

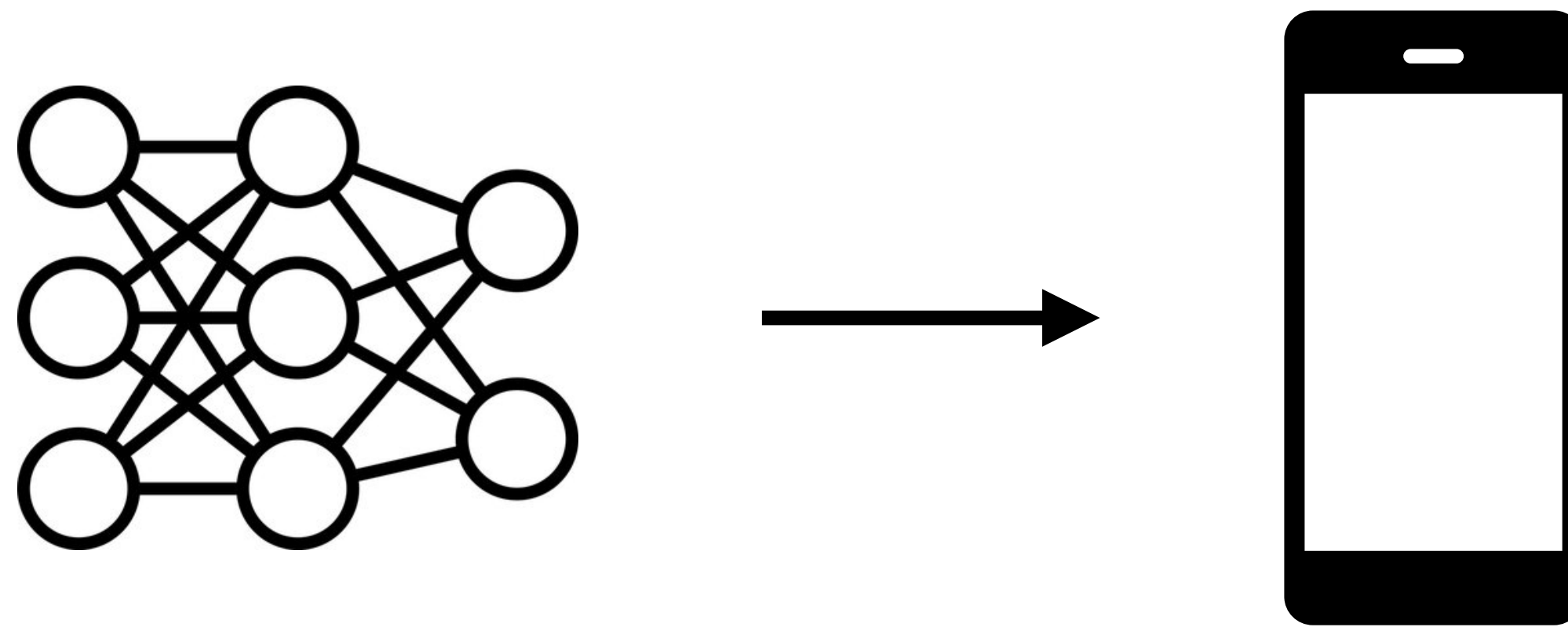
Уменьшение размеров нейронных сетей

План

- Прунинг
- Квантизация
- Дистилляция

Мотивация

- С каждым годом нейронные сети становятся больше
- При этом часто хочется использовать модели локально на портативных устройствах
- Для этого нужно научиться уменьшать их размеры без потери качества

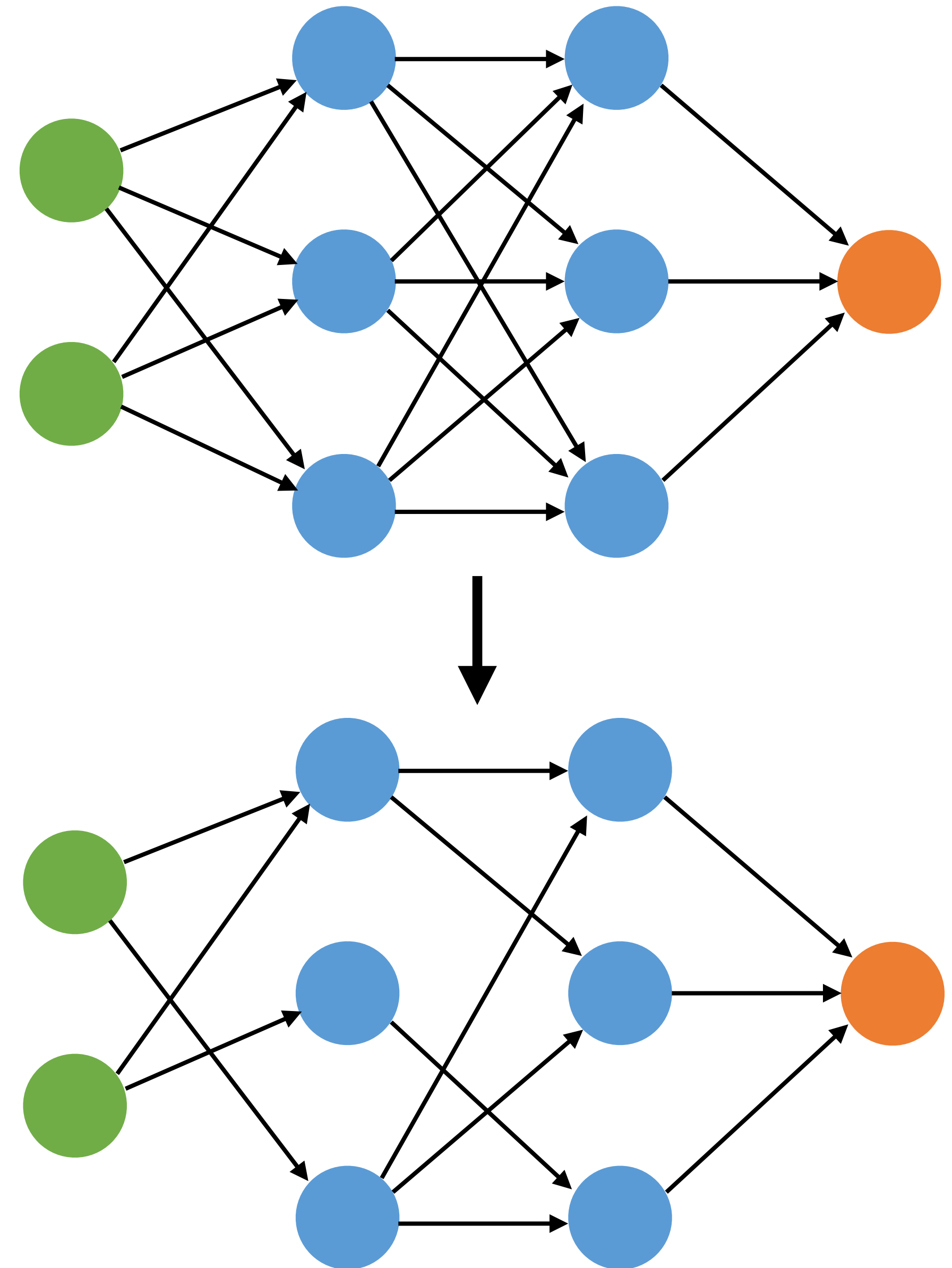


Прунинг

Нейронные сети часто состоят из миллионов параметров.

При этом оказывается, что не все параметры нужны.

Прунинг – удаление части весов модели без значимых потерь в качестве.



