



лекция

Pruning

Курс «Ускорение нейросетей»

спикер



Андрей
Щербин



Александр
Гончаренко

→ enot.ai

Pruning — удаление определенных частей модели

→ Как выбирать части
модели для пруинга?

→ Какие это могут быть
части?

Содержание

Типы пруинга →

Содержание

Типы пруинга → Критерии важности →

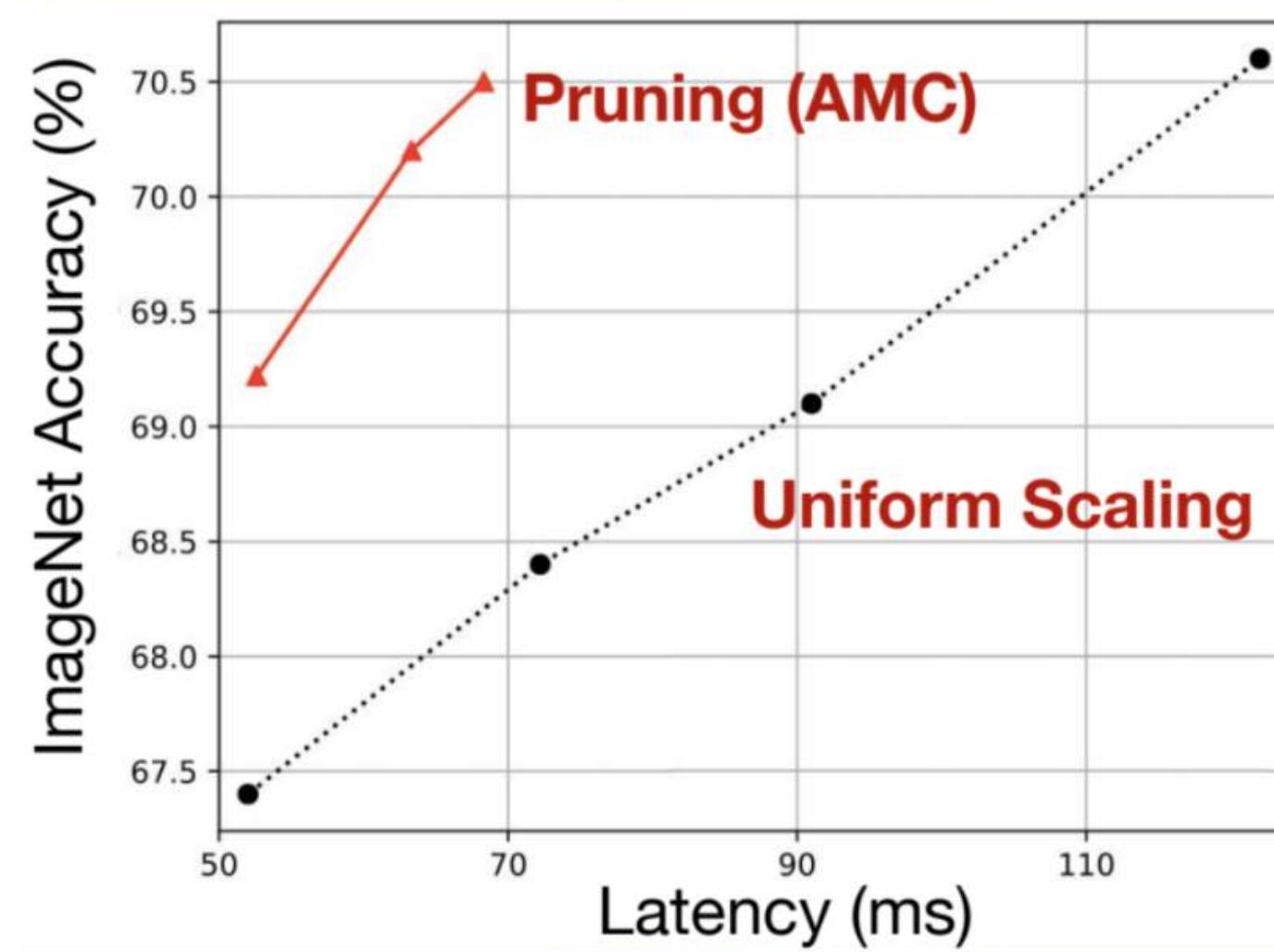
Содержание

Типы пруинга → Критерии важности →
Лотерейные билеты →

Содержание

Типы пруинга → Критерии важности →
Лотерейные билеты → **Практические
рекомендации** →

Почему пруним, а не скейлим архитектуру?



AMC: AutoML for Model Compression and Acceleration on Mobile Devices^[1]

Типы пруニングа

«Оси» прунинга

Гибкость

→ Uniform / Non-Uniform

Структура

→ Structured / Unstructured

Масштаб

→ Слои / Attention heads / Filters / Weights
with constraint / Weights without
constraints

Тактика применения

→ One-shot / Iterative

Критерий

«Оси» прунинга

Гибкость

→ Uniform / Non-Uniform

Структура

→ Structured / Unstructured

Масштаб

→ Слои / Attention heads / Filters / Weights
with constraint / Weights without
constraints

Тактика применения

→ One-shot / Iterative

Критерий

Гибкость

Uniform vs Non Uniform

Uniform

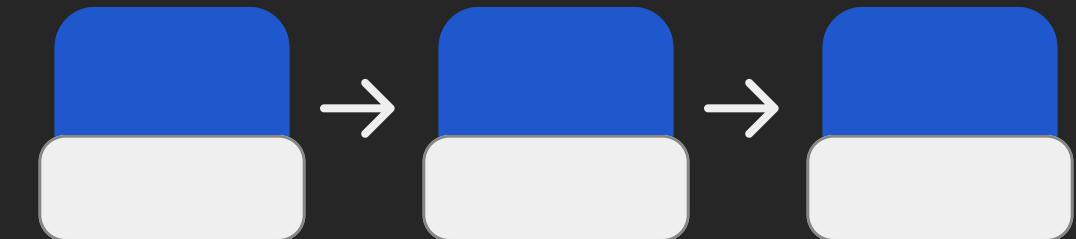
- На каждом слое выбрасываем одинаковый процент

Non Uniform

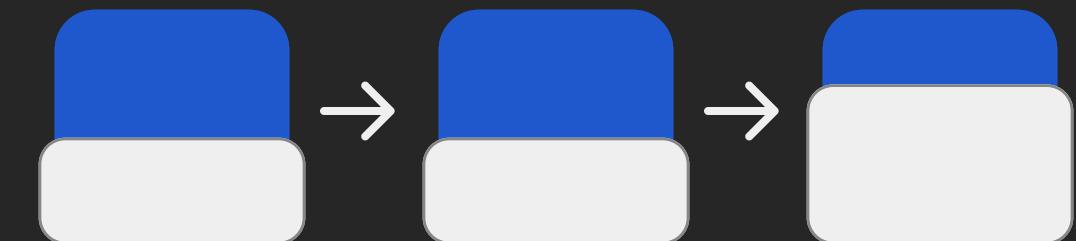
- Global
Сортируем то, что будем прунить по важности, выбрасываем самые бесполезные

→ Individual

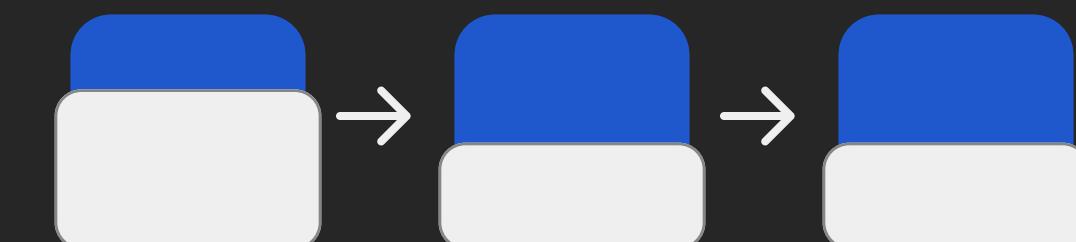
- Pruning ratio подбирается на каждом слое отдельно



Uniform



Global



Individual

Гибкость

- Uniform** → Даёт гарантированное ускорение
 - Может сильно повредить точности
- Non Uniform Global** → Даёт большую точность
 - Может не давать ускорения при том же уровне прореживания
- Non Uniform Individual** → Обеспечивает наибольший trade-off точность / скорость
 - Очень непрост в реализации, нет готовых open source решений

«Оси» прунинга

Гибкость

→ Uniform / Non-Uniform

Структура

→ Structured / Unstructured

Масштаб

→ Слои / Attention heads / Filters / Weights
with constraint / Weights without
constraints

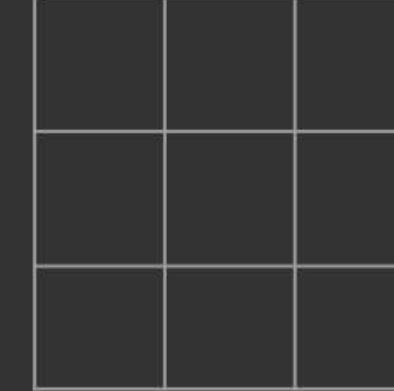
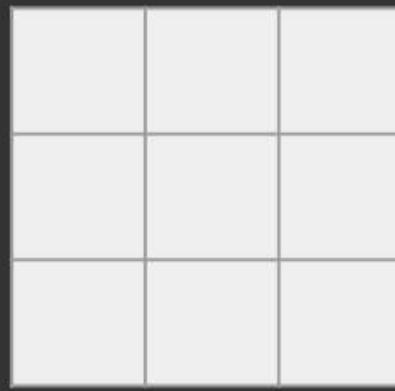
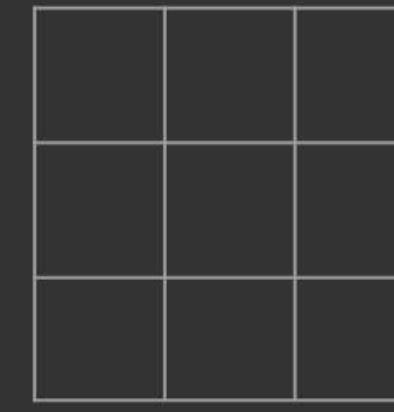
Тактика применения

→ One-shot / Iterative

Критерий

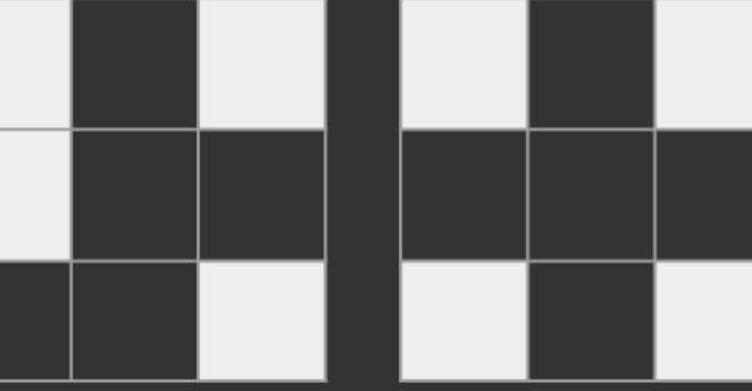
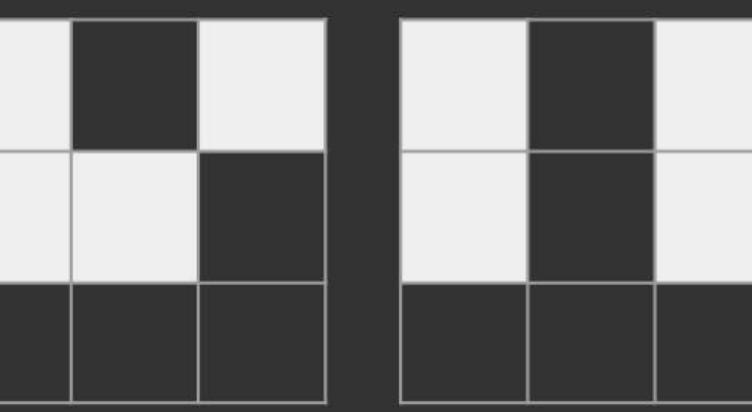
Структурированный vs Неструктурированный

→ Структурированный дает ускорение на большем количестве устройств



Структурированный

Удаляем наименее важные каналы



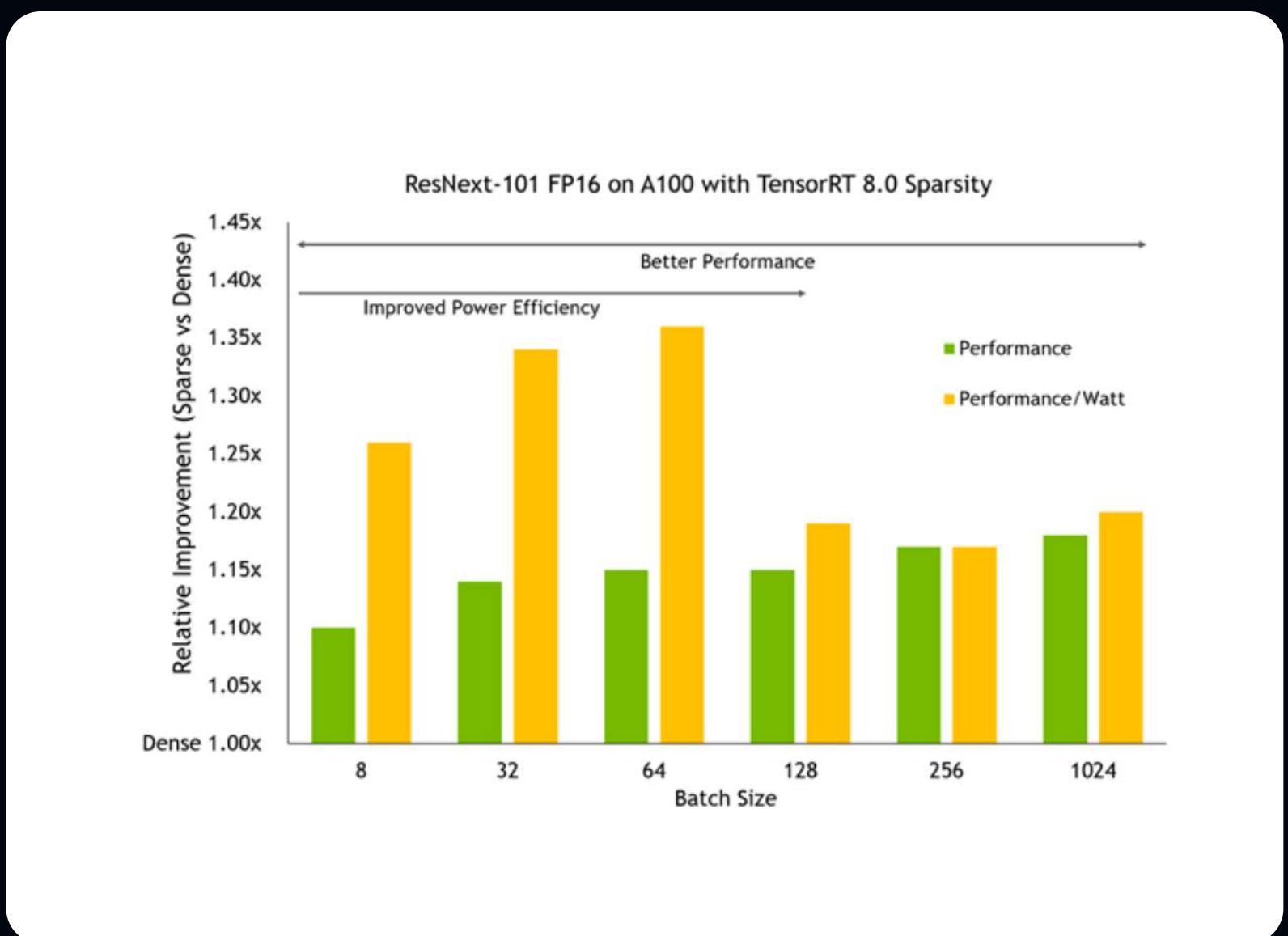
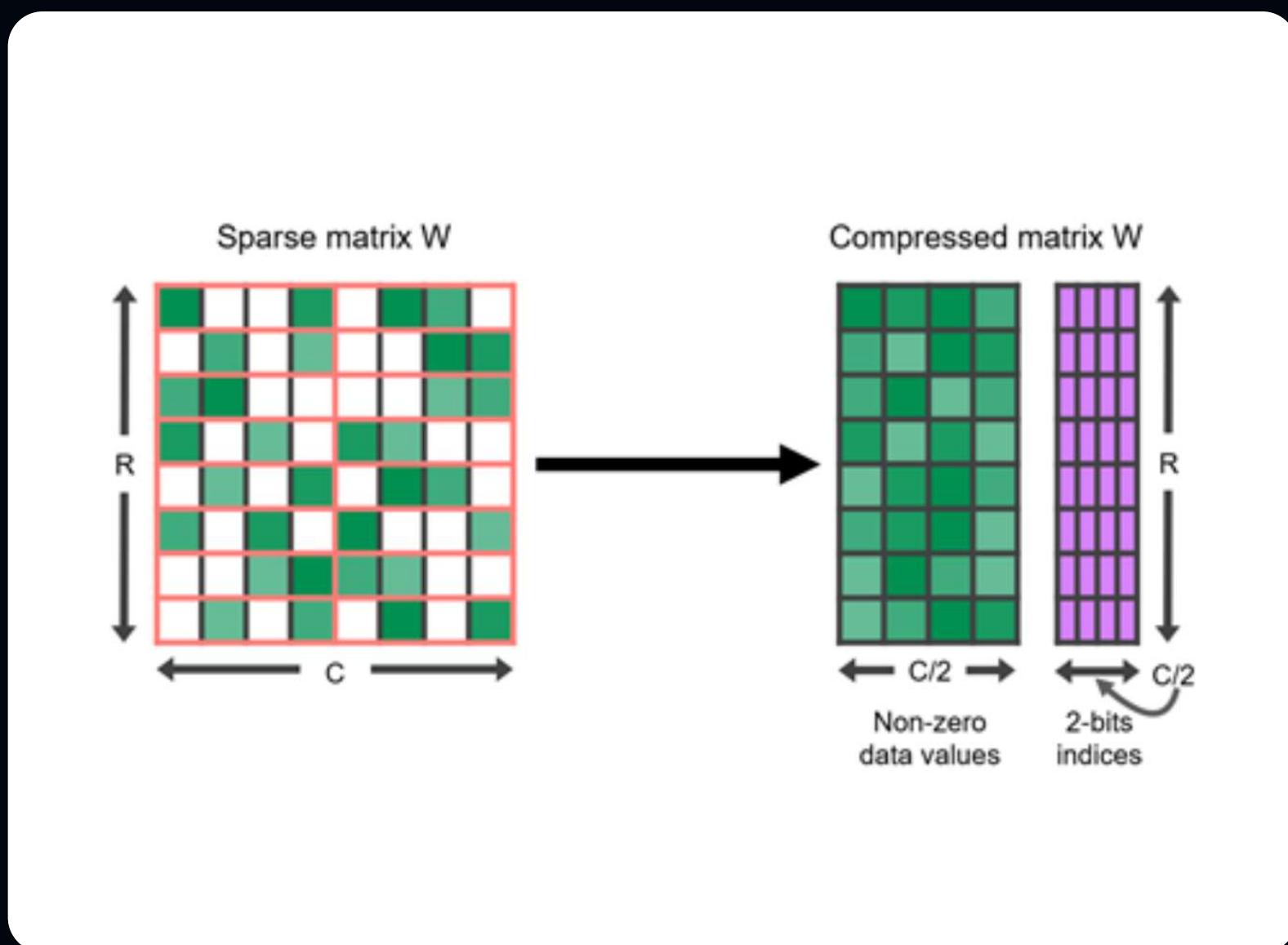
Неструктурированный

