# What's Hidden in a Randomly Weighted Neural Network?

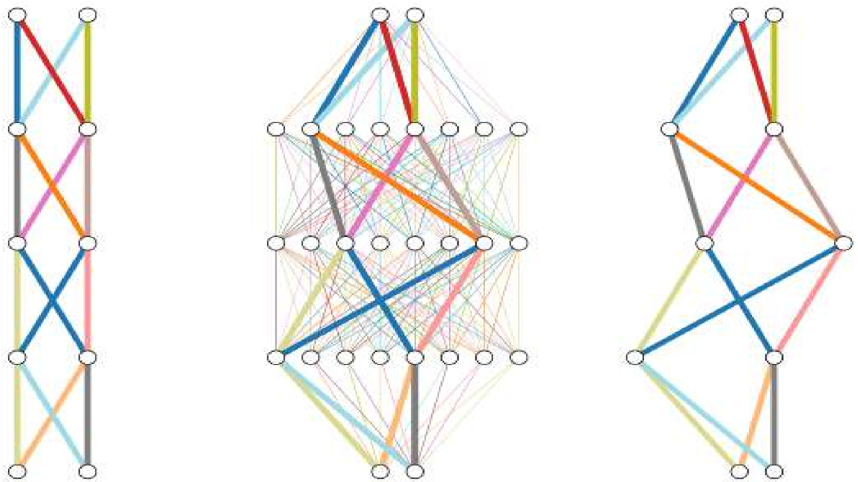
Huihua Yang | 04/15/2020

# Introduction

1. **arXiv:1911.13299v2 [cs.CV] 31 Mar 2020.**

2. **"Randomly weighted neural networks contain subnetworks which achieve impressive performance without ever modifying the weight values."**

# The Existence of Good Subnetworks.



A neural network $\tau$ which achieves good performance

Randomly initialized neural network $N$

A subnetwork $\tau'$ of $N$

Figure 1. If a neural network with random weights (center) is sufficiently overparameterized, it will contain a subnetwork (right) that perform as well as a trained neural network (left) with the same number of parameters.

**The Intuition of the existence of good subnetworks.**

**The probability is extremely small, but it is still nonzero.**

**How to find a good subnetwork?**

# The edge-popup Algorithm

A fully connected neural network consists of layers $1, ..., L$ where layer $\ell$ has $n_\ell$ nodes $\mathcal{V}^{(\ell)} = \{v_1^{(\ell)}, ..., v_{n_\ell}^{(\ell)}\}$. We let $\mathcal{I}_v$ denote the input to node $v$ and let $\mathcal{Z}_v$ denote the output, where $\mathcal{Z}_v = \sigma(\mathcal{I}_v)$ for some non-linear activation function $\sigma$

The input to neuron $v$ in layer $\ell$ is:

$$\mathcal{I}_v = \sum_{u \in \mathcal{V}^{(\ell-1)}} w_{uv} \mathcal{Z}_u \qquad (2)$$

where $w_{uv}$ are the network parameters for layer $\ell$.

the weights $w_{uv}$ for layer $\ell$ are initialized by independently sampling from distribution $\mathcal{D}_\ell$.

keep the weights at their random initialization, and optimize to find a subnetwork $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We then compute the input of node $v$ in layer $\ell$ as

$$\mathcal{I}_v = \sum_{(u,v) \in \mathcal{E}} w_{uv} \mathcal{Z}_u \qquad (3)$$

where $\mathcal{G}$ is a subgraph of the original fully connected network

For each weight $w_{uv}$ in the original network, learn a popup score $s_{uv}$ . choose the subnetwork G by selecting the weights in each layer which have the top-k% highest scores. Where k is a parameter.

$$\mathcal{I}_v = \sum_{u \in \mathcal{V}^{(\ell-1)}} w_{uv} \mathcal{Z}_u h(s_{uv}) \qquad (4)$$

where $h(s_{uv}) = 1$ if $s_{uv}$ is among the top $k\%$ highest scores in layer $\ell$ and $h(s_{uv}) = 0$ otherwise.

