One Initialization to Rule Them All?

Stefan Wijnja, Ellis Wierstra, Thomas Hamburger

January 28, 2020

Neural networks are very cool

But they can be very demanding

- Biggest network has 16 million neurons
  - A powerful desktop needs 12 days to train 100.000 neurons
  - Would take more than 5 years to train
- Le et al (2011)
  - Cluster of 1,000 servers
  - 16,000 CPU cores
  - Ran for 3 days

The Lottery Ticket Hypothesis

A randomly initialized, dense neural-network contains a subnetwork that is initalized such that – when trained in isolation – it can match the test accuracy of the original network after training for at most the same number of iterations.

[@frankle2019]

The Lottery Ticket Hypothesis (redux)

A big net contains a small net that can match the big net.

Finding winning tickets by pruning

- Pruning
- Magnitude-based pruning
- Iterative magnitude-based pruning

Status quo on pruning

TODO

- [@frankle2019]
  - "Winning tickets"
- *80-90% size reduction*
  - Learn faster
  - Higher test accuracy
- [@morcos2019]
  - Finding winning tickets is computationally expensive
    - Solution: possible to reuse WTs across datasets
    - Generalise without performance loss

Our question

How do initialization algorithms compare when looking for winning tickets?

Method

Plan

1. Pick dataset.
2. Pick model.
3. Pick initialization methods.
4. **Write the code.**
5. **Do lots of experiments.**
6. Answer the question.

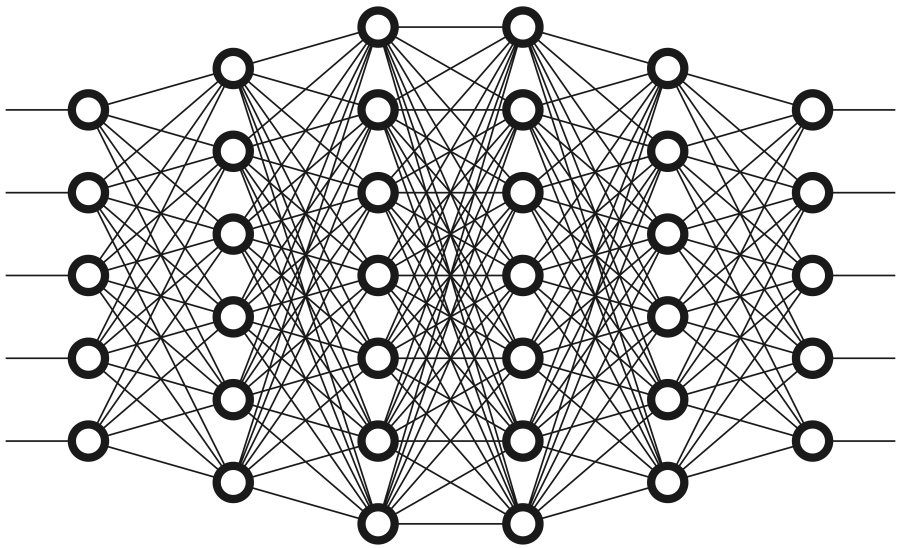
Method

Dataset: MNIST

- Handwritten digits
- 28×28 pixels
- Training set: 60k
- Test set: 10k

Method

Model: LeNet [@lecun1998]



- Fully connected
- Two hidden layers: 300 & 100 neurons $\rightarrow$ 266k weights.
- Leaky ReLU (negative slope: 0.05)