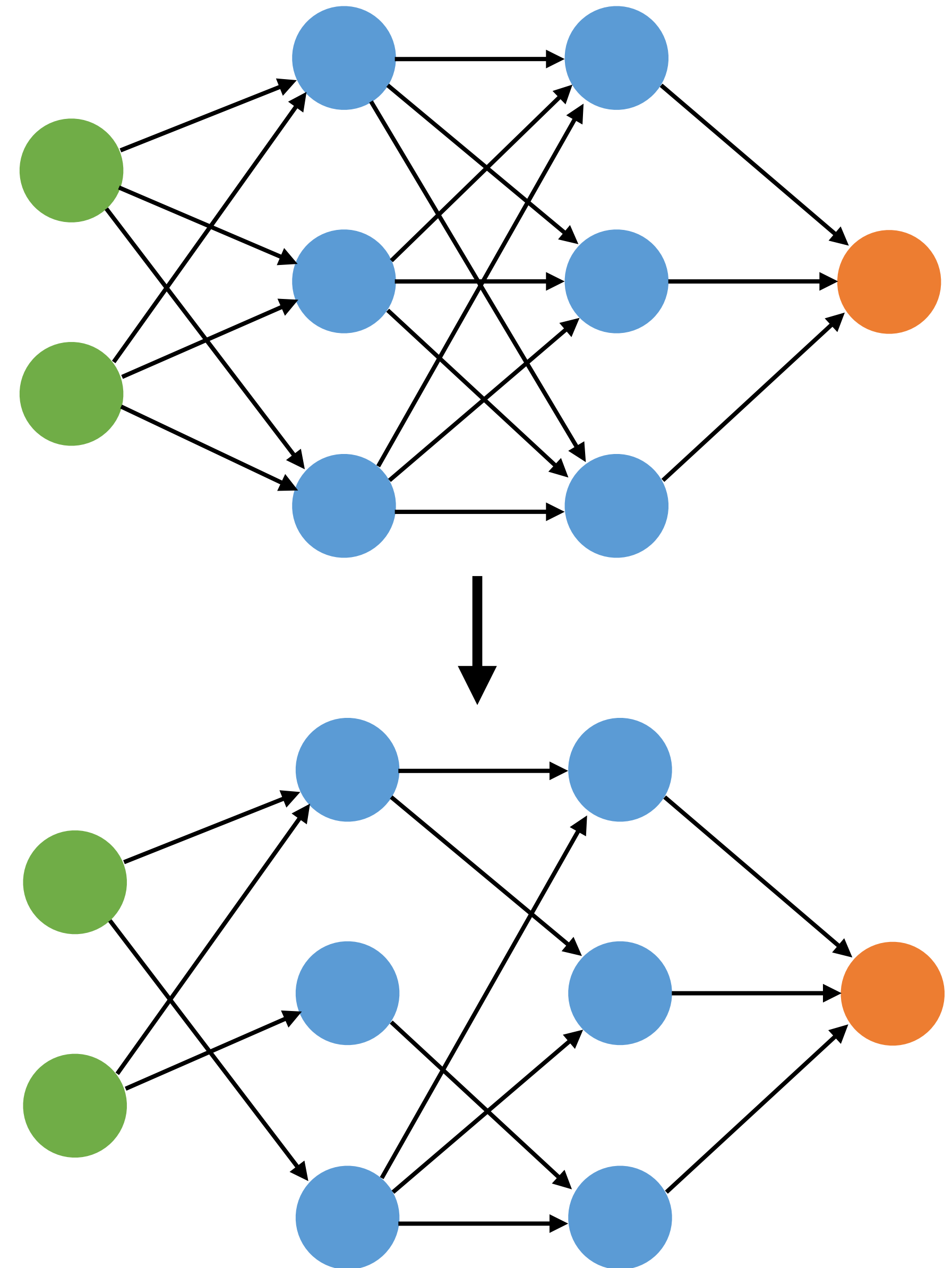
Прунинг

Нейронные сети часто состоят из миллионов параметров.

При этом оказывается, что не все параметры нужны.

Прунинг – удаление части весов модели без значимых потерь в качестве.

Dropout – частный случай прунинга



Критерии удаления весов

- **Магнитуда весов:** удаляем веса с наименьшим абсолютным значением (самый популярный метод)

Критерии удаления весов

- **Магнитуда весов:** удаляем веса с наименьшим абсолютным значением (самый популярный метод)
- **Дисперсия выходов нейронов:** если выход не меняется для разных входов, то нейрон можно заменить на константу и добавить в сдвиг

Критерии удаления весов

- **Магнитуда весов:** удаляем веса с наименьшим абсолютным значением (самый популярный метод)
- **Дисперсия выходов нейронов:** если выход не меняется для разных входов, то нейрон можно заменить на константу и добавить в сдвиг
- **Величина активации нейрона:** удаляем нейрон, если он никогда не активируется
Не требуется допущение равенства входных значений как в магнитуде весов.

Критерии удаления весов

- **Магнитуда весов:** удаляем веса с наименьшим абсолютным значением (самый популярный метод)
- **Дисперсия выходов нейронов:** если выход не меняется для разных входов, то нейрон можно заменить на константу и добавить в сдвиг
- **Величина активации нейрона:** удаляем нейрон, если он никогда не активируется
Не требуется допущение равенства входных значений как в магнитуде весов.
- **Корреляция между весами:** если веса одного нейрона сильно коррелируют, то можно оставить только один

Критерии удаления весов

- **Магнитуда весов:** удаляем веса с наименьшим абсолютным значением (самый популярный метод)
- **Дисперсия выходов нейронов:** если выход не меняется для разных входов, то нейрон можно заменить на константу и добавить в сдвиг
- **Величина активации нейрона:** удаляем нейрон, если он никогда не активируется
Не требуется допущение равенства входных значений как в магнитуде весов.
- **Корреляция между весами:** если веса одного нейрона сильно коррелируют, то можно оставить только один
- L_0 -регуляризация: мотивируем модель загулять веса.

$$\|w\|_0 = \sum_i [w_i \neq 0]$$

Обучение разреженных моделей

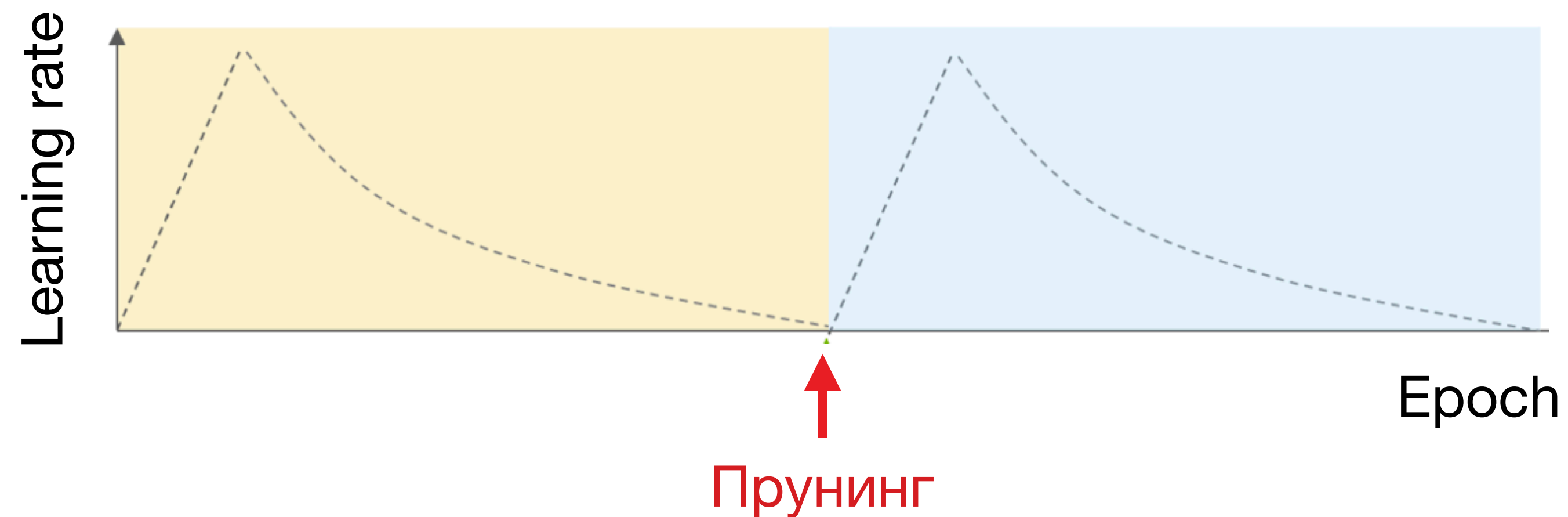
После удаления весов модель необходимо дообучить, чтобы перенастроить оставшиеся веса.