# The edge-popup Algorithm

the gradient to $s_{uv}$ as

$$\hat{g}_{s_{uv}} = \frac{\partial \mathcal{L}}{\partial \mathcal{I}_v} \frac{\partial \mathcal{I}_v}{\partial s_{uv}} = \frac{\partial \mathcal{L}}{\partial \mathcal{I}_v} w_{uv} \mathcal{Z}_u \tag{5}$$

where $\mathcal{L}$ is the loss we are trying to minimize. The scores $s_{uv}$ are then updated via stochastic gradient descent with learning rate $\alpha$. If we ignore momentum and weight decay then we update $s_{uv}$ as

$$\tilde{s}_{uv} = s_{uv} - \alpha \frac{\partial \mathcal{L}}{\partial \mathcal{I}_v} w_{uv} \mathcal{Z}_u \tag{6}$$

where $\tilde{s}_{uv}$ denotes the score after the gradient step

# Experiment-setup

1. They used two different distributions for the weights in their network:

- **Kaiming Normal**: $N_k = N(0, \sqrt{2/n_{l-1}})$, where N denotes the normal distribution.

- **Signed Kaiming Constant** which we denote $U_k$. Here we set each weight to be a constant and randomly choose its sign to be $+$ or $-$. The constant we choose is the standard deviation of Kaiming Normal, and as a result the variance is the same. We use the notation $U_k$ as we are sampling uniformly from the set $\{-\sigma_k, \sigma_k\}$ where $\sigma_k$ is the standard deviation for Kaiming Normal (*i.e.* $\sqrt{2/n_{\ell-1}}$).

2. Datasets: CIFAR-10 & ImageNet.
3. Network: simple VGG-like architectures of varying depth.
4. Optimize with Adam & SGD

Table 1. The structure of Network

| Model | Conv2 | Conv4 | Conv6 | Conv8 |
|---|---|---|---|---|
| Conv Layers | 64, 64, pool | 64, 64, pool<br>128, 128, pool | 64, 64, pool<br>128, 128, pool<br>256, 256, pool | 64, 64, pool<br>128, 128, pool<br>256, 256, pool<br>512, 512, pool |
| FC | 256, 256, 10 | 256, 256, 10 | 256, 256, 10 | 256, 256, 10 |

# Experiment-Varying the k% of Weights

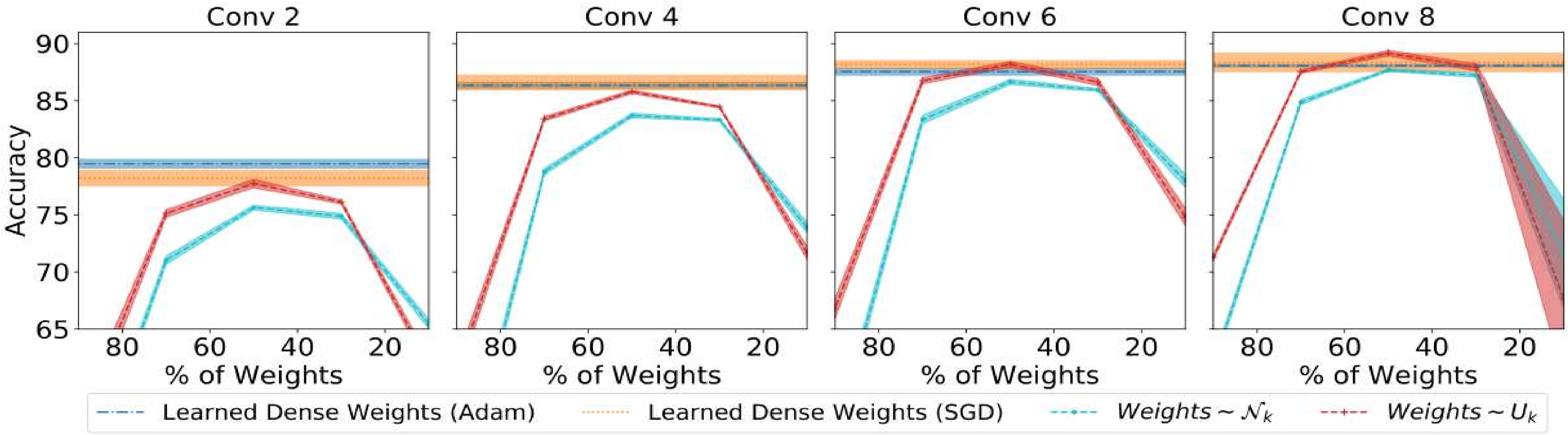



Figure 3. **Going Deeper:** Experimenting with shallow to deep neural networks on CIFAR-10 [13]. As the network becomes deeper, we are able to find subnetworks at initialization that perform as well as the dense original network when trained. The baselines are drawn as a horizontal line as we are not varying the % of weights. When we write *Weights* $\sim \mathcal{D}$ we mean that the weights are randomly drawn from distribution $\mathcal{D}$ and are never tuned. Instead we find subnetworks with size (% of Weights)/100 * (Total # of Weights).