Method

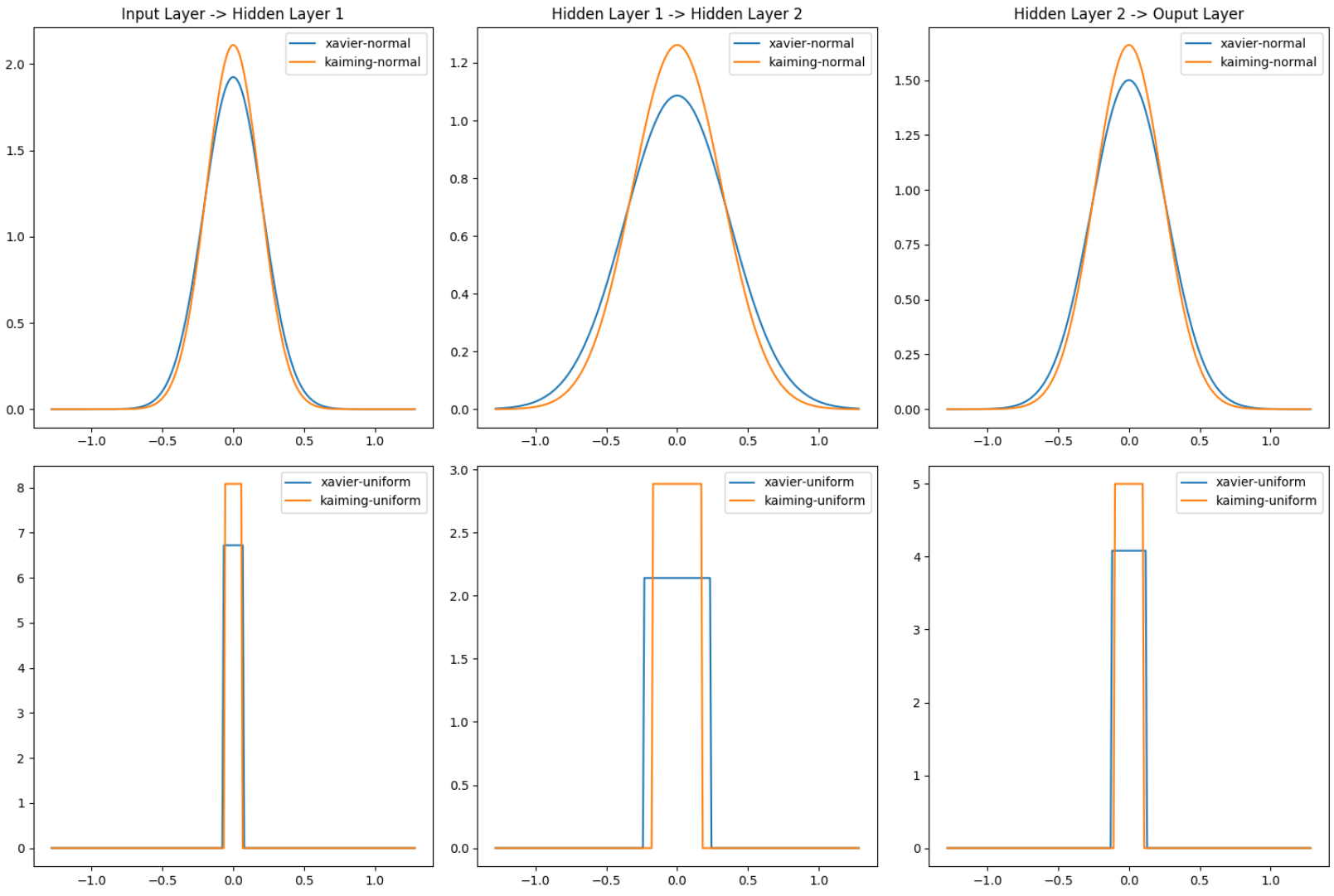
Initialization Methods

	Xavier	Kaiming
$\mathcal{U}(-a, a)$	$\sqrt{\frac{6}{\text{fan\_in}+\text{fan\_out}}}$	$\sqrt{\frac{3}{\text{fan\_mode}}}$
$\mathcal{N}(0, \text{var})$	$\sqrt{\frac{2}{\text{fan\_in}+\text{fan\_out}}}$	$\frac{1}{\sqrt{\text{fan\_mode}}}$

- `fan_in`/`fan_out`: number of inputs to/outputs from a neuron. `fan_mode`: `fan_in` or `fan_out`.

