EFFECTIVE TRAINING OF SPARSE NEURAL NETWORKS UNDER GLOBAL SPARSITY CONSTRAINT

Anonymous authors
Paper under double-blind review

ABSTRACT

Weight pruning is an effective technique to reduce the model size and inference time for deep neural networks in real-world deployments. However, since magnitudes and relative importance of weights are very different for different layers of a neural network, existing methods rely on either manual tuning or handcrafted heuristic rules to find appropriate pruning rates individually for each layer. This approach general leads to suboptimal performance. In this paper, by directly working on the probability space, we propose an effective network sparsification method called *probabilistic masking* (ProbMask), which solves a natural sparsification formulation under global sparsity constraint. The key idea is to use probability as a global criterion for all layers to measure the weight importance. An appealing feature of ProbMask is that the amounts of weight redundancy can be learned automatically via our constraint and thus we avoid the problem of tuning pruning rates individually for different layers in a network. Extensive experimental results on CIFAR-10/100 and ImageNet demonstrate that our method is highly effective, and can outperform previous state-of-the-art methods by a significant margin, especially in the high pruning rate situation. Notably, the gap of Top-1 accuracy between our ProbMask and existing methods can be up to 10%. As a by-product, we show ProbMask is also highly effective in identifying supermasks, which are subnetworks with high performance in a randomly weighted dense neural network.

1 INTRODUCTION

Weight pruning (Han et al., 2015a) is a popular technique for alleviating the weight redundancy in deep neural networks (DNNs) to improve inference efficiency and decrease computation demands. Typical pruning algorithms usually prune the unimportant weights by developing proper criteria. It is repeatedly reported in the literature (Guo et al., 2016; Liu et al., 2018; Zeng & Urtasun, 2018; Li et al., 2016) that by pruning one can reduce the neural network size and improve the inference efficiency significantly with quite slight or even negligible loss on performance, which makes deploying large-scale DNNs on equipment with limited computational and memory budget possible.

What can serve as a suitable global comparator to measure weight importance and identify sparsity level for different layers is a long-standing problem (Gale et al., 2019) though impressive results have been achieved. We know that the core module in pruning is the explicit or implicit criterion for identifying the redundant weights, and it is difficult to develop a global criterion for the weights in all the layers. For example, in Han et al. (2015a), the authors propose a simple yet effective criterion, i.e., for each layer it prunes all the weights below a certain threshold in a fully trained network. The threshold is obtained by sorting weight by its magnitude and retrieving the weight magnitude at the target pruning rate. The criterion is weight magnitude in this case. Notice that the magnitudes of the weights across layers could be quite different and different layers could have different amount of redundancy. If we use a global threshold for all the layers, then almost all the weights in certain layers could be pruned in order to achieve high enough pruning ratio, which will be verified in Section 4.3. Thus, we need to set a proper threshold or pruning ratio for each layer individually. In the networks with numerous layers, it is very difficult and even impossible to find the optimal thresholds or pruning ratios for all the layers manually. One reasonable compromise for such dilemma is to set sparsity level uniformly for different layers. However, this results in imperfect weight allocation obviously and gives unsatisfactory results on high pruning rates. Recently Kusupati et al. (2020) also notice the drawback of uniform weight allocation and gives considerable improvements by learnable sparsity. However, its performance is limited by both the modification of original network pruning problem and the imperfect manual choice of threshold function.

In this paper, to address the above limitations, we propose an effective network sparsification method called *probabilistic masking* (ProbMask). Firstly, we know that network pruning can be naturally formulated into a problem of finding a sparse binary mask $\boldsymbol{m}$ as well as the weights at the same time to minimize the empirical loss (1). If the component m_i is equal to 0, it means that the corresponding weight is pruned. However, it is a discrete optimization problem and hard to solve. We notice that if we view the components $\boldsymbol{m}_i$ in the mask as independent Bernoulli random variables with probability s_i being 1 and probability $(1-s_i)$ being 0 and reparameterize them w.r.t. its probability, then the loss in problem (1) would become continuous over the probability space. Due to the nature of probability, probability can be used as a global criterion in all the layers. Therefore, we can control the model size via forcing the sum of all the probabilities s_i of the mask smaller than a proper value, leading to a global sparsity constraint in the probability space. In this way, the discrete optimization problem (1) is transformed into a constrained expected loss minimization problem (2) over a probability space, which is continuous. Finally, we adopt the Gumbel-Softmax trick to solve the continuous problem. As the optimizer goes on, the probabilities s_i would converge to either 0 or 1, i.e., $\boldsymbol{m}$ would become close to a deterministic sparse mask. Thus, a fully trained mask would have quite low variance, making the loss of the sampled sparse network according to $\boldsymbol{m}$ close to the excepted loss in problem (2). Another appealing feature of our proposed method is that the amount of weight redundancy in each layer can be identified automatically by our global sparsity constraint and thus we do not need to choose different pruning ratios for different layers.

Experimental results on network pruning and supermask (Zhou et al., 2019) finding demonstrate that our method is much more effective than the state-of-the-art methods on both small scale datasets and large scale datasets and can outperform them with a significant margin when the pruning rate is high.

The contribution and novelty of ProbMask can be summarized as follows:

1) We provide evidence showing that probability can serve as a suitable global comparator to measure weight importance and identify sparsity level for different layers, which is a long-standing problem (Gale et al., 2019).