Зануляем наименее важные веса

Типы прунинга

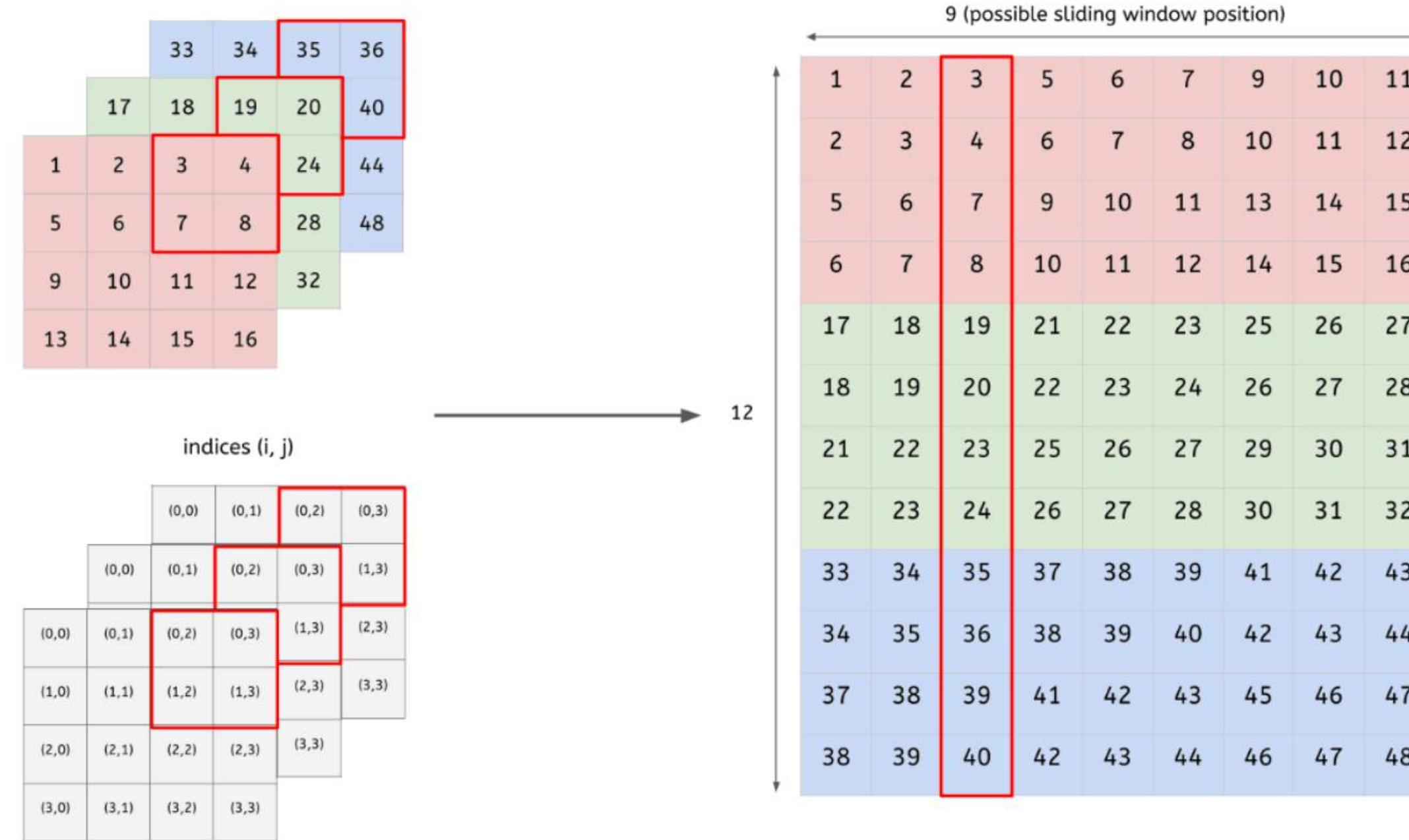
Semi-structured 2-4 pruning Nvidia

→ Поддерживается видеокартами Nvidia, начиная с поколения Ampere

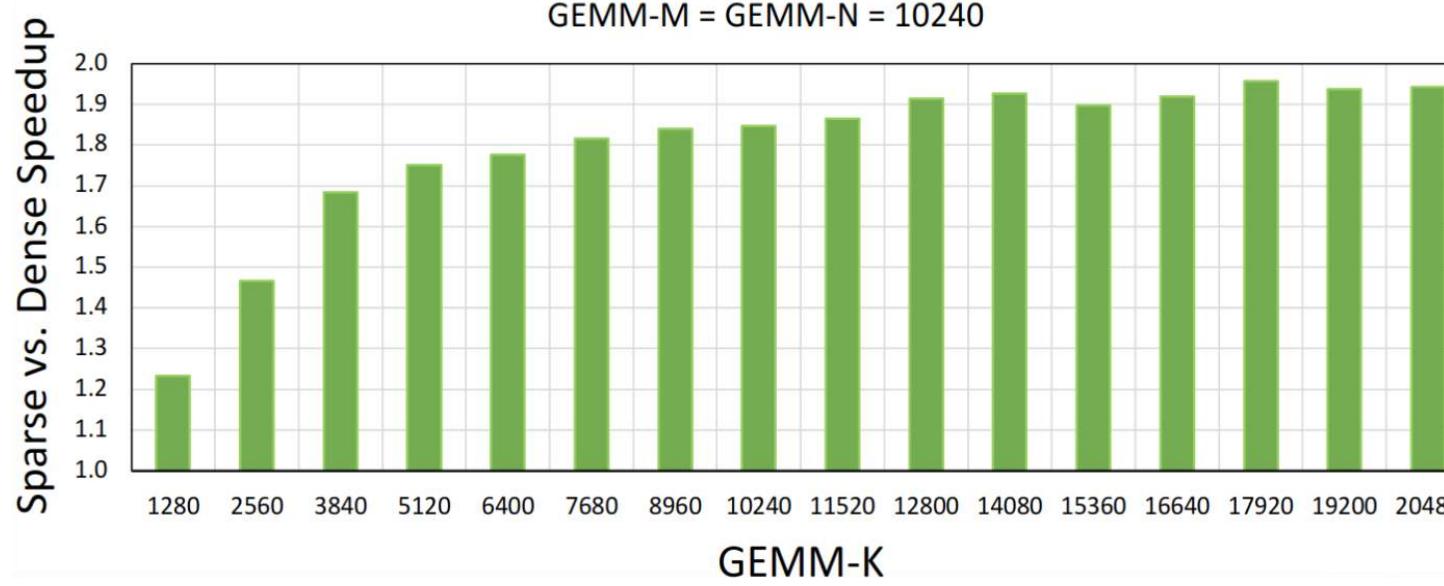


Типы прунинга

Im2col



Что дает на практике



Network	Accuracy		
	Dense FP16	Sparse FP16	Sparse INT8
ResNet-34	73.7	73.9	73.7
ResNet-50	76.1	76.2	76.2
ResNet-50 (WSL)	81.1	80.9	80.9
ResNet-101	77.7	78.0	77.9
ResNeXt-50-32x4	77.6	77.7	77.7
ResNeXt-101-32x16	79.7	79.9	79.9
ResNeXt-101-32x16 (WSL)	84.2	84.0	84.2
DenseNet-121	75.5	75.3	75.3
DenseNet-161	78.8	78.8	78.9
Wide ResNet-50	78.5	78.6	78.5
Wide ResNet-101	78.9	79.2	79.1
Inception v3	77.1	77.1	77.1
Xception	79.2	79.2	79.2
VGG-11	70.9	70.9	70.8
VGG-16	74.0	74.1	74.1
VGG-19	75.0	75.0	75.0
SUNet-128	75.6	76.0	75.4
SUNet-7-128	76.4	76.5	76.3
DRN26	75.2	75.3	75.3
DRN-105	79.4	79.5	79.4

«Оси» прунинга

Гибкость

→ Uniform / Non-Uniform

Структура

→ Structured / Unstructured

Масштаб

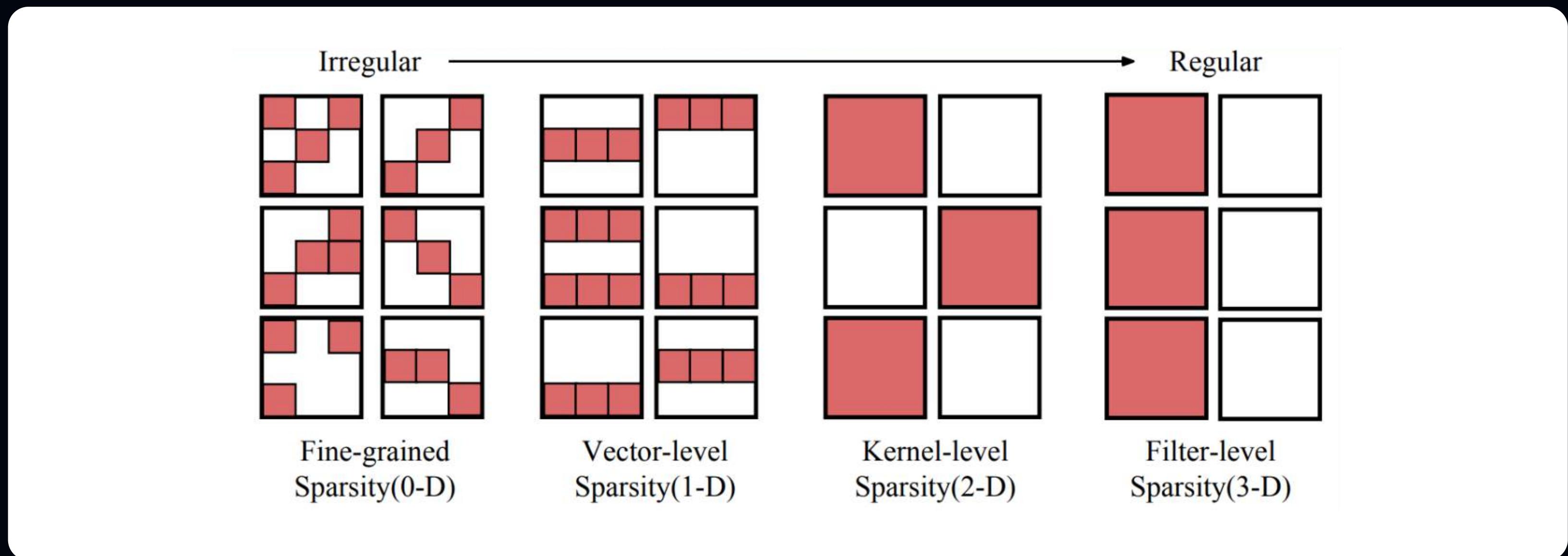
→ Слои / Attention heads / Filters / Weights
with constraint / Weights without
constraints

Тактика применения

→ One-shot / Iterative

Критерий

Масштаб



Exploring the Granularity of Sparsity in Convolutional Neural Networks [2]

Масштаб

Чем выше гранулярность,
тем сложнее затюнить сетку,
но тем выше прирост по скорости.

«Оси» прунинга

Гибкость

→ Uniform / Non-Uniform

Структура

→ Structured / Unstructured

Масштаб

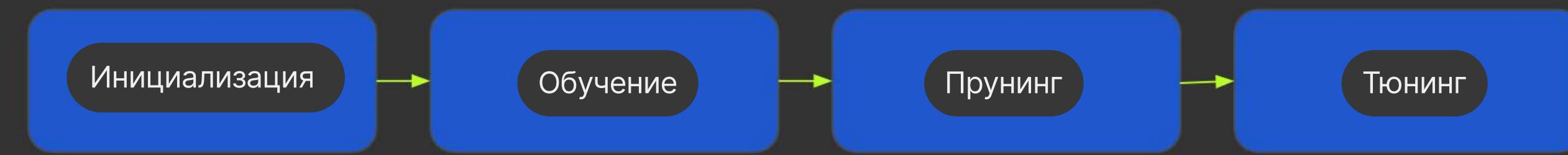
→ Слои / Attention heads / Filters / Weights
with constraint / Weights without
constraints

Тактика применения

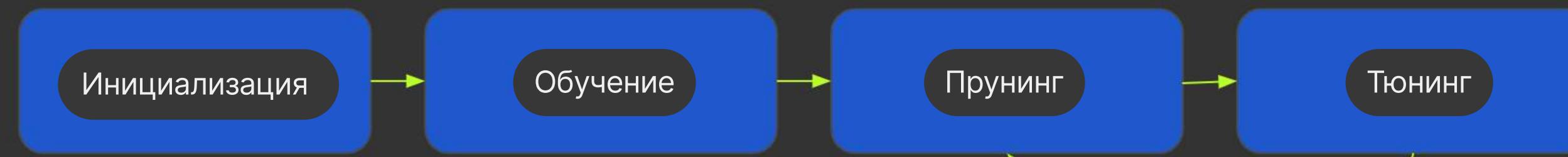
→ One-shot / Iterative

Критерий

Тактика применения прунинга

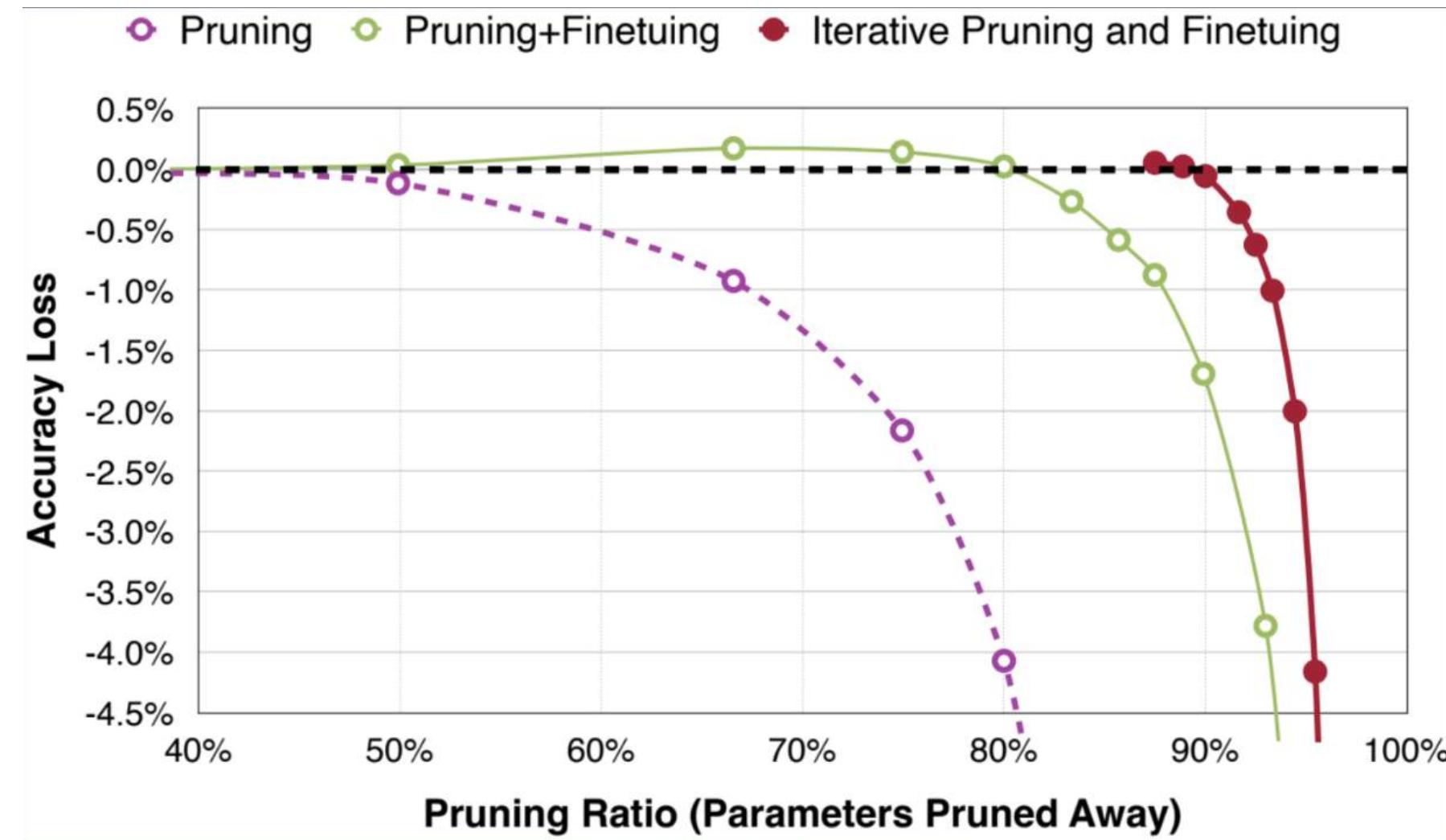


One shot



Iterative

А какая вообще разница?



Learning both Weights and Connections for Efficient Neural Networks^[4]

Тактика применения

One-Shot

→ Более быстро, но менее точно

Iterative

→ Более точно, но медленно.
Рекомендуется применять 2-5
итераций, не более

«Оси» прунинга

Гибкость

→ Uniform / Non-Uniform

Структура

→ Structured / Unstructured

Масштаб

→ Слои / Attention heads / Filters / Weights
with constraint / Weights without
constraints

Тактика применения

→ One-shot / Iterative

Критерий

Критерии прунинга

Data free

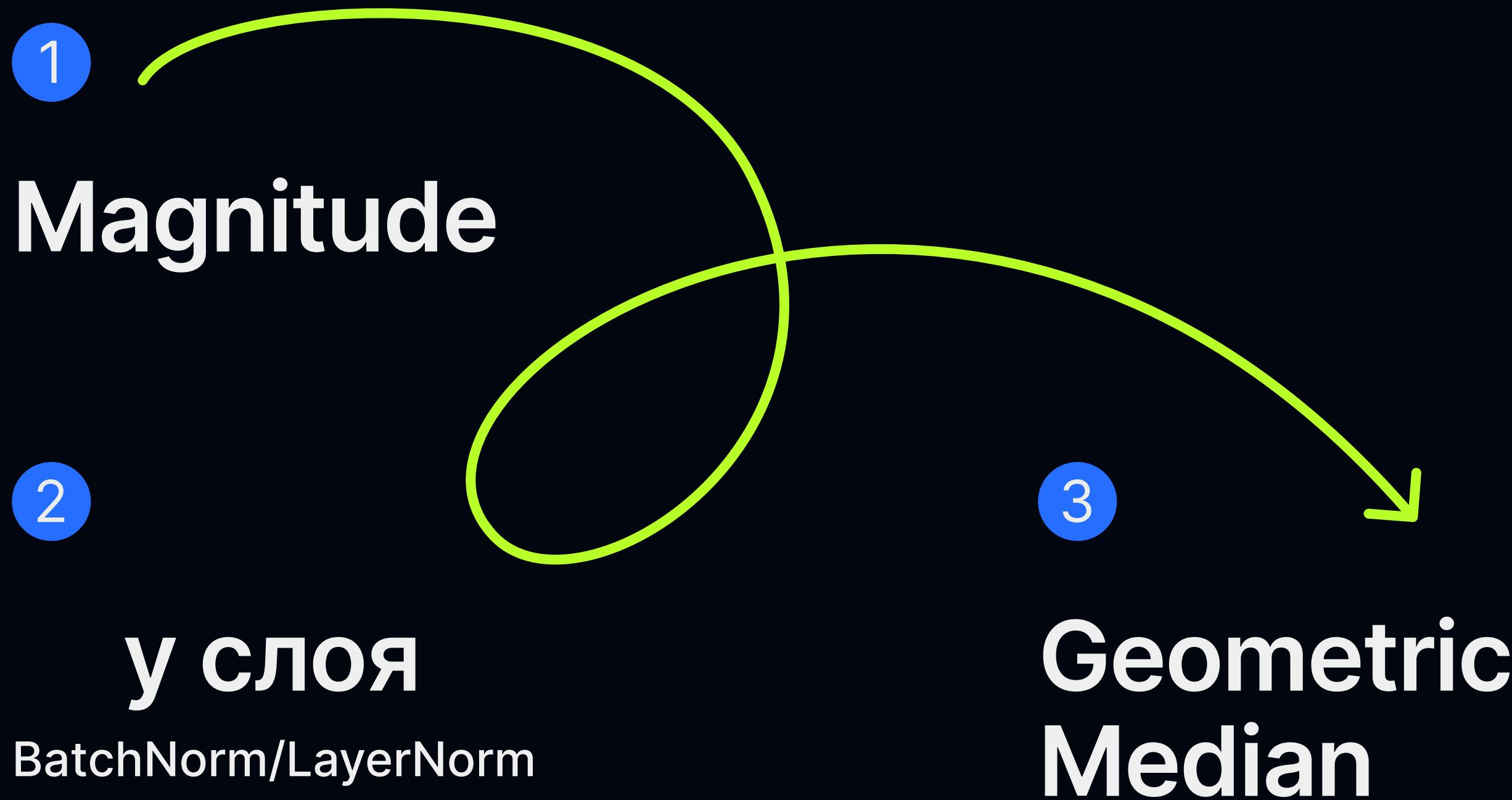
- ✓ Не нужно ничего вычислять во время прунинга
- ✓ Не нужна дополнительная память, хранится только модель
- ✗ Низкая точность

Data aware

- ✓ Более точные
- ✓ Могут учитывать контекст целевой задачи
- ✗ Необходима процедура вычисления лосса
- ✗ Требуют память для хранения/вычисления

Data Free

Критерии пруニングа



Magnitude pruning

Гипотеза

- чем меньше вес (фильтр), тем меньше он влияет на результат слоя

Работает

- в случае неструктурированного и структурированного пруинга

Критерий

- L_1/L_2 — норма фильтра / веса

$$W.\ shape = [InC, OutC, K, P]$$

$$L_1 : \min |w_{i,j,k,p}| \text{ wrt } i \in InC, j \in OutC, k \in K, p \in P$$

$$L_2 : \min w_{i,j,k,p}^2 \text{ wrt } i \in InC, j \in OutC, k \in K, p \in P$$

$$W.\ shape = [InC, OutC, K, P]$$

$$L_1 : \min \sum_{i,k,p} |w_{i,j,k,p}| \text{ wrt } j \in OutC$$

$$L_2 : \min \sqrt{\sum_{i,k,p} w_{i,j,k,p}^2} \text{ wrt } j \in OutC$$

Batch Norm scale [9]