Method

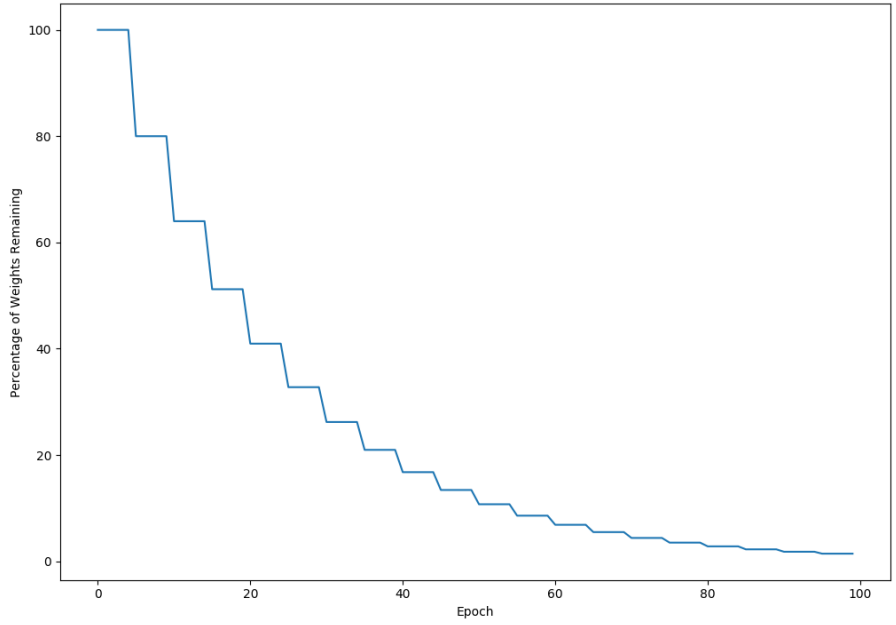
Initialization Methods' Probability Density Functions



Method

Training

1. Initialize network and save weights.
2. Train for 5 epochs.
3. Trim smallest 20% of the weights.
4. Reset the weights to the saved ones.
5. `GO TO 2`