2) We present a natural formulation of global sparsity constraint, and an optimization method that is practically effective. Our solution fixes the training and testing performance discrepancy problem observed in practice, which led to the failure of previous methods (Louizos et al., 2017) on ImageNet (Gale et al., 2019).

3) We demonstrate the effectiveness of using probability as global comparator on small-scale and large-scale problems and various models and achieve state-of-the-art results on Top-1 accuracy and accuracy-versus-FLOPS curve.

4) We show ProbMask can also serve as a powerful tool for identifying supermasks, which are subnetworks with high performance in a randomly weighted dense neural network, and we achieve state-of-the-art results on Top-1 accuracy on CIFAR-100 under high pruning rates.

Notations: Let $\|\cdot\|_0$, $\|\cdot\|_1$ and $\|\cdot\|_2$ be the ℓ_0, ℓ_1 and ℓ_2 norm of a real valued vector, respectively. We denote $\mathbf{1} \in \mathbb{R}^n$ to be a vector with all components equal to 1. In addition, $\{0,1\}^n$ is the set of n-dimensional vectors with each coordinate valued in $\{0,1\}$.

2 RELATED WORKS

Below, we first review the related work on network pruning. Then we review training methods for obtaining sparse networks which can be divided into two groups: dense-to-sparse training and sparse-to-sparse training.

2.1 NETWORK PRUNING

Network Pruning (Han et al., 2015b; Guo et al., 2016; Zeng & Urtasun, 2018; Li et al., 2016; Luo et al., 2017; He et al., 2017; Zhu & Gupta, 2017; Kang & Han, 2020; Wang et al., 2019; Renda et al., 2020; Ye et al., 2020) has been extensively studied in recent years to reduce the model size and improve the inference efficiency of deep neural networks. Since it is a widely-recognized property that modern neural networks are always over-parameterized, pruning methods are developed to remove unimportant parameters in the fully trained dense networks to alleviate such redundancy. According to the granularity of pruning, existing pruning methods can be roughly divided into two categories, i.e., unstructured pruning and structured pruning. The former one is also called weight pruning, which removes the unimportant parameters in an unstructured way, that is, any element in the weight tensor could be removed. The latter one removes all the weights in a certain group together, such as kernel and filter. Since structure is taken into account in pruning, the pruned networks obtained by structured pruning are available for efficient inference on standard computation devices. In both structured and unstructured pruning methods, their key idea is to propose a proper implicit or explicit criterion (e.g., magnitude of the weight (Han et al., 2015a;b; Guo et al., 2016; Zhu & Gupta, 2017; Frankle & Carbin, 2018; Mostafa & Wang, 2019; Bellec et al., 2017; Mocanu et al., 2018; Wortsman et al., 2019), scores based on Hessian, momentum or gradient (Menick & Elsen; Lee et al., 2018; Zeng & Urtasun, 2018; LeCun et al., 1990; Hassibi & Stork, 1993; Dettmers & Zettlemoyer, 2019)) to evaluate the importance of the weight, kernel or filter and then remove the unimportant ones. The results in the literature (Guo et al., 2016; Liu et al., 2018; Zeng & Urtasun, 2018; Li et al., 2016; Renda et al., 2020; Kusupati et al., 2020; Menick & Elsen; Wortsman et al., 2019) demonstrate that pruning methods can significantly improve the inference efficiency of DNNs with minimal performance degradation, making the deployment of modern neural networks on resource limited devices possible.

2.2 Dense-to-sparse and Sparse-to-sparse Training

We follow the convention of Kusupati et al. (2020) to divide training algorithms for obtaining sparse networks into two groups: dense-to-sparse training and sparse-to-sparse training. Dense-to-sparse training starts with a dense network and obtains a sparse network at the end of the training (Han et al., 2015b; Zhu & Gupta, 2017; Molchanov et al., 2017; Frankle & Carbin, 2018; Renda et al., 2020; Xiao et al., 2019; Srinivas et al., 2017; Louizos et al., 2017; Wortsman et al., 2019). ProbMask belongs to the group of dense-to-sparse training. Han et al. (2015a); Zhu & Gupta (2017); Frankle & Carbin (2018); Renda et al. (2020) follows the idea of using weight magnitude as the criterion. Zhu & Gupta (2017) manually set a uniform sparsity budget for different layers. Renda et al. (2020) achieves strong results but needs multiple rounds of pruning and retraining. Xiao et al. (2019) assigns auxiliary scores to weights and use it as the criterion. Xiao et al. (2019) suffers from the bias induced by the approximation of the step function and will have gradient vanishing problem when using ReLU and SoftPlus as the approximator. This makes the auxiliary scores hard to act as a global criterion. Srinivas et al. (2017); Louizos et al. (2017); Molchanov et al. (2017) base its criterion on reparameterization of probability and have the most connections with our work. We will fully discuss them in the next subsection.

Sparse-to-sparse training starts with a sparse network and maintain the sparsity during training (Bellec et al., 2017; Mocanu et al., 2018; Mostafa & Wang, 2019; Menick & Elsen; Dettmers & Zettlemoyer, 2019). It uses criterion like weight magnitude, weight gradient magnitude, momentum of weight to reallocate sparsity through training. Conceptually, sparse-to-sparse training can reduce the computational cost during training but it is hard to take effect without the support of sparse convolution framework on GPU. The performance of sparse-to-sparse training generally falls behind dense-to-sparse training under the same setting as shown in Wang et al. (2020); Kusupati et al. (2020).

