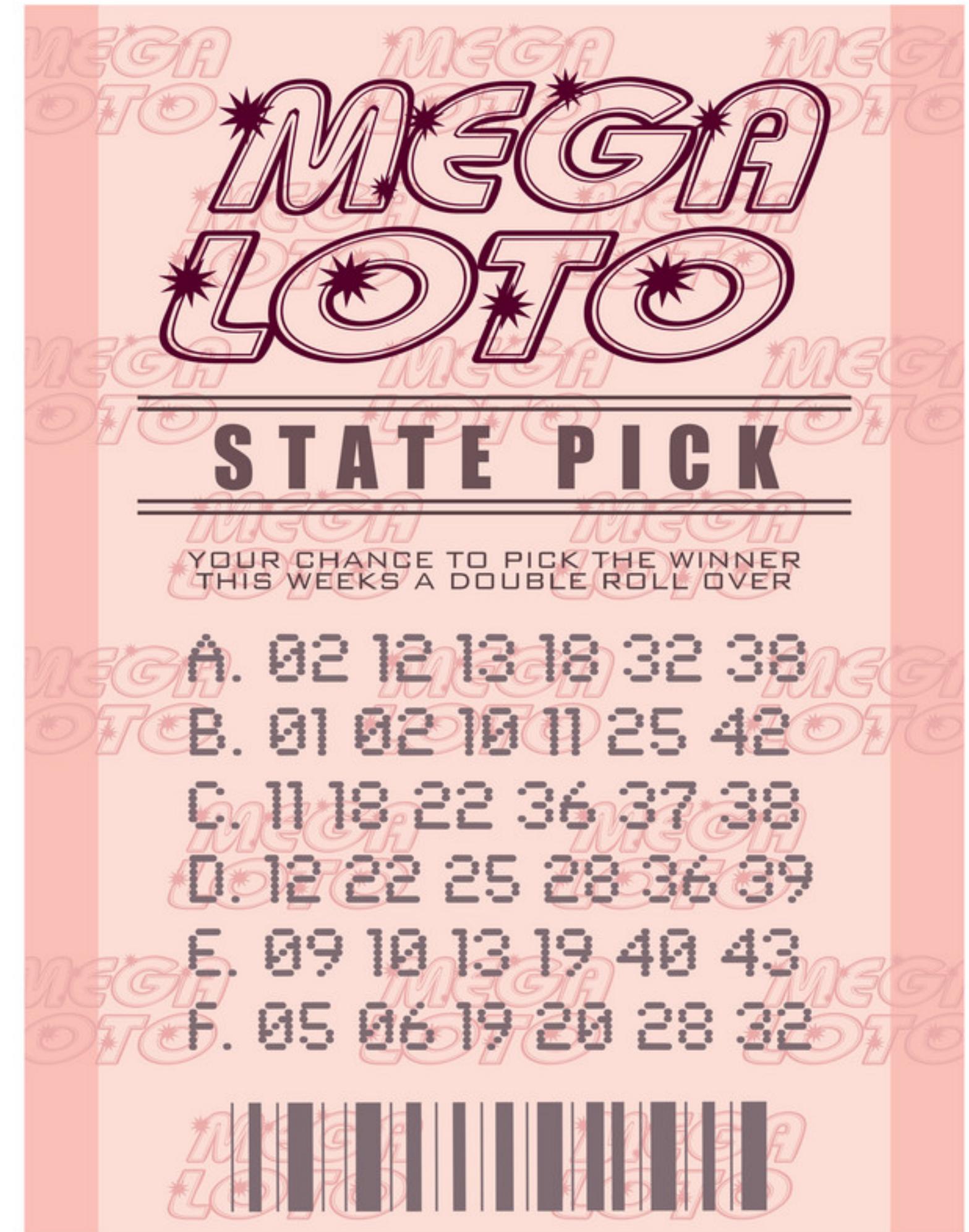


THE LOTTERY TICKET HYPOTHESIS

БПМИ192 НИС

Кокорина Юлия



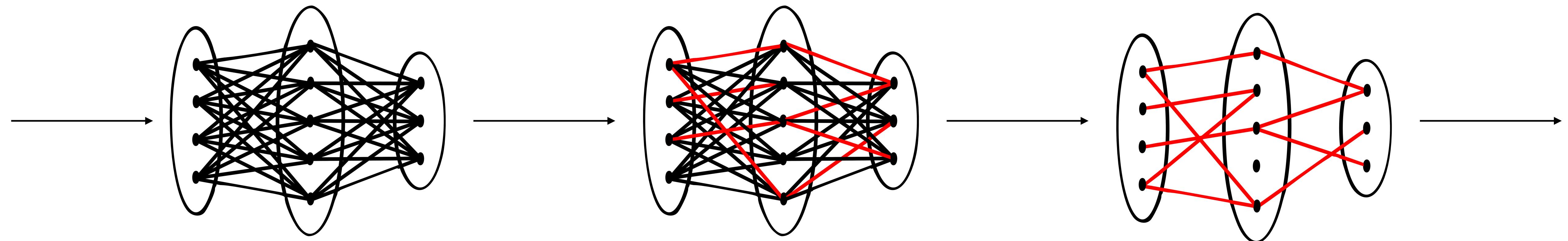
Зачем нужно уменьшать количество параметров в нейросетях?

- Уменьшить затраты по памяти
- Увеличить скорость работы
- Ускорить процесс обучения

Как обычно уменьшают количество параметров?

1. Обучают полную нейросеть
2. Как-то выбирают, какие параметры можно выкинуть
3. Дообучают то, что осталось

Чего нам хочется: избежать первого пункта, сразу учить маленькую сеть



Основная гипотеза

Пусть дана нейросеть $f(x, \theta)$ с заданными параметрами инициализации θ_0 . Тогда существует подсеть ($m \in \{0, 1\}^{|\theta|}$), такая что если учить с нуля $f(x, \theta \odot m)$ с параметрами инициализации $\theta_0 \odot m$, то:

1. Наименьшая ошибка на валидации достигается на меньшем числе итераций
2. Ошибка на валидации меньше, чем у исходной нейросети
3. $\|m\|_0 \ll |\theta|$

Такую подсеть мы будем называть «выигрышным билетом».

Базовый алгоритм поиска «выигрышного билета»

1. Как-то инициализируем параметры исходной сети - θ_0
2. Учим ее j итераций, получаем параметры θ_j
3. Обрезаем $p\%$ параметров, получаем маску m
4. Наш выигрышный билет – $f(x, \theta_0 \odot m)$

Архитектуры

<i>Network</i>	Lenet	Conv-2	Conv-4	Conv-6	Resnet-18	VGG-19
<i>Convolutions</i>				64, 64, pool 64, 64, pool 128, 128, pool	16, 3x[16, 16] 128, 128, pool 256, 256, pool	2x64 pool 2x128 pool, 4x256, pool 4x512, pool, 4x512
<i>FC Layers</i>	300, 100, 10	256, 256, 10	256, 256, 10	256, 256, 10	avg-pool, 10	avg-pool, 10
<i>All/Conv Weights</i>	266K	4.3M / 38K	2.4M / 260K	1.7M / 1.1M	274K / 270K	20.0M
<i>Iterations/Batch</i>	50K / 60	20K / 60	25K / 60	30K / 60	30K / 128	112K / 64
<i>Optimizer</i>	Adam 1.2e-3	Adam 2e-4	Adam 3e-4	Adam 3e-4	← SGD 0.1-0.01-0.001 Momentum 0.9 →	
<i>Pruning Rate</i>	fc20%	conv10% fc20% conv10% fc20% conv15% fc20%	conv20% fc0%	conv20% fc0%		