1. Worst accuracy when k approaches 0 or 100.
2. The best accuracy occurs when k $\in$ [30,70]

# Experiment-Varying the Width

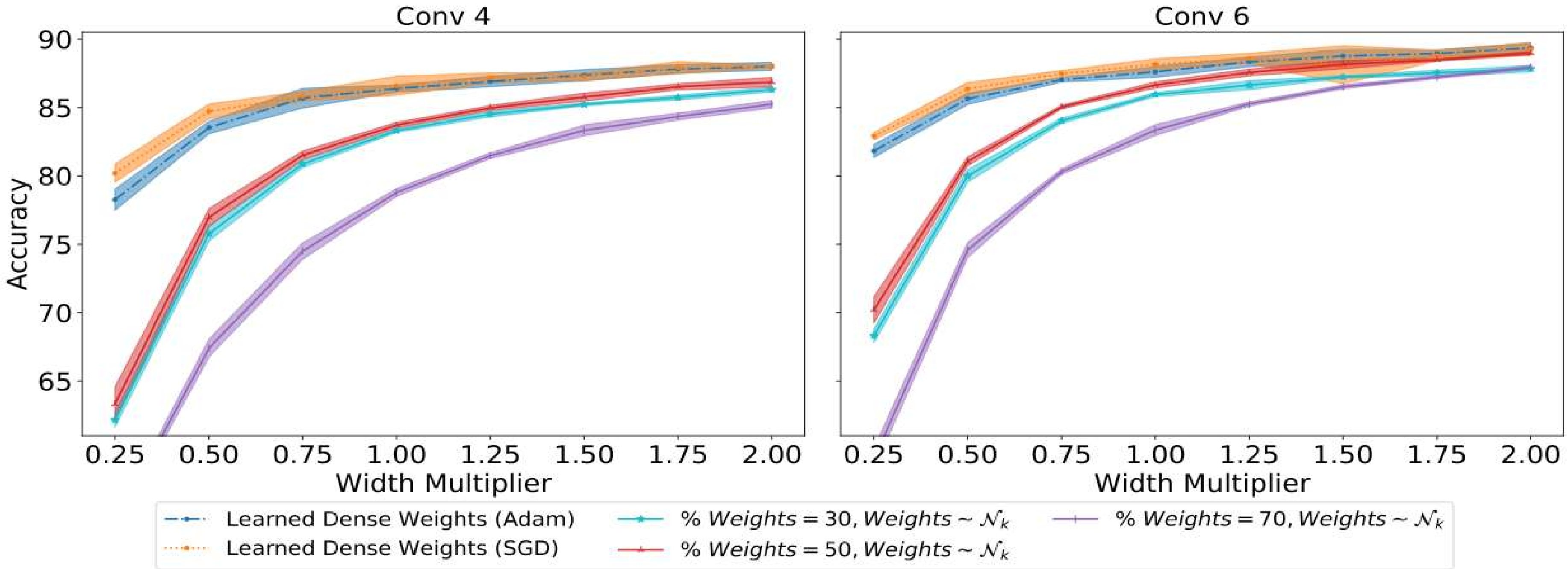



Figure 4. **Going Wider:** Varying the width (*i.e.* number of channels) of Conv4 and Conv6 for CIFAR-10 [13]. When Conv6 is wide enough, a subnetwork of the randomly weighted model (with $\%Weights = 50$) performs just as well as the full model when it is trained.

1. As the width multiplier increases, the gap of accuracy shrinks.
2. When Conv6 is wide enough, a subnetwork of the randomly weighted model (with %Weights = 50) performs just as well as the dense model when it is trained.