3 Effective Sparsification with Global Sparsity Constraint

Below, we present our proposed network sparsification framework and the method for solving the minimization problem in the framework.

3.1 A Probabilistic Sparsification Framework

Let $\mathcal{D}$ be a dataset consisting of N i.i.d. samples $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$, $\boldsymbol{w} \in \mathbb{R}^n$ be the weights of a neural network. We denote $\boldsymbol{m} \in \{0,1\}^n$ to be the masks of the weights. $m_i = 0$ means the weight w_i is pruned and otherwise w_i is kept. The problem of training sparse neural networks can be naturally formulated into the following empirical risk minimization problem:

$$\min_{\boldsymbol{w}\in\mathbb{R}^n,\boldsymbol{m}\in\{0,1\}^n} \mathcal{L}(\boldsymbol{w},\boldsymbol{m}) = \frac{1}{N}\sum_{i=1}^{N} \ell\left(h\left(\mathbf{x}_i;\boldsymbol{w}\circ\boldsymbol{m}\right),\mathbf{y}_i\right), s.t.\ \|\boldsymbol{m}\|_1 \leq K, \tag{1}$$

where $h(\cdot;\boldsymbol{w}\circ\boldsymbol{m})$ is output of the pruned network with $\circ$ being the element-wise product of two vectors, and $\ell(\cdot,\cdot)$ is the loss function, e.g, the squared loss for regression or cross entropy loss for classification. $K = kn$ is the model size we want to reduce the network to, i.e., the number of remaining weights after pruning, here k is the remaining ratio of model weights. Different from the existing pruning methods, in this framework, the model size is controlled by a single constraint which avoids tuning the pruning rate for each layer. However, since the objective is discrete with respect to the mask $\boldsymbol{m}$, problem (1) is hard to solve and thus cannot be applied in practice.

We notice that if we view each component of mask $\boldsymbol{m}$ as a binary random variable and reparameterize problem (1) with respect to the distributions of this random variable, then problem (1) can be reformulated into an excepted loss minimization problem over the weight and probability spaces, which is continuous. Specifically, we can view m_i as a Bernoulli random variable with probability s_i to be 1 and $1 - s_i$ to be 0, that is $m_i \sim \text{Bern}(s_i)$, where $s_i \in [0,1]$. Assuming the variables m_i are independent, then we can get the distribution function of $\boldsymbol{m}$, i.e., $p(\boldsymbol{m}|\boldsymbol{s}) = \Pi_{i=1}^{n}(s_i)^{m_i}(1-s_i)^{(1-m_i)}$. Thus, the model size can be controlled by the sum of the probabilities s_i, i.e., $\mathbf{1}^\top\boldsymbol{s}$, since $\mathbb{E}_{\boldsymbol{m}\sim p(\boldsymbol{m}|\boldsymbol{s})}\|\boldsymbol{m}\|_0 = \sum_{i=1}^{n} s_i$. Then the discrete constraint $\|\boldsymbol{m}\|_0 \leq K$ in problem (1) can be transformed into $\mathbf{1}^\top\boldsymbol{s} \leq K$ with each $s_i \in [0,1]$, which is continuous and convex. Therefore, by reparameterizing with respect to $\mathbf{s}$, problem (1) can be reformulated into the following excepted loss minimization problem:

$$\min_{\boldsymbol{w}\in\mathbb{R}^n,\boldsymbol{s}} \mathbb{E}_{\boldsymbol{m}\sim p(\boldsymbol{m}|\boldsymbol{s})}\mathcal{L}(\boldsymbol{w},\boldsymbol{m}) := \frac{1}{N}\sum_{i=1}^{N} \ell\left(h\left(\mathbf{x}_i;\boldsymbol{w}\circ\boldsymbol{m}\right),\mathbf{y}_i\right), s.t.\ \mathbf{1}^\top\boldsymbol{s} \leq K \text{ and } \boldsymbol{s}\in[0,1]^n \tag{2}$$

Discussion. We do not reparameterize problem (1) in the logit space (probability is generated by sigmoid transformation of logit), since we need a *simple feasible region*, in a sense that the projection on this region can be solved very efficiently. Another benefit of directly optimizing on the probability space is that we can avoid gradient vanishing induced by sigmoid transformation. Moreover, our framework has the following appealing features:

- The constraints in problem (2) can be rewritten as $\|\boldsymbol{s}\|_1 \leq K$ and $\boldsymbol{s}\in[0,1]^n$. Due to this ℓ_1 norm constraint, the optimal $\boldsymbol{m}$ is sparse, and most of m_i would be either 0 or 1 with high probability, making $\boldsymbol{m}$ converge to a deterministic mask. Therefore, $\boldsymbol{m}$ after training would have a quite low variance and thus the loss of a randomly sampled $\boldsymbol{m}$ would close to the expected loss in Eqn.(2).
- Compared with problem (1), problem (2) is continuous. Moreover, the feasible region of problem (2) is quite simple, which is actually the intersection of the cube $[0,1]^n$ and the half space $\mathbf{1}^\top\boldsymbol{s} \leq K$. For such simple set, the projection operator has an explicit expression, please see Theorem 1 for the details. This makes it possible to adopt the efficient optimization algorithms such as projected gradient descent to solve problem (2).
- In our framework, the problem is reparameterized with respect to probability, which can be used as a global criterion to measure the importance of the weights in different layers. The amount of redundancy in each layer of the neural network, can be learned automatically in the process of solving problem (2). Therefore, we avoid to set pruning ratio for each layer manually. The benefits of such nice feature on the model size and the accuracy of the final pruned network will be verified in the experiments.