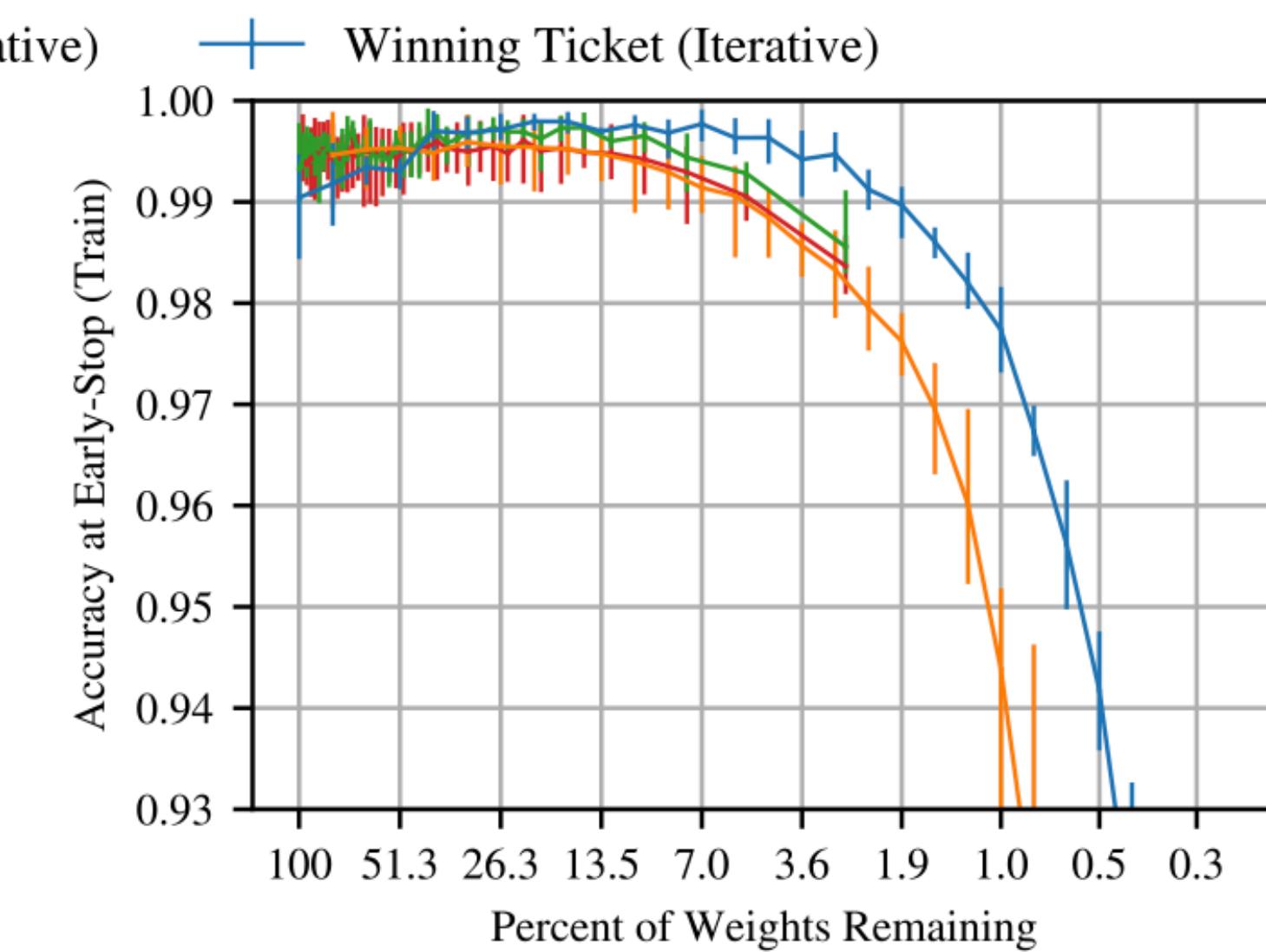
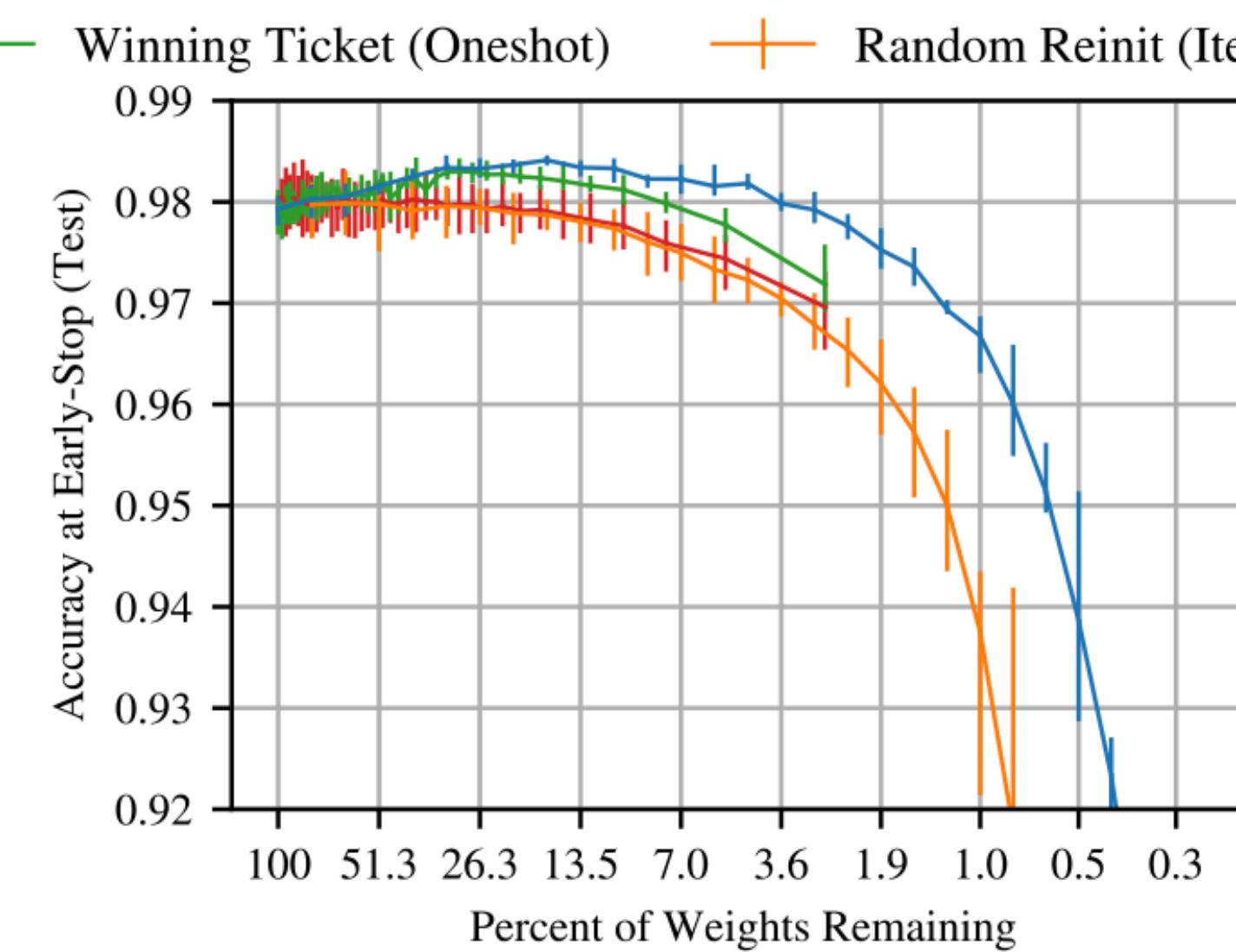
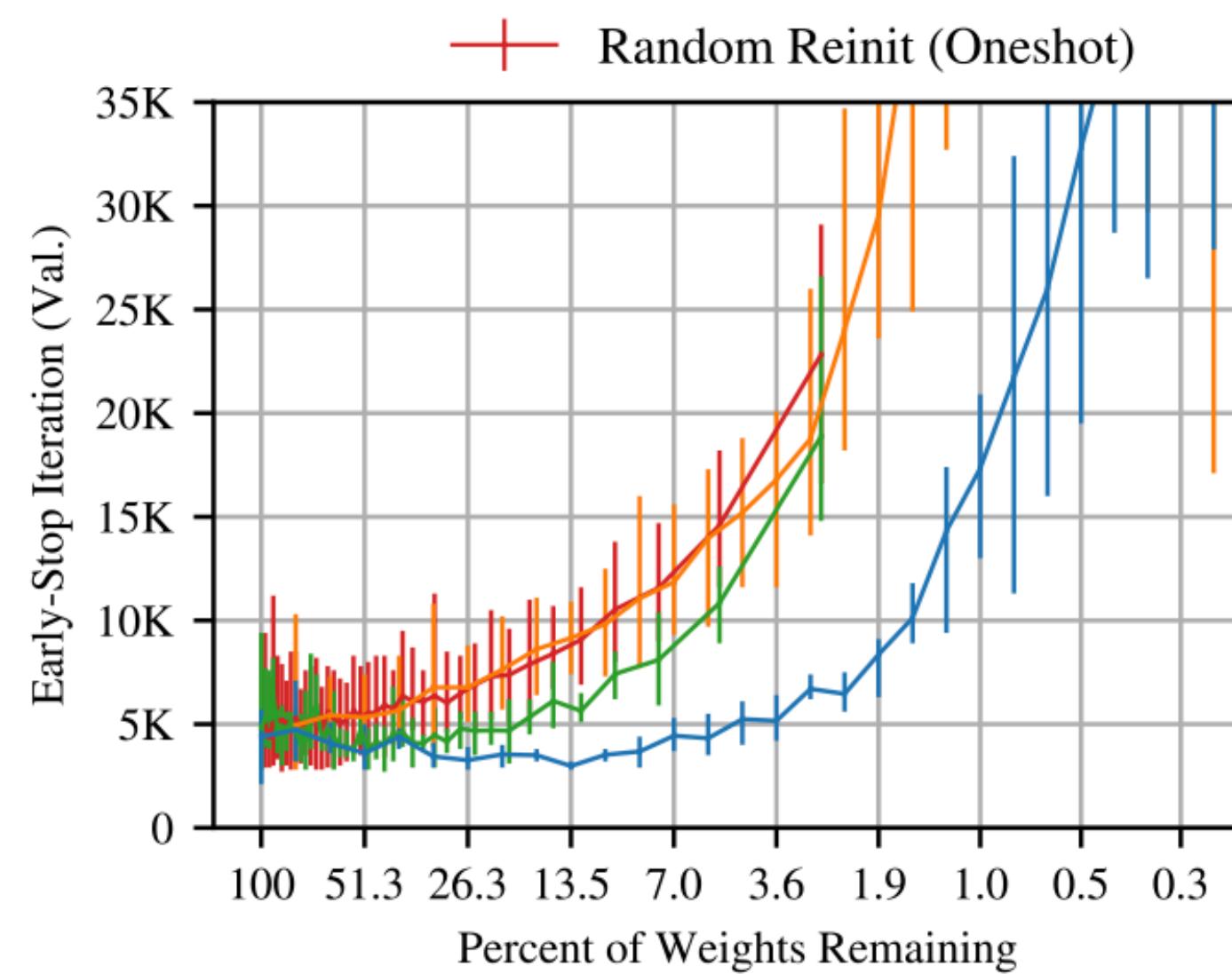
MNIST:



CIFAR10:

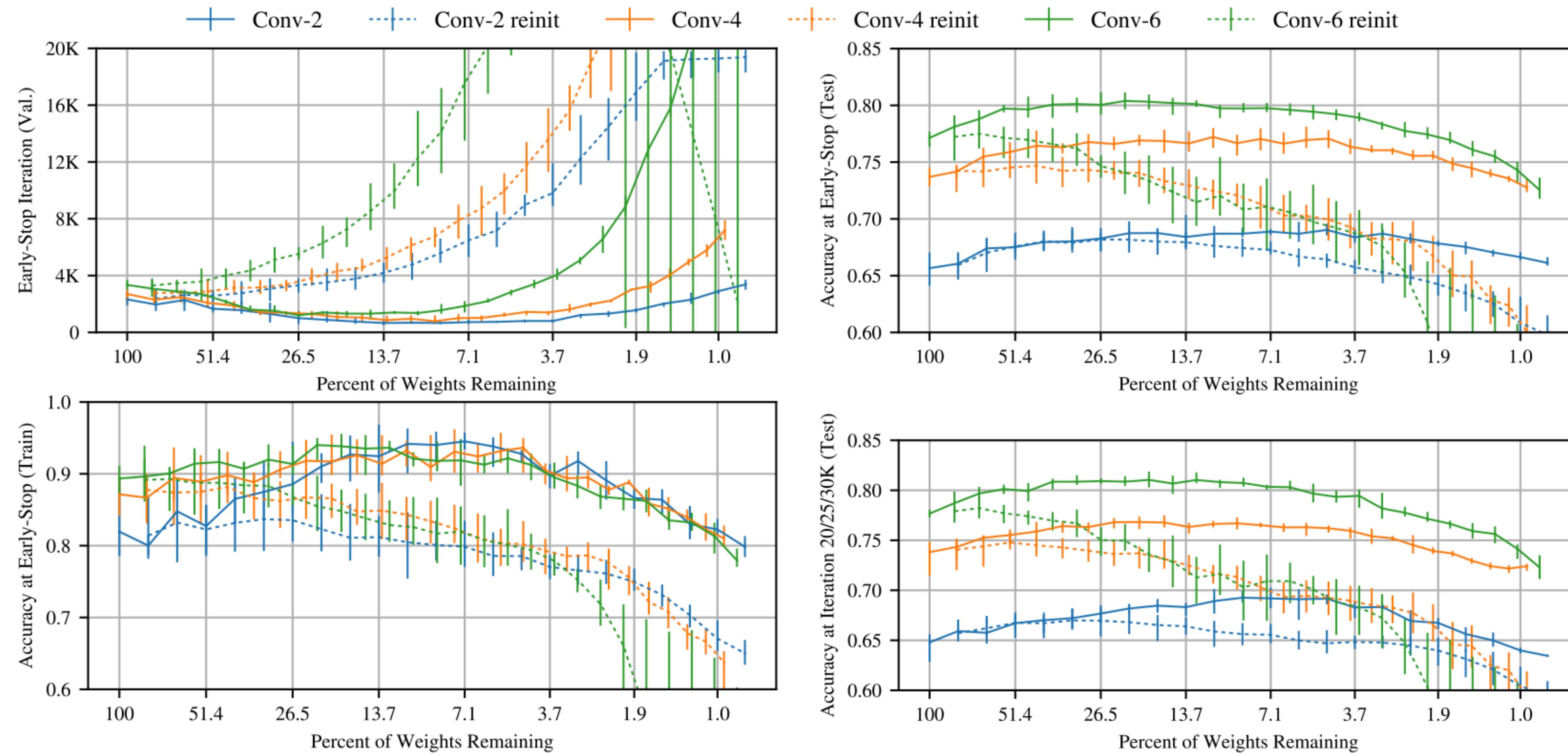


Результаты для Lenet

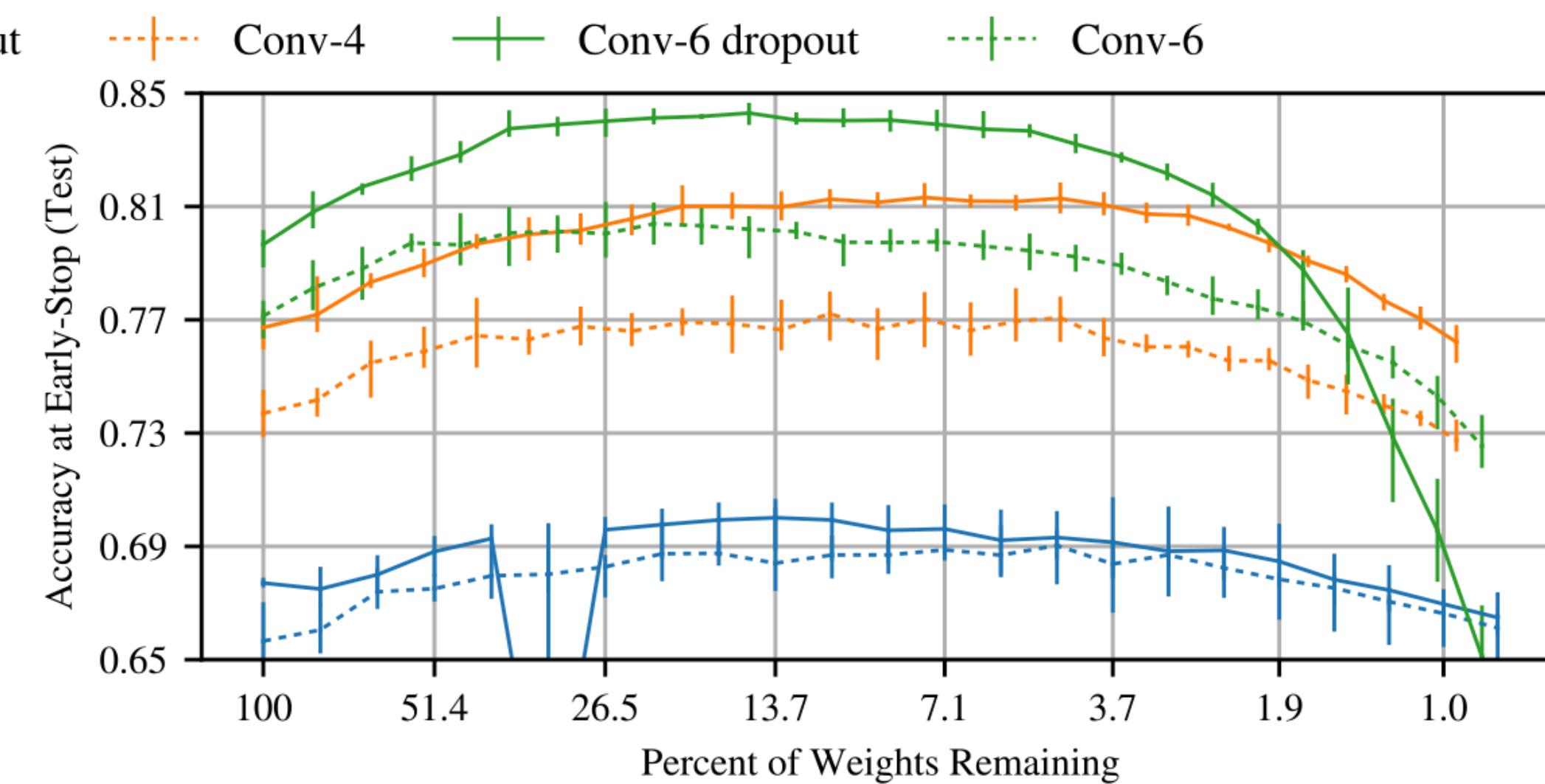
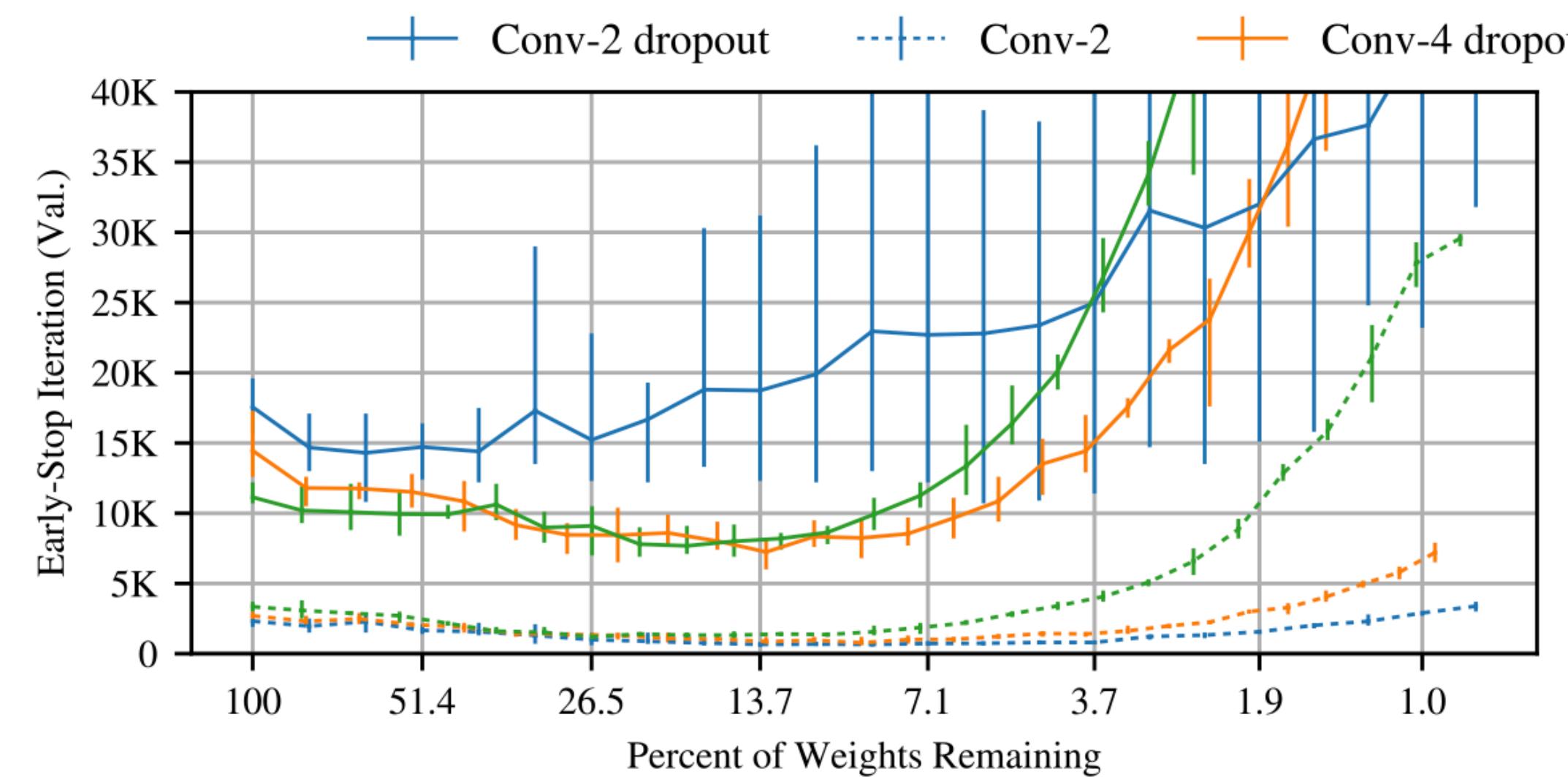


Вывод: инициализация важна, итеративная обрезка лучше, чем «one-shot»