Помимо этого можно удалять веса в разные моменты времени:

- После обучения
- Во время обучения

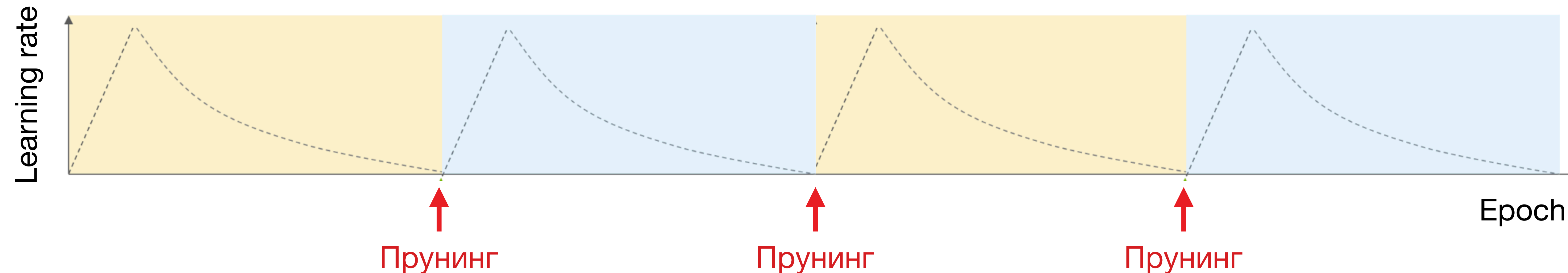
Прунинг после обучения

- Обучаем модель до сходимости
- Обрезаем веса
- Дообучаем



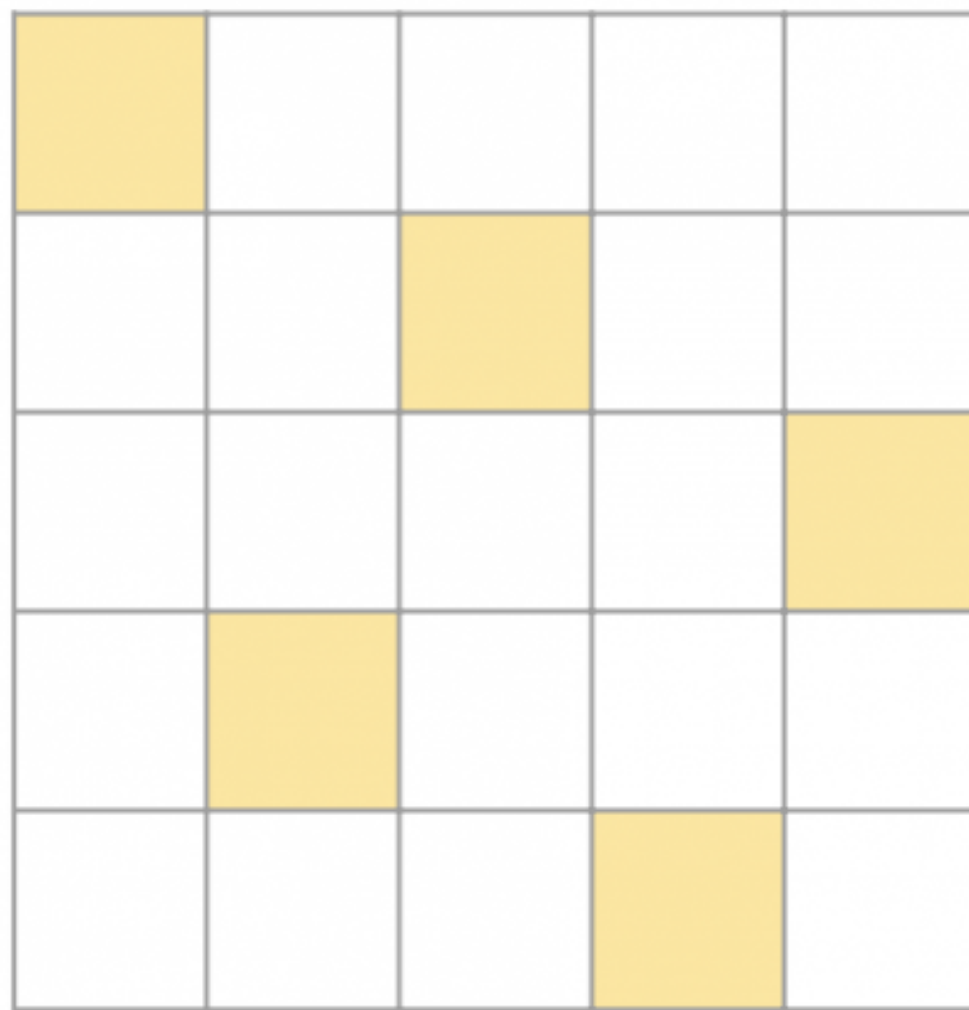
Прунинг во время обучения

- Обрезаем долю весов каждые несколько итераций обучения
- Больше настраиваемых параметров
- Обычно работает лучше одного обрезания

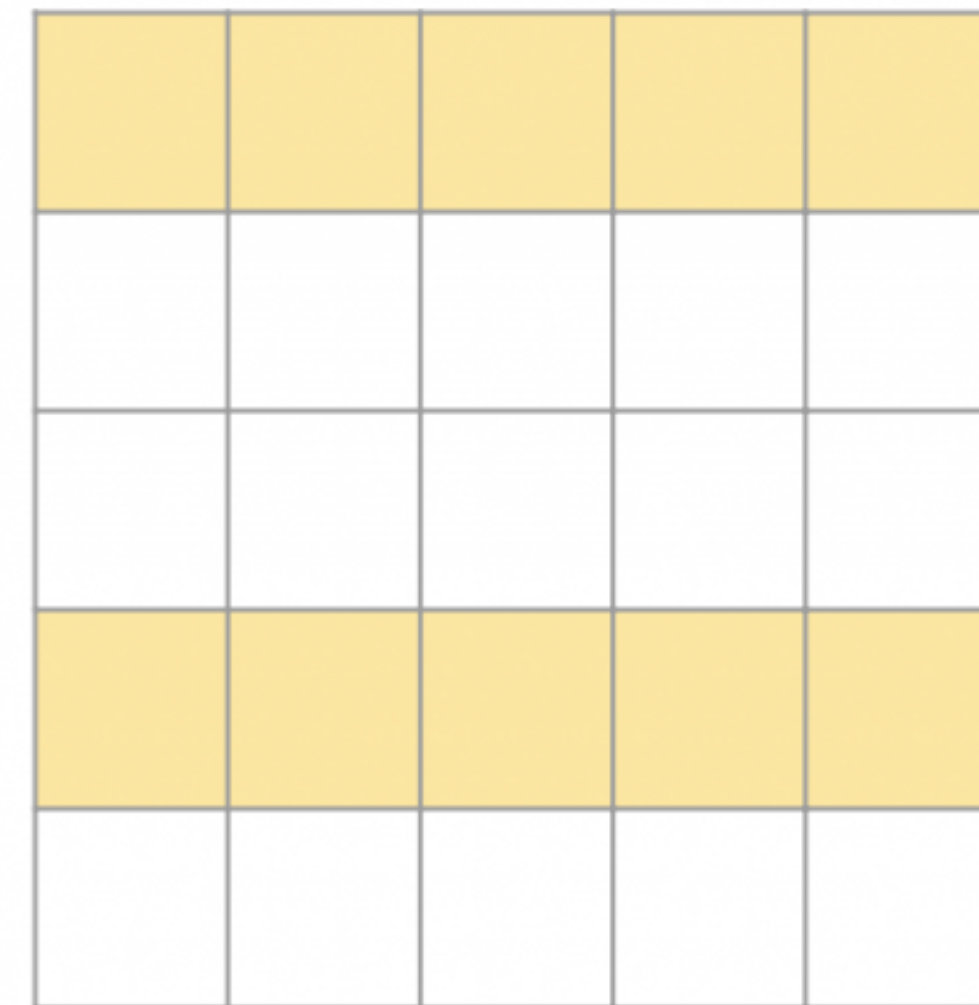


Структурированный прунинг

- Операции умножения матриц параллелизованны на уровне векторов
- При удалении отдельных весов, сеть становится разреженной, но прироста в скорости нет
- Для значимого ускорения нужно удалять целые структуры: векторы матриц, слои сети



Неструктурированный



Структурированный

Недостатки прунинга

- Прунинг позволяет уменьшить число весов на 30-50%, но это не дает большой прирост в скорости из-за реализации слов моделей
- Прунинг работает хорошо для старых неоптимизированных архитектур и для современных моделей он не так полезен
- Заранее нельзя гарантировать, что та или иная техника сработает