3.2 Optimization with Projected Gradient Descent

Below, we present our training method for problem (2). We train both the weights $\boldsymbol{w}$ and the probability $\boldsymbol{s}$ at the same time, and finally sparse network by sampling according to the final probability $\boldsymbol{s}$. We adopt projected gradient descent(PGD) as the optimizer and the details are as follows.

[Gradient Computation] The difficulty lies in computing the gradient of the expected loss with respect to the probability. Therefore, in this paper, we adopt Gumbel-Softmax trick to calculate the gradient, with which the gradient w.r.t. weights and probability can be calculated in the following form:

$$\nabla_{\boldsymbol{s},\boldsymbol{w}}\mathbb{E}_{p(\boldsymbol{m}|\boldsymbol{s})}\mathcal{L}\left(\boldsymbol{w},\boldsymbol{m}\right) = \mathbb{E}_{\boldsymbol{g_0},\boldsymbol{g_1}}\nabla_{\boldsymbol{s},\boldsymbol{w}}\mathcal{L}\Big(\boldsymbol{w},\mathbb{1}\big(\log(\boldsymbol{s}) - \log(\mathbf{1}-\boldsymbol{s}) + \boldsymbol{g_1} - \boldsymbol{g_0} \geq 0\big)\Big) \tag{3}$$

$$\approx \mathbb{E}_{\boldsymbol{g_0},\boldsymbol{g_1}}\nabla_{\boldsymbol{s},\boldsymbol{w}}\mathcal{L}\Big(\boldsymbol{w},\sigma\big(\frac{\log(\boldsymbol{s}) - \log(\mathbf{1}-\boldsymbol{s}) + \boldsymbol{g_1} - \boldsymbol{g_0}}{\tau}\big)\Big) \tag{4}$$

$$\approx \frac{1}{I}\sum_{i=1}^{I}\nabla_{\boldsymbol{s},\boldsymbol{w}}\mathcal{L}\Big(\boldsymbol{w},\sigma\big(\frac{\log(\boldsymbol{s}) - \log(\mathbf{1}-\boldsymbol{s}) + \boldsymbol{g_1}^{(i)} - \boldsymbol{g_0}^{(i)}}{\tau}\big)\Big), \tag{5}$$

where $\mathbb{1}(\boldsymbol{A}) \in \{0,1\}^n$ is the indicator function. $\boldsymbol{g_0}$ and $\boldsymbol{g_1}$ are two random variables in $\mathbb{R}^n$, with each element i.i.d sampled from $\text{Gumbel}(0,1)$ distribution. $\boldsymbol{g_1}^{(i)}$ and $\boldsymbol{g_0}^{(i)}$ with $i = 1, 2, \ldots, I$ are $2I$ sampled instances. $\sigma(\cdot) : \mathbb{R}^n \to (0,1)^n$ here is the element-wise sigmoid function, i.e., $\sigma(\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{x}}}$ for any $\boldsymbol{x} \in \mathbb{R}^n$. τ is a temperature annealing parameter decreasing during training, making $\sigma(\cdot)$ become closer to the indicator function. We find in the experiments that this would help the mask converge to a deterministic one. The proof of the equations above are given in appendix.

From Eqn.(4) to Eqn.(5), multiple networks are sampled to obtain a steady gradient flow with a low variance. The constraint $\|\boldsymbol{m}\|_1 \leq K$ can be transformed into $\|\boldsymbol{s}\|_1 \leq K$ in sight of the fact that $\mathbb{E}(\|\boldsymbol{m}\|_1) = \|\boldsymbol{s}\|_1$ and the probabilities would converge to either 1 or 0 during the training process.

[Projected Gradient Descent] We denote the feasible region of problem (2) as C, that is $C = \{\boldsymbol{s} \mid \|\boldsymbol{s}\|_1 \leq K \text{ and } \boldsymbol{s} \in [0,1]^n\}$. The theorem below shows that the projection of a vector onto C can be calculated efficiently, which makes PGD applicable.

Theorem 1. *For each vector $\boldsymbol{z}$, its projection $\boldsymbol{s}$ in the set C can be calculated as follows:*

$$\boldsymbol{s} = \min(1, \max(0, \boldsymbol{z} - v_2^*\mathbf{1})).$$

where $v_2^* = \max(0, v_1^*)$ with v_1^* being the solution of the following equation

$$\mathbf{1}^\top[\min(1, \max(0, \boldsymbol{z} - v_1^*\mathbf{1}))] - K = 0. \tag{6}$$

The equation (6) can be solved by bisection method very efficiently.