Результаты для Conv-2/4/6

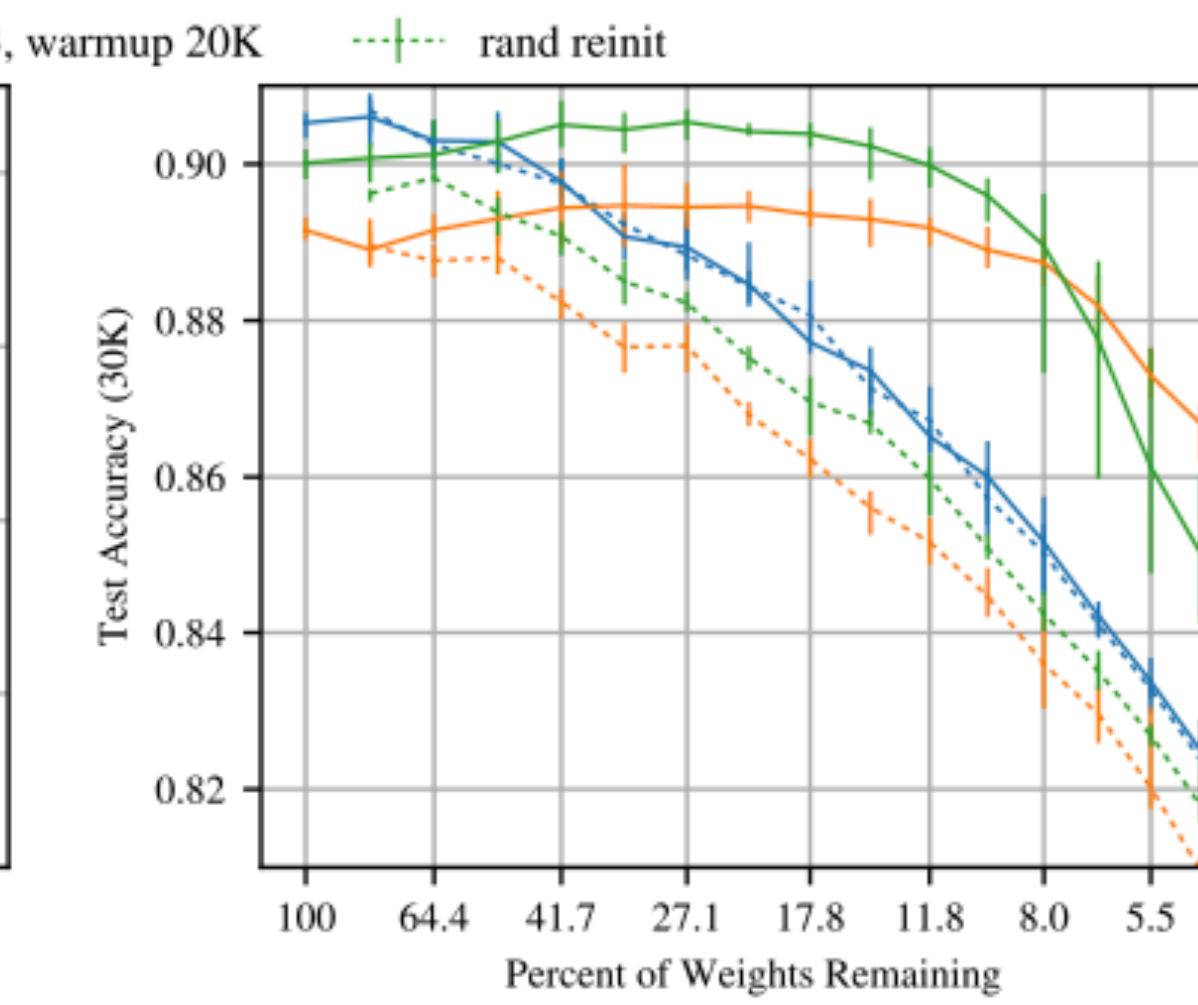
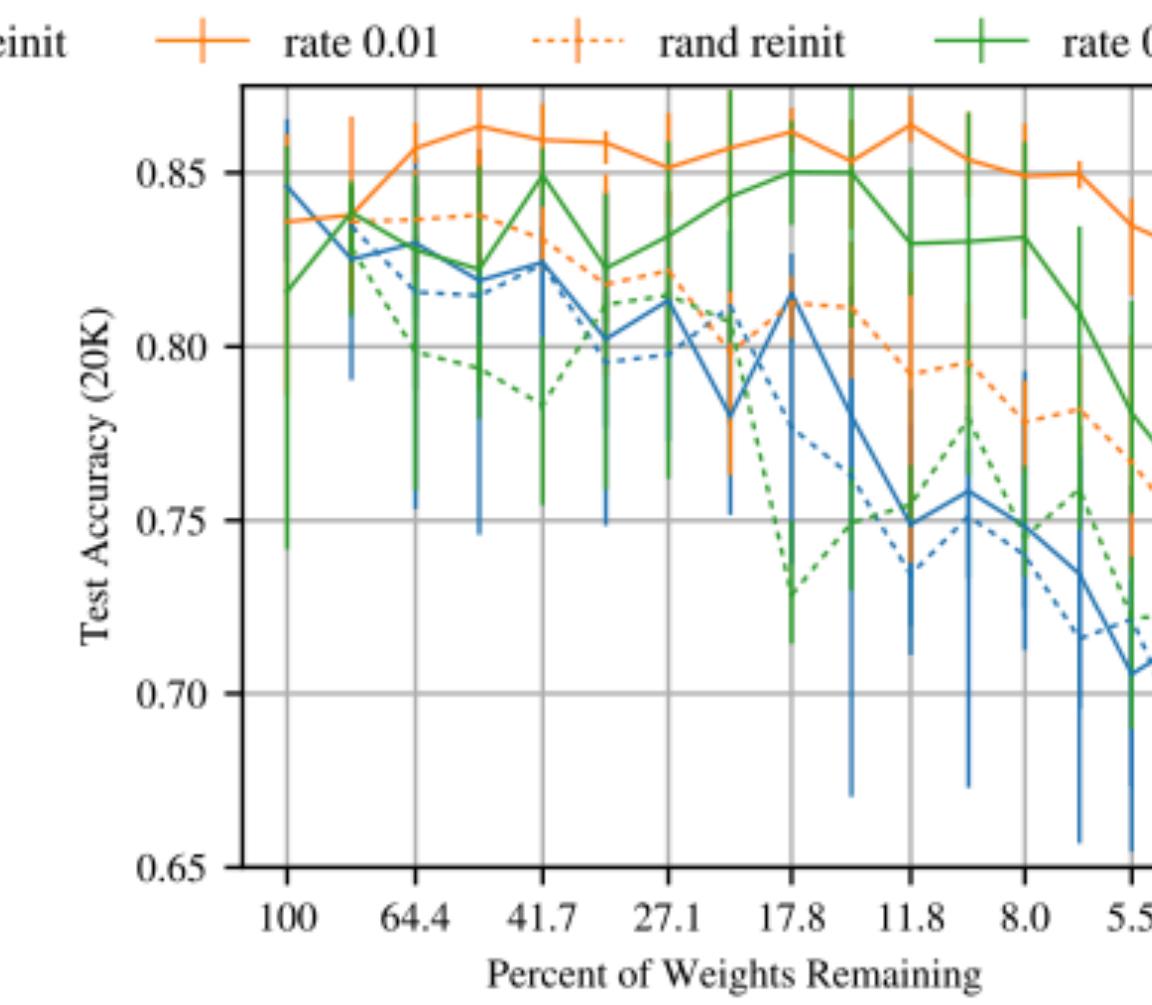
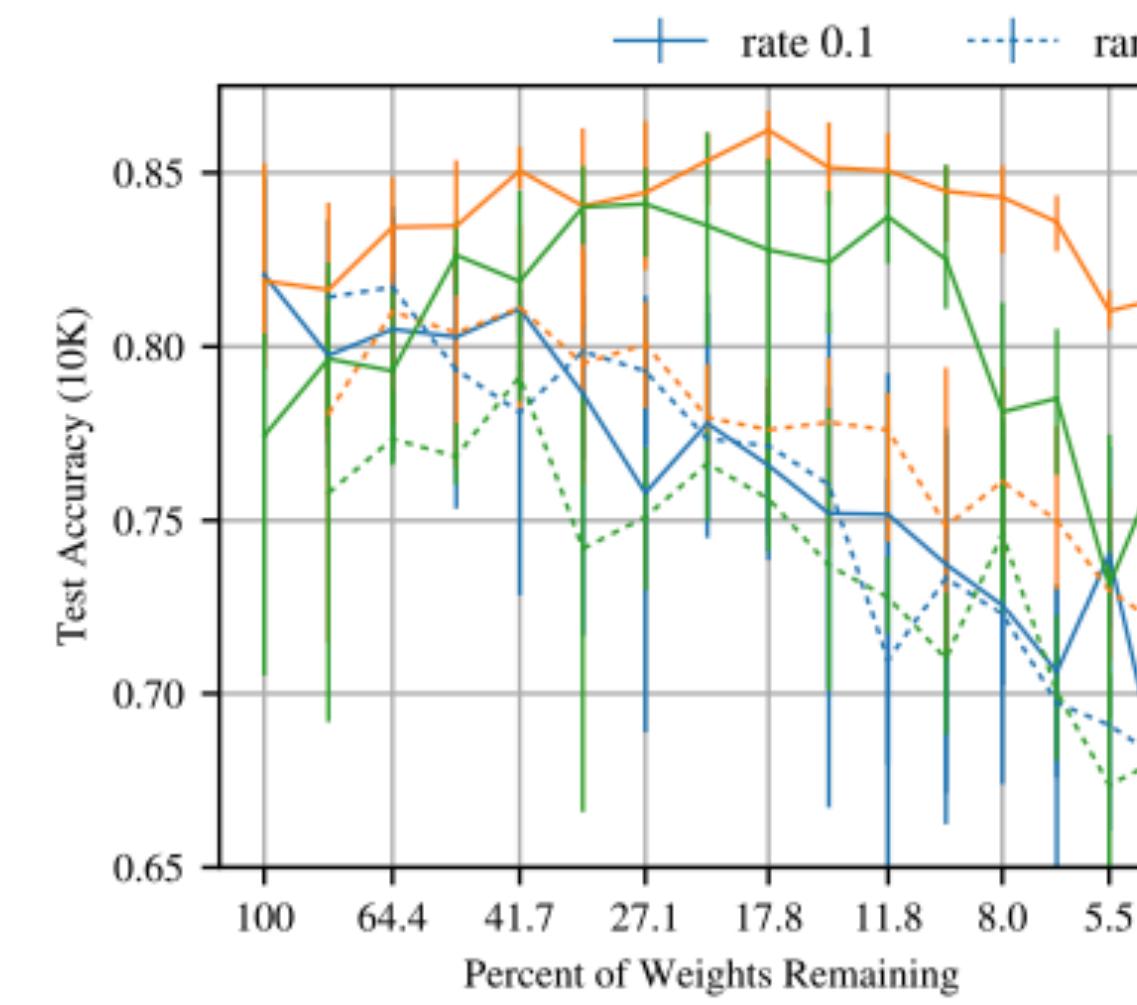