План

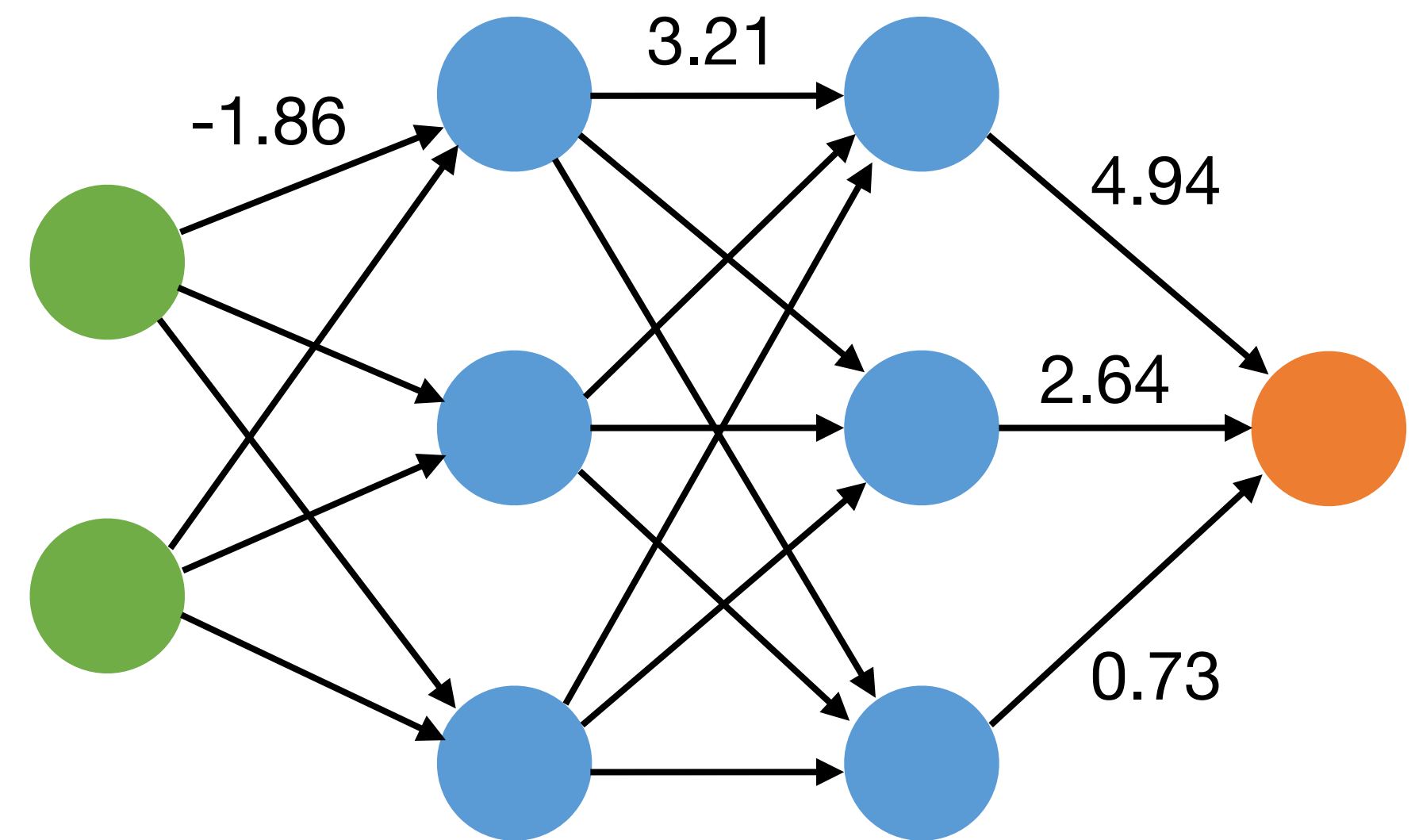
- Прунинг
- **Квантизация**
- Дистилляция

Квантизация

- По умолчанию нейронные сети работают с числами в формате float32.
- Это вещественные числа, которые занимают 32 бита.
- Чем больше бит используется для представления числа, тем больше нужно
 - Памяти на хранение
 - Времени на операции сложения и умножения

Квантизация – уменьшение точности (битности) чисел.

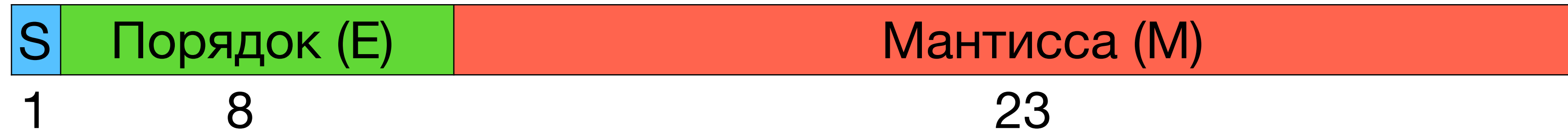
Чаще всего используется **после** обучения.



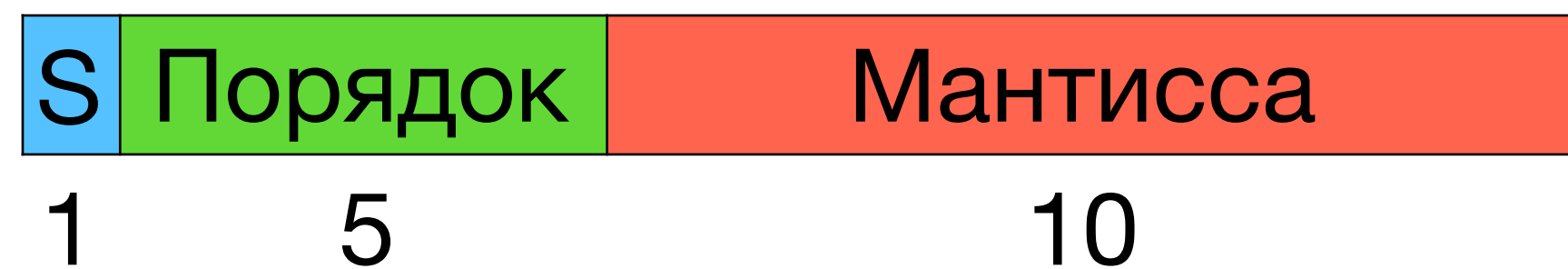
Как машины видят числа?

float32 (single-precision)

знак

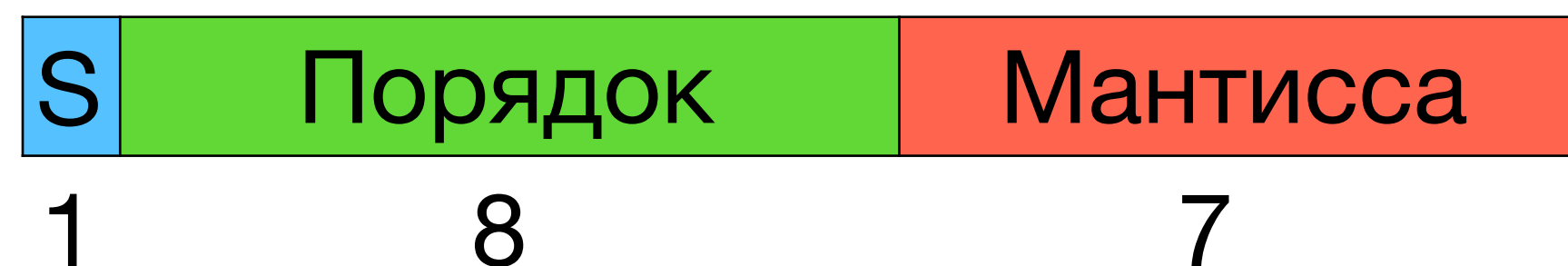


float16 (half-precision)