# Experiment-Varying the Width

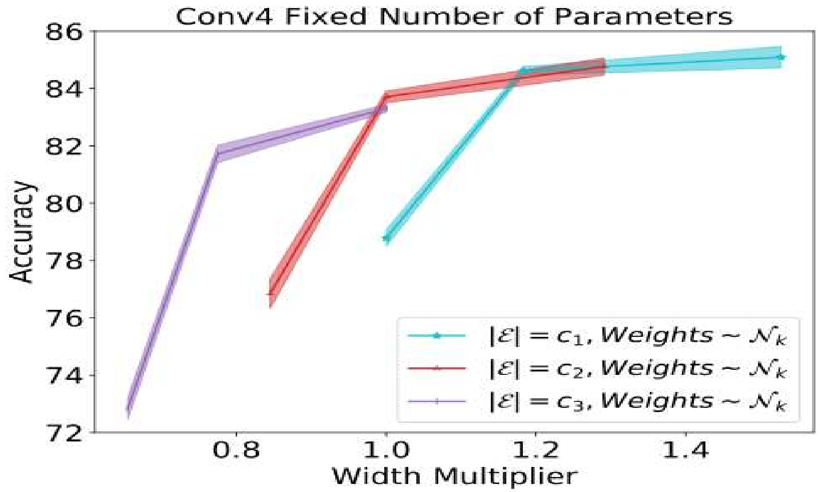



Figure 5. Varying the width of Conv4 on CIFAR-10 [13] while modifying $k$ so that the # of Parameters is fixed along each curve. $c_1, c_2, c_3$ are constants which coincide with # of Parameters for $k = [30, 50, 70]$ for width multiplier 1.

1. When the # of parameters is fixed, increasing the width and therefore the search space leads to better performance.
2. The boost in performance is not solely from the subnetwork having more parameters.

# Experiment- Effect of The Distribution

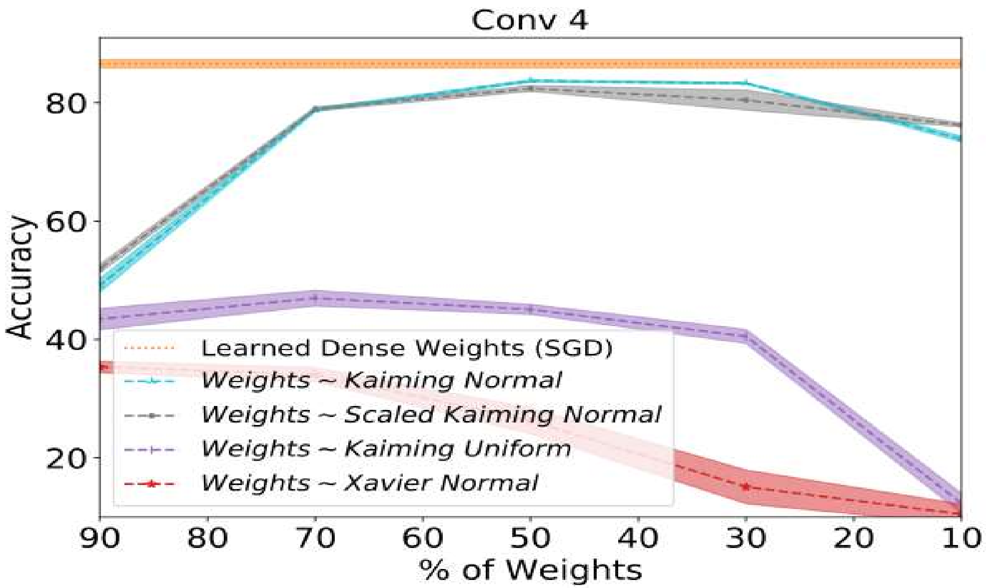



Figure 7. Testing different weight distributions on CIFAR-10 [13].