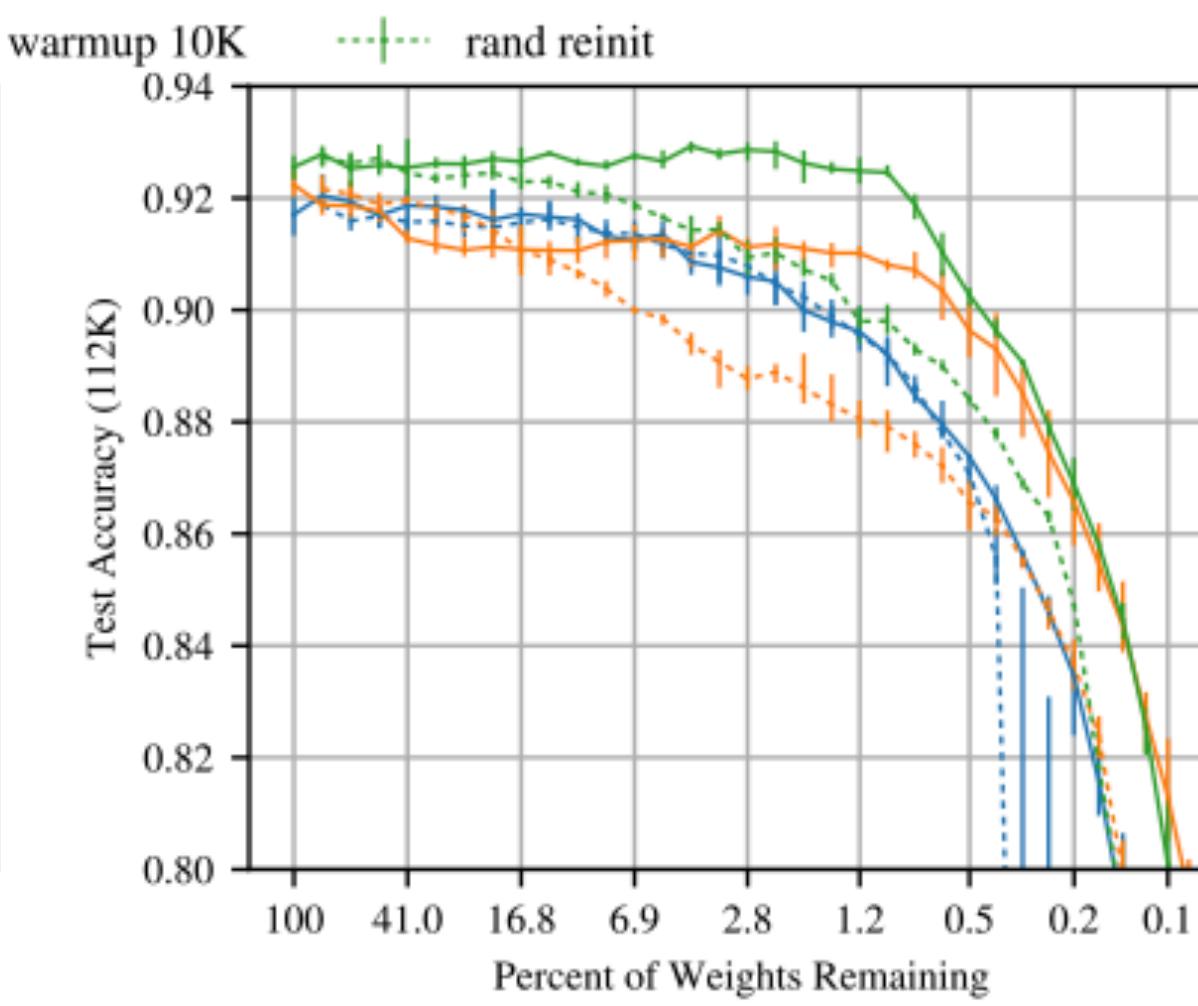
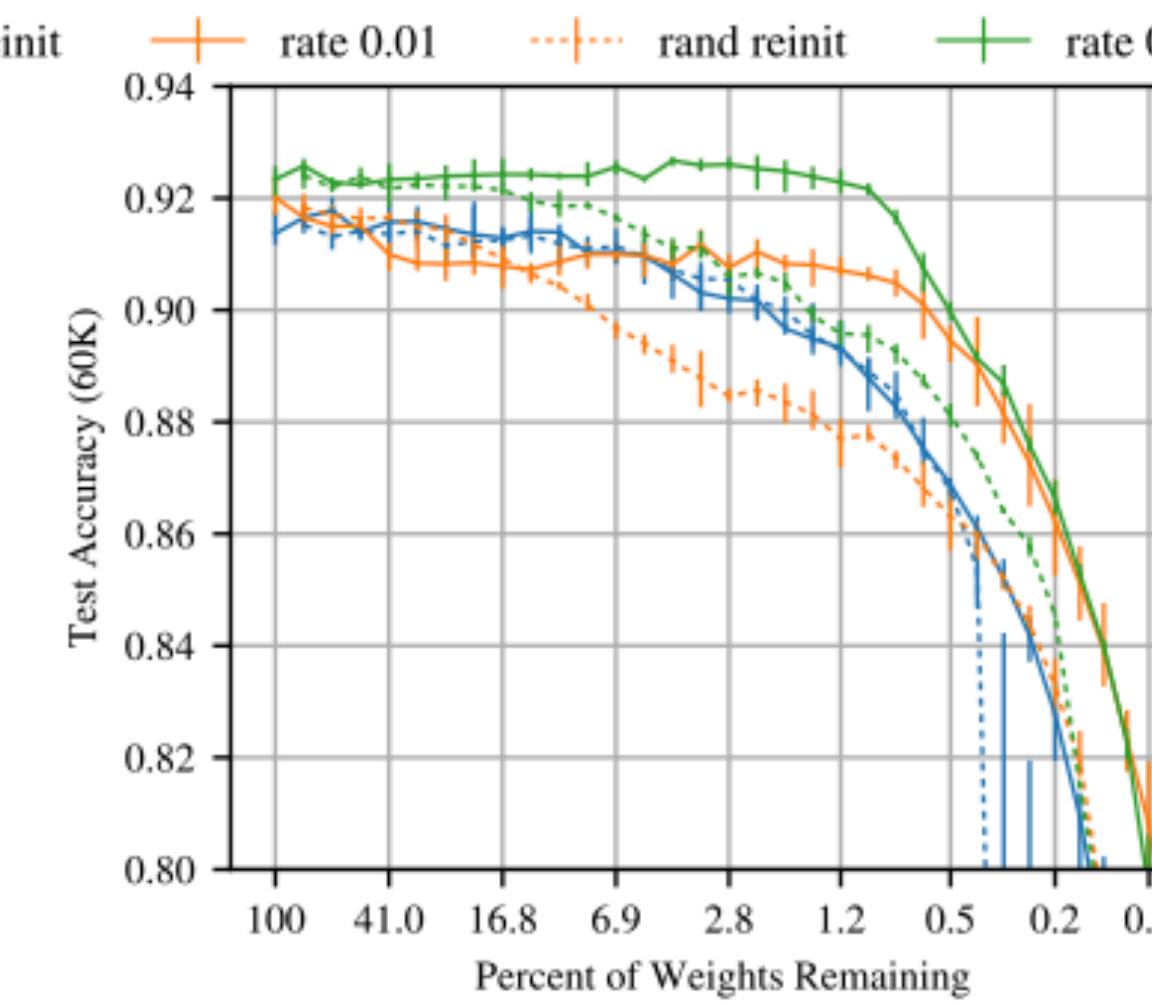
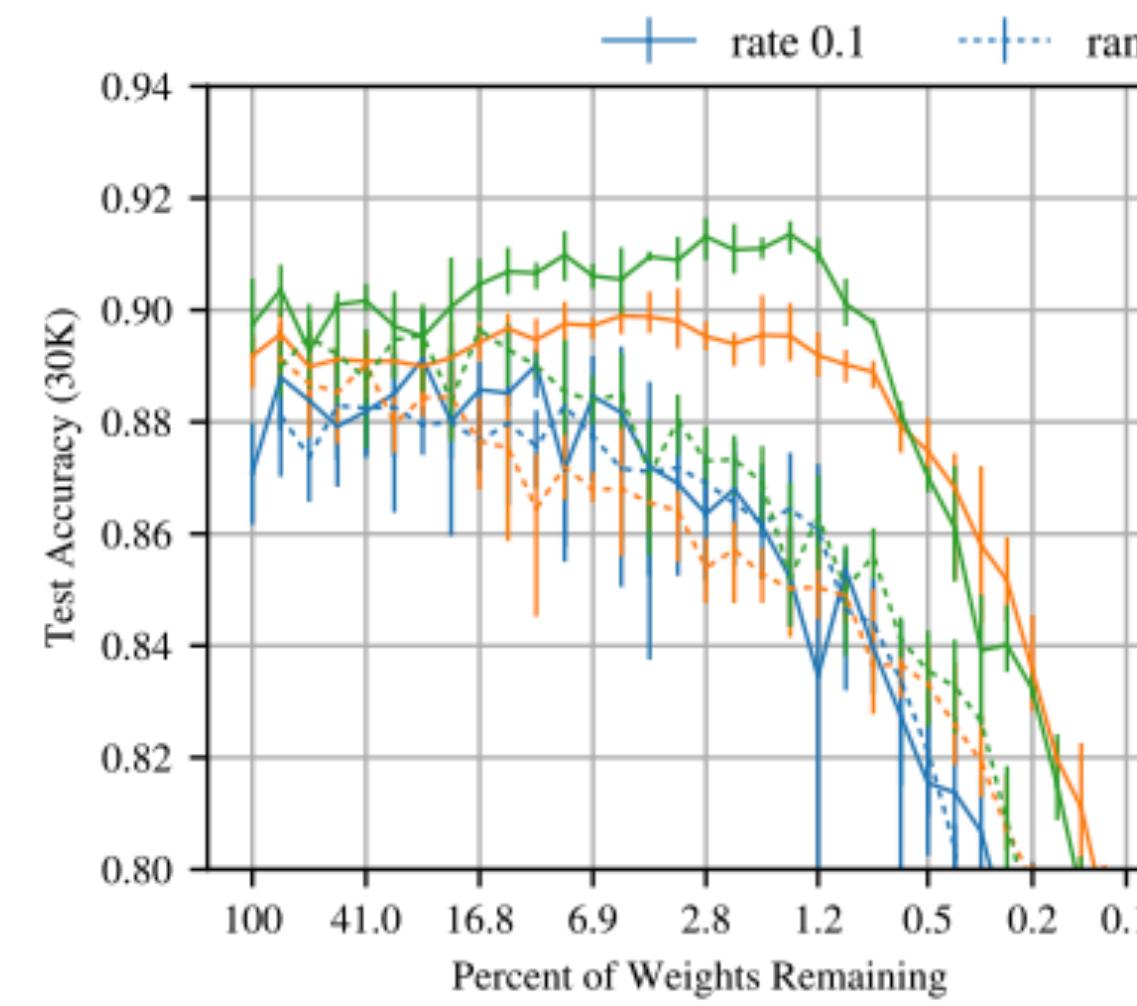