Algorithm 1 Probabilistic Masking (ProbMask)

```
Input: target remaining ratio $k_f$, a dense network $\boldsymbol{w}$.
 1: Initialize $\boldsymbol{w}$, assign probabilities $\boldsymbol{s}$ to weights $\boldsymbol{w}$, let $\boldsymbol{s} = \mathbf{1}$ and $\tau = k = 1$.
 2: for training epoch $t = 1, 2 \ldots T$ do
 3:     Decrease the temperature annealing parameter by $\tau = 0.97(1 - t/T) + 0.03$.
 4:     Update $k$ according to Eqn.(7).
 5:     for each training iteration do
 6:         Sample mini batch of data $\mathcal{B} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_B, \mathbf{y}_B)\}$.
 7:         Generate $\boldsymbol{g_1}^{(i)}$ and $\boldsymbol{g_0}^{(i)}$ with each element sampled from Gumbel(0, 1), $i = 1, 2, \ldots, I$.
 8:         $\boldsymbol{s} \leftarrow \text{proj}_C(\boldsymbol{z})$, with $\boldsymbol{z} = \boldsymbol{s} - \eta\frac{1}{I}\sum_{i=1}^{I}\nabla_{\boldsymbol{s}}\mathcal{L}_{\mathcal{B}}\Big(\boldsymbol{w}, \sigma\big(\frac{\log(\boldsymbol{s})-\log(\mathbf{1}-\boldsymbol{s})+\boldsymbol{g_1}^{(i)}-\boldsymbol{g_0}^{(i)}}{\tau}\big)\Big)$.
 9:         $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\frac{1}{I}\sum_{i=1}^{I}\nabla_{\boldsymbol{w}}\mathcal{L}_{\mathcal{B}}\Big(\boldsymbol{w}, \sigma\big(\frac{\log(\boldsymbol{s})-\log(\mathbf{1}-\boldsymbol{s})+\boldsymbol{g_1}^{(i)}-\boldsymbol{g_0}^{(i)}}{\tau}\big)\Big)$
10:     end for
11: end for
12: return A pruned network $\boldsymbol{w} \circ \boldsymbol{m}$ by sampling a mask $\boldsymbol{m}$ from the distribution $p(\boldsymbol{m}|\boldsymbol{s})$.
```

Now we can apply PGD to solve problem (2), the detailed steps are given in Algorithm 1.

[Gradually Increased Pruning Rate] Following (Zhu & Gupta, 2017), we increase the pruning rate to make a smooth transformation from dense to extremely sparse status. That is, we let

$$k = k_f + (1 - k_f)\left(1 - \frac{t - t_1}{t_2 - t_1}\right)^3 \text{ for } t \in \{t_1, t_1 + 1, \ldots, t_2\}, \tag{7}$$

t is the current epoch number. t_1 and t_2 are two integers. k keeps 1 before epoch t_1 and k_f after epoch t_2. k_f is the targeted remaining ratio.

4 Experiment

In this section, we conduct a series of experiments to evaluate the performance of our proposed method. We divide the experiments into two parts. In part one, we conduct lots of relatively small-scaled experiments on CIFAR-10/100 datasets with modern architectures VGG19 (Simonyan & Zisserman, 2014) and ResNet32 (He et al., 2016) to verify some appealing properties of our method. In part two, we verify the superiority of our method over state-of-the-art methods by conducting experiments on ImageNet (Deng et al., 2009). We choose six representative methods PBW (Pruning by Weight, Han et al. (2015b)), MLPrune (Zeng & Urtasun, 2018), RIGL (Menick & Elsen), STR (Kusupati et al., 2020), DNW(Maddison et al., 2016), GMP (Zhu & Gupta, 2017)) as baselines. PBW (Han et al., 2015b) is a classic magnitude-based pruning method. MLPrune (Zeng & Urtasun, 2018) is a latest Hessian-based pruning method showing overall better performance (Wang et al., 2020) against various sparse-to-sparse training methods (SET (Mocanu et al., 2018), DEEPR (Bellec et al., 2017), DSR (Mostafa & Wang, 2019)), so we compare with these sparse-to-sparse training methods implicitly in CIFAR experiments. DNW (Wortsman et al., 2019), GMP (Zhu & Gupta, 2017), STR (Kusupati et al., 2020) are state-of-the-art methods on dense-to-sparse training. RIGL (Menick & Elsen) is the state-of-the-art sparse-to-sparse training method. Due to the space limitation, we postpone the experiment of identifying supermasks and experimental configurations and into appendix.

4.1 VGG19 and ResNet32 on CIFAR-10/100

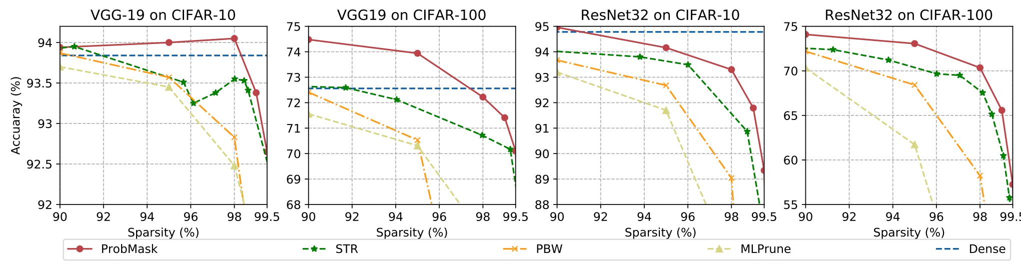


Figure 1: Comparison of Top-1 Accuracy on CIFAR-10/100.

Table 1 presents the detailed accuracy of PBW, MLPrune and ProbMask at different pruning ratios. It is very hard to accurately tune weight decay parameter in STR to obtain the desired pruning ratio. Therefore we tune the weight decay parameter manually to make it have roughly the same pruning ratio range with ProbMask, i.e., 90% to 99.9%.