Работает

→ Только в случае структурированного
прунинга — выбрасываем каналы целиком

Критерий

→ $\min \gamma_i$

$$BN(x_i) = \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} * \gamma_i + \beta_i$$

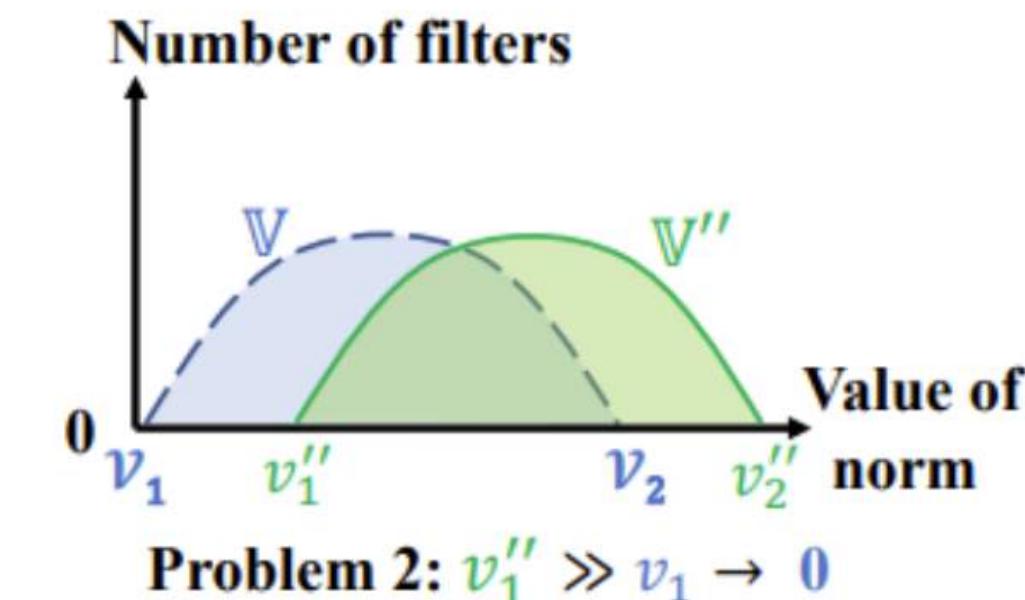
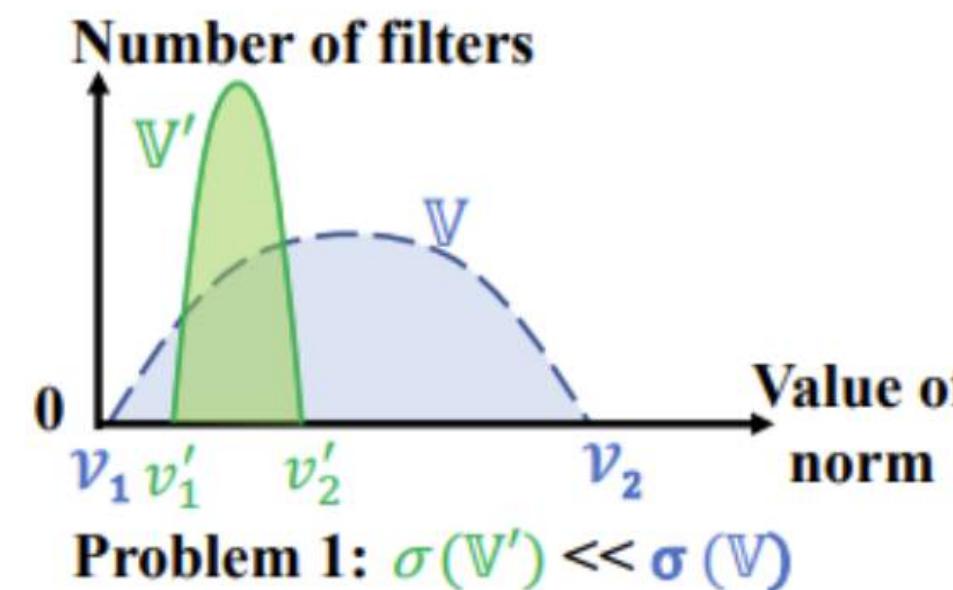
Geometric median [10]

Проблема 1

→ Маленькая дисперсия у «маленьких» фильтров → нельзя найти нормально порог отбрасывания

Проблема 2

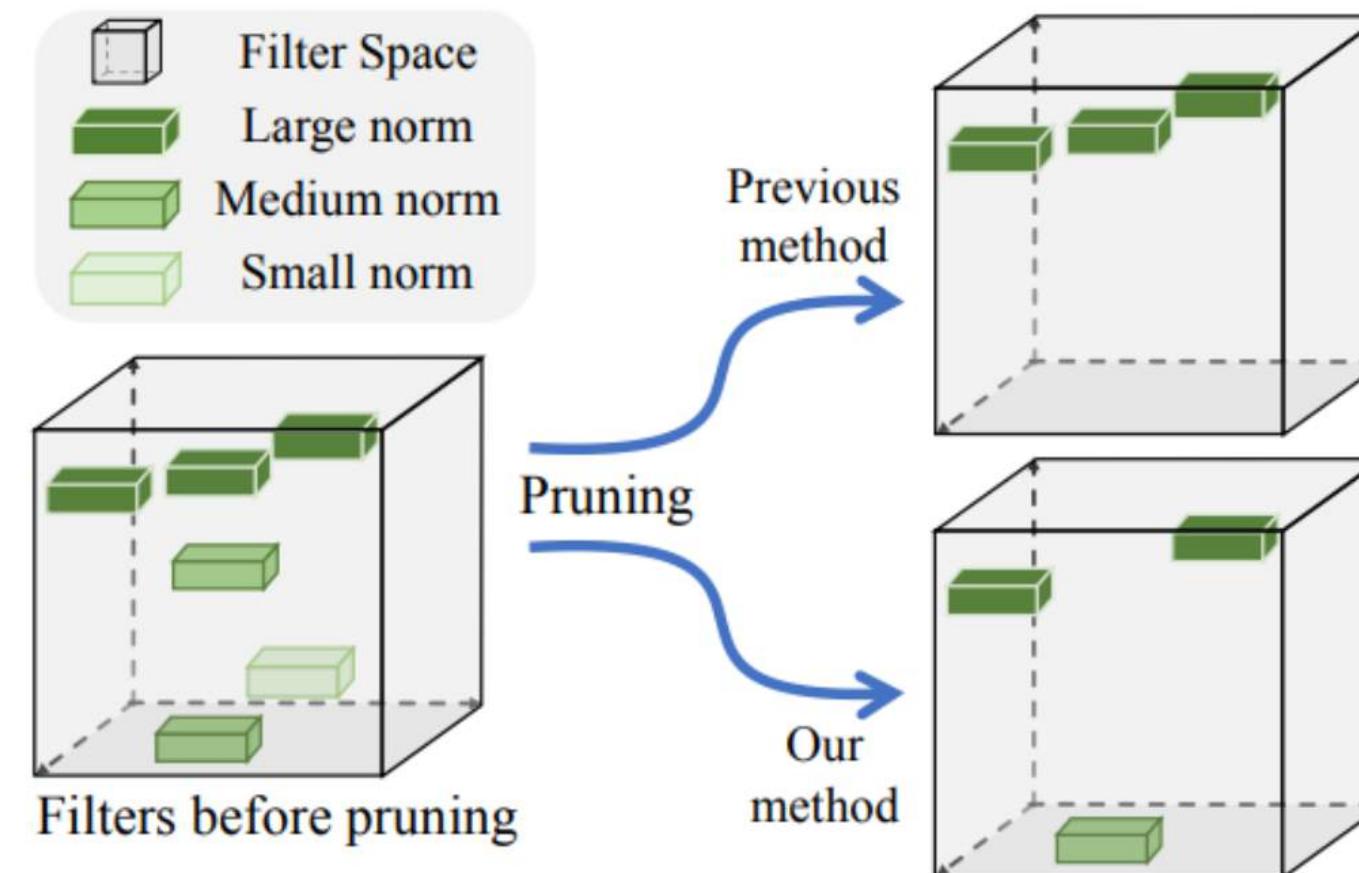
→ Маленькая норма не такая уж и маленькая



Geometric median

Интуиция

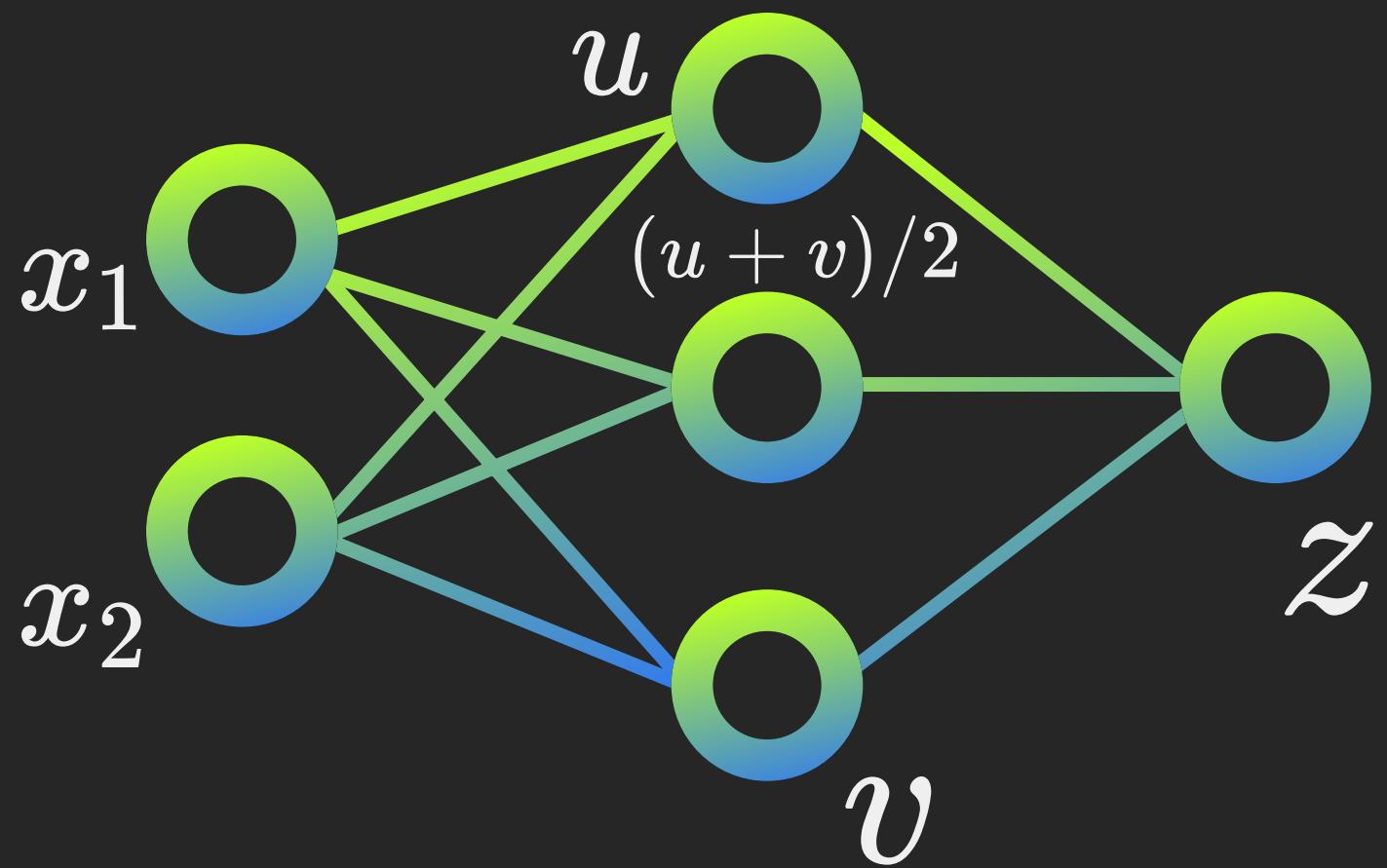
→ Давайте прунить те фильтры, которые ничего нового не приносят



Geometric median

Интуиция

→ Давайте прунить те фильтры, которые ничего нового не приносят



$$\begin{aligned}O_u &= x_1 * u_1 + x_2 * u_2 \\O_v &= x_1 * v_1 + x_2 * v_2 \\O_{\text{median}} &= 0.5[x_1 * [u_1 + v_1] + x_2 * [u_2 + v_2]]\end{aligned}$$

Geometric median

Интуиция

→ Давайте прунить те фильтры, которые ничего нового не приносят

$$O_u z_1 + O_{median} z_2 + O_v z_3 =$$

$$[x_1 * u_1 + x_2 * u_2]z_1 + [x_1 * v_1 + x_2 * v_2]z_3 + 0.5[x_1 * [u_1 + v_1] + x_2 * [u_2 + v_2]]z_2 =$$

Geometric median

Интуиция

→ Давайте прунить те фильтры, которые ничего нового не приносят

$$O_u z_1 + O_{median} z_2 + O_v z_3 =$$

$$[x_1 * u_1 + x_2 * u_2]z_1 + [x_1 * v_1 + x_2 * v_2]z_3 + 0.5[x_1 * [u_1 + v_1] + x_2 * [u_2 + v_2]]z_2 =$$

$$x_1 u_1 z_1 + x_2 u_2 z_1 + x_1 v_1 z_3 + x_2 v_2 z_3 + 0.5x_1 u_1 z_2 + 0.5x_1 v_1 z_2 + 0.5x_2 u_2 z_2 + 0.5x_2 v_2 z_2 =$$

Geometric median

Интуиция

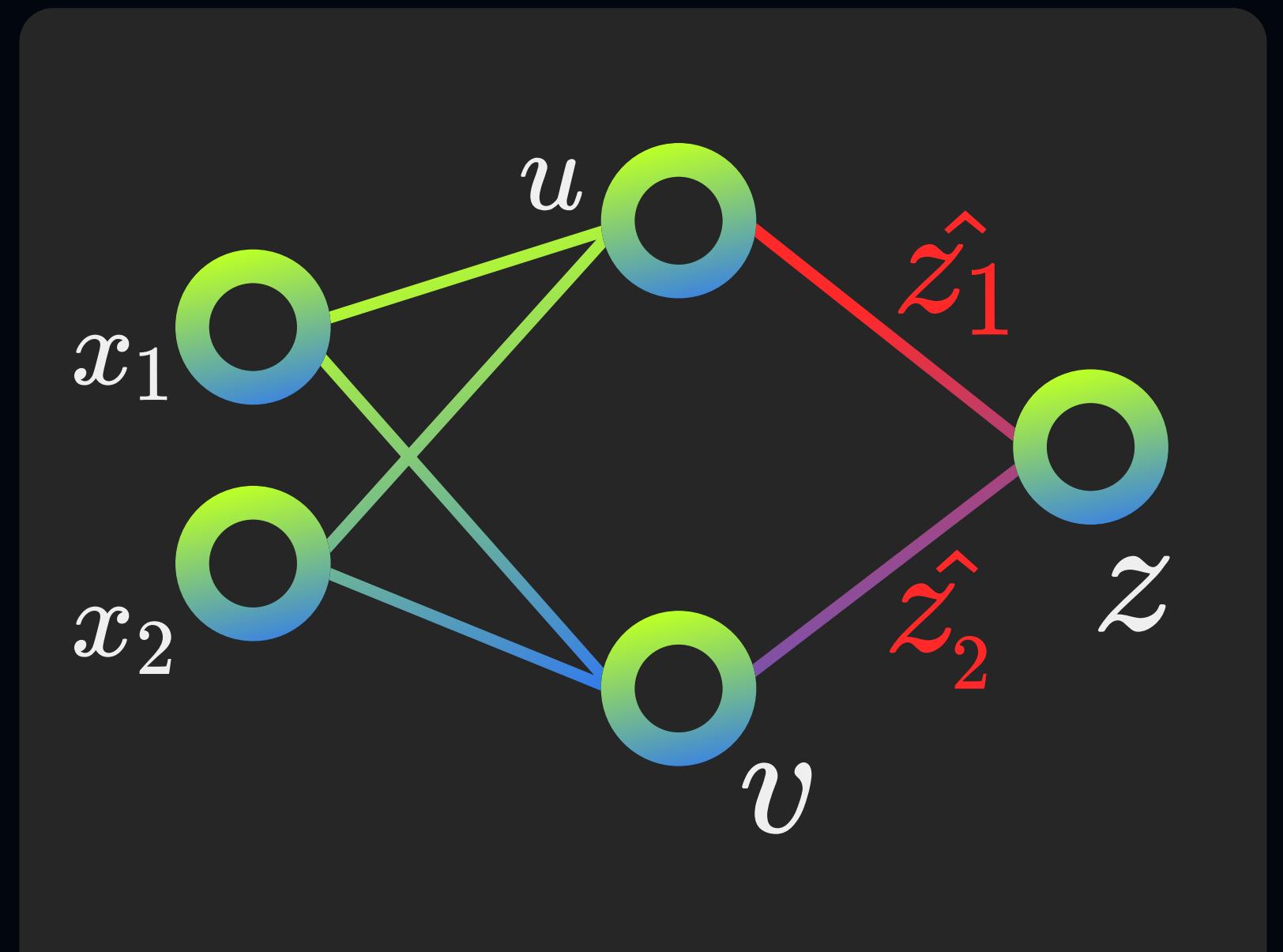
→ Давайте прунить те фильтры, которые ничего нового не приносят

$$\begin{aligned} O_u z_1 + O_{median} z_2 + O_v z_3 &= \\ [x_1 * u_1 + x_2 * u_2] z_1 + [x_1 * v_1 + x_2 * v_2] z_3 + 0.5[x_1 * [u_1 + v_1] &+ x_2 * [u_2 + v_2]] z_2 = \\ x_1 u_1 z_1 + x_2 u_2 z_1 + x_1 v_1 z_3 + x_2 v_2 z_3 + 0.5 x_1 u_1 z_2 + 0.5 x_1 v_1 z_2 + 0.5 x_2 u_2 z_2 &+ 0.5 x_2 v_2 z_2 = \\ = x_1 u_1 [z_1 + 0.5 z_2] + x_2 u_2 [z_1 + 0.5 z_2] + x_1 v_1 [z_3 + 0.5 z_2] + x_2 v_2 [z_3 + 0.5 z_2] &= \end{aligned}$$

Geometric median

Интуиция

→ Давайте прунить те фильтры, которые ничего нового не приносят



$$O_u z_1 + O_{\text{median}} z_2 + O_v z_3 = \\ x_1 u_1 \hat{z}_1 + x_2 u_2 \hat{z}_1 + x_1 v_1 \hat{z}_3 + x_2 v_2 \hat{z}_3$$