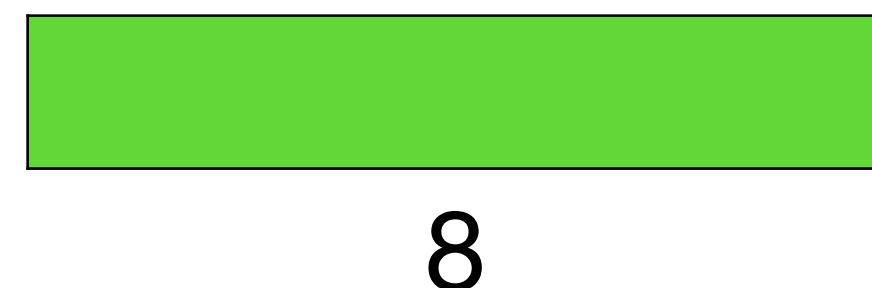
$$x = (-1)^S \cdot (1, M) \cdot 2^{E-127}$$

bf16 (brain float)



Для представления больших чисел

int8



Квантизация

Чаще всего точность понижается с **float32** до

- **float16**
- **int8** (float8 не используется, потому что он почти нигде не определен)

Квантизация в **float16** выполняется в лоб обрезанием мантиссы до нужного размера

Квантизация в **int8** работает намного хитрее

float32 -> int8

Пусть x – число в float32, x_q – квантизованное x в int8

$$x_q = \text{round}\left(\frac{x}{s} + z\right)$$

- s – масштабирующий коэффициент; положительное число в float32
- z – сдвиг; он хранится в int8 и соответствует 0 в исходном float32 числе

Такая квантизация называется **асимметричной** и используется, когда $x_q \in [0, 255]$

Как найти s и z ?

Если мы знаем диапазон $[a, b]$ значений x и $x_q \in [a_q, b_q]$, то

$$a = s(a_q - z)$$

$$b = s(b_q - z)$$

Тогда

$$s = \frac{b - a}{b_q - a_q} \quad z = \frac{ba_q - ab_q}{b - a}$$

Как найти s и z ?

Если мы знаем диапазон $[a, b]$ значений x и $x_q \in [a_q, b_q]$, то

$$a = s(a_q - z)$$

$$b = s(b_q - z)$$

Тогда

$$s = \frac{b - a}{b_q - a_q} \quad z = \frac{ba_q - ab_q}{b - a}$$

Важно убедиться, что 0 переводится без погрешностей. То есть что z – целое число. Поэтому

$$z = \text{round} \left(\frac{ba_q - ab_q}{b - a} \right)$$

Как найти s и z ?

Если мы знаем диапазон $[a, b]$ значений x и $x_q \in [a_q, b_q]$, то

$$a = s(a_q - z)$$

$$b = s(b_q - z)$$

Тогда

$$s = \frac{b - a}{b_q - a_q} \quad z = \frac{ba_q - ab_q}{b - a}$$

Важно убедиться, что 0 переводится без погрешностей. То есть что z – целое число. Поэтому

$$z = \text{round}\left(\frac{ba_q - ab_q}{b - a}\right)$$

Если попадаете $x \notin [a, b]$, то округляем его до ближайшей границы.

$$x_q = \text{clip}\left(\text{round}\left(\frac{x}{s} + z\right), a_q, b_q\right)$$

Симметричная квантизация

Используется, если границы симметричны, $x \in [-a, a]$.

Для квантизации берется $x_q \in [-127, 127]$.

Так мы теряем одно значение квантизованной переменной, но избавляемся от сдвига

$$x_q = \textit{round}(s \cdot x)$$

Как найти границы a и b ?

Квантизовывать нужно веса модели и входные значения (активации).

- Для весов легко – они не меняются
- Для активаций есть три способа:
 - Динамическая квантизация
 - Статичная квантизация
 - Обучение с учетом квантизации

Как найти границы a и b ?

Квантизовывать нужно веса модели и входные значения (активации).

- Для весов легко – они не меняются
- Для активаций есть три способа:
 - Динамическая квантизация – считаем границы на лету для каждой активации
 - Статичная квантизация – заранее оцениваем границы прогоняя n примеров
 - Обучение с учетом квантизации – используем квантизацию во время обучения и находим границы по тренировочным примерам

Как найти границы a и b ?

Квантизовывать нужно веса модели и входные значения (активации).

- Для весов легко – они не меняются
- Для активаций есть три способа:
 - Динамическая квантизация – считаем границы на лету для каждой активации (максимальная точность, ниже скорость)
 - Статичная квантизация – заранее оцениваем границы прогоняя n примеров (ниже точность, максимальная скорость)
 - Обучение с учетом квантизации – используем квантизацию во время обучения и находим границы по тренировочным примерам (хорошая точность, максимальная скорость, модель настраивается под квантизацию)

Калибровка значений границ

Min-max: хорошо работает для весов модели

a = минимальное найденное значение

b = максимальное найденное значение

Moving average min-max: хорошо работает для активаций

a = скользящее среднее минимальное значение

b = скользящее среднее максимальное значение

Калибровка значений границ

Min-max: хорошо работает для весов модели

a = минимальное найденное значение

b = максимальное найденное значение

Moving average min-max: хорошо работает для активаций

a = скользящее среднее минимальное значение

b = скользящее среднее максимальное значение

Histogram-based – формируем гистограммы из всех значений активаций, находим границы одним из 3-х способов:

- **Entropy:** минимизируем KL-дивергенцию между распределениями квантизованных и исходных значений
- **MSE:** минимизируем квадрат разности значений бинов гистограмм
- **Перцентиль:** выбираем границы по перцентильям значений активаций

Операции с квантизованными числами

Сложение:

$$y = x_1 + x_2 = s_1(x_{q1} - z_1) + s_2(x_{q2} - z_2) = s_y(y_q - z_y)$$

$$y_q = z_y + \frac{1}{s_y} \left(s_1(x_{q1} - z_1) + s_2(x_{q2} - z_2) \right)$$

Операции с квантизованными числами

Сложение:

$$y = x_1 + x_2 = s_1(x_{q1} - z_1) + s_2(x_{q2} - z_2) = s_y(y_q - z_y)$$
$$y_q = z_y + \frac{1}{s_y} \left(s_1(x_{q1} - z_1) + s_2(x_{q2} - z_2) \right)$$

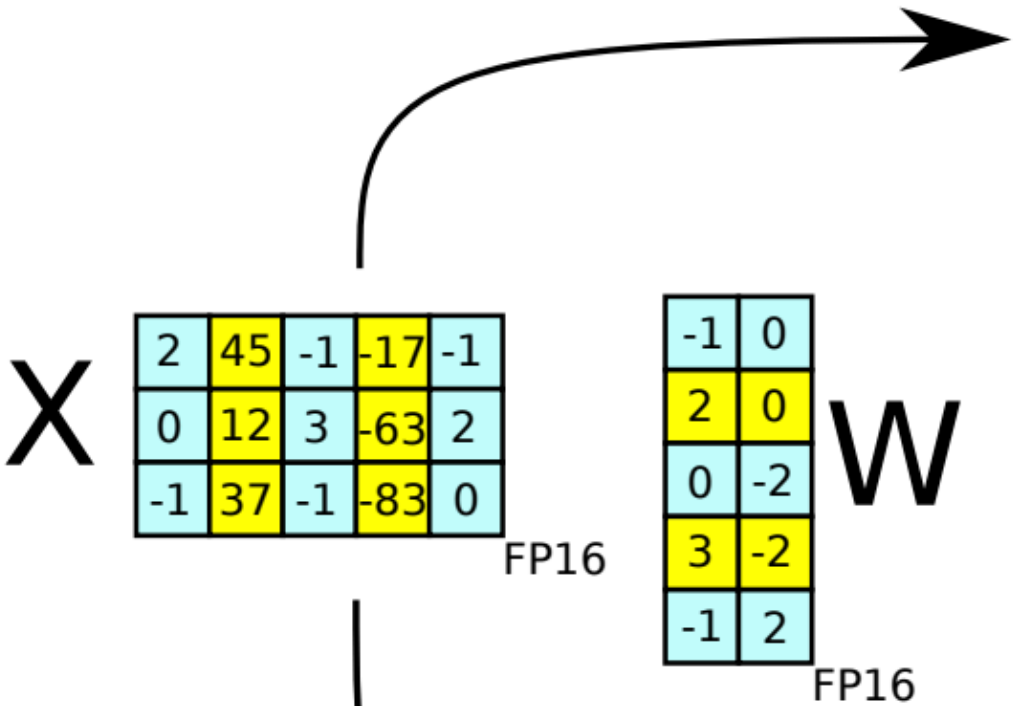
Умножение:

$$y = x_1 \cdot x_2 = s_1(x_{q1} - z_1) \cdot s_2(x_{q2} - z_2) = s_y(y_q - z_y)$$
$$y_q = z_y + \frac{s_1 s_2}{s_y} (x_{q1} - z_1)(x_{q2} - z_2)$$

