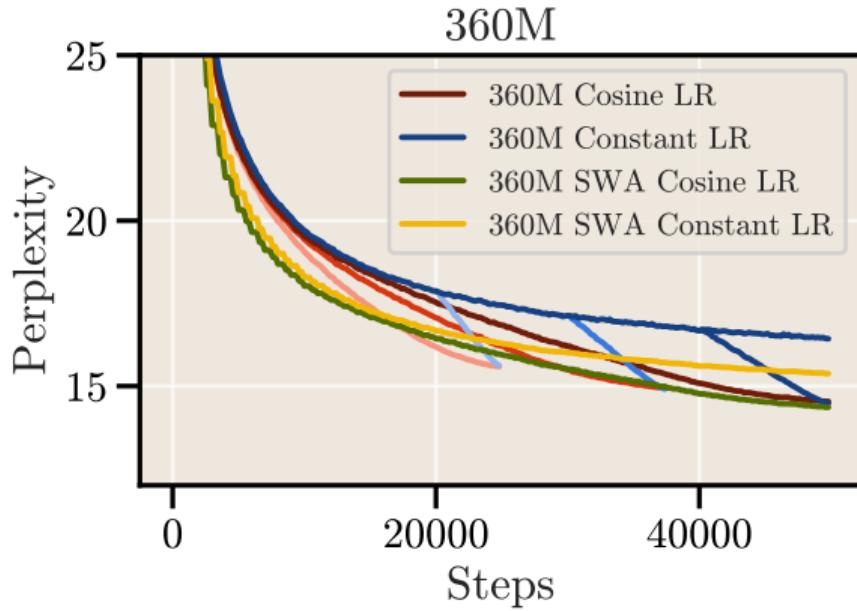
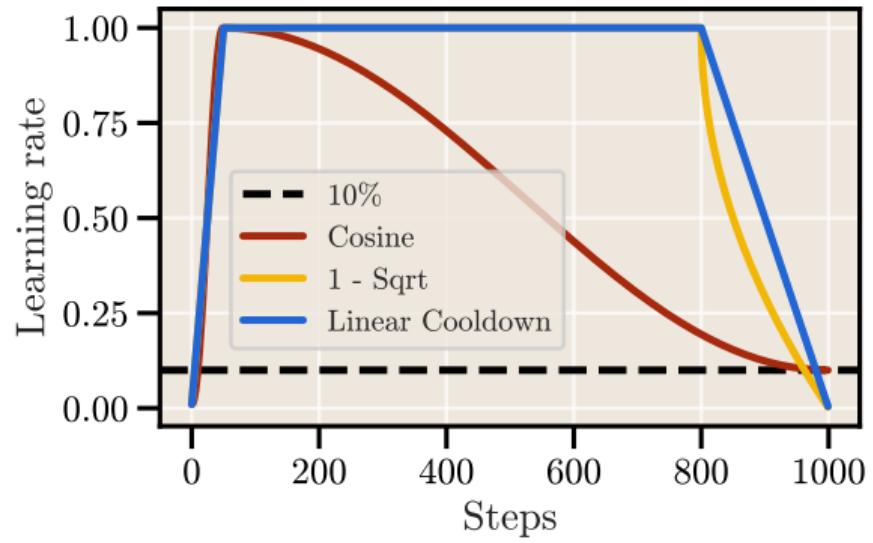


Рис. 13: gradual warmup

<sup>19</sup>Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour

## Cooldown<sup>20 21</sup>



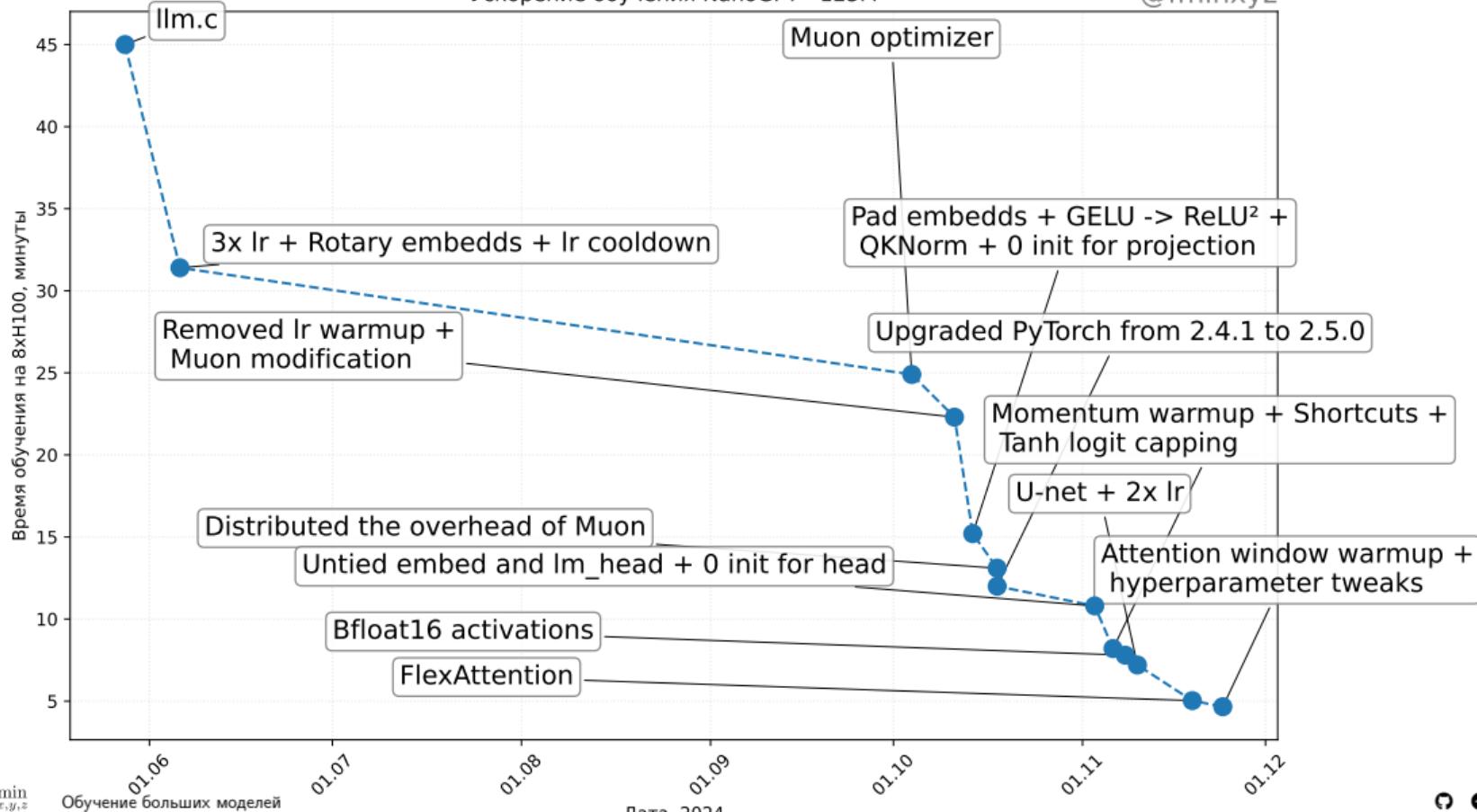
<sup>20</sup>Scaling Laws and Compute-Optimal Training Beyond Fixed Training Durations