The results in both Table 1 and Figure 1 demonstrate that our ProbMask can steadily outperform the baselines and the superiority becomes more significant at higher pruning ratios. From Table 1, we can see that when prune rate come to 99.5% or higher on CIFAR-10/100, PBW and MLPrune would seriously degrade or even collapse, while our ProbMask can still achieve significantly higher. Figure 1 shows that on CIFAR-100 with VGG19, the gap between ProbMask and STR would be roughly 2% on average when the remaining ratio is in the range of $[0.9, 0.98]$. Pruning ResNet32 is more challenging since VGG19 has about 10 times parameters than ResNet32. In this case, the gap becomes more significant especially at high pruning ratios, which can be up to 5% on CIFAR-100 experiments. The superiority of ProbMask over such high prune ratios attributes to our global sparsity constraint, allowing us to have non-uniform sparsity budgets across layers. This will be further validated in the ablation study in Section 4.3.

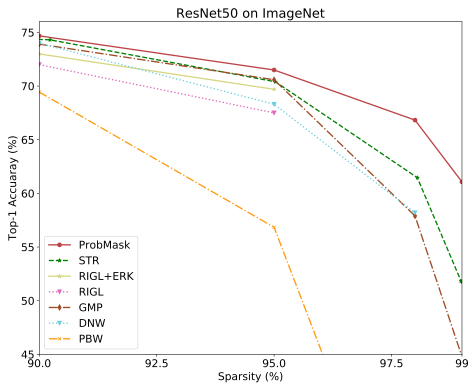
4.2 ResNet50 on ImageNet-1K

In this section, we evaluate the performance of our ProbMask on ImageNet with ResNet50. Table 2 and Figure 2 report the detailed accuracy at different pruning ratios. ProbMask steadily outperform state-of-the-art methods with a large margin, especially when the pruning ratio is high than 98%. Notably the gap comes up to 5% at 98% sparsity and 10% at 99% sparsity. DNW and GMP allocate uniform sparsity budget. They present reasonably good performance at 90% sparsity while

Dataset	CIFAR-10						CIFAR-100					
Ratio	90%	95%	98%	99%	99.5%	99.9%	90%	95%	98%	99%	99.5%	99.9%
VGG19	93.84	-	-	-	-	-	72.56	-	-	-	-	-
PBW	93.87	93.57	92.83	90.89	10.00	10.00	72.41	70.53	58.91	1.00	1.00	1.00
MLPrune	93.70	93.45	92.48	91.44	88.18	65.38	71.56	70.31	66.77	60.10	50.98	5.58
ProbMask	**93.94**	**94.00**	**94.05**	**93.38**	**92.65**	**89.79**	**74.48**	**73.94**	**72.22**	**71.41**	**70.10**	**60.41**
ResNet32	94.78	-	-	-	-	-	75.94	-	-	-	-	-
PBW	93.67	92.68	89.04	77.03	73.03	38.64	72.19	68.42	58.23	43.00	20.75	5.96
MLPrune	93.20	91.70	85.64	76.88	67.66	36.09	70.33	61.73	37.86	22.38	13.85	5.50
ProbMask	**94.96**	**94.16**	**93.30**	**91.79**	**89.34**	**76.87**	**74.09**	**73.06**	**70.35**	**65.57**	**57.25**	**26.72**

Table 1: Accuracy of VGG19 and ResNet32 on CIFAR-10/100 at different pruning ratios.

fall behind ProbMask by about 9% at 98% sparsity. This validates our previous claim that identifying weight allocation for different layers really matters. Uniform sparsity budget is a reasonable compromise but obviously don't give a perfect solution. STR attempts to learn weight allocation for different layers but don't give perfect results. ProbMask presents much better perfomance on high spasity regions, leading to a gap about 10% percent at 99% sparsity. With the global comparable nature of probability, ProbMask easily learns a much better weight allocation scheme for different layer. We also compare ProbMask with Sparse VD (Molchanov et al., 2017) on sparsity 90%. Sparse VD finds a subnet with 73.84% Top-1 Acuucracy, a weaker result than ProbMask. We also observe noticable fluctuations between different runs, and this can be expected because Sparse VD adopts crude cut-off practice rather than sampling. This inevitably results in perfomance gap in training and testing phases. ProbMask learns a deterministic mask at the end of training, fixing the training and testing performance discrepancy problem. Figure 2 also reports the accuracy-versus-FLOPs for ProbMask and compared methods. It shows that ProbMask finds a smaller mask with comparable accuracy and FLOPs and achieve state-of-the-art result on it.



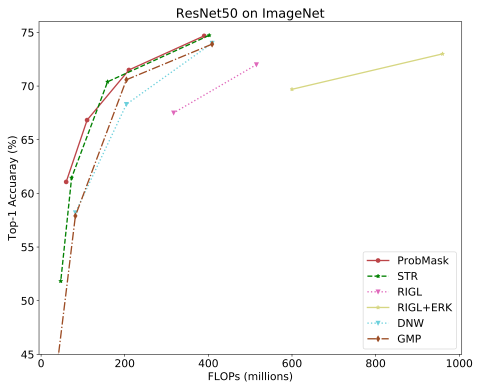


Figure 2: ProbMask comfortably beats state-of-the-art methods in all sparsity regions. Notably, the gap comes up to 5% at 98% sparsity and 10% at 99% sparsity. ProbMask obtains a smaller sparse network with comparable accuracy and FLOPs, still achieving state-of-the-art result on accuracy-vesus-FLOPs curve.

4.3 Ablation Study

In this section, we will show global comparability of probability adopted by ProbMask to measure weight importance, the superiority of our global sparsity constraint over layer-wise constraint, and the convergence of scores to an almost deterministic mask.