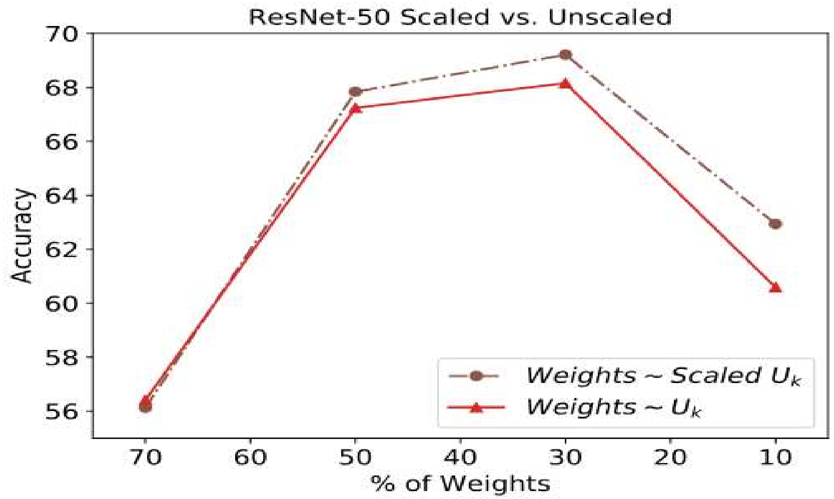



Figure 10. Examining the effect of using the "Scaled" initialization detailed in Section 4.5 on ImageNet.

1. The distribution that the random weights are sampled from is very important to the authors' algorithm.(Figure 7)
2. The choice of the random distribution matters more for ImageNet. The "Scaled" distribution on ImageNet performs much better.(Figure 10)

# Experiment- ImageNet Experiments

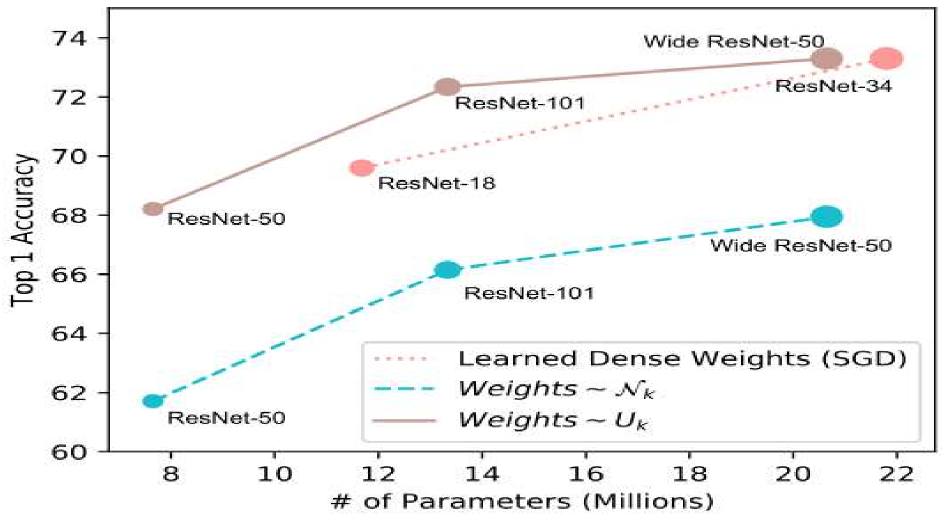



Figure 8. Testing our Algorithm on ImageNet [4]. We use a fixed $k = 30\%$, and find subnetworks within a randomly weighted ResNet-50 [9], Wide ResNet-50 [32], and ResNet-101. Notably, a randomly weighted Wide ResNet-50 contains a subnetwork which is smaller than, but matches the performance of ResNet-34. Note that for the non-dense models, # of Parameters denotes the size of the subnetwork.