LLM.int8()

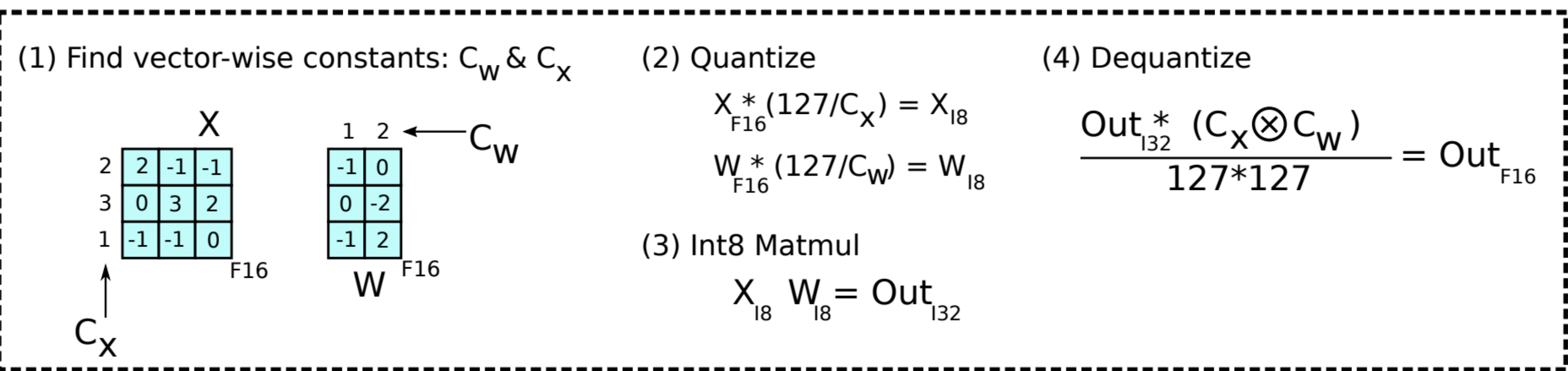
Динамическая квантизация. Используется в Hugging Face (bitsandbytes).

LLM.int8()

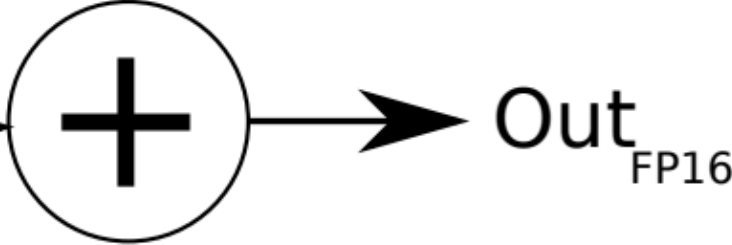
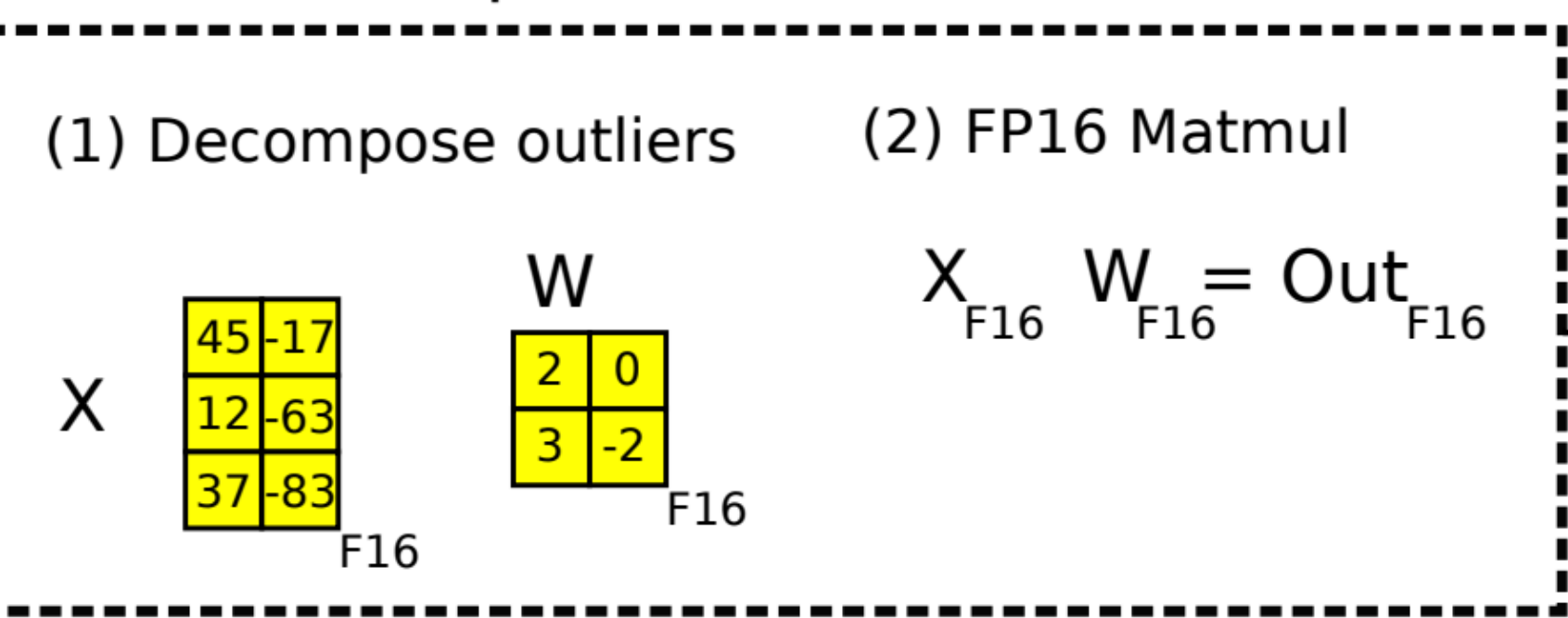