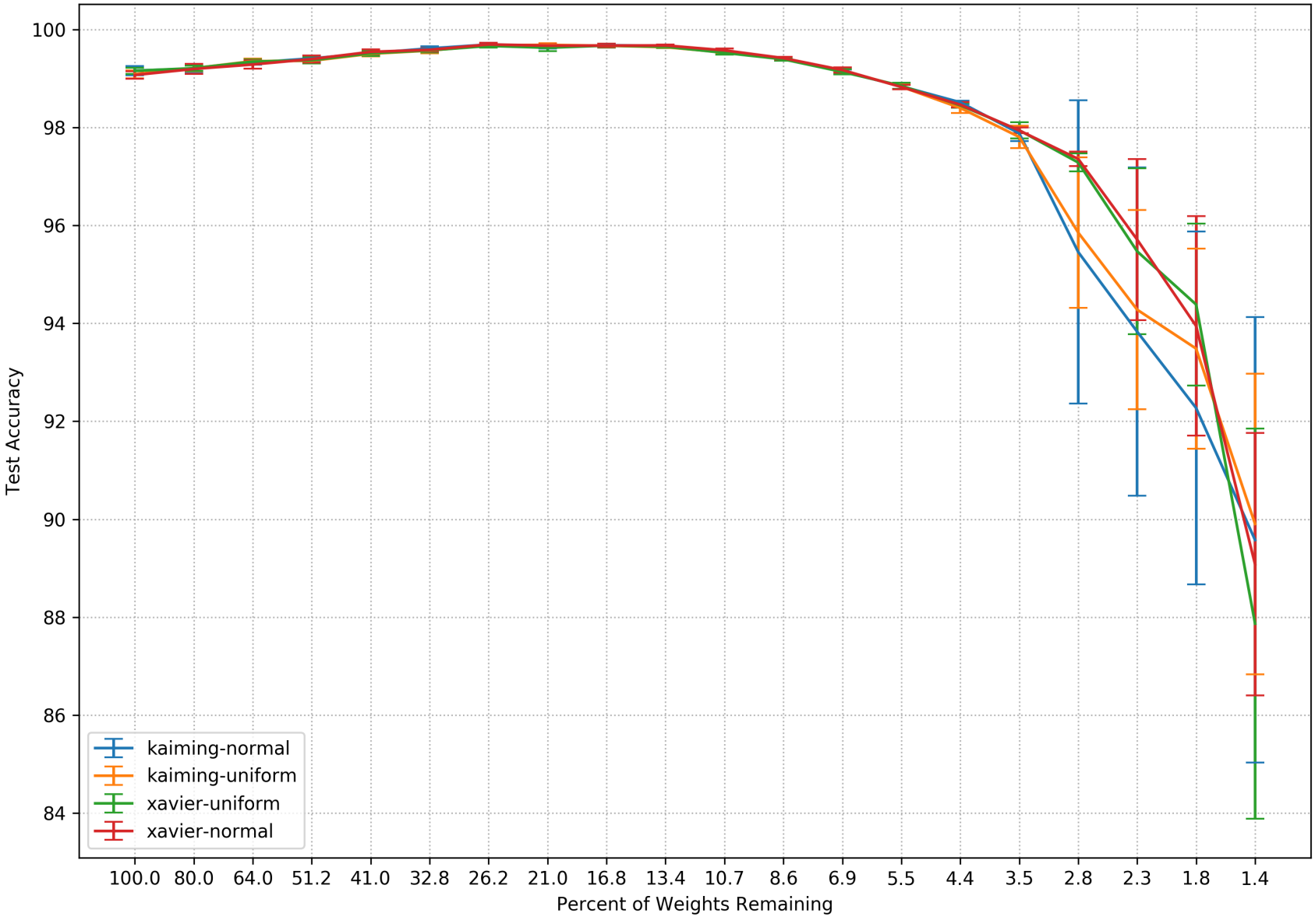


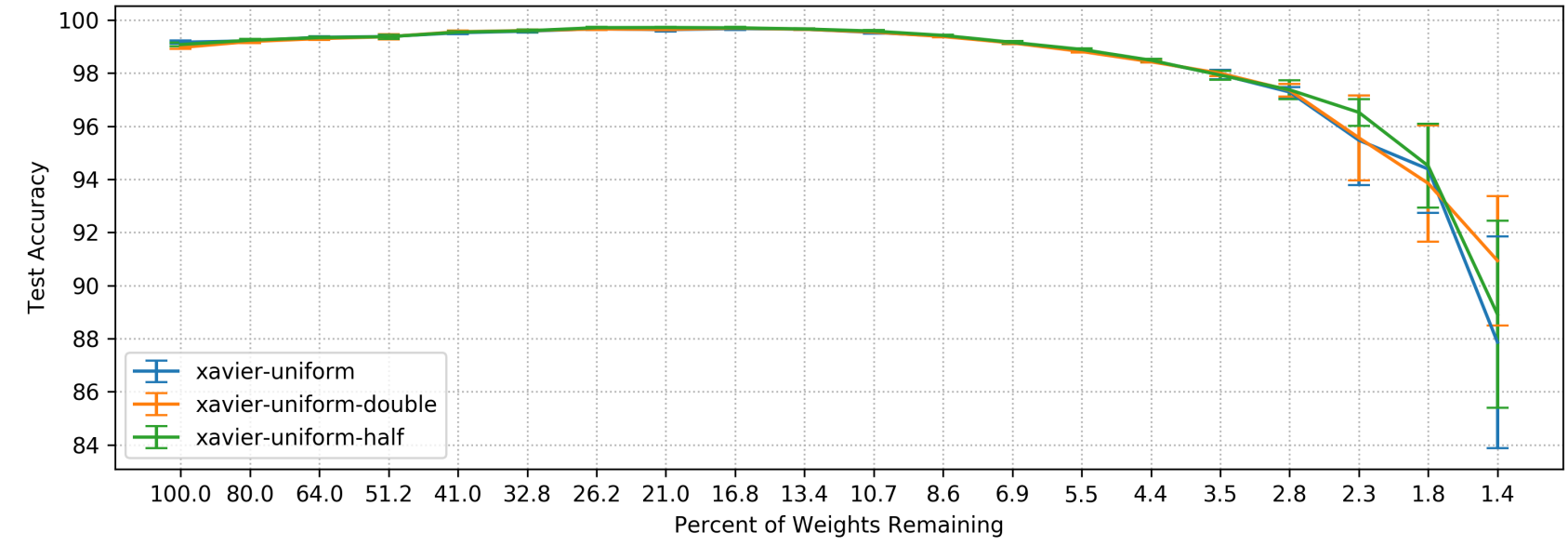
Method

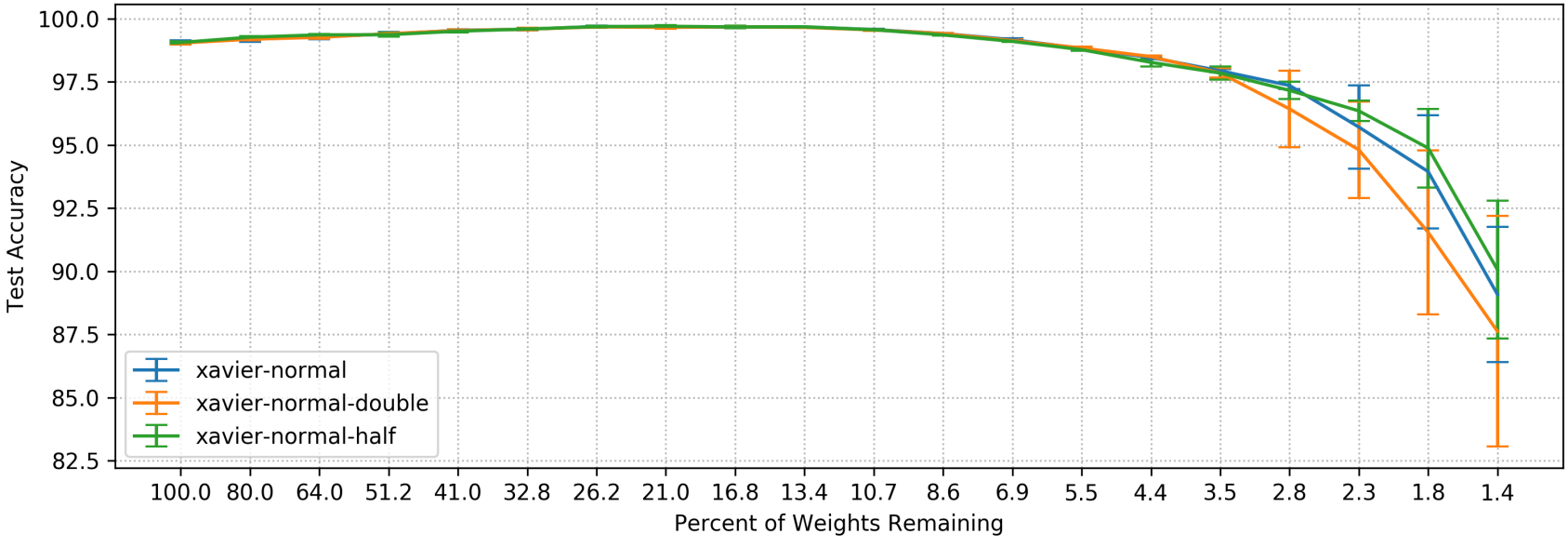
Testing

- Save loss and accuracy on the training and testing set after every epoch.

Results


Test Accuracy
100
98
96
94
92
90
88
86
84
100.0 80.0 64.0 51.2 41.0 32.8 26.2 21.0 16.8 13.4 10.7 8.6 6.9 5.5 4.4 3.5 2.8 2.3 1.8 1.4
Percent of Weights Remaining
kaiming-normal
kaiming-uniform
xavier-uniform
xavier-normal


Test Accuracy
100
98
96
94
92
90
88
86
84
xavier-uniform
xavier-uniform-double
xavier-uniform-half
100.0 80.0 64.0 51.2 41.0 32.8 26.2 21.0 16.8 13.4 10.7 8.6 6.9 5.5 4.4 3.5 2.8 2.3 1.8 1.4
Percent of Weights Remaining


Test Accuracy
100.0
97.5
95.0
92.5
90.0
87.5
85.0
82.5
xavier-normal
xavier-normal-double
xavier-normal-half
100.0 80.0 64.0 51.2 41.0 32.8 26.2 21.0 16.8 13.4 10.7 8.6 6.9 5.5 4.4 3.5 2.8 2.3 1.8 1.4
Percent of Weights Remaining

Question

How do initialization algorithms compare when looking for winning tickets?

Answer*

$$\text{Xavier} > \text{Kaiming}$$

$$\text{Uniform} \approx \text{Normal}$$

$$\text{Narrow} > \text{wide}$$

Further Research

- Grid search other datasets/models/pruning settings.
- Are pruning method and initialization method connected?

Questions?

- Thanks
  - Andrei Apostol
  - Putri van der Linden and Lieuwe Rekker
  - BrainCreators
- Stats
  - ~599 lines of code
  - 122 training runs
- Frameworks
  - PyTorch
- Can I reproduce your results?
  - Please do. Email s.wijnja@me.com

References