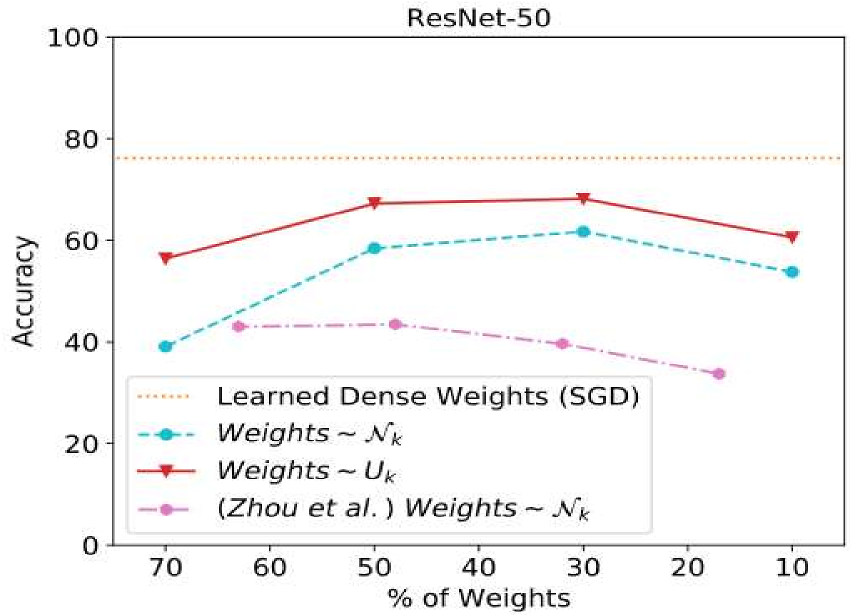



Figure 9. Examining the effect of % *weights* on ImageNet for `edge-popup` and the method of Zhou *et al*.

1. A randomly weighted Wide ResNet-50 contains a subnetwork that is smaller than but matches the accuracy of ResNet-34 when trained on ImageNet.
2. k∈[30,70] performs best though 30 now provides the best performance

Rensselaer

why not change the world?®

# Experiment-Comparison with Zhou et al.

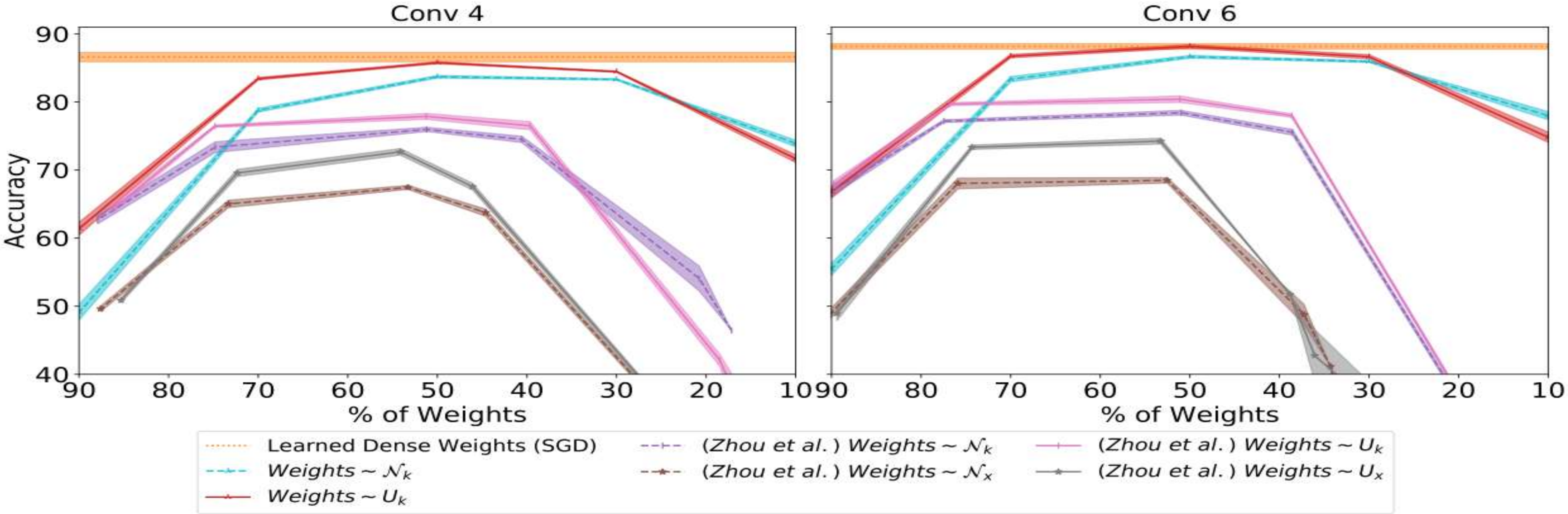



Figure 6. Comparing the performance of `edge-popup` with the algorithm presented by Zhou *et al.* [33] on CIFAR-10 [13].

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the super-mask, 2019. 1, 2, 3, 5, 6, 7

Zhou's points: winning tickets achieve better than random performance without training. Lottery Tickets and Super-masks

1. A significant improvement by changing Xavier normal to Kaiming normal with Zhou's algorithm.
2. The author's accuracy exceed the Zhou's even using Kaiming normal.