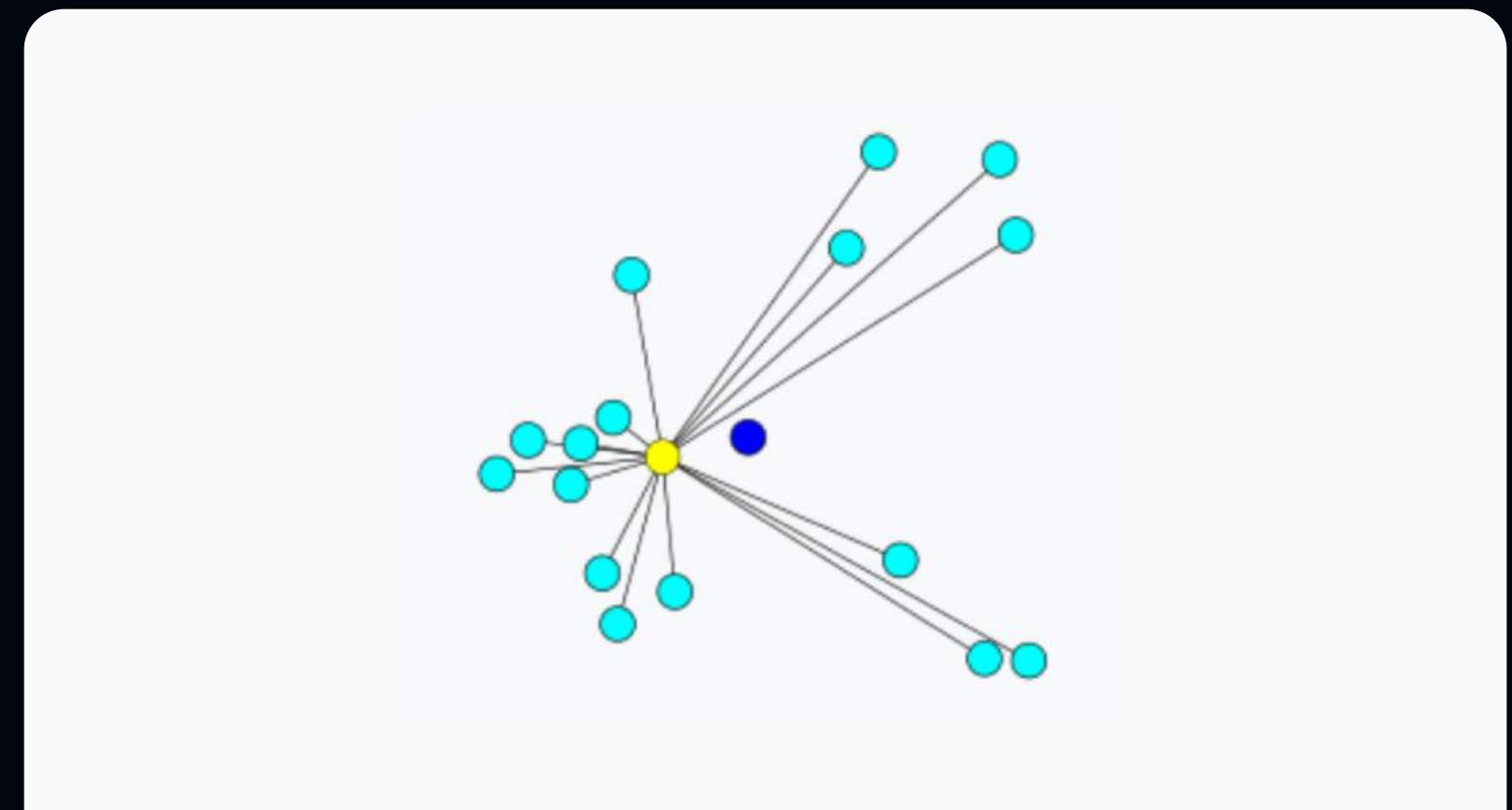
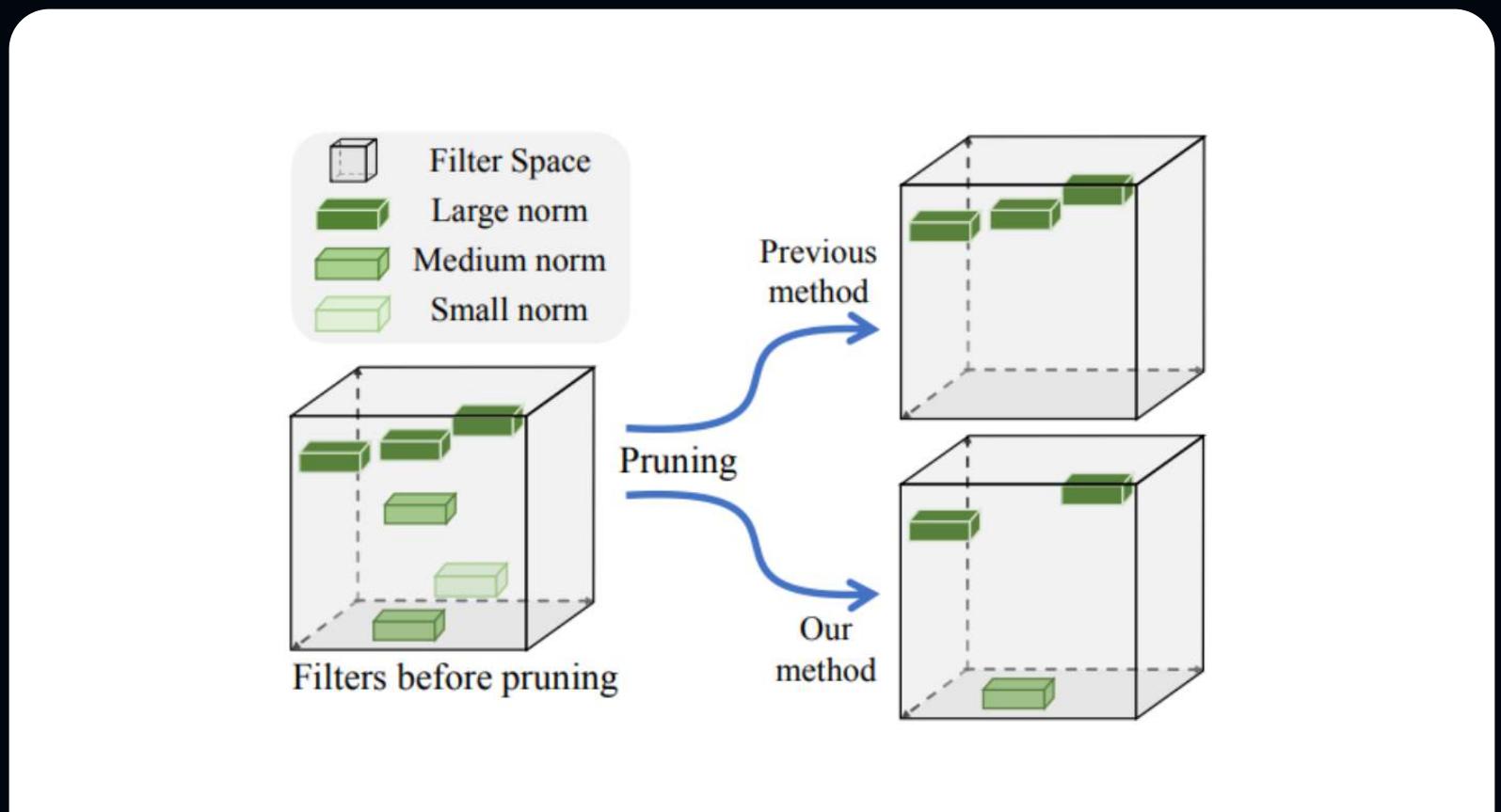
Geometric median

Интуиция

- Давайте прунить те фильтры, которые ничего нового не приносят

Критерий

- Geometric median — точка, расстояние до которой самое маленькое для каждой из других точек



Geometric median

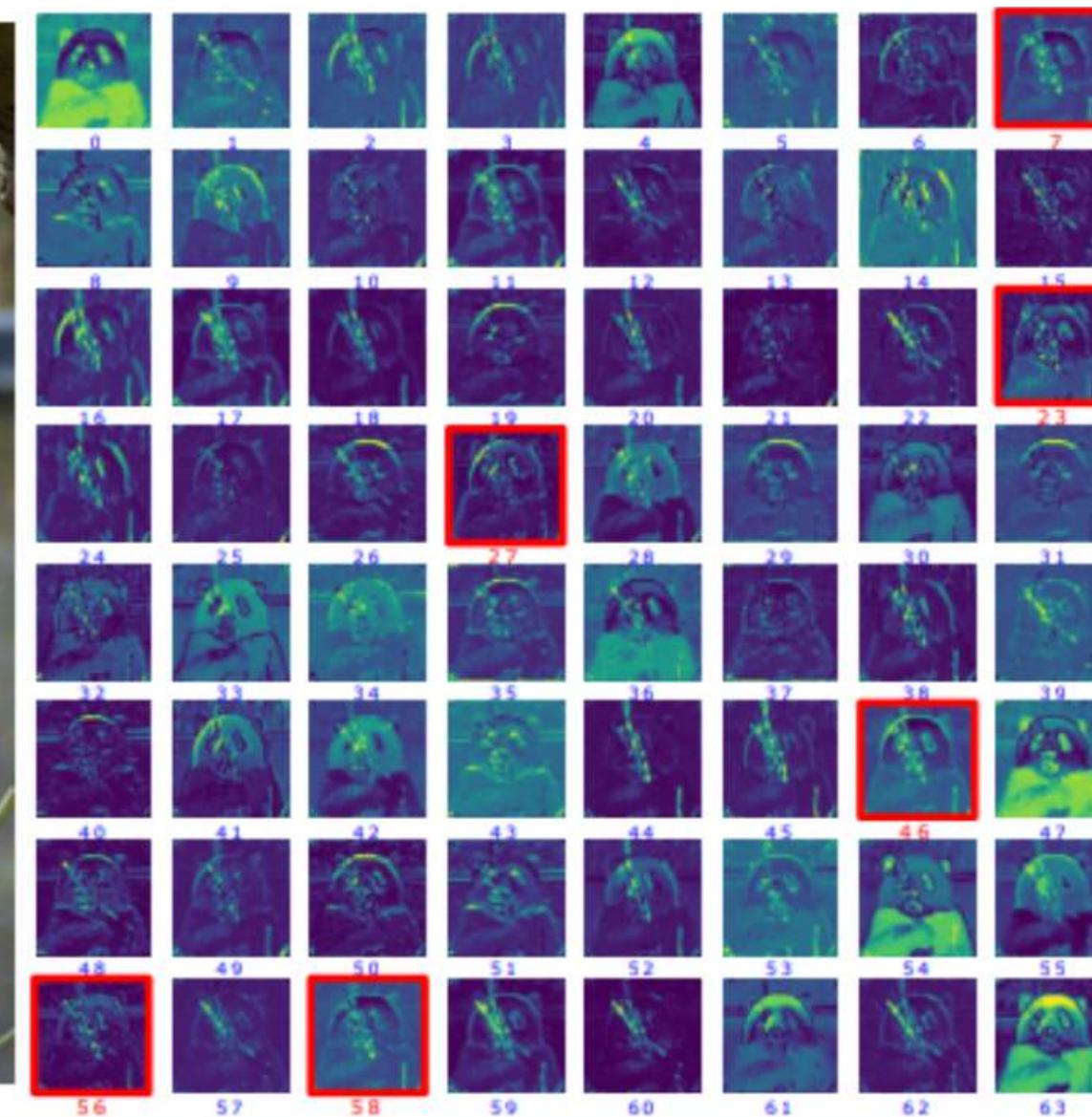
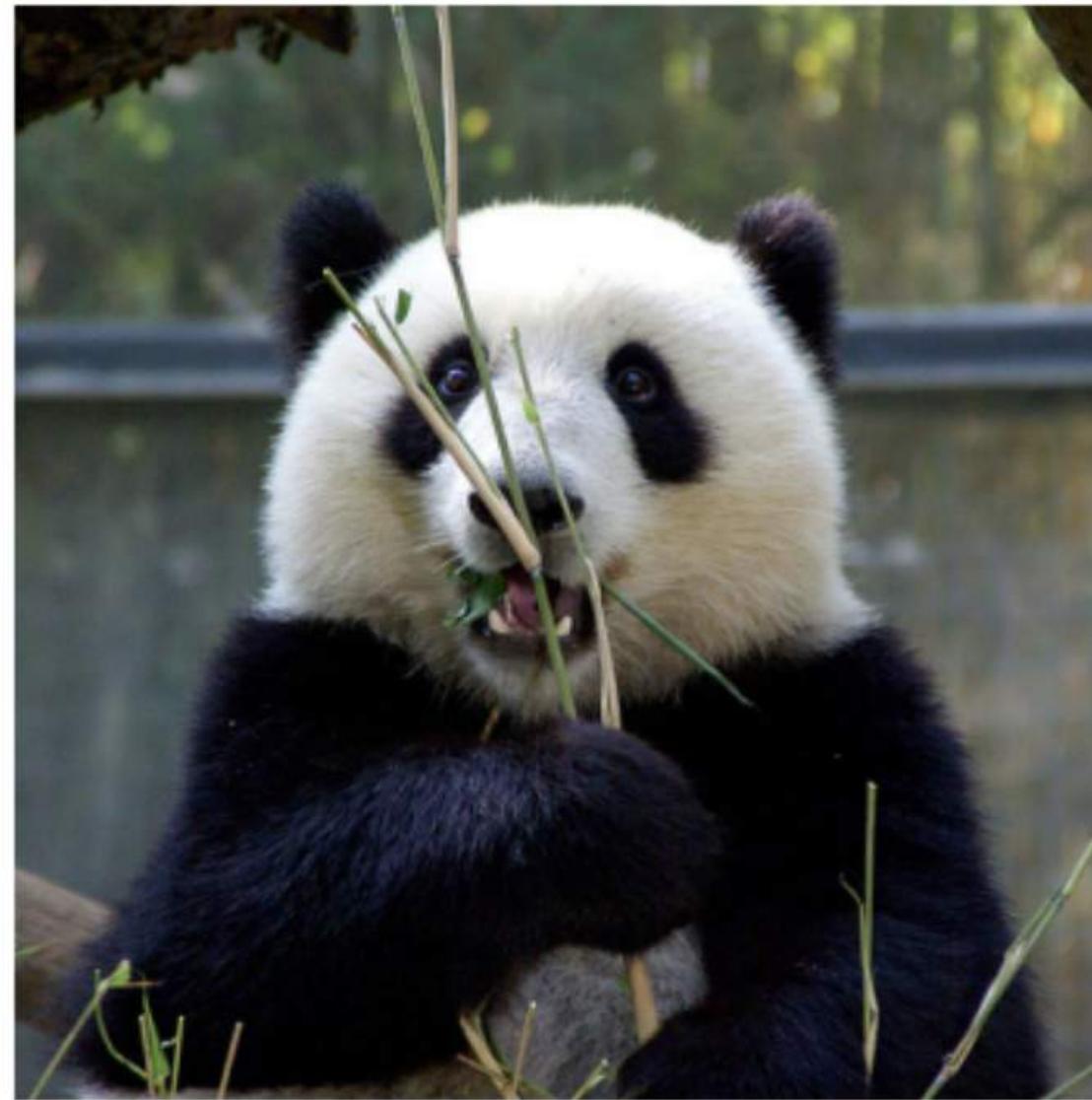
Критерий

→ geometric median — точка, расстояние до которой самое маленькое для каждой из других точек

$$\begin{aligned}\mathcal{F}_{i,x^*} &= \arg \min_x \sum_{j' \in [1, N_{i+1}]} \|x - \mathcal{F}_{i,j'}\|_2, \text{ s.t. } x \in \{\mathcal{F}_{i,1}, \dots, \mathcal{F}_{i,N_{i+1}}\} \\ &\stackrel{\text{def}}{=} \arg \min_x g(x), \text{ s.t. } x \in \{\mathcal{F}_{i,1}, \dots, \mathcal{F}_{i,N_{i+1}}\}\end{aligned}$$

Geometric median

→ Выкидываем красные боксы (очертания панды и бамбука), так как есть похожие признаки: очертания бамбука (22) и панды (4)



Data Aware

Основная идея

Рассматривать **нейронную сеть + лосс**
как обычную функцию и применить
к ней разложение Тейлора

Прунинг по разложению в ряд Тейлора

Основная идея

нейронная сеть + loss

это обычная функция многих переменных. Разложим ее в ряд Тейлора вблизи некоторого значения.

$$f(x) = \sum_{p=0}^P \frac{f^{(p)}(a)}{p!} (x - a)^p + R_p(x)$$

Пруниг по разложению в ряд Тейлора

Важно:

в качестве, переменной, относительно которой раскладываем в ряд Тейлора - это веса

переписали формулу Тейлора →

$$L(w) = \sum_{p=0}^P \frac{L^{(p)}(w_0)}{p!} (w - w_0)^p + R_p(w)$$

Разложили до 1 порядков и
посмотрели значение в точке 0 →

$$L(w=0) = L(w_0) - w_0 \frac{\partial L}{\partial w}(w_0) + R_p(w)$$

Модуль берем, потому что нам важно,
чтобы loss не поменялся →

$$\min |L(w=0) - L(w_0)| = \min \left| w_0 \frac{\partial L}{\partial w}(w_0) \right|$$

Чем хорош данный критерий?

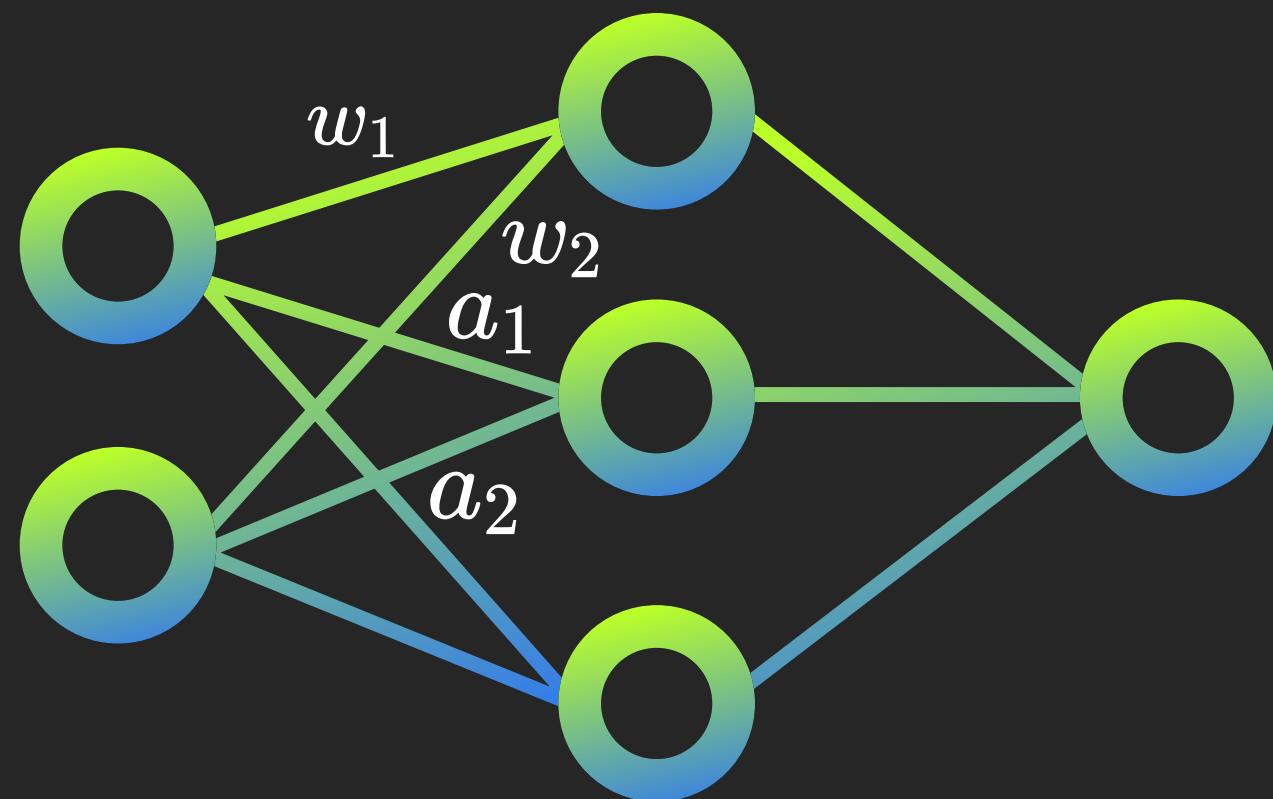
- ① Теоретически обоснован
(ну в каком-то виде)
- ② Глобальный, то есть можно сравнивать веса между собой на любом слое
- ③ Часто встречается, выведен независимо в нескольких работах

$$\text{mins}_w = \min \left| w \frac{\partial L}{\partial w} \right|$$

Модификация для структурированного пруинга

Структурированный пруинг — это когда выбрасываем веса / фильтры целиком

Критерий: сумма индивидуальных критериев всех весов



$$S_W = \left| w_1 \frac{\partial L}{\partial w_1} \right| + \left| w_2 \frac{\partial L}{\partial w_2} \right|$$

$$S_A = \left| a_1 \frac{\partial L}{\partial a_1} \right| + \left| a_2 \frac{\partial L}{\partial a_2} \right|$$