[Global Comparability of Probability] Table 1 shows that PBW and MLPrune collapse on CIFAR-10/100 when the pruning ratio is as high as 99.9%. To explore the reason, we plot the remaining ratio across layers at pruning ratio of 90% and 99.9% on CIFAR-10 in Figure 3. It shows that

Dataset	ImageNet			
Ratio	90%	95%	98%	99%
ResNet50	77.01	-	-	-
PBW (Han et al., 2015a)	69.44	56.84	22.46	5.98
MLPrune (Zeng & Urtasun, 2018)	60.98	30.89	3.16	0.77
GMP (Zhu & Gupta, 2017)	73.91	70.59	57.90	44.78
DNW (Wortsman et al., 2019)	74.00	68.30	58.20	-
STR (Kusupati et al., 2020)	74.31	70.40	61.46	50.35
RIGL (Menick & Elsen)	72.00	67.50	-	-
ProbMask	**74.68**	**71.50**	**66.83**	**61.07**

Table 2: Accuracy of ResNet50 on ImageNet at different pruning ratios. ProbMask steadily beats previous state-of-the-art methods on Hessian-based pruning, weight magnitude pruning, dense-to-sparse training and sparse-to-sparse training. RIGL improves with the help of ERK (Erdós-Rényi-Kernel) but will result in doubling the FLOPs at inference time, so we put it in Figure 2).

when the pruning ratio is high, PBW and MLPrune prune almost all the weights in certain layers with remaining ratio approaching 10^{-6}, e.g., layer 10 and 11 in VGG 19 and layer 23, 24 and 26 in ResNet32. Such pruned networks would never achieve good performance even with finetuning. The reason is that the proposed weight importance measure in PBW and MLPrune are not globally comparable. To be precise, although the weight importance scores in different layers have been normalized in MLPrune, their magnitudes are still quite different, which is verified in Figure 4. Thus a global threshold could remove almost all the weights in certain layers in order to achieve high enough pruning ratio. Figure 3 also shows that the pruning ratio of ProbMask varies in a proper range. This attributes to the probability that we adopt as the weight importance measure, which is globally comparable and enables us to use a global sparsity constraint. That is, by training under the global sparsity constraint, ProbMask can learn optimal sparsity budget allocation automatically for all the layers and we do not need to set proper pruning ratio for each layer manually.

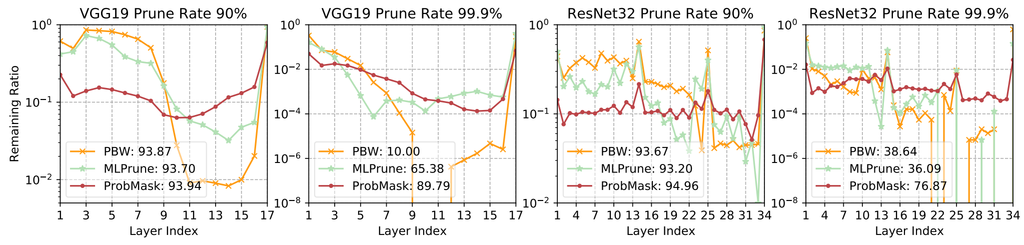


Figure 3: Remaining ratio in each layer on CIFAR-10.

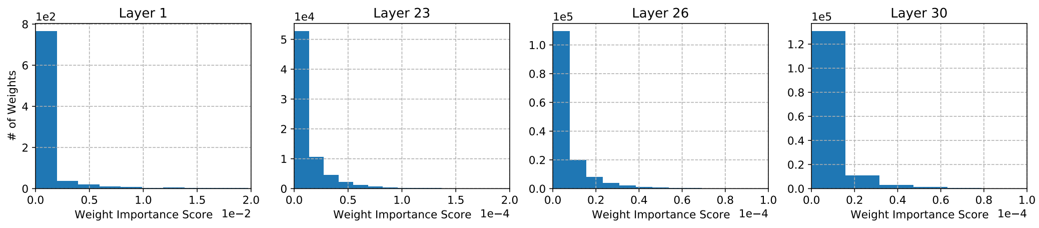


Figure 4: Weight importance score histogram of ResNet32 from MLPrune with pruning rate 99.9%.

[Superiority of Global Sparsity Constraint over Layer-wise Constraint] To show this superiority, we evaluate the performance of ProbMask under global sparsity constraint and layer-wise sparsity constraint. In layer-wise constraint, we force all the pruning ratios in each layer to be equal and also equal to the one in the global constraint. The experiment is conducted on CIFAR-10 with ResNet32 and the results are given in Table 3. It shows that the gap of the accuracy under global and layer-wise constraints are small when prune rate is lower than 95%, however, it would grow up rapidly when the pruning ratio is larger than 98%. For example, when the pruning ratio is 99.9%, the accuracy of global sparsity constraint can be up to 57.75% higher than the layer-wise one.

Table 3: Comparing layerwise sparsity budget and global sparsity budget of ProbMask on ResNet32.