Regular values
Outliers

8-bit Vector-wise Quantization

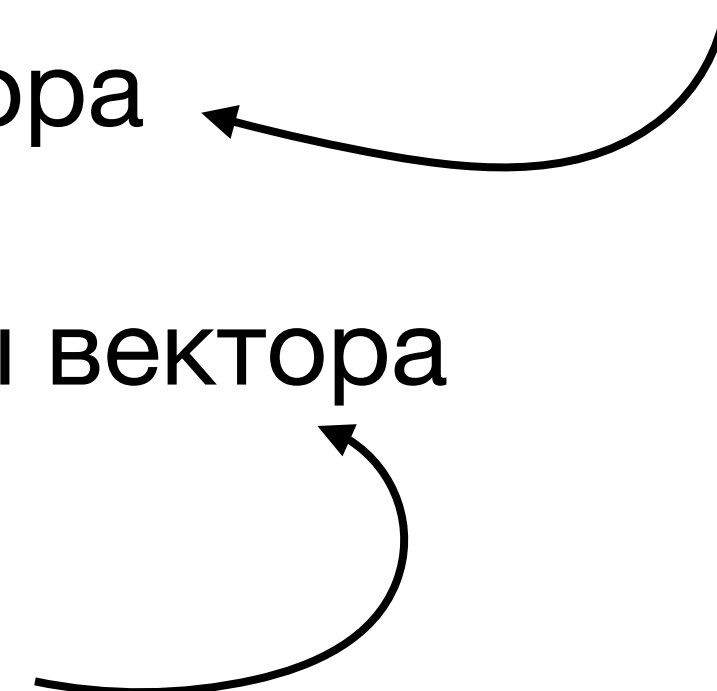


16-bit Decomposition



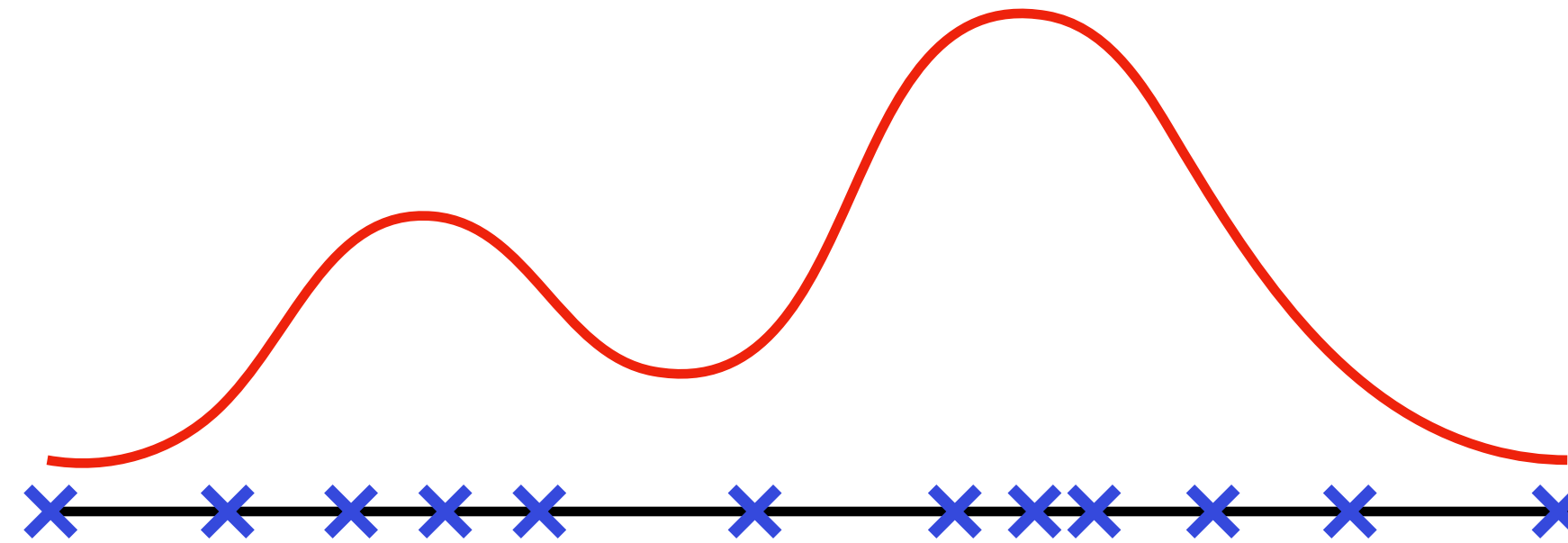
Per-tensor и per-channel квантизация

Так как параметры квантизации занимают место, хочется минимизировать их число

- **Per-tensor:** храним одну пару (s; z) для каждого тензора
 - **Per-channel:** храним по паре для каждой координаты вектора
- Меньше памяти
- Выше точность
- 
- A diagram with two curved arrows. One arrow starts at the text 'Меньше памяти' and points to the 'Per-tensor' bullet point. The other arrow starts at the text 'Выше точность' and points to the 'Per-channel' bullet point.

Нелинейная квантизация

Обычно значения весов и активация распределены неравномерно.
Полезно увеличивать точность там, где плотность больше.



Для этого есть несколько способов, но они используются реже из-за сложности.

Практические советы

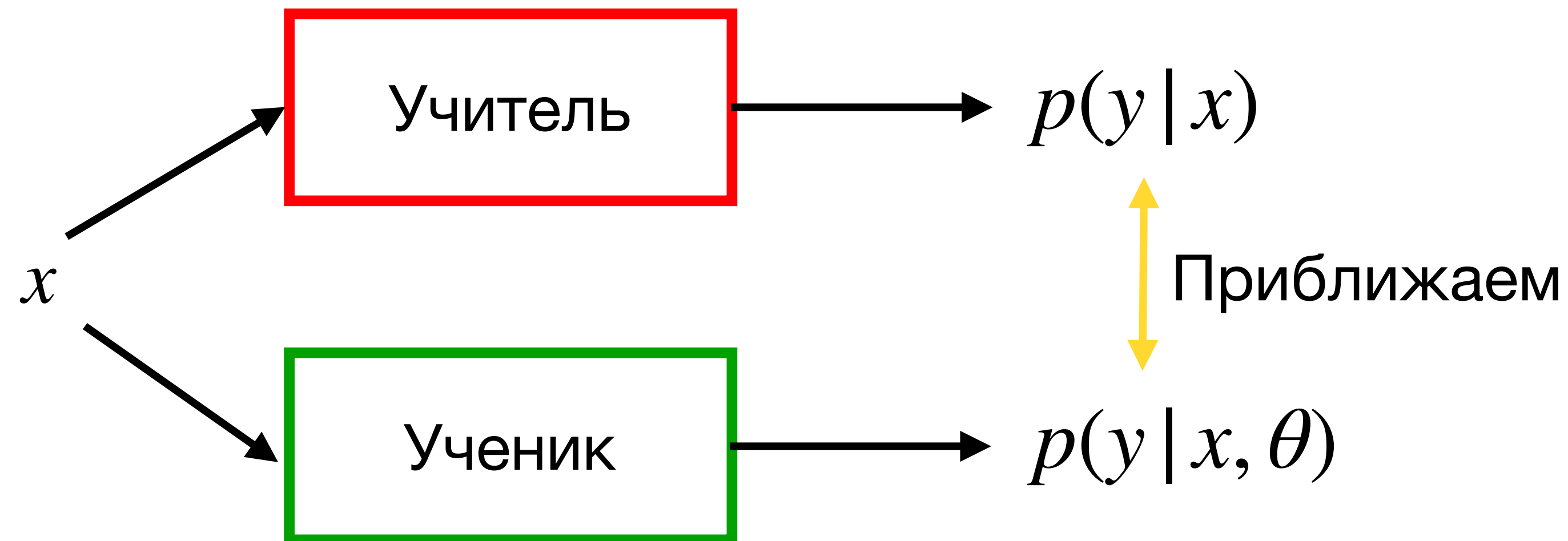
- Не все операции нужно квантизовывать.
Важнее всего **матричные умножения**, так как они самые дорогие.
Перемножение двух матриц $A \in \mathbb{R}^{n \times m}$ и $B \in \mathbb{R}^{m \times p}$ требует около $2nmp$ операций.
- Для Layer Norm лучше не применять квантизацию, слой после этого может работать не стабильно
- Лучше всего начать с динамической квантизации.
Если скорости не хватает, то перейти к статической.
В конце, если не хватает точности, то к обучению с учетом квантизации.

План

- Прунинг
- Квантизация
- **Дистилляция**

Дистилляция

- Дистилляция знаний – процесс переноса знаний из большой обученной модели (учителя) в маленькую модель (ученика) с минимальной потерей качества
- Дистилляция реализуется путем приближения выходов ученика к выходам учителя



Как именно приближать?

Для приближения вероятностей минимизируется одна из двух ошибок:

- **KL-дивергенция**

$$KL(p^s \| p^t) = \sum_{k=1}^K p_k^s \log \frac{p_k^s}{p_k^t}$$

- **Кросс-энтропия**

$$CE(p^t, p^s) = - \sum_{k=1}^K p_k^t \log p_k^s$$

Как именно приближать?

Для приближения вероятностей минимизируется одна из двух ошибок:

- **KL-дивергенция**

$$KL(p^s \| p^t) = \sum_{k=1}^K p_k^s \log \frac{p_k^s}{p_k^t}$$

- **Кросс-энтропия**

$$\begin{aligned} CE(p^t, p^s) &= - \sum_{k=1}^K p_k^t \log p_k^s \propto \\ &\propto \sum_{k=1}^K p_k^t \log p_k^t - \sum_{k=1}^K p_k^t \log p_k^s = KL(p^t \| p^s) \end{aligned}$$

Сглаживание вероятностей

- Нейронные сети очень часто бывают слишком уверены в предсказаниях
- Распределение p^t оказывается почти вырожденным
- Приближение выходов к такому распределению не лучше обычного обучения с учителем

В дистилляции используются **сглаженные** вероятности.

$$\begin{aligned} p_{\tau}^t &= \textit{softmax}(l^t, T = \tau) \\ p_{\tau}^s &= \textit{softmax}(l^s, T = \tau) \end{aligned} \quad \tau > 1$$

Такие метки называются **мягкими**.