S_W vs S_A

Модификация для структурированного пруинга

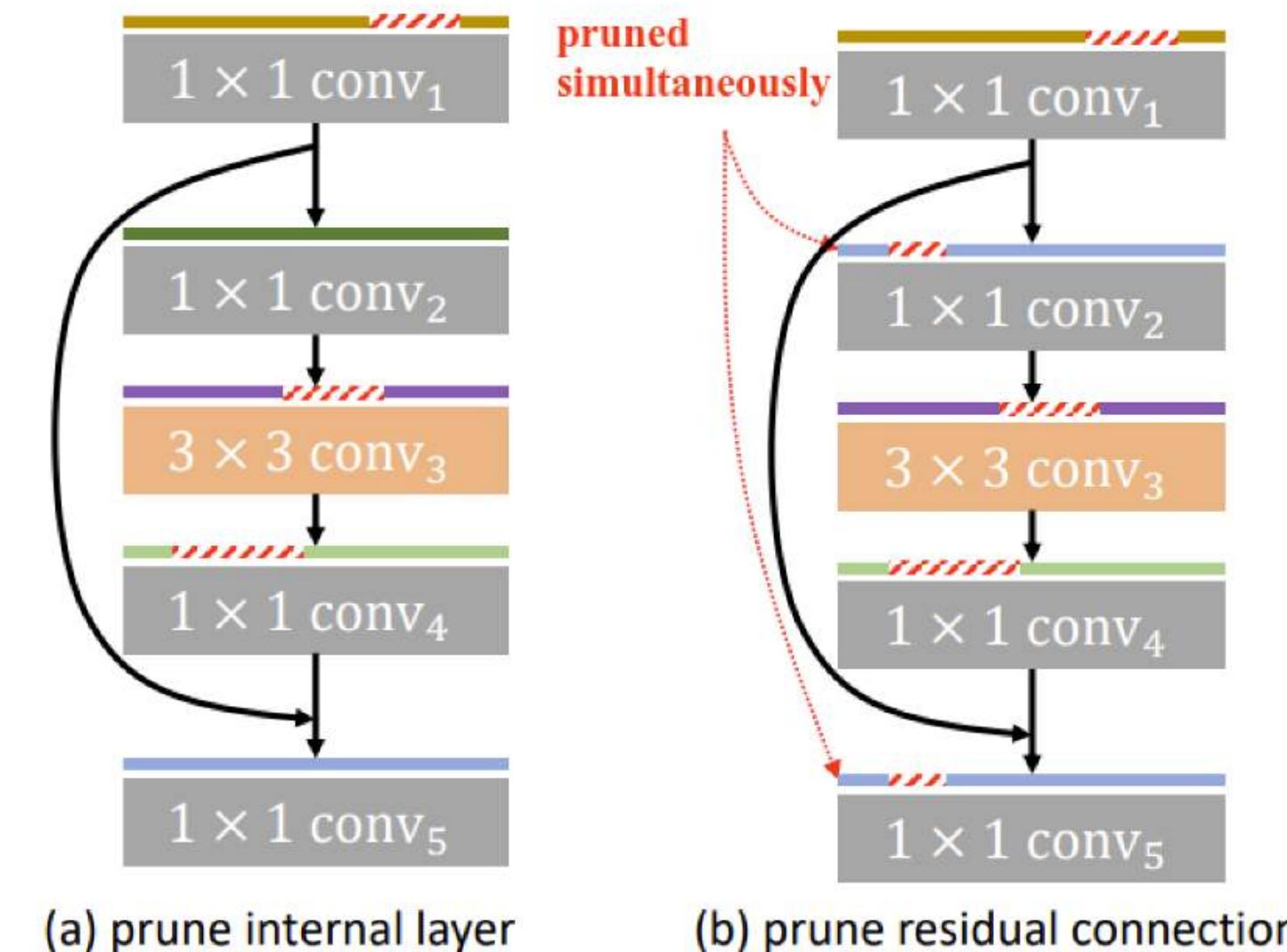
Структурированный пруинг — это когда выбрасываем веса / фильтры целиком

$$\min s_W = \min \left| \sum_{w_j \in W} w_j \frac{\partial L}{\partial w}(w_j) \right|$$

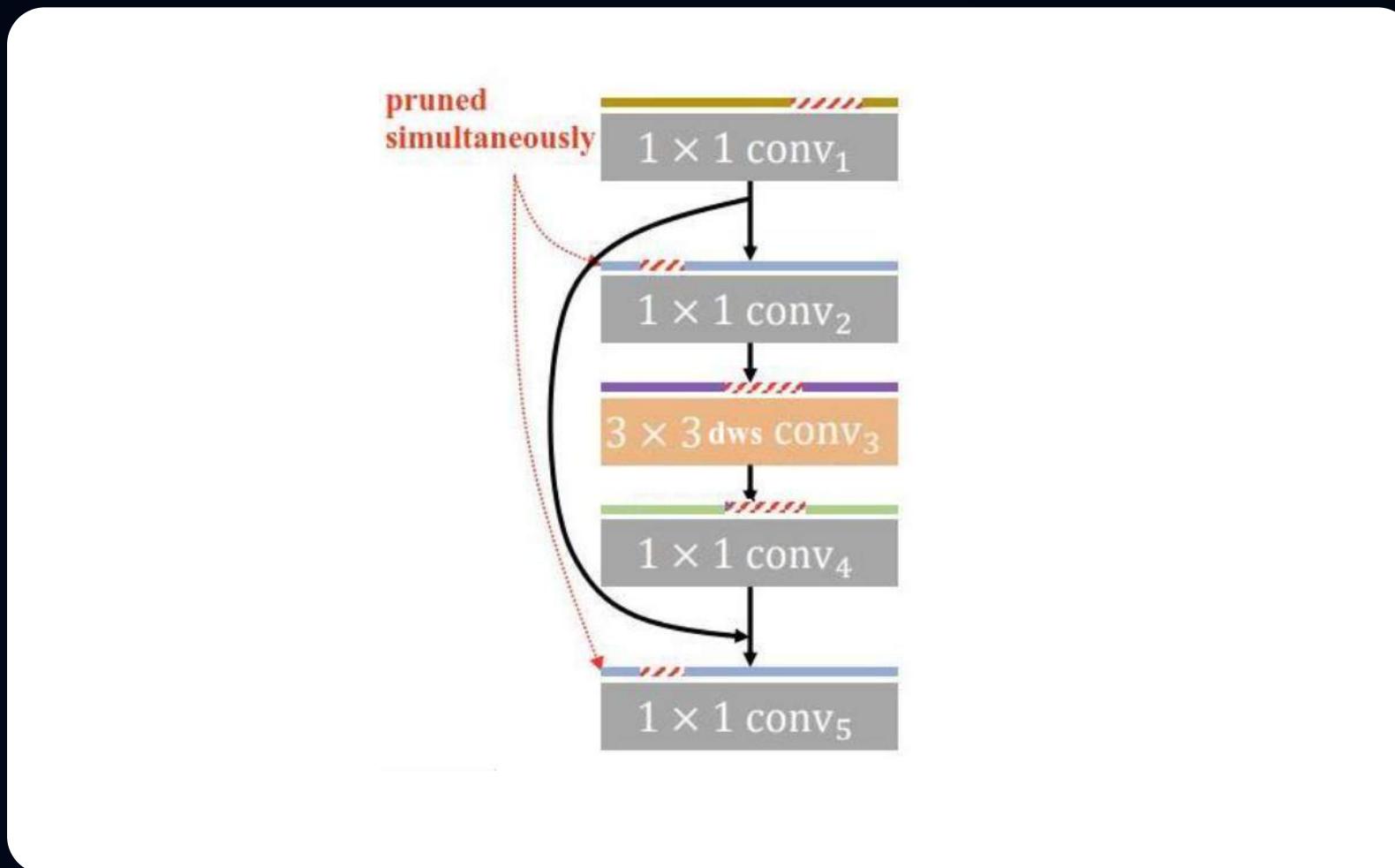
$$\min s_W = \min \sum_{w_j \in W} \left[w_j \frac{\partial L}{\partial w}(w_j) \right]^2$$

$$\min s_W = \min \sum_{w_j \in W} \left| w_j \frac{\partial L}{\partial w}(w_j) \right|$$

Как обрабатывать skip connection



Как обрабатывать skip connection



$$\begin{aligned}W_{\text{conv } 1} &= [C_1, \text{InC}, 1, 1] \\W_{\text{conv } 2} &= [\text{InC}, \text{OutC}, 1, 1] \\W_{\text{dws_conv } 3} &= [\text{OutC}, 1, 3, 3] \\W_{\text{conv } 4} &= [\text{OutC}, \text{InC}, 1, 1] \\W_{\text{conv } 5} &= [\text{InC}, C_2, 1, 1]\end{aligned}$$

$$\text{baseline_macs} = C_1 * \text{InC} + \text{InC} * \text{OutC} + 9 * \text{OutC} + \text{OutC} * \text{InC} + \text{InC} * C_2$$

Почему это важно?

Пруним **внутренние каналы**

$$\frac{\frac{C * InC + InC * (OutC-1) + 9 * (OutC-1) + (OutC-1) * InC + InC * C2}{baseline_macs}}{\frac{9 + 2 * InC}{baseline_macs}} = \frac{\frac{baseline_macs - 9 - InC * 2}{baseline_macs}}{\frac{9 + 2 * InC}{baseline_macs}} = 1 -$$

Пруним **skip connection**

$$\frac{\frac{C * (InC - 1) + (InC - 1) * OutC + 9 * OutC + (InC - 1) * OutC + (InC - 1) * C2}{baseline_macs}}{\frac{C + 2 * OutC + C2}{baseline_macs}} = [OutC = 6ImC] = 1 - \frac{\frac{baseline_macs - C - 2 * OutC - C2}{baseline_macs}}{\frac{C + 12ImC + C2}{baseline_macs}} =$$

Hessian pruning

Разложим до 2 порядка

$$\Delta L = L(w + \Delta w) - L(w) = g^T \Delta w + \frac{1}{2} \Delta w^T H \Delta w + O(\|\Delta w\|^3)$$

Hessian pruning

Гипотеза

→ Сеть обучена, находится в оптимуме,
выбросим первые порядки

$$\Delta L = L(w + \Delta w) - L(w) = g^T \cancel{\Delta w} + \frac{1}{2} \Delta w^T H \Delta w + O(\|\Delta w\|^3)$$

Hessian pruning

Надо посчитать гессиан. Как?

1 Гипотеза

веса независимые → гессиан
диагональный

2 Гипотеза

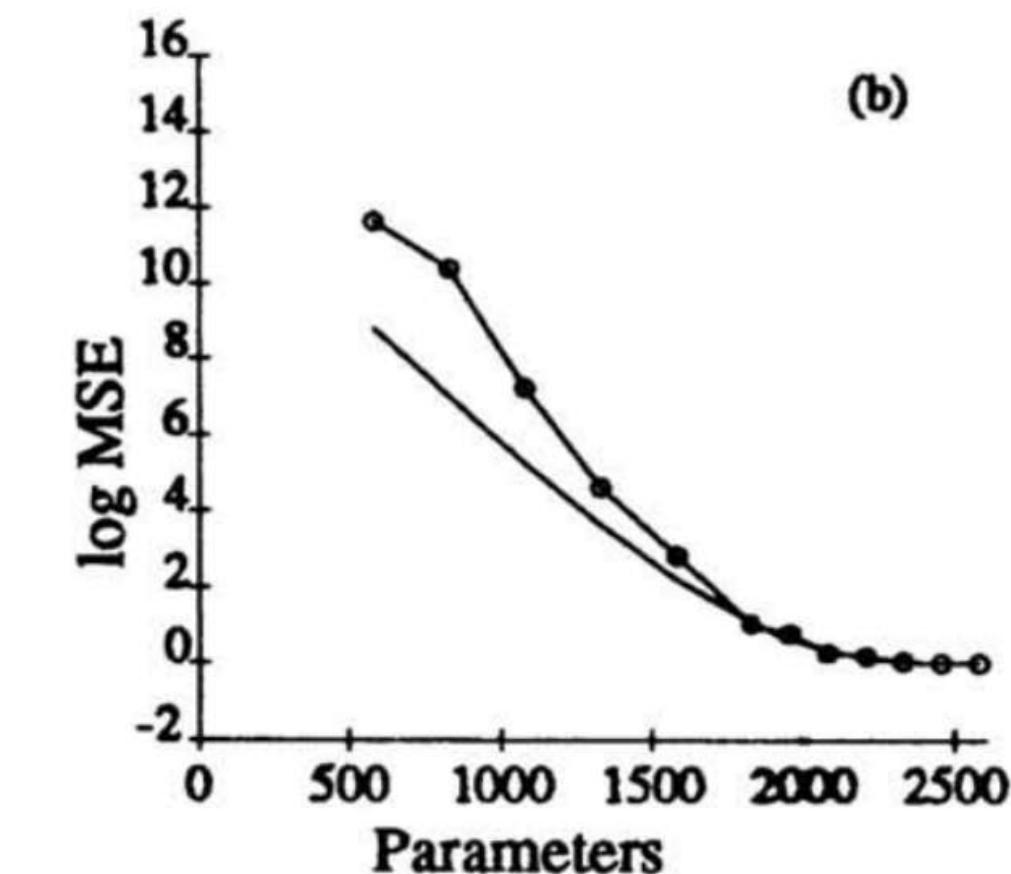
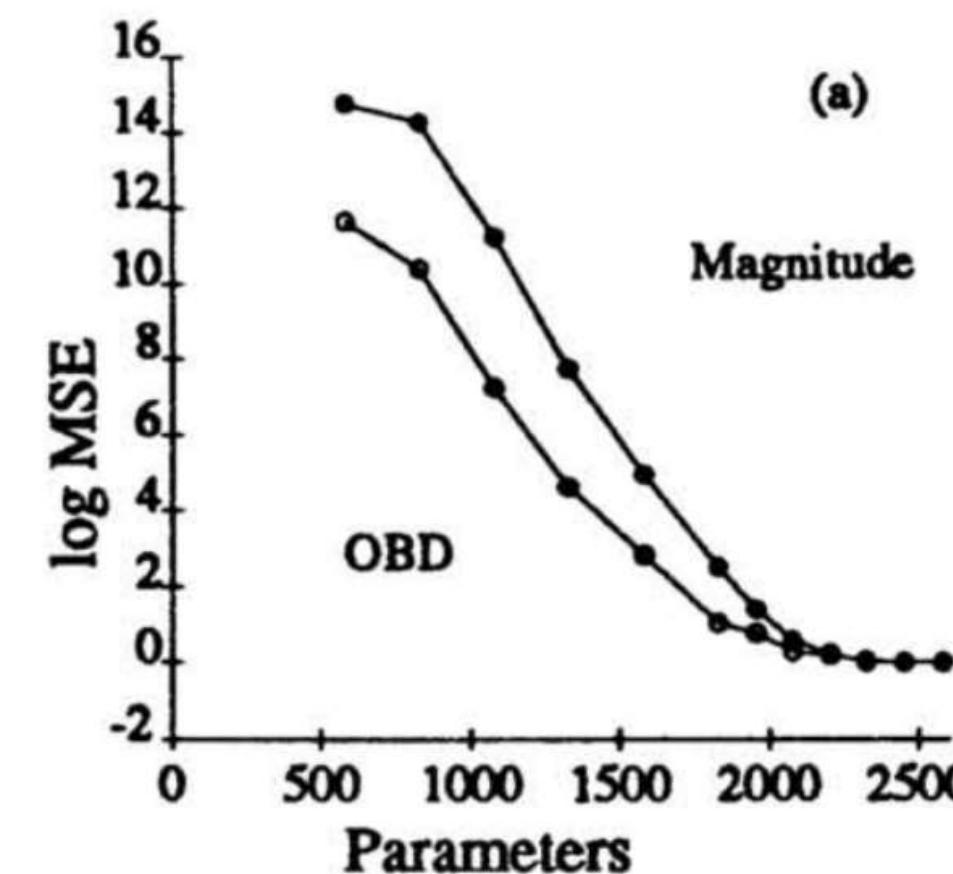
Веса внутри слоя не являются
независимыми, однако, на разных
слоях являются независимыми →
гессиан блочно-диагональный

Диагональный гессиан

OBD [12]

1. Сделал Ян Ле Кун
2. Одна из первых работ по прунингу
3. Проверяли на MNIST

$$\delta E = \frac{1}{2} \sum_i h_{ii} \delta u_i^2$$



Optimal Brain Damage

Yann Le Cun, John S. Denker and Sara A. Solla
AT&T Bell Laboratories, Holmdel, N. J. 07733

Недиагональный гессиан

Критерии прунинга

1

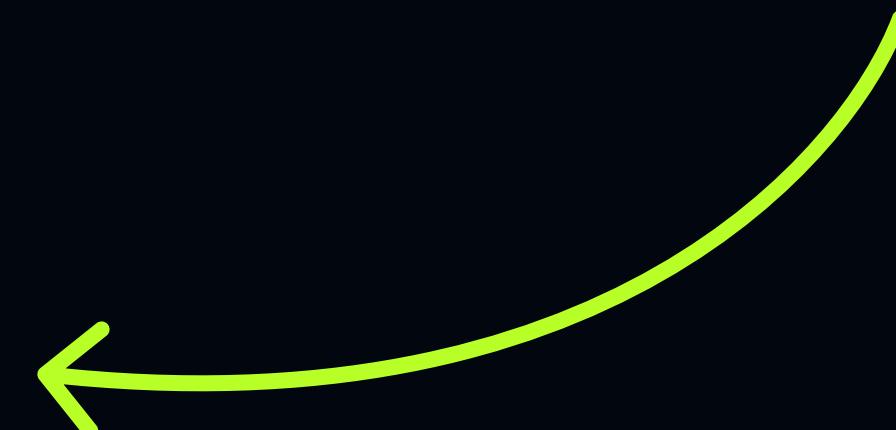
Выбрасываем всего лишь
один вес + MSE-loss

2

Fisher approximation

3

K-FAC



Optimal Brain Surgeon [13]

1

Сделано в прошлом веке

2

Поставили задачу удаления одного веса в более общем виде, чем Ле Кун

3

Нашли не только формулу минимального веса, но еще способ «пересчитывать» оставшиеся веса так, чтобы δL была ЕЩЕ меньше

4

Гессиан необязательно диагональный

4

В оригинальной работе все сделали для MSE \rightarrow гессиан превращается в ковариационную матрицу $X^T X$

$$\min_q \left\{ \min_{\delta w} \left(\frac{1}{2} \delta w^T \cdot H \cdot \delta w \right) \mid e_q^T \cdot \delta w + w_q = 0 \right\}$$

$$\delta w = - \frac{u_q}{[H^{-1}]_{qq}} H^{-1} \cdot e_q$$
$$L_q = \frac{1}{2} \frac{u_q^2}{[H^{-1}]_{qq}}$$

Аппроксимация Фишера

WoodFisher^[14]

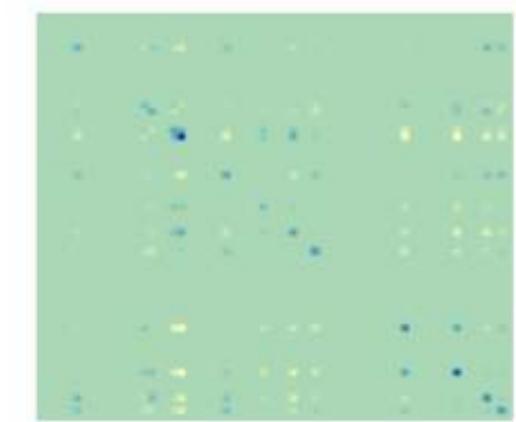
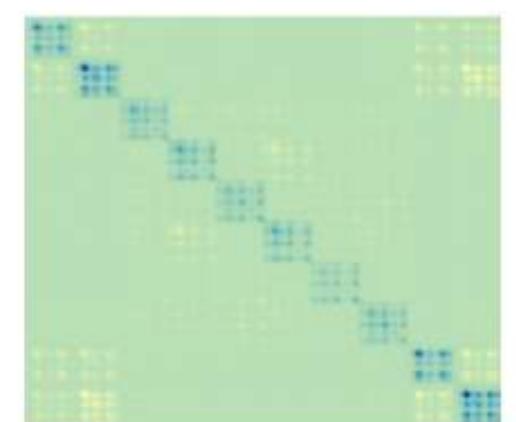
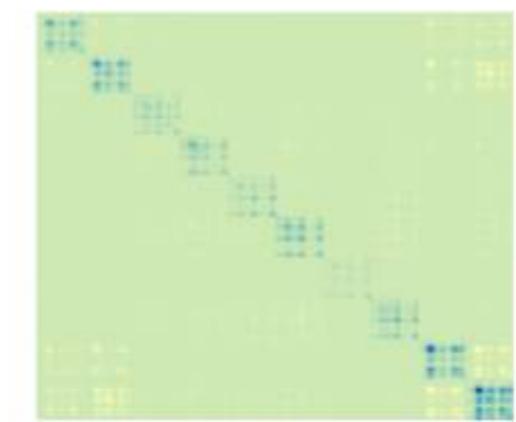
→ Вычисление гессиана можно заменить
на вычисление матрицы Фишера
Fisher Information Matrix, FIM

$$L(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \ell(\mathbf{y}_n, f(\mathbf{x}_n; \mathbf{w}))$$

$$\hat{F}^{(a)} \frac{1}{N} \sum_{n=1}^N \underbrace{\nabla \ell(\mathbf{y}_n, f(\mathbf{x}_n; \mathbf{w}))}_{\nabla \ell_n} \nabla \ell(\mathbf{y}_n, f(\mathbf{x}_n; w))^\top$$

Аппроксимация Фишера

→ Визуально очень
похожи

(a) $\mathbf{H}_{2,2}$ (b) $\widehat{\mathcal{F}}_{2,2}$ (c) $\mathbf{H}_{3,3}$ (d) $\widehat{\mathcal{F}}_{3,3}$