Связь с dropout

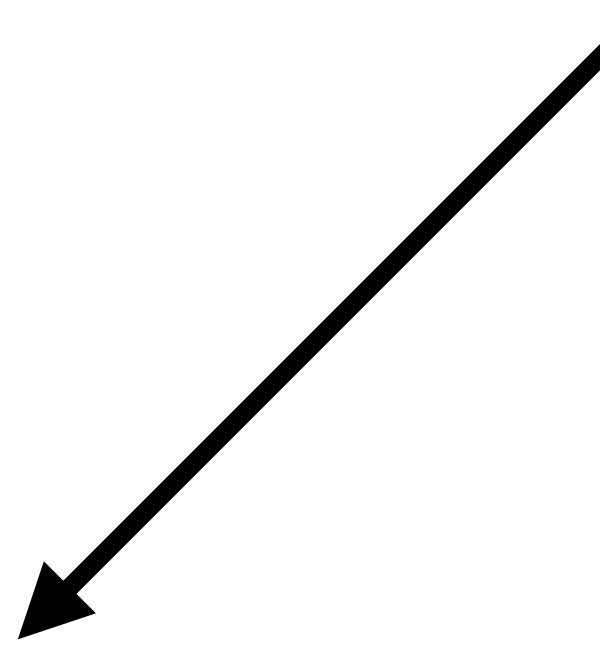


Вывод: с dropout учится дольше, но результат лучше

Результаты для VGG-19, ResNet

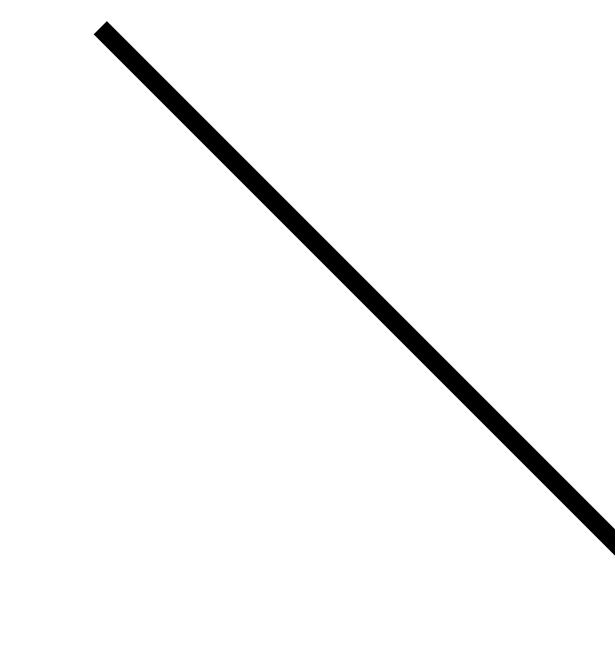


Варианты итеративной обрезки



With continued training

Продолжаем учить от весов,
которые получились на
предыдущем шаге

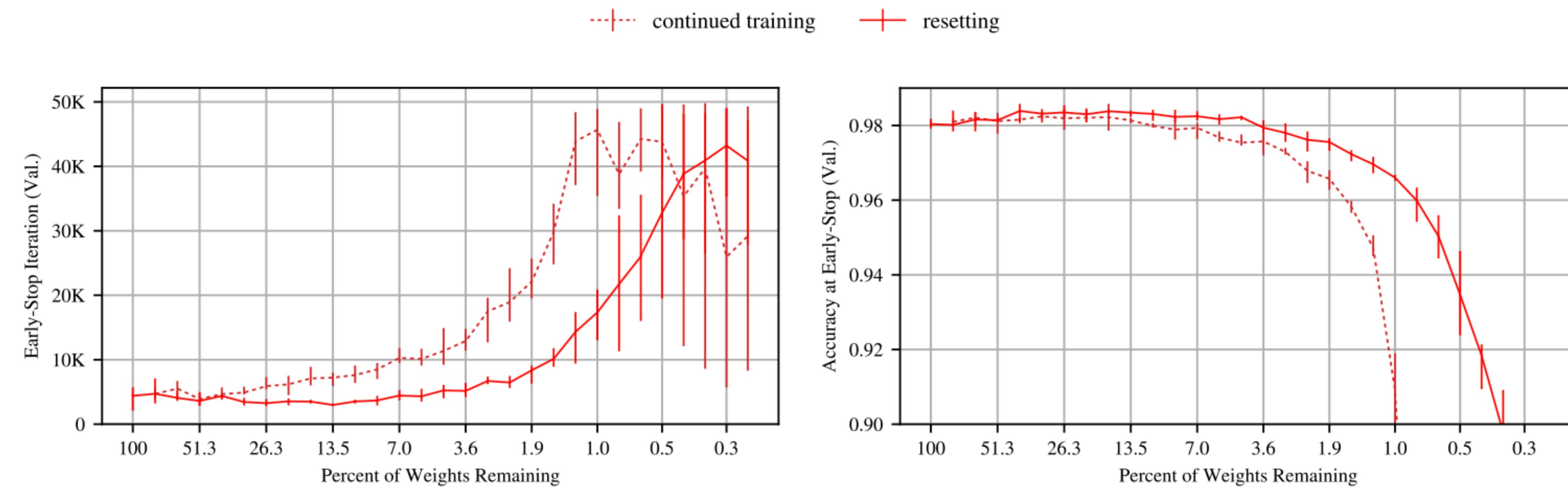


With resetting

Каждую итерацию возвращаем
веса к исходным