<sup>21</sup>Scaling Vision Transformers

# NanoGPT speedrun

Ускорение обучения NanoGPT - 125M

@fminxyz



## Работают ли трюки, если увеличить размер модели?

### Scaling up the NanoGPT (124M) speedrun



Рис. 15: Источник

## Работают ли трюки, если увеличить размер модели?



Рис. 16: Источник

Спасибо за внимание!

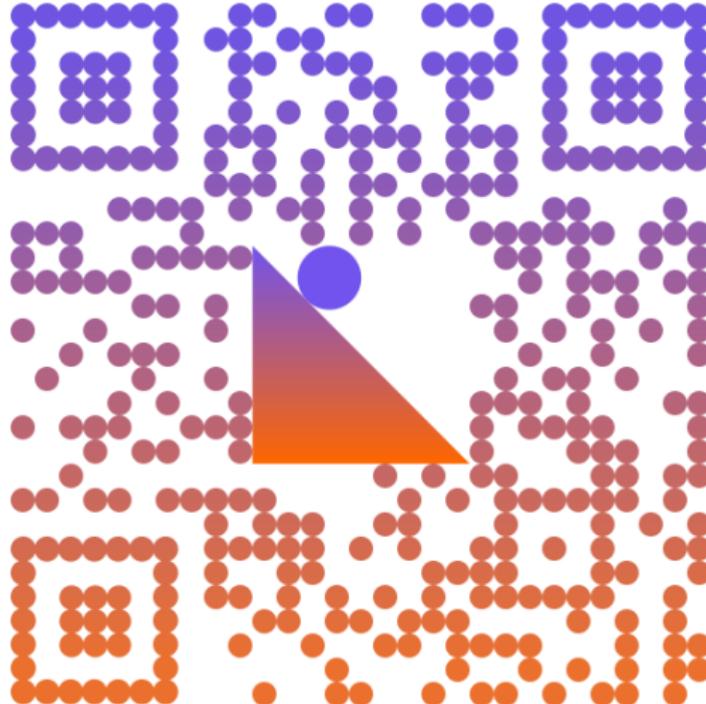
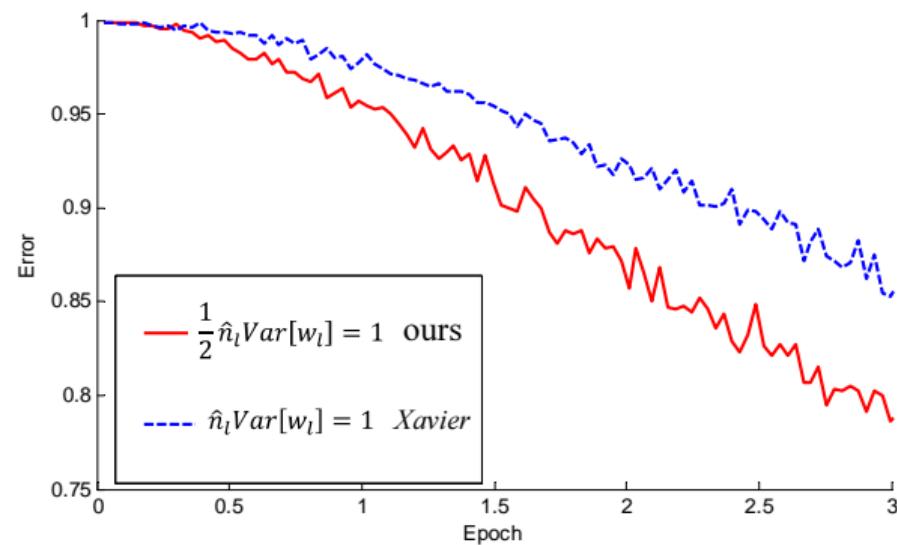


Рис. 17: Мои контакты

Запас

# Влияние инициализации весов нейронной сети на сходимость методов

22



<sup>22</sup>Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

## Методы уменьшения дисперсии? Не всё так просто на практике<sup>23</sup>

<sup>23</sup>On the Ineffectiveness of Variance Reduced Optimization for Deep Learning