Вычисление \hat{F}^{-1}

→ Нам нужен H^{-1} (а точнее, F^{-1}). Как найти? Воспользоваться формулой Woodbury Matrix Identity

$$\hat{F}_{n+1} = \hat{F}_n + \frac{1}{N} \nabla \ell_{n+1} \nabla \ell_{n+1}^\top, \quad \text{where } \hat{F}_0 = \lambda I_d.$$

Notes the *dampening* term, i.e., a positive scalar λ times the identity I_d to make \hat{F}_n positive definite. Then, the recurrence for calculating the inverse of empirical Hessian is

$$\hat{F}_{n+1}^{-1} = \hat{F}_n^{-1} - \frac{\hat{F}_n^{-1} \nabla \ell_{n+1} \nabla \ell_{n+1}^\top \hat{F}_n^{-1}}{N + \nabla \ell_{n+1}^\top \hat{F}_n^{-1} \nabla \ell_{n+1}}, \quad \text{where } \hat{F}_0^{-1} = \lambda^{-1} I_d.$$

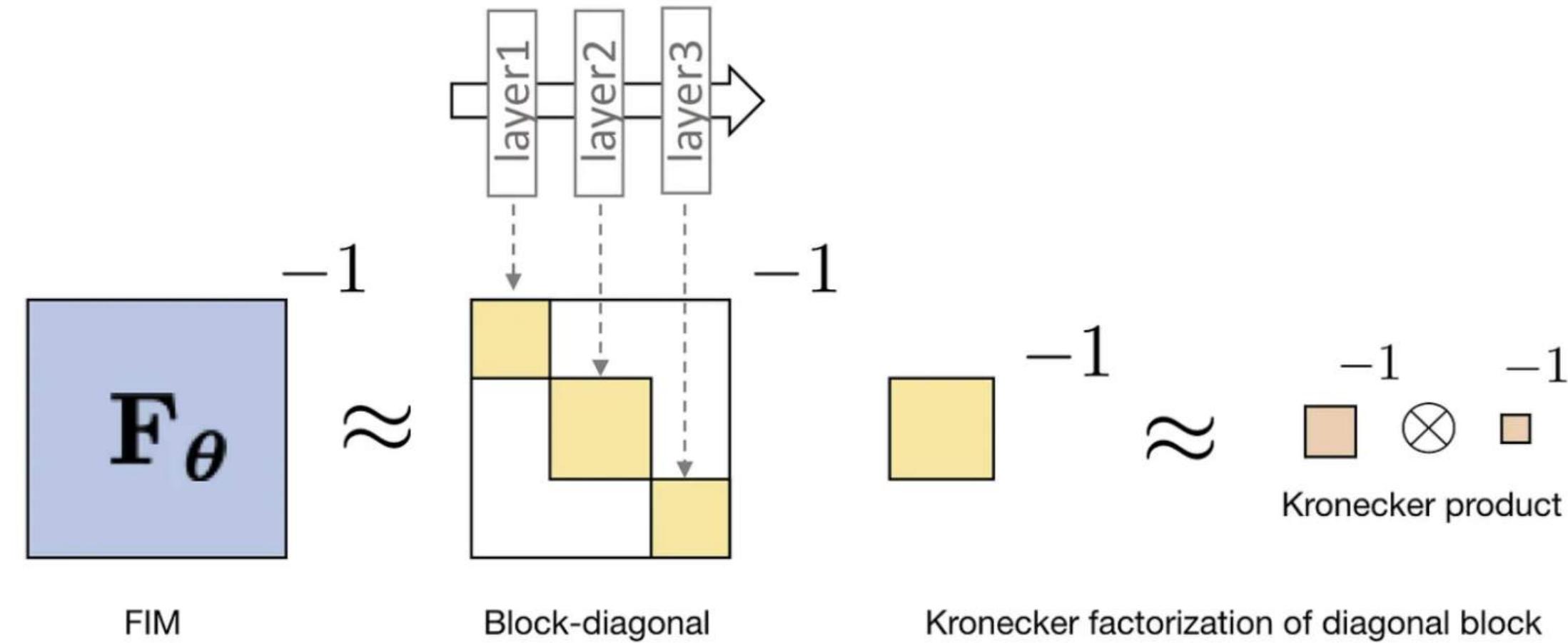
WoodFisher

WoodFisher = OBS, только добавили
оценку гессиана для произвольной
функции потерь

K-FAC [15]

Основная идея

Даже если представить себе, что FIM блочно диагональный, то блоки все равно большие. Как быть? Апроксимируем блоки.



K-FAC

$$\mathbf{A} \otimes \mathbf{B} := \begin{pmatrix} [\mathbf{A}]_{1,1} \mathbf{B} & \cdots & [\mathbf{A}]_{1,n} \mathbf{B} \\ \vdots & \ddots & \vdots \\ [\mathbf{A}]_{m,1} \mathbf{B} & \cdots & [\mathbf{A}]_{m,n} \mathbf{B} \end{pmatrix} \in \mathbb{R}^{ma \times nb}$$

$\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{a \times b}$: Kronecker factors

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$$

K-FAC

Рассмотрим линейный слой

$$\rightarrow \quad y = Wx$$

Обозначим

$$\rightarrow \quad \nabla_W \mathcal{L} \text{ производную по весу}$$

$$\rightarrow \quad \nabla_y \mathcal{L} \text{ производную по выходным активациям}$$

Допустим, что x и $\nabla_y \mathcal{L}$ статистически независимы, тогда:

$$\begin{aligned} \mathbf{F} &= \mathbb{E}\left[(\nabla_W \mathcal{L})(\nabla_W \mathcal{L})^\top\right] = \mathbb{E}\left[\left(\text{vec}(\nabla_y \mathcal{L})x^\top\right)\left(\text{vec}(\nabla_y \mathcal{L})x^\top\right)^\top\right] = \\ &= \mathbb{E}\left[\left(x \otimes \nabla_y \mathcal{L}^\top\right)\left(x \otimes \nabla_y \mathcal{L}^\top\right)^\top\right] = \mathbb{E}\left[\left(xx^\top\right) \otimes (\nabla_y \mathcal{L} \nabla_y \mathcal{L}^\top)\right] = \\ &= \mathbb{E}\left[\left(xx^\top\right)\right] \otimes \mathbb{E}\left[\left(\nabla_y \mathcal{L} \nabla_y \mathcal{L}^\top\right)\right] = X \otimes G \end{aligned}$$

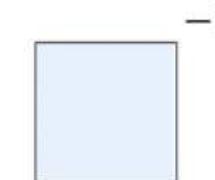
K-FAC

All layers in AlexNet

60,000,000 parameters

Fisher information matrix

$$\mathbf{F}_{\theta} \in \mathbb{R}^{60,000,000 \times 60,000,000}$$

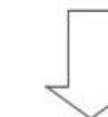


Final layer of AlexNet

Input dimension: 4,096

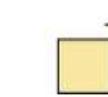
Output dimension: 1,000

4,096,000 parameters



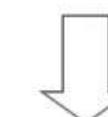
Fisher block

$$\mathbf{F}_i \in \mathbb{R}^{4,096,000 \times 4,096,000}$$

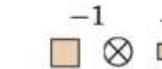


Kronecker factors

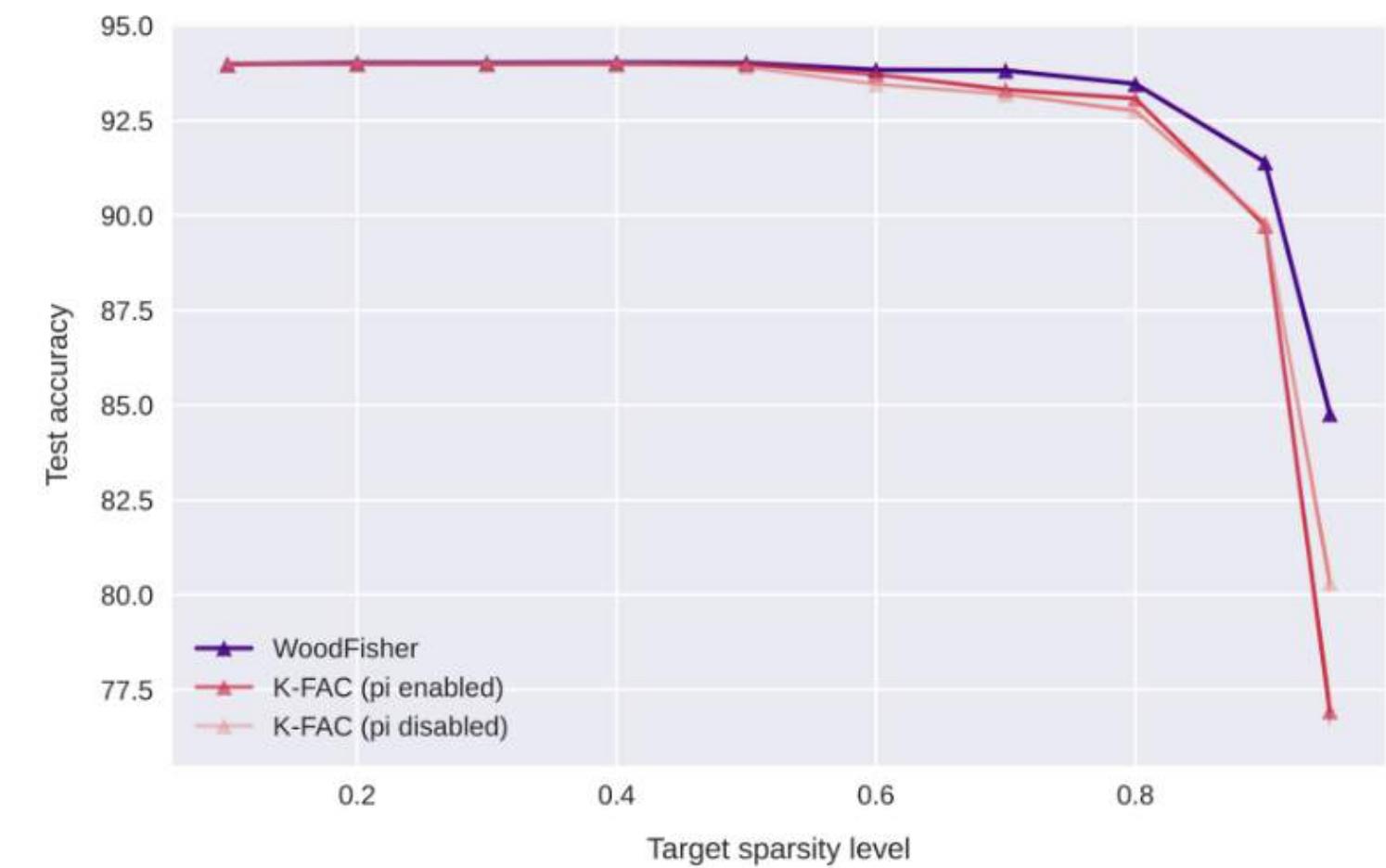
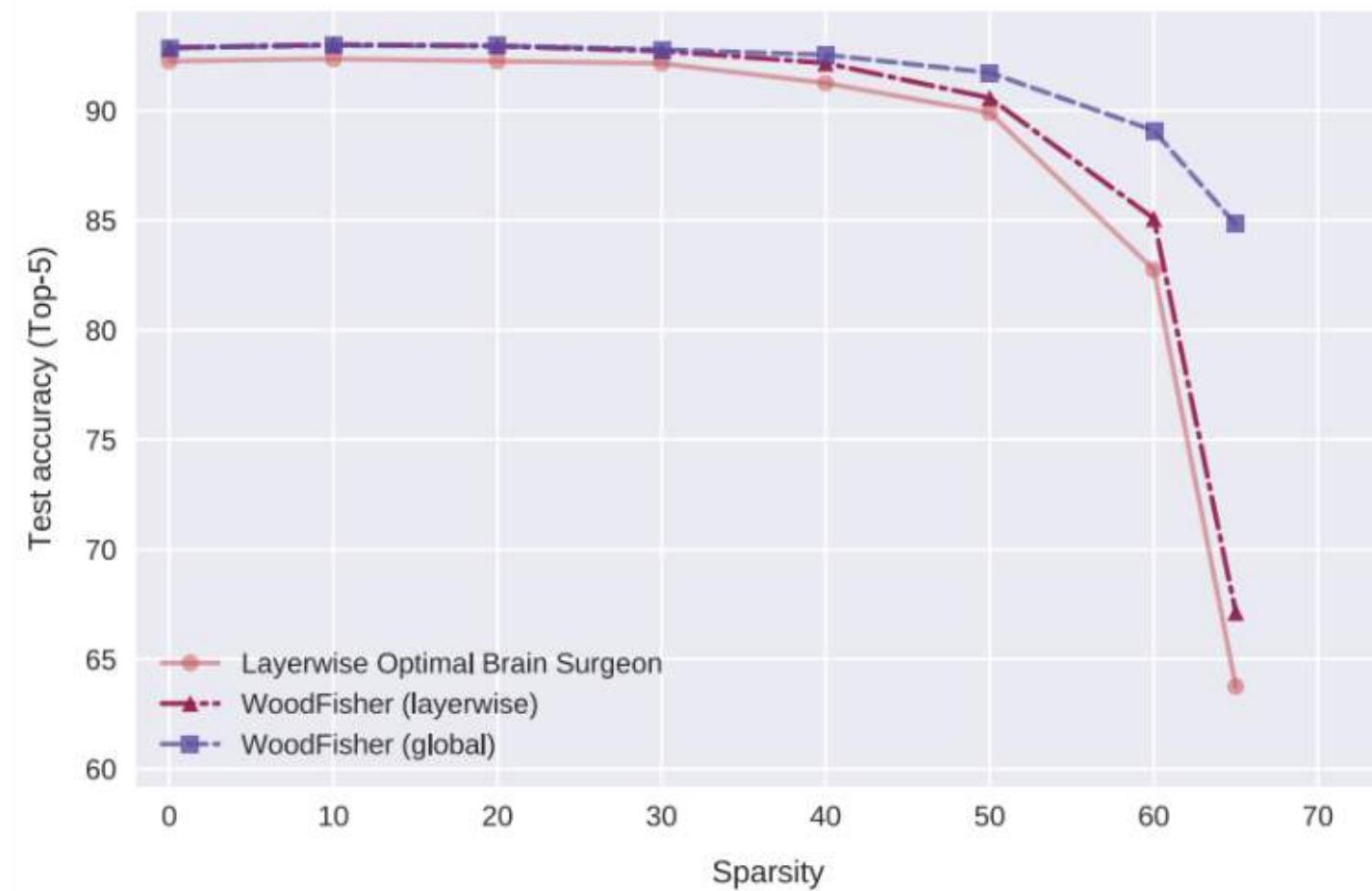
$$\mathbf{X} \in \mathbb{R}^{4,096 \times 4,096}$$



$$\mathbf{G} \in \mathbb{R}^{1,000 \times 1,000}$$



K-FAC



Критерии прунинга

Критерии пруинга

Data free

- ✓ Не нужно ничего вычислять во время пруинга
- ✓ Не нужна дополнительная память, хранится только модель
- ✓ Применяются единообразным образом ко всем моделям
- ✗ Более низкая точность сети после тюнинга

Data aware

- ✓ Более высокая точность сети после тюнинга
- ✓ Учитывают «контекст» задачи
- ✗ Необходима процедура вычисления лосса
- ✗ Могут требовать много памяти для хранения

Гипотеза лотерейных билетов

Можно ли найти оптимальную архитектуру перед тренировкой?

«The neural networks we typically train have subnetworks (at non-trivial sparsities) at initialization that can train to full accuracy in the same number of steps as the original network»

Можно ли найти оптимальную архитектуру перед тренировкой?

«The neural networks we typically train have subnetworks (at non-trivial sparsities) at initialization that can train to full accuracy in the same number of steps as the original network»

Можно ли найти оптимальную архитектуру перед тренировкой?

«The neural networks we typically train have subnetworks (at non-trivial sparsities) at initialization that can train to full accuracy in the same number of steps as the original network»

THE LOTTERY TICKET HYPOTHESIS: FINDING SPARSE, TRAINABLE NEURAL NETWORKS^[6]

Можно ли найти оптимальную архитектуру перед тренировкой?

«The neural networks we typically train have subnetworks (at non-trivial sparsities) at initialization that can train to full accuracy in the same number of steps as the original network»

THE LOTTERY TICKET HYPOTHESIS: FINDING SPARSE, TRAINABLE NEURAL NETWORKS [6]

Можно ли найти оптимальную архитектуру перед тренировкой?

«The neural networks we typically train have subnetworks (at non-trivial sparsities) at initialization that can train to full accuracy in the same number of steps as the original network»

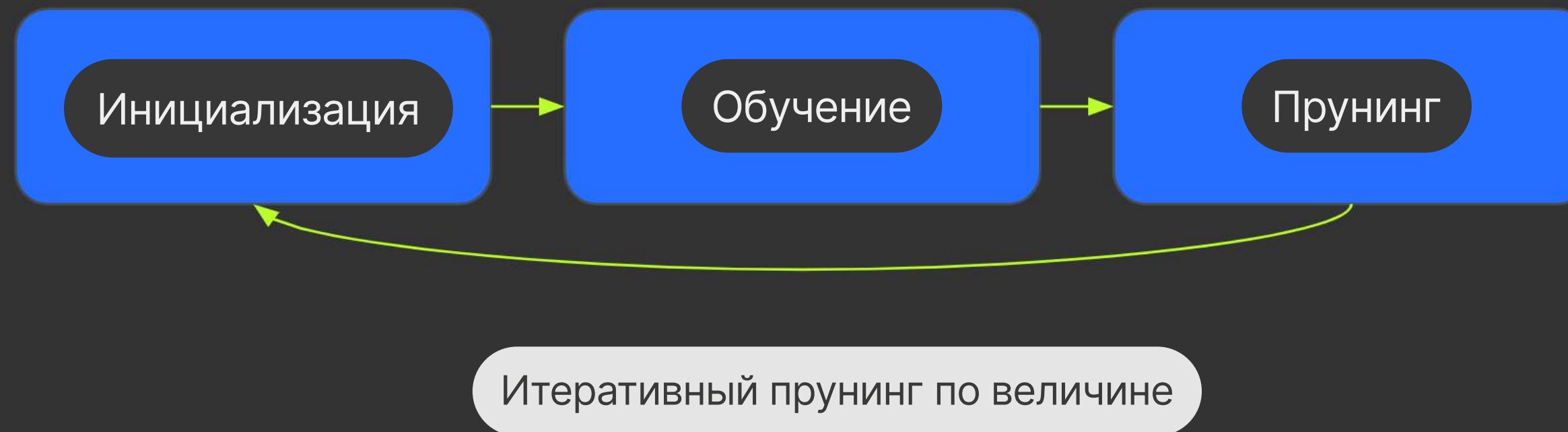
Как искать выигрышные билеты?