Варианты итеративной обрезки

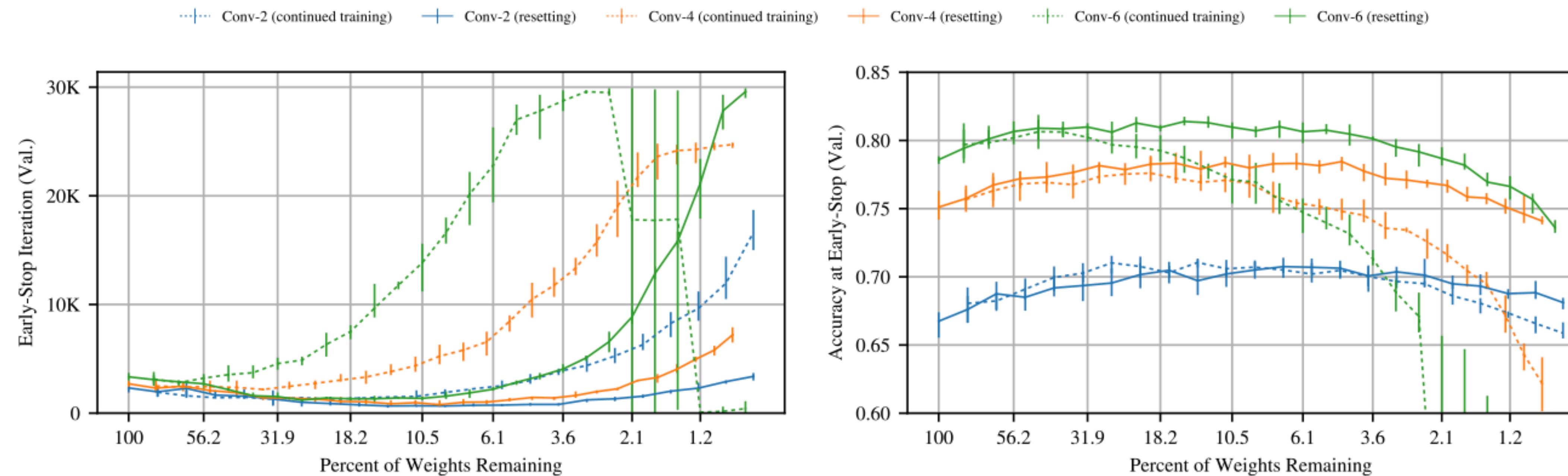
Результаты для Lenet



Вывод: 2й вариант лучше

Варианты итеративной обрезки

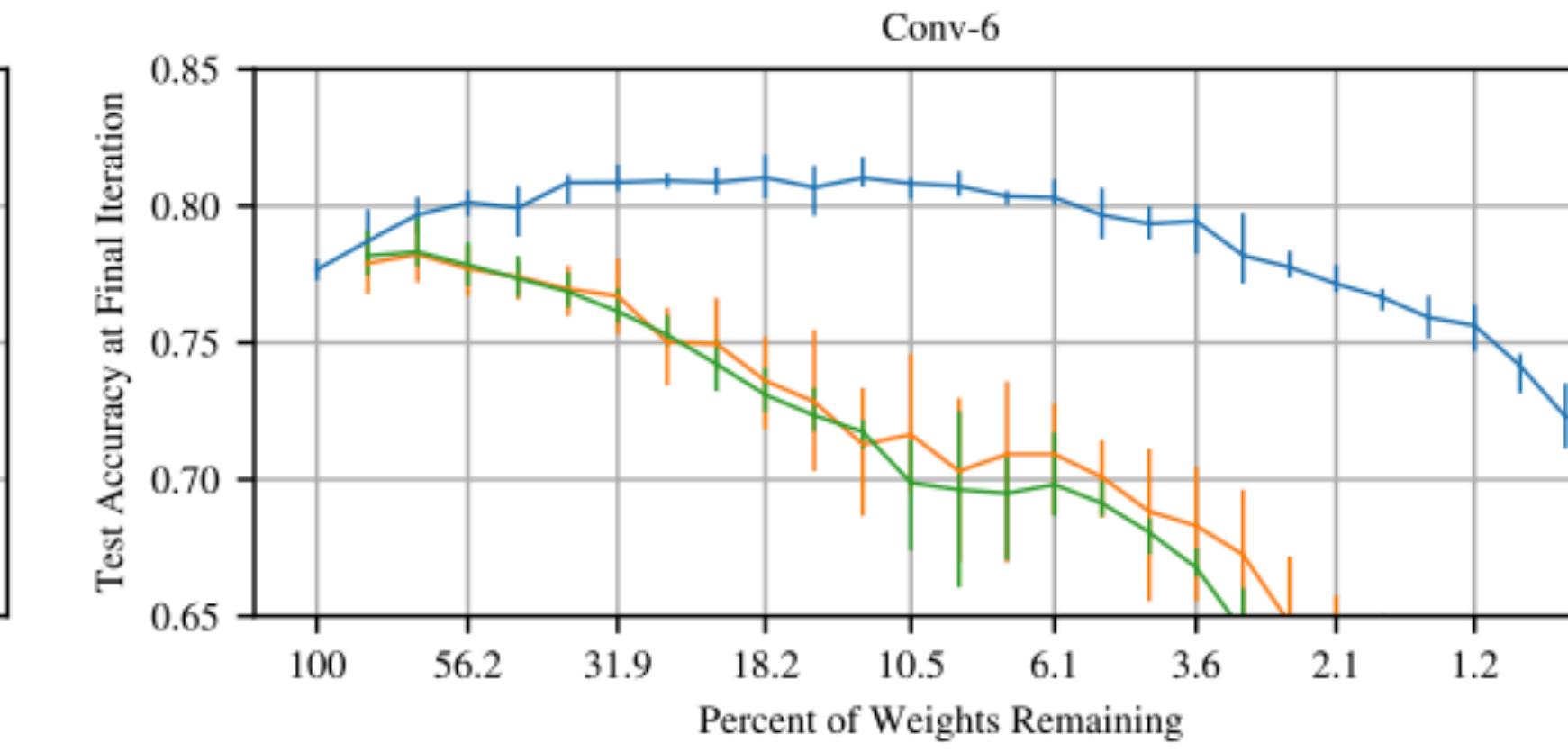
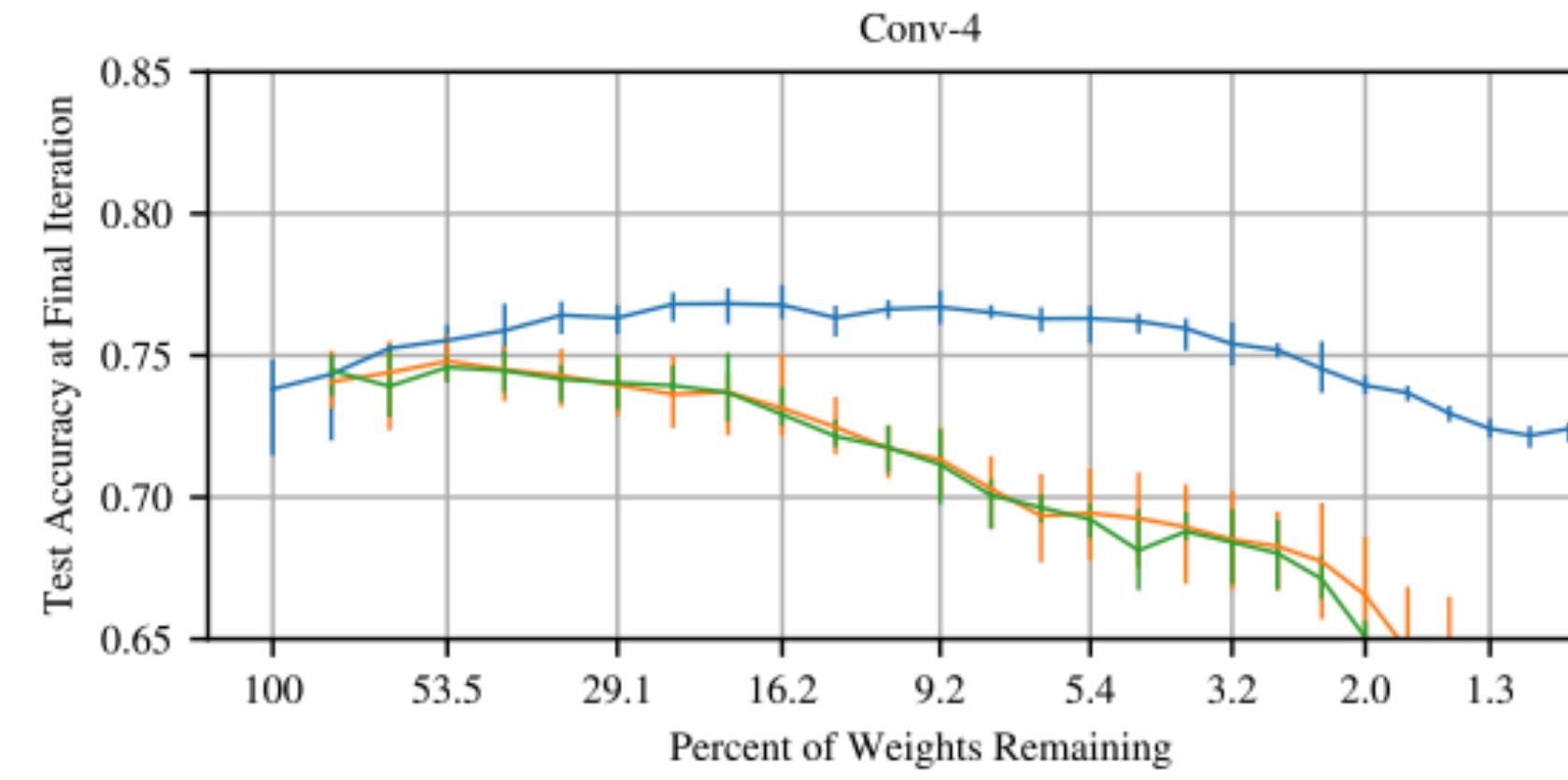
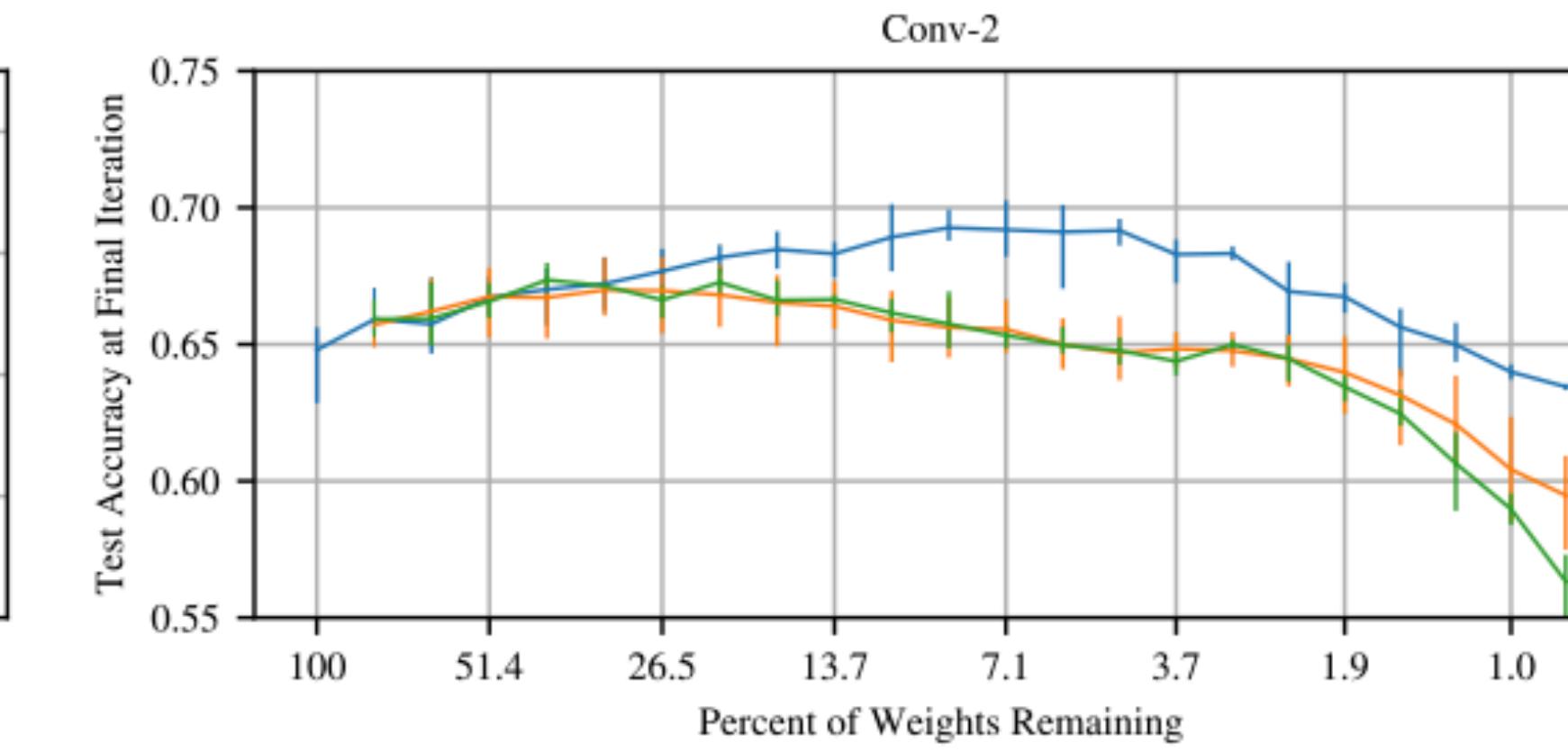
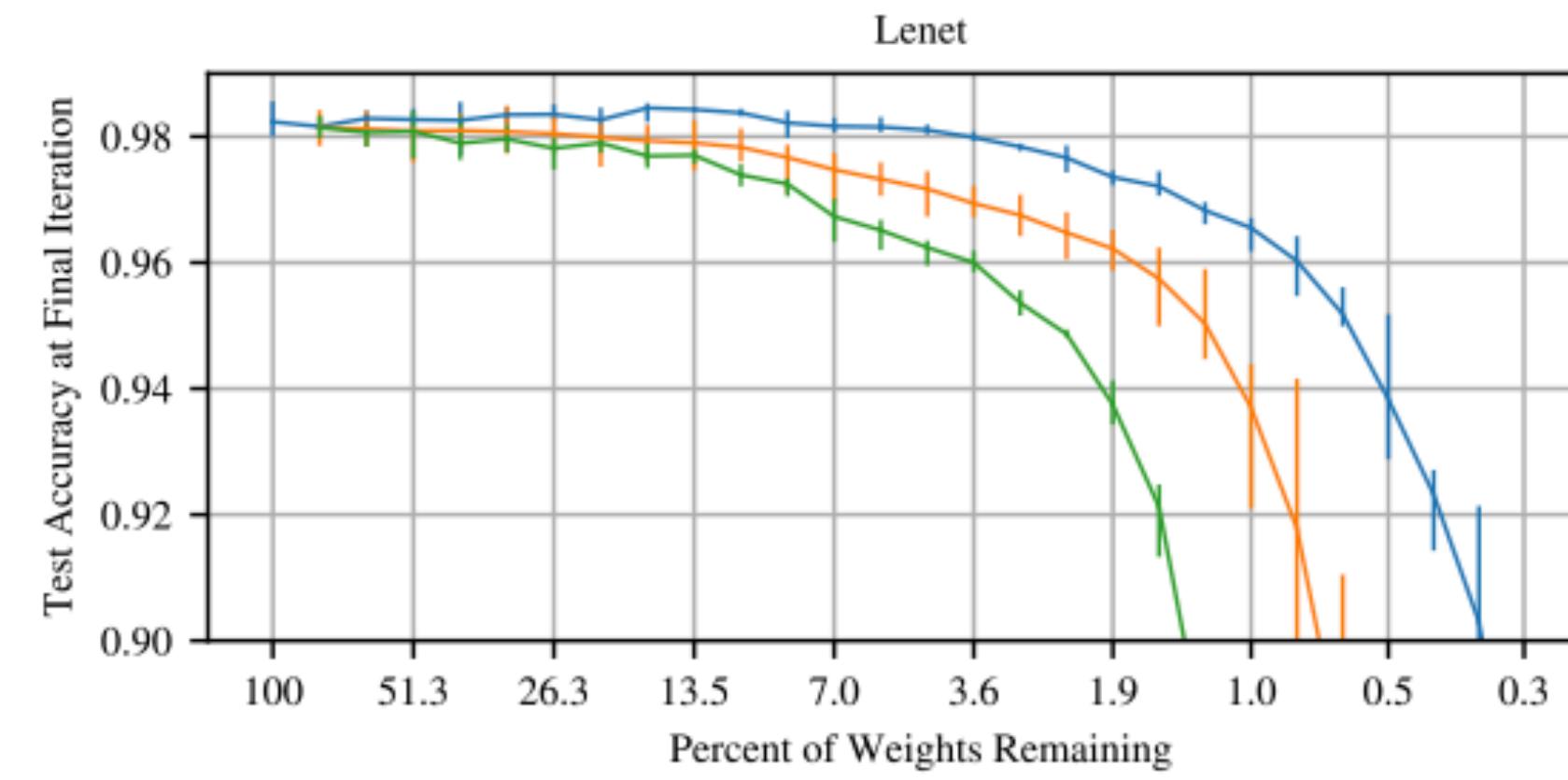
Результаты для Conv-2/4/6



Изучаем особенности «выигрышных билетов»

Важна ли структура сама по себе?