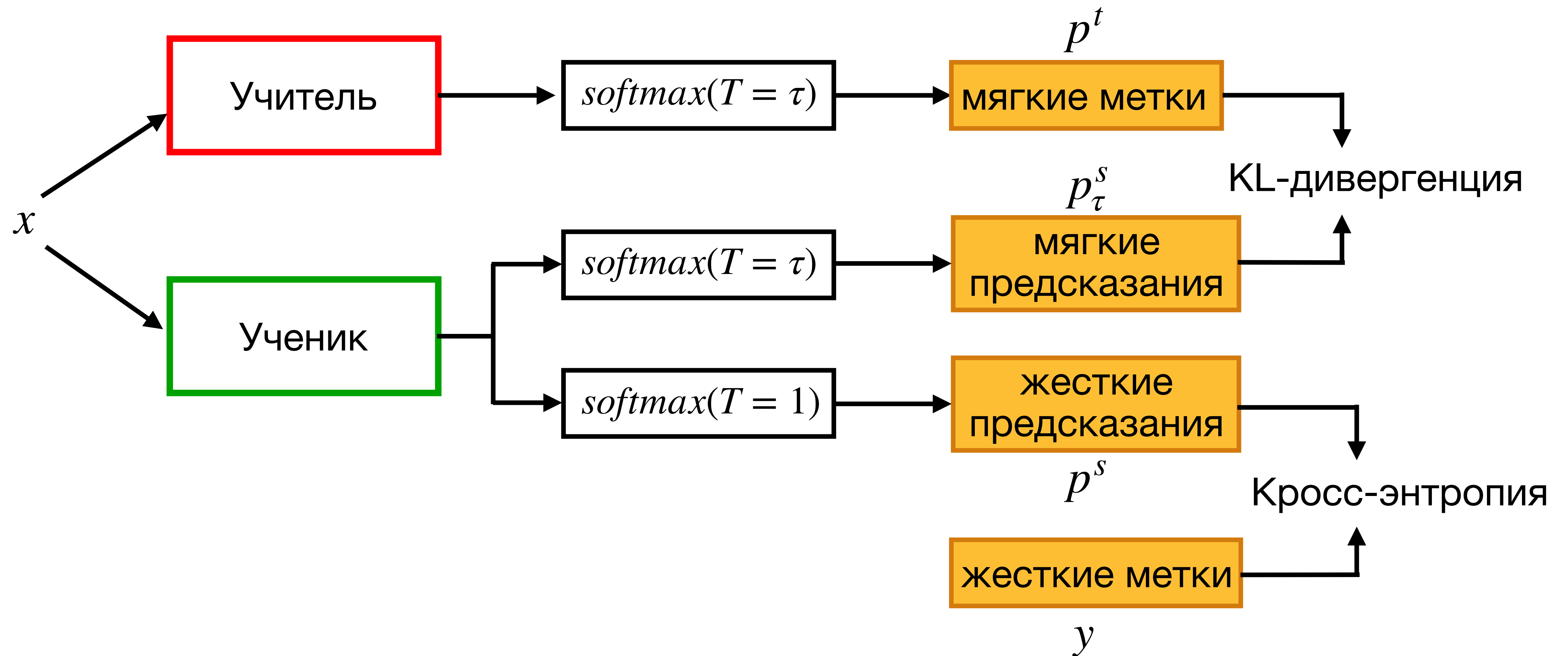
Жесткие метки

Так как у нас есть доступ к правильным ответам, добавляем так же стандартную ошибку.

$$CE(y, p^s) = - \sum_{k=1}^K [y = k] \log p_k^s$$

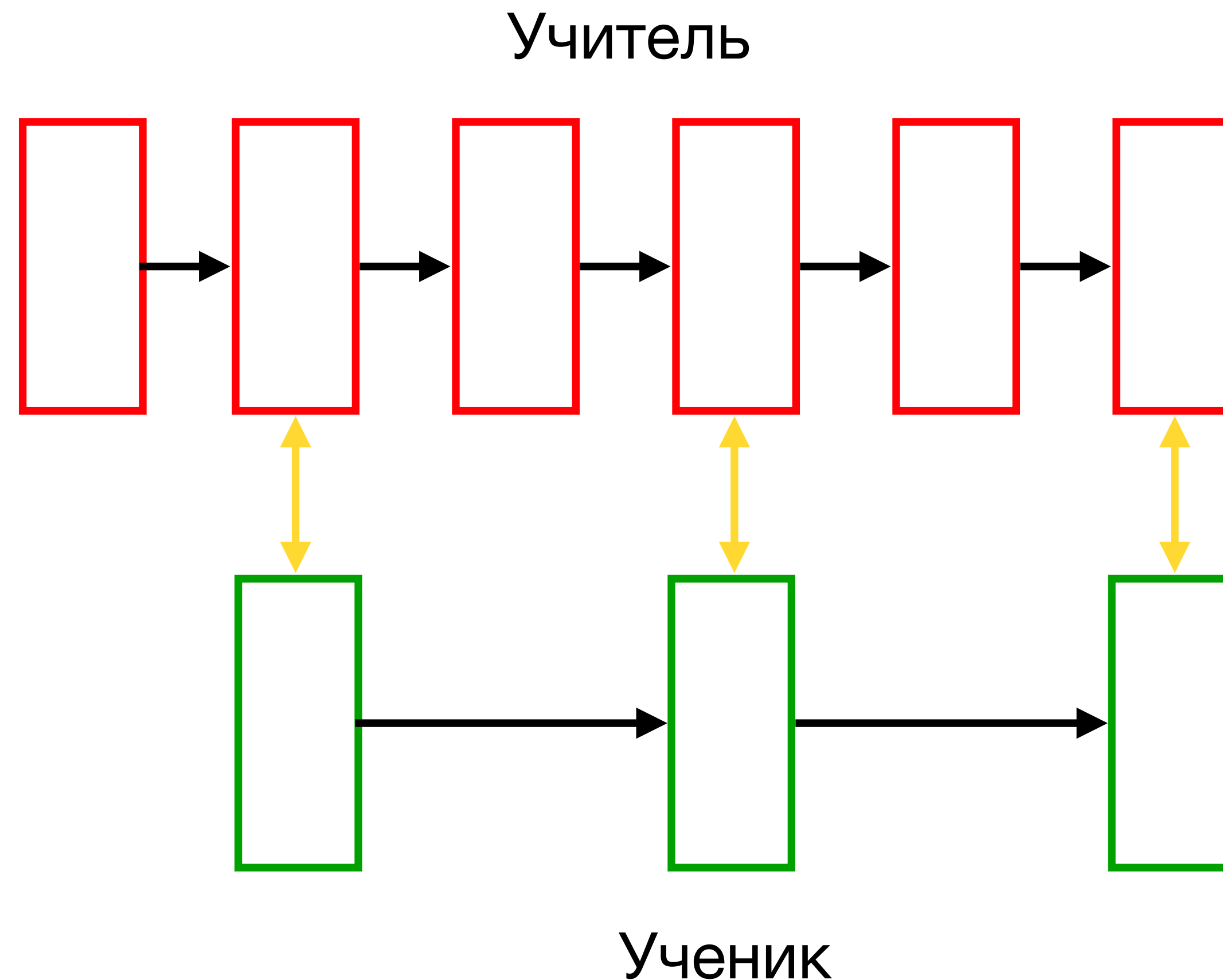
Правильные ответы y называются **жесткими** метками.

Схема обучения



Приближение промежуточных слоев

Мы приближаем только выходы моделей, но есть же еще выходы слоев!
Будем выходы слоев ученика приближать к выходам учителя



Приближение слоев внимания

Матрицы внимания имеют одинаковый размер: $[H, seq_len, seq_len]$

Введем ошибку для минимизации расстояния между ними

$$L_{attn} = \frac{1}{H} \sum_{h=1}^H \|A_h^s - A_h^t\|_2^2$$

Приближение выходов слоев

Размер выхода слоя может изменяться у разных моделей, поэтому добавим проекцию $W \in \mathbb{R}^{[d_s \times d_t]}$.

$$L_{hidn} = \|H^s W - H^t\|_2^2$$

Матрица W обучается вместе с параметрами ученика.

Дистилляция с задачей предобучения

Почти все трансформерные модели обучаются в два этапа: предобучение и дообучение.

В таком случае лучше сначала дистиллировать ученика на задаче предобучения, а затем на задаче дообучения.