Ответ

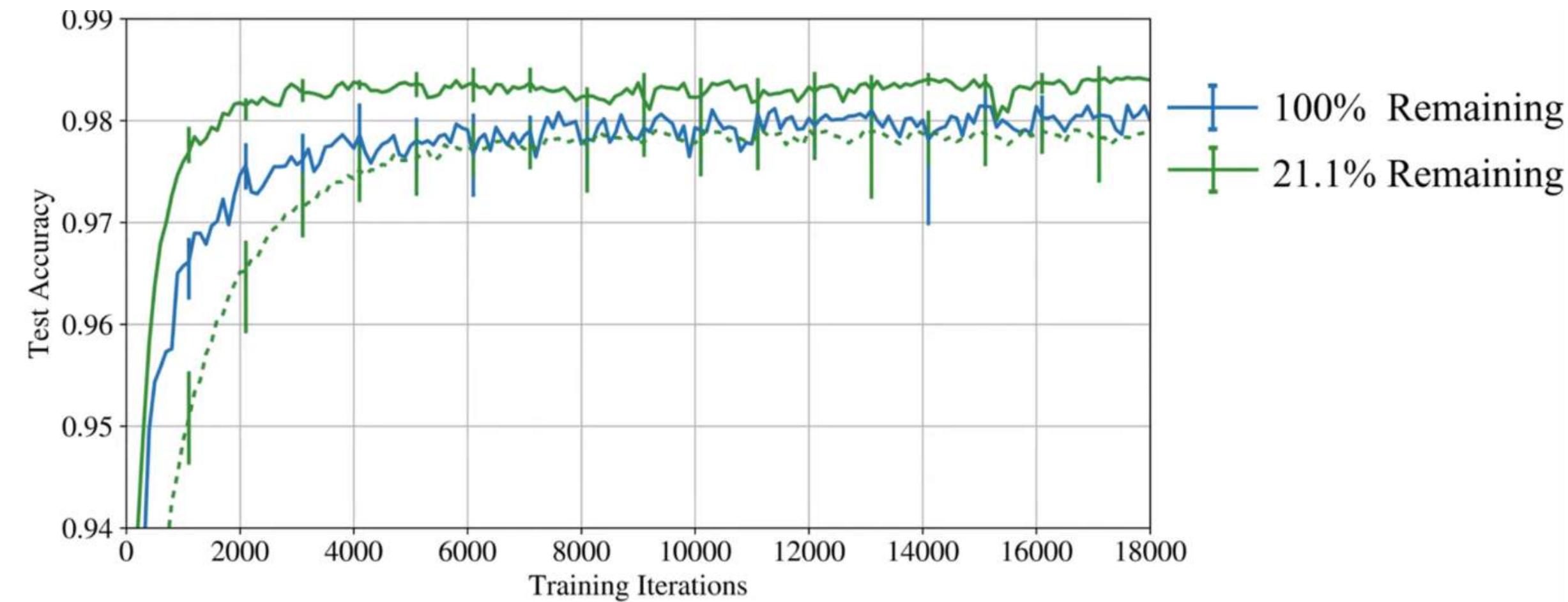
- Обучить модель,
а потом запрунить
Не работает с большими моделями

Виды пайплайна прунинга

→ Гипотеза о лотерейных билетах
Для прунига весов неструктурным образом



Начальные веса против случайных

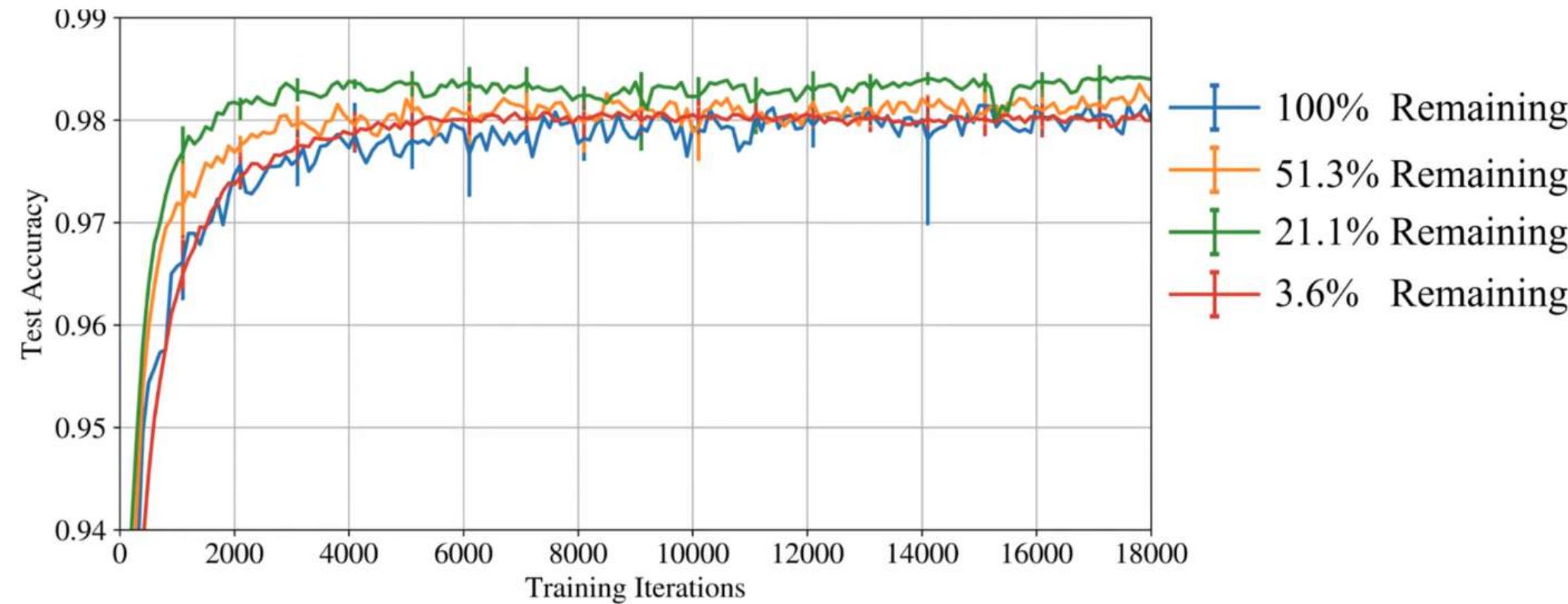


LeNet 300-100-10 for MNIST

fully-connected

300K parameters

Продолжаем заняться

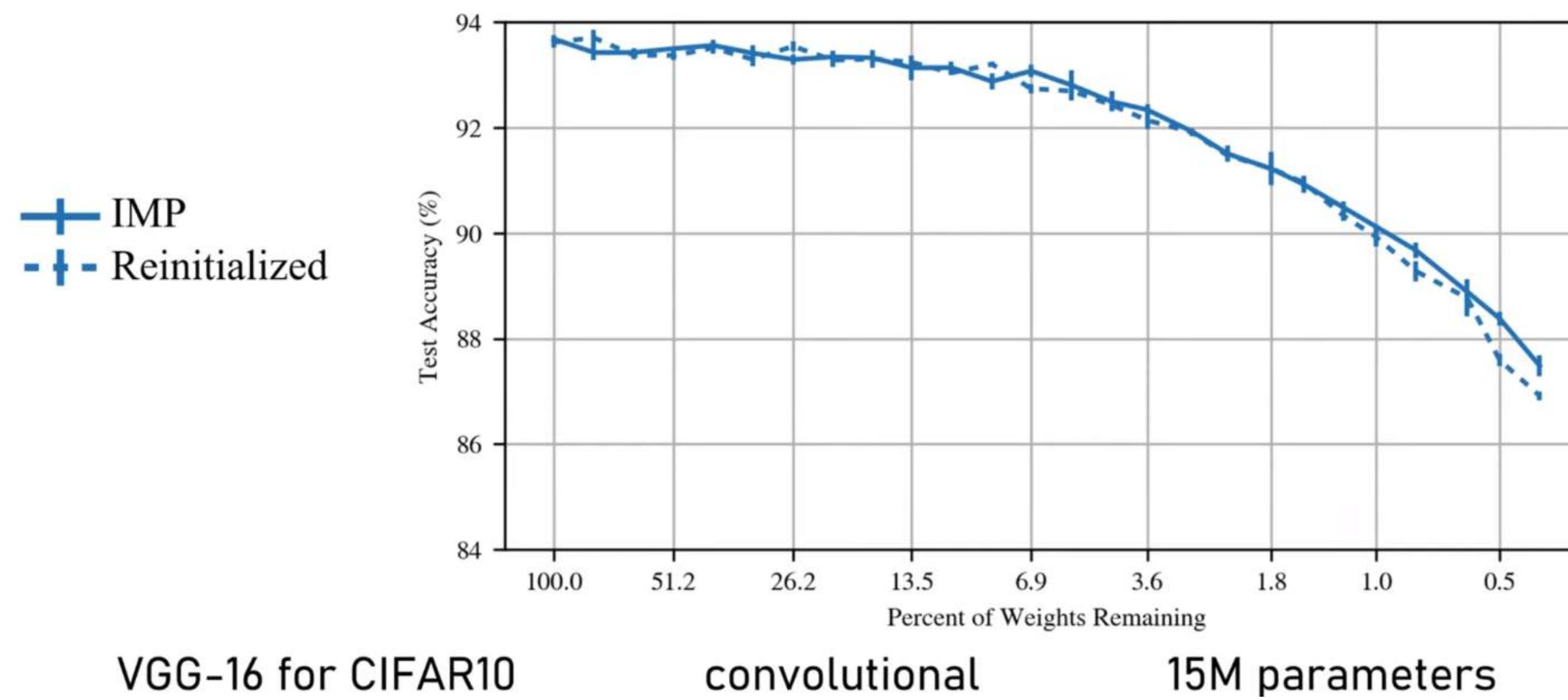


LeNet 300-100-10 for MNIST

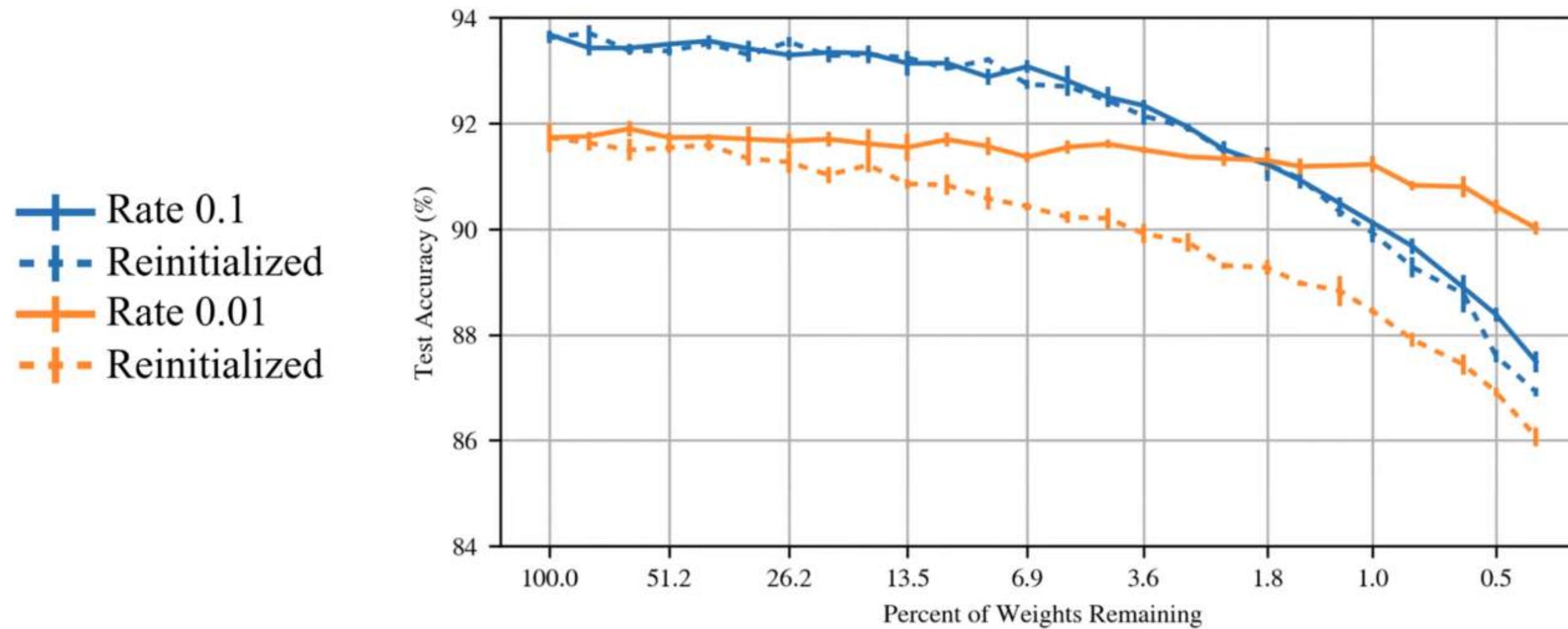
fully-connected

300K parameters

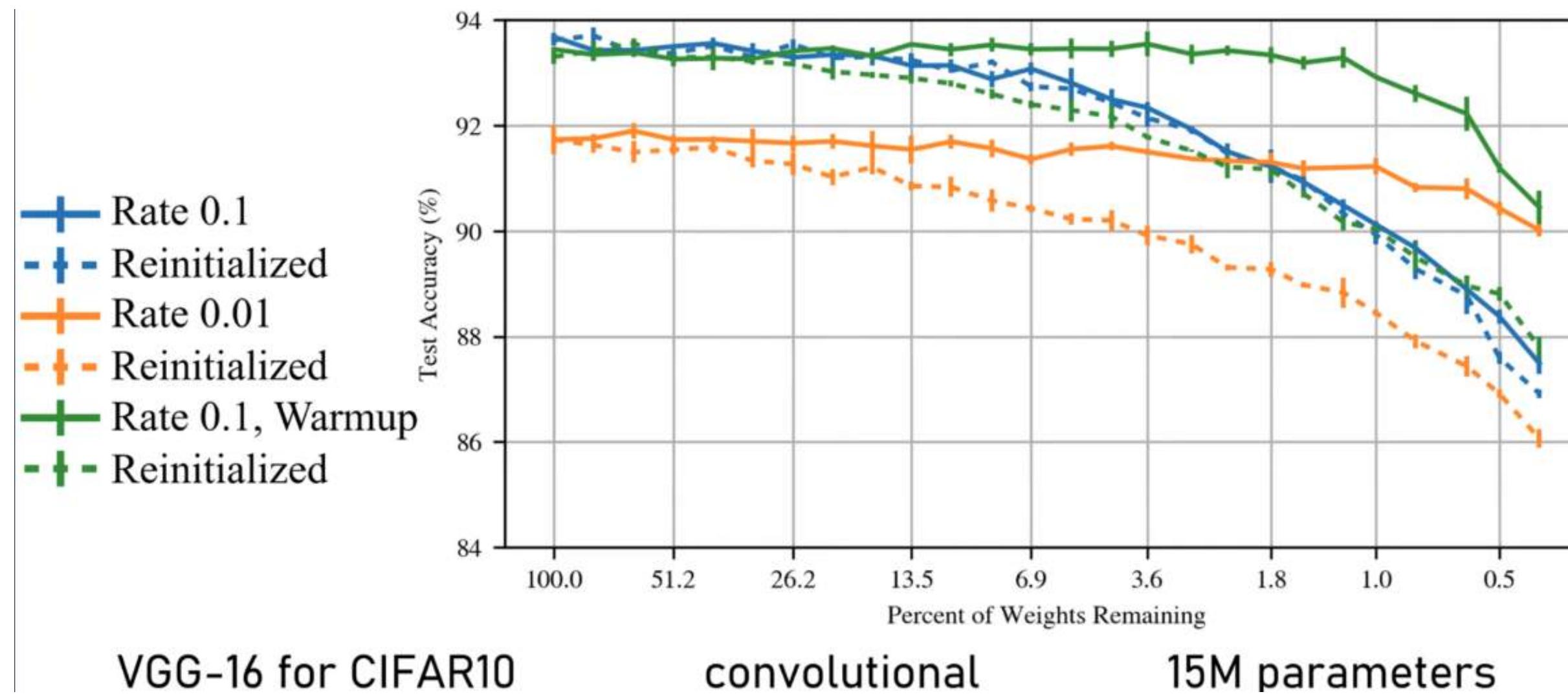
А что с большими моделями?



Слишком большой Learning rate



С warmup-ом еще лучше



Почему так происходит

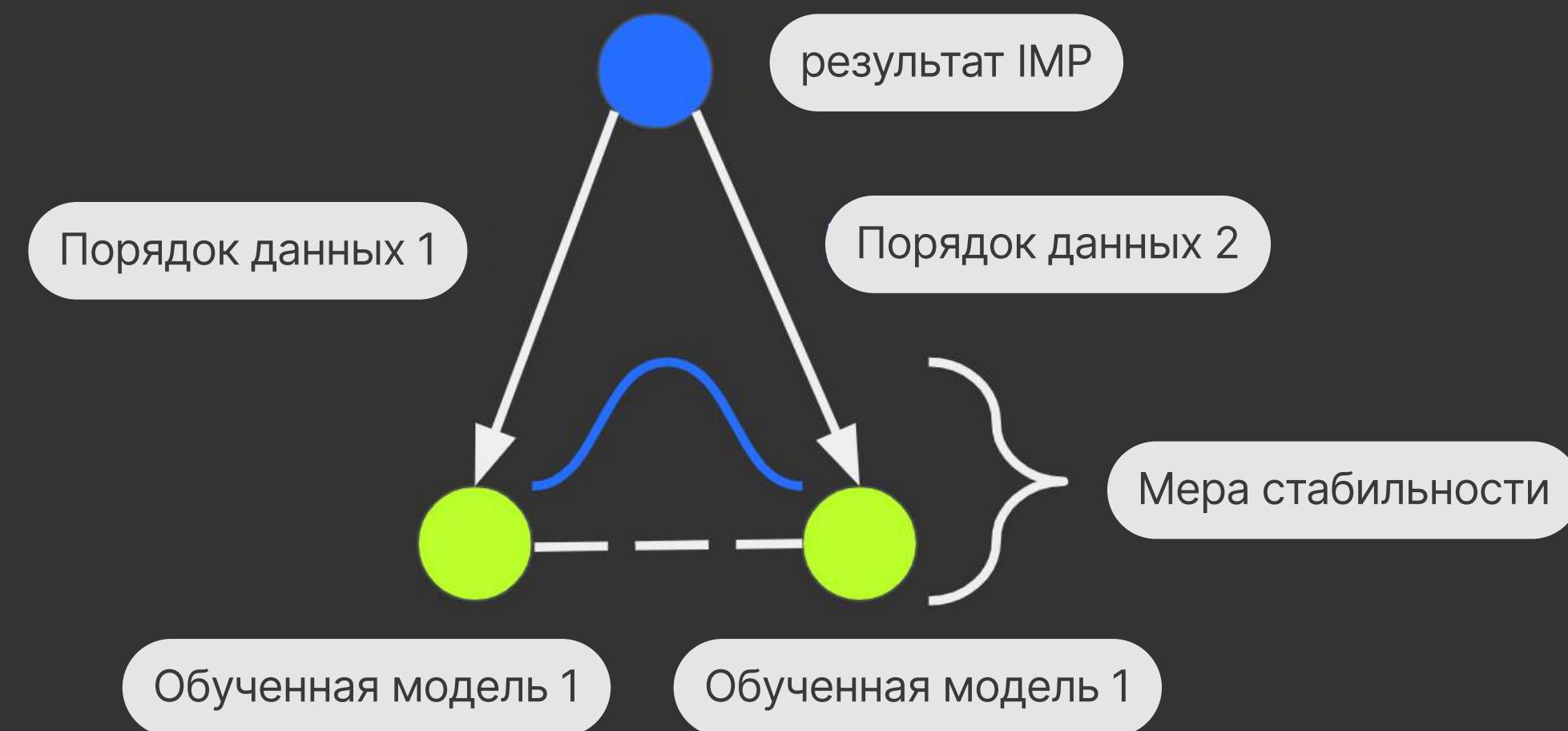
Гипотеза

Модели, найденные с помощью IMP,
чувствительны к шуму от SGD в начале
обучения

Доказательство

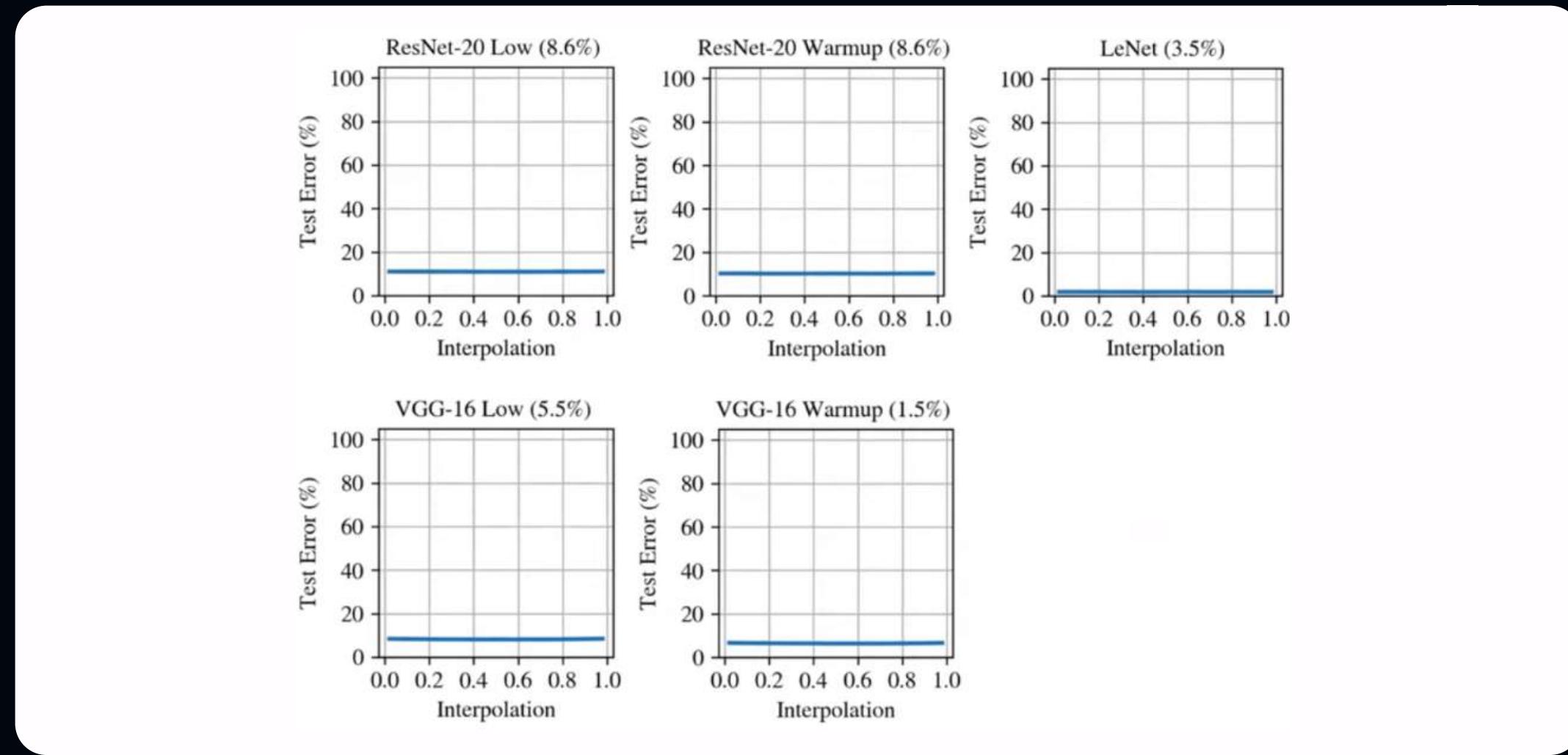
Показать, что уровень шума от SGD
влияет на точность моделей IMP