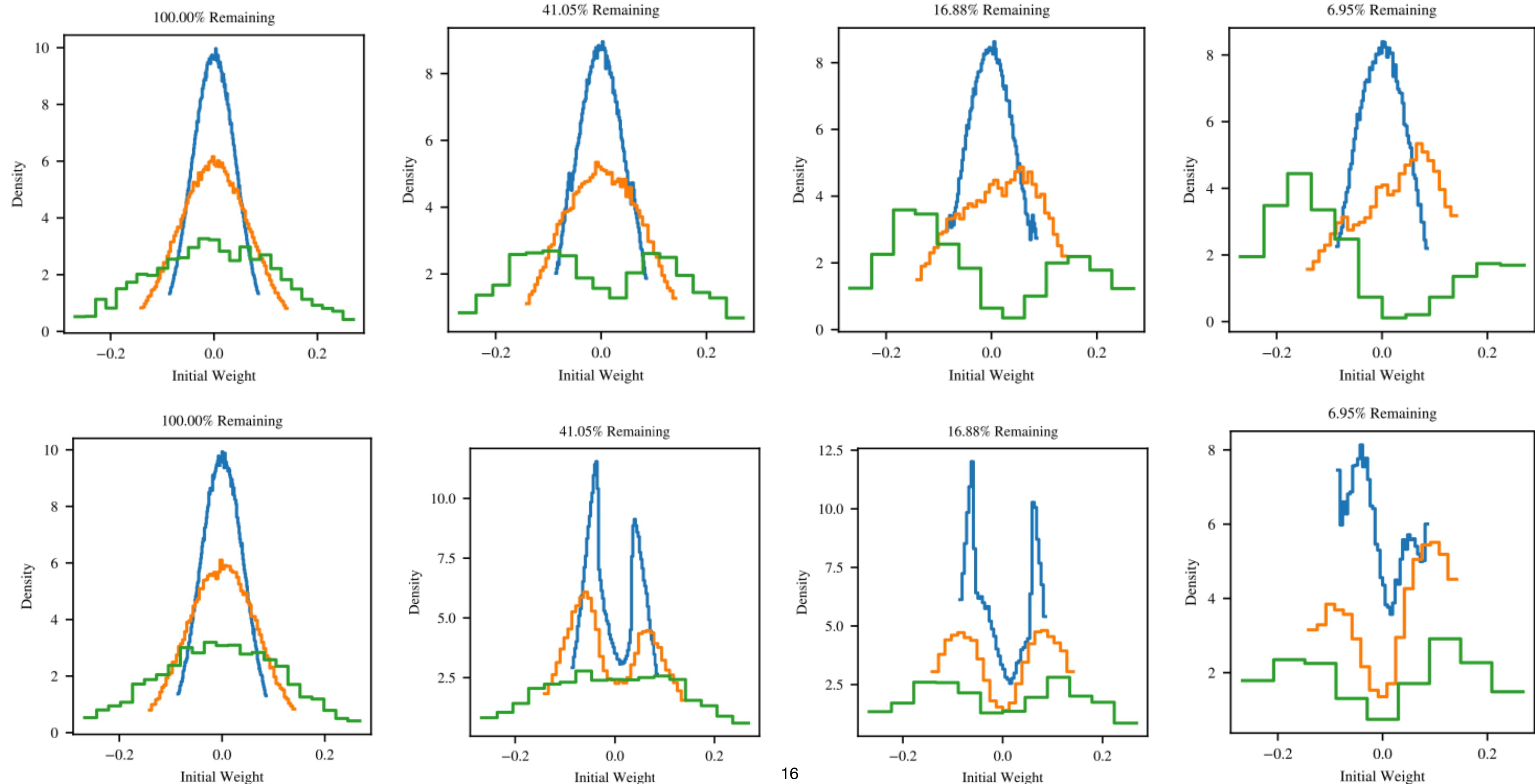
+ original initialization + random reinitialization + random sparsity



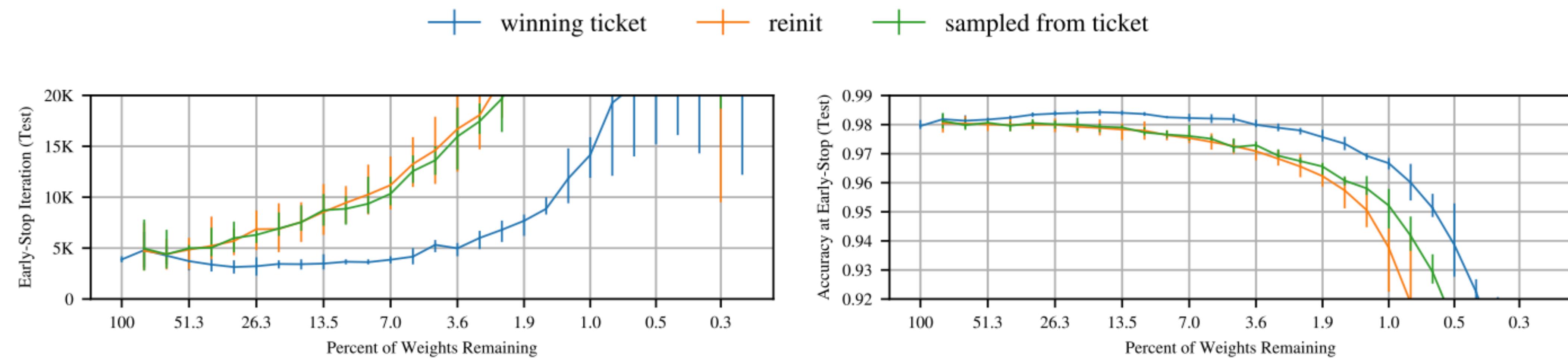
Вывод: зависит от данных

Посмотрим на распределения параметров

5



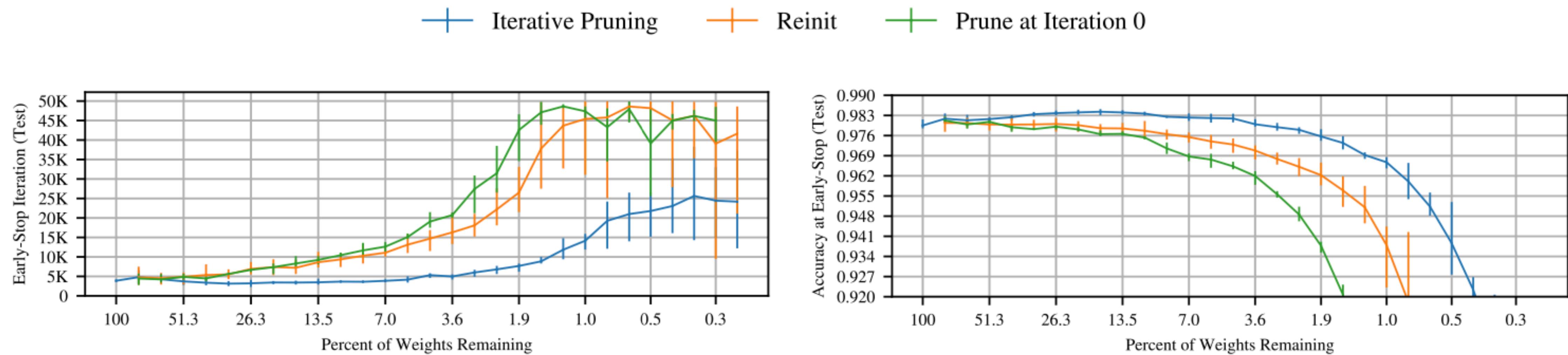
А что, если сэмплировать из этого распределения?



Вывод: не сильно помогает

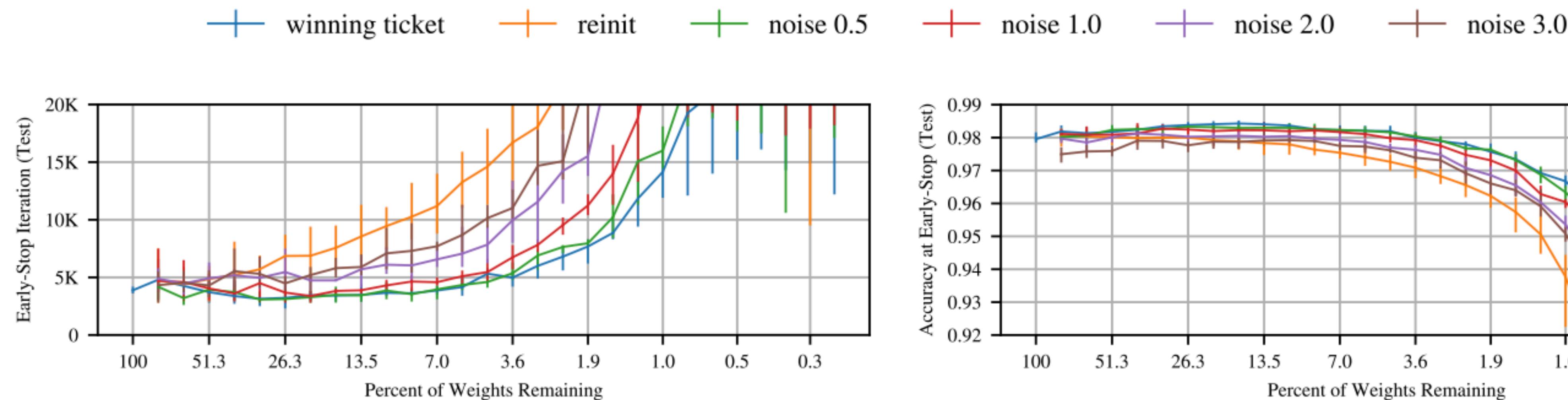
5

А что, если обрезать маленькие веса еще до обучения?



Вывод: совсем плохо

А что, если добавить шум?



Вывод: шум не так сильно влияет

Выводы

- Некоторые сети можно уменьшить в 10 раз, а качество не ухудшится
- Очень важно, что веса остаются теми же
- Структура «выигрышных билетов» может что-то сказать о данных(возможно)
- «Выигрышные билеты» имеют лучшую обобщающую способность
- Пока все еще непонятно, как находить их без обучения большой модели

Источники

- <https://arxiv.org/abs/1803.03635>