Linear model connectivity

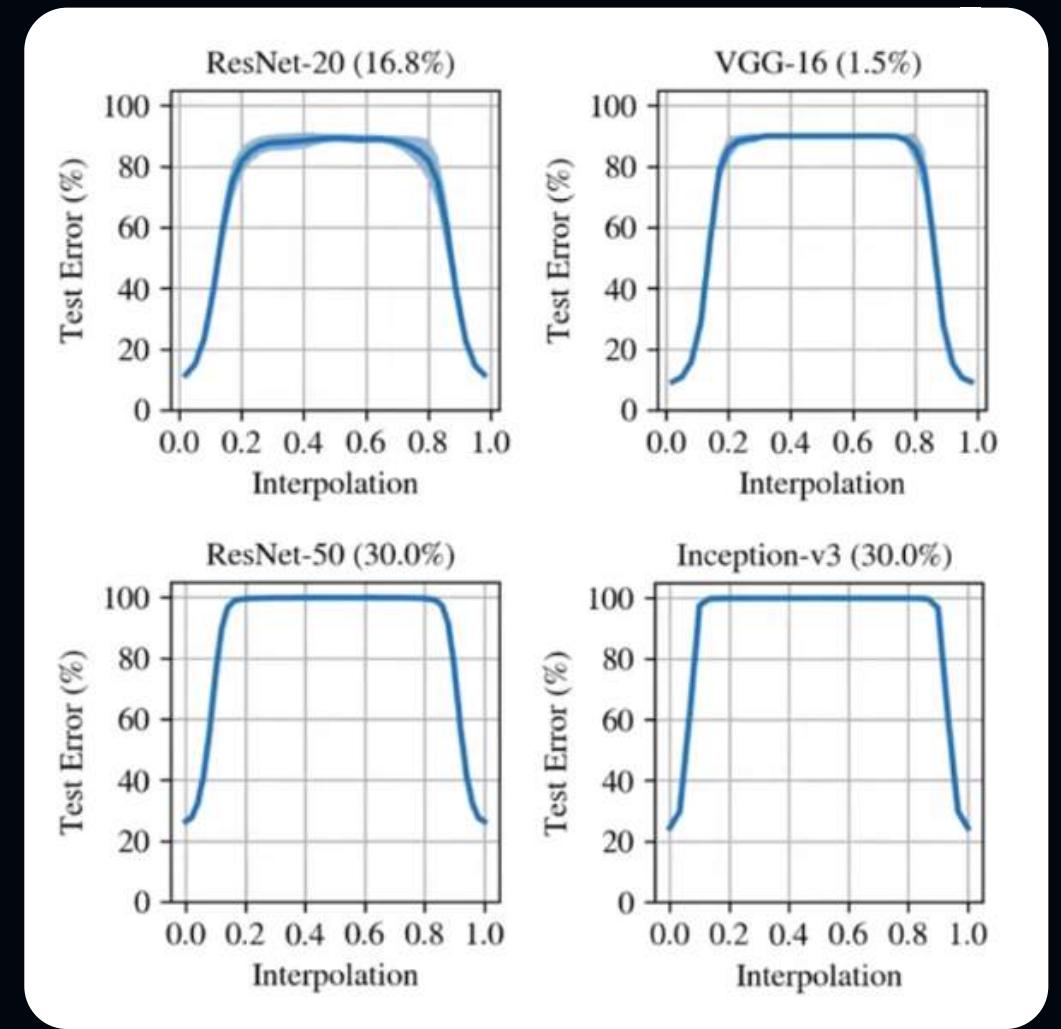


It works to estimate IMP perspective

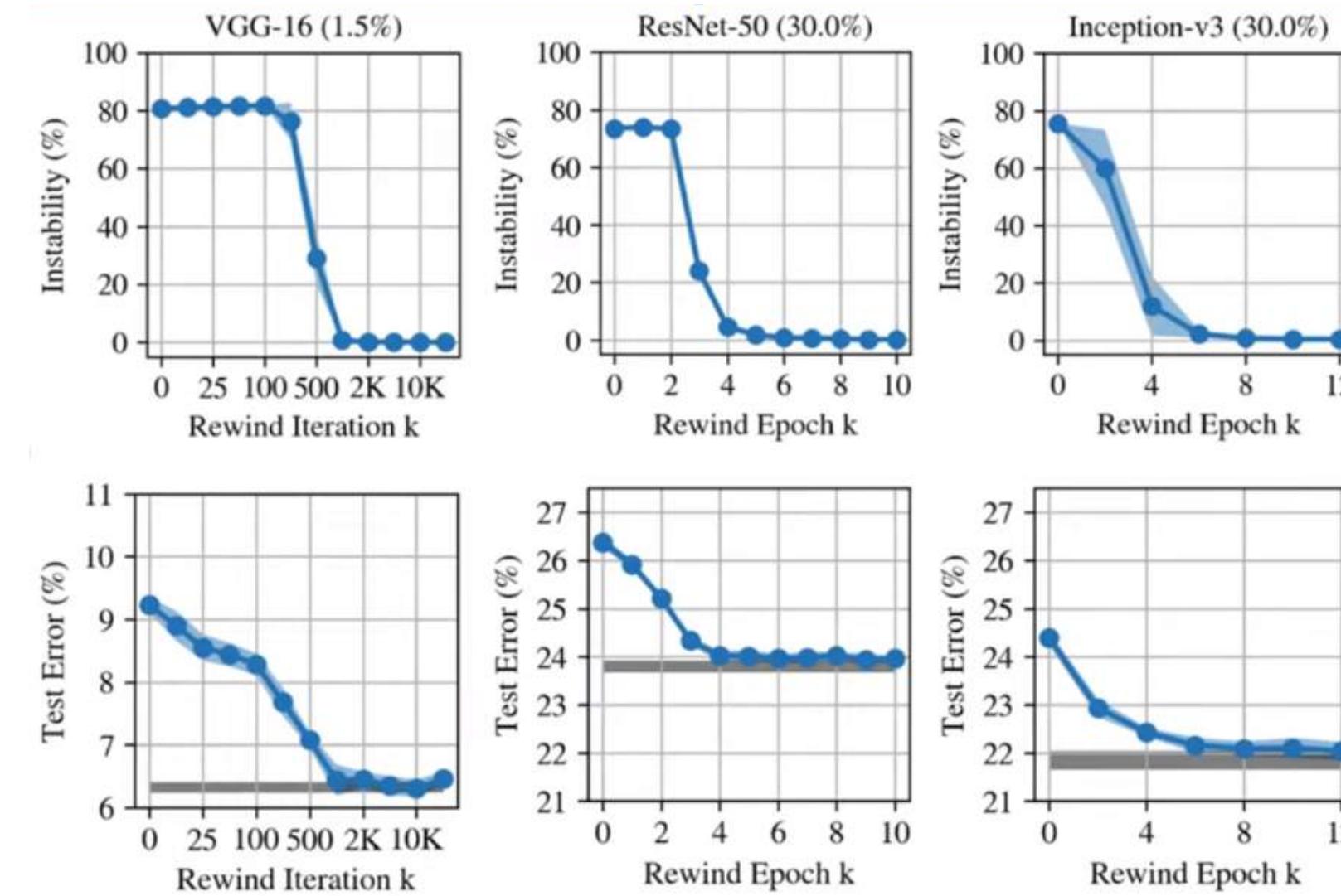
IMP succeed



IMP fails



Early IMP applying is the way to find winning tickets



Практические советы

Как это выглядит в коде Для библиотеки torch_pruning

```
1 taylor_criteria = tp.importance.GroupTaylorImportance()
2
3 pruner = tp.pruner.MetaPruner(
4     distilled_model,
5     example_inputs=input_example,
6     importance=taylor_criteria,
7     pruning_ratio=0.4,
8     global_pruning=True,
9     ignored_layers=ignored_layers,
10 )
11
12 for i in range(iterative_steps):
13     for group in pruner.step(interactive=True): # Warning: groups must be handled sequentially. Do not keep them as a list.
14         print(group)
15         # do whatever you like with the group
16         dep, idxs = group[0] # get the idxs
17         target_module = dep.target.module # get the root module
18         pruning_fn = dep.handler # get the pruning function
19         group.prune()
20         # group.prune(idxs=[0, 2, 6]) # It is even possible to change the pruning behaviour with the idxs parameter
21         macs, nparams = tp.utils.count_ops_and_params(model, example_inputs)
22         # finetune your model here
23         # finetune(model)
24         # ...|
```

Пруниг групп

```
# 0. prepare your model and example inputs
model = resnet18(pretrained=True).eval()
example_inputs = torch.randn(1,3,224,224)

# 1. build dependency graph for resnet18
DG = tp.DependencyGraph().build_dependency(model, example_inputs=example_inputs)

# 2. Select some channels to prune. Here we prune the channels indexed by [2, 6, 9].
pruning_idxs = pruning_idxs=[2, 6, 9]
pruning_group = DG.get_pruning_group(model.conv1, tp.prune_conv_out_channels, idxs=pruning_idxs)

# 3. prune all grouped layer that is coupled with model.conv1
if DG.check_pruning_group(pruning_group):
    pruning_group.prune()
```

ПРУНИНГ ГРУПП

Вопрос на понимание

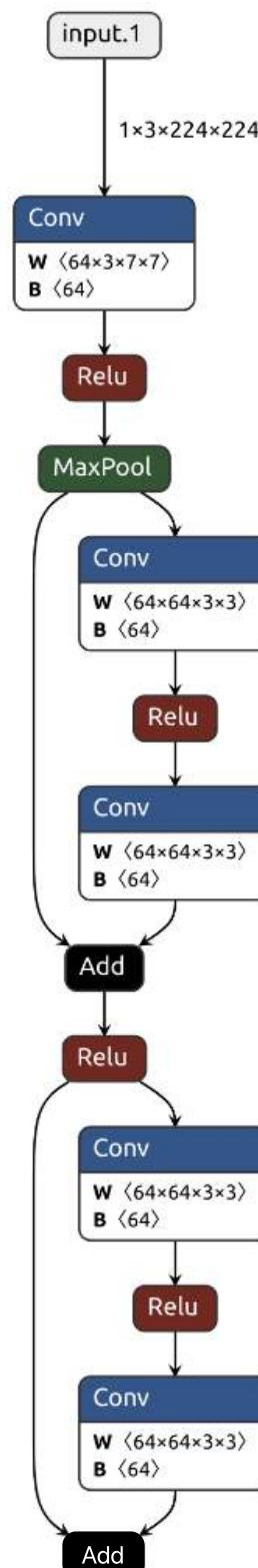
→ Какие каналы запрунятся?

```
# 0. prepare your model and example inputs
model = resnet18(pretrained=True).eval()
example_inputs = torch.randn(1,3,224,224)

# 1. build dependency graph for resnet18
DG = tp.DependencyGraph().build_dependency(model, example_inputs=example_inputs)

# 2. Select some channels to prune. Here we prune the channels indexed by [2, 6, 9].
pruning_idxs = pruning_idxs=[2, 6, 9]
pruning_group = DG.get_pruning_group(model.conv1, tp.prune_conv_out_channels, idxs=pruning_idxs)

# 3. prune all grouped layer that is coupled with model.conv1
if DG.check_pruning_group(pruning_group):
    pruning_group.prune()
```



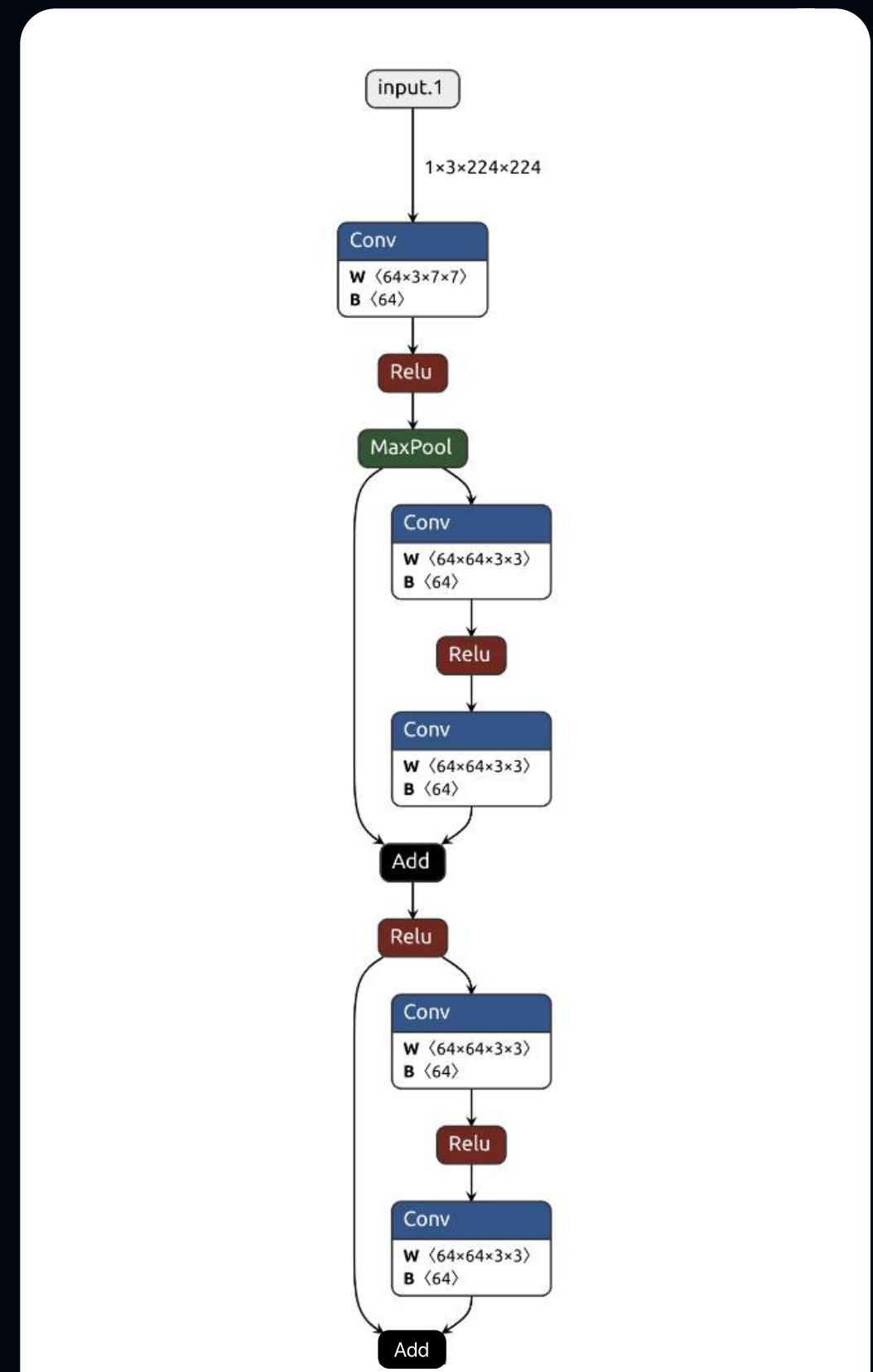
ПРУНИНГ ГРУПП

```
# 0. prepare your model and example inputs
model = resnet18(pretrained=True).eval()
example_inputs = torch.randn(1,3,224,224)

# 1. build dependency graph for resnet18
DG = tp.DependencyGraph().build_dependency(model, example_inputs=example_inputs)

# 2. Select some channels to prune. Here we prune the channels indexed by [2, 6, 9].
pruning_idxs = pruning_idxs=[2, 6, 9]
pruning_group = DG.get_pruning_group( model.conv1, tp.prune_conv_out_channels, idxs=pruning_idxs )

# 3. prune all grouped layer that is coupled with model.conv1
if DG.check_pruning_group(pruning_group):
    pruning_group.prune()
```



Рекомендации по дообучению

- ✓ Используйте `warmup` на первых эпохах тюнинга
- ✓ Используйте `learning rate` меньше чем для `baseline`
- ✓ Используйте больший `batch_size`
Запруненная модель требует меньше памяти
- ✓ Помните, что при сильном прунинге точность может не восстановиться
- ✓ Дистиллируйте с незапруненной модели

Как заставить модель прунитьсѧ

1 Избавьтесь от констант, связанных с пруящеися размерностью

Каналы/головы

2 Вычисляйте параметры для `reshape/view` на основе актуального числа каналов

Ультимативный гайд по прунигу

- ✓ Сначала применяйте пруинг, а уже потом ищите глубину
- ✓ Используя равномерный пруинг, найдите максимальное ускорение при устраивающей вас точности — delta
- ✓ Prune ratio на каждом слое — это гиперпараметр, он ищется также как и любой другой гиперпараметр. Optuna в помощь
- ✓ Используя глобал пруинг, найдите максимальное ускорение при устраивающей вас точности — delta
- ✓ Искомая архитектура лежит в заданных пределах по pruning ratio
- ✓ Не пруньте сетку так, чтобы у вас были **очень** сильные болтлнеки по каналам
Пример: 256-1-256

Ультимативный гайд по прунигу

- ✓ Если у вас такой все же возник, рекомендуется удалить данный слой, дообучить сетку и приступить к прунигу заново
- ✓ Первый слой самый дорогой в плане вычислений, но и самый ценный в плане информации. Не обрезайте его больше чем в половину
- ✓ **Пользуйтесь дистилляцией!!!**
Все практические советы про дистилляцию
Дистилляция с адаптерами, с какого слоя стягивать могут быть применимы и тут

Домашнее задание

Magnitude vs Taylor criteria

Segformer ↗



Архитектура

Human Matting
Dataset ↗



Датасет

Выяснить, какая из двух предложенных конфигураций
прунинга дает наибольшее ускорение модели при максимальной
просадке mean_iou 1%



Задача

Список литературы

Список литературы

-
- | | | |
|--|---|---|
| 1. AMC: AutoML for Model Compression and Acceleration
on Mobile Devices | arxiv.org/abs/1802.03494 | ↗ |
| 2. Exploring the Granularity of Sparsity in Convolutional
Neural Networks | openaccess.thecvf.com/
content_cvpr_2017_workshops/w29/papers/
Mao_Exploring_the_Granularity_CVPR_2017_paper.pdf | ↗ |
| 3. Accelerating Sparse Deep Neural Networks | arxiv.org/abs/2104.08378 | ↗ |
| 4. Learning both Weights and Connections for Efficient
Neural Networks | arxiv.org/abs/1506.02626 | ↗ |
| 5. DepGraph: Towards Any Structural Pruning | openaccess.thecvf.com/content/CVPR2023/papers/
Fang_DepGraph_Towards_Any_Structural_Pruning_
CVPR_2023_paper.pdf | ↗ |
| 6. THE LOTTERY TICKET HYPOTHESIS: FINDING SPARSE,
TRAINABLE NEURAL NETWORKS | arxiv.org/abs/1803.03635 | ↗ |
-

-
- 7. Stabilizing the Lottery Ticket Hypothesis arxiv.org/pdf/1903.01611.pdf ↗
 - 8. Linear Mode Connectivity and the Lottery Ticket Hypothesis arxiv.org/abs/1912.05671 ↗
 - 9. Batch-Normalization-based Soft Filter Pruning for Deep Convolutional Neural Networks ieeexplore.ieee.org/document/9305319 ↗
 - 10. Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration arxiv.org/abs/1811.00250 ↗
 - 11. Importance Estimation for Neural Network Pruning arxiv.org/abs/1906.10771 ↗
 - 12. Optimal Brain Damage https://proceedings.neurips.cc/paper_file/paper/1989file/6c9882bbac1c7093bd25041881277658-Paper.pdf ↗
 - 13. Optimal Brain Surgeon www.babak.caltech.edu/pubs/conferences/00298572.pdf ↗
 - 14. WoodFisher arxiv.org/abs/2004.14340 ↗
 - 15. K-FAC прунинг openreview.net/pdf?id=r1g5b2RcKm ↗
-