Dataset	CIFAR-10					
Ratio	90%	95%	98%	99%	99.5%	99.9%
ResNet32	94.78	-	-	-	-	-
Layerwise Sparsity Budget	94.89	94.09	92.64	90.89	74.6	19.12
Global Sparsity Budget	**94.96**	**94.16**	**93.30**	**91.79**	**89.34**	**76.87**

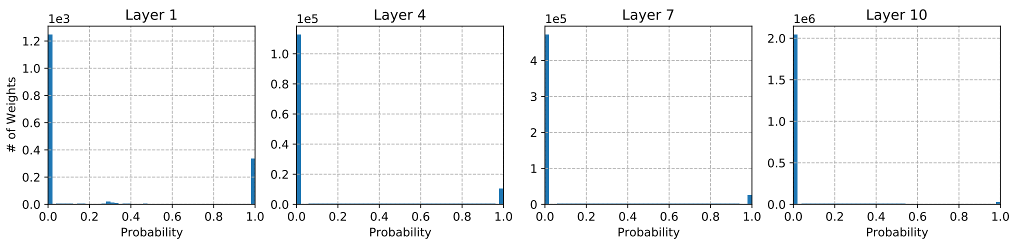


Figure 5: Probability histogram of VGG19 trained by ProbMask on CIFAR-10 at pruning rate 90%.

[Convergence to Deterministic Mask] To show that the mask trained by our ProbMask can converge to a deterministic mask after training, we randomly choose some layers from VGG19 and present their distribution of the probability value after training in Figure 5. We can see that after training, almost all of the probabilities $\boldsymbol{s}_i$ can converge to either 0 and 1, leading to a deterministic mask. This nice feature attributes to ℓ_1 norm in our global sparsity constraint over the probability space and the temperature annealing technique.

5 CONCLUSION

This paper proposes an effective network sparsification method ProbMask and demonstrate state-of-the-art results on various models and datasets. We provide evidence that probability can serve as a suitable global comparator to measure weight importance and solve the training and testing performance discrepancy problem observed in practice. ProbMask can also serve as a powerfull tool for identifying subnetworks with high performance in a randomly weighted dense neural network.

REFERENCES

Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. *arXiv preprint arXiv:1711.05136*, 2017.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In *Advances in neural information processing systems*, pp. 1379–1387, 2016.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015b.

Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pp. 164–171, 1993.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.

Minsoo Kang and Bohyung Han. Operation-aware soft channel pruning using differentiable masks. *arXiv preprint arXiv:2007.03938*, 2020.

Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. *arXiv preprint arXiv:2002.03231*, 2020.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pp. 598–605, 1990.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pp. 5058–5066, 2017.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Utku Evci1 Trevor Gale1 Jacob Menick and Pablo Samel Castro1 Erich Elsen. Rigging the lottery: Making all tickets winners.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. *arXiv preprint arXiv:1701.05369*, 2017.

Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. *arXiv preprint arXiv:1902.05967*, 2019.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11893–11902, 2020.

Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Suraj Srinivas, Akshayvarun Subramanya, and R Venkatesh Babu. Training sparse neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 138–145, 2017.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*, 2019.

Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. Discovering neural wirings. In *Advances in Neural Information Processing Systems*, pp. 2684–2694, 2019.

Xia Xiao, Zigeng Wang, and Sanguthevar Rajasekaran. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In *Advances in Neural Information Processing Systems*, pp. 13681–13691, 2019.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

Mao Ye, Chengyue Gong, Lizhen Nie, Denny Zhou, Adam Klivans, and Qiang Liu. Good subnetworks provably exist: Pruning via greedy forward selection. *arXiv preprint arXiv:2003.01794*, 2020.

Wenyuan Zeng and Raquel Urtasun. Mlprune: Multi-layer pruning for automated neural network compression. 2018.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems*, pp. 3597–3607, 2019.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

A Appendix

In this appendix, we present the general experimental configurations, results of applying ProbMask for finding supermasks, proof for equation 3, and proof for theorem 1.

A.1 Experimental Configurations

Dataset	CIFAR	ImageNet
GPUs	1	4
Batch Size	256	256
Epochs	300	100
Weight Optimizer	SGD	SGD
Weight Learning Rate	0.1	0.256
Weight Momentum	0.9	0.875
Probability Optimizer	Adam	Adam
Probability Learning Rate	**6e-3**	**6e-3**
t_1	48	16
t_2	180	60
Warmup	✗	✓
Label Smoothing	✗	0.1

Table 4: The bold-face probability learning rate 6e-3 is the **only** hyperparameter obtained by grid search on CIFAR-10 experiments on a small size network Conv-4 (Frankle & Carbin, 2018) and applied directly to larger datasets and networks. This demonstrates the generality of our proposed ProbMask to different datasets, different networks and different tasks, i.e., pruning networks and finding supermasks. Other hyperparameters are applied following the same practice of previous works (Ramanujan et al., 2020; Kusupati et al., 2020; Liu et al., 2018; Zhu & Gupta, 2017). The channels of ResNet32 for CIFAR experiments are doubled following the same practice of Wang et al. (2020). The temperature annealing scheme follows the same practice of Xie et al. (2018)

A.2 Applying ProbMask for Finding Supermasks in Randomly Weighted Neural Networks

Previous works on supermasks, i.e, subnetworks achieving good performance with weights fixed at random state, focus on sparsity region [10%, 90%]. Here, we would like to explore the performance of supermasks with higher sparsity, [90%, 99%]. We conduct experiments on modern architeture ResNet32 and dataset CIFAR-100, a harder task than CIFAR-10 where a large portion of previous experiments are conducted. In this experiment, weights are fixed at initialization state by Kaiming Normal He et al. (2015). Hyperparameters follow the same as previous CIFAR experiments. According to Figure 6, we observe that ProbMask easily scales to ultra sparse region with about 50% accuracy and 2% remaining weights, while state-of-the-art method edge-popup Ramanujan et al. (2020) collapse with less than 30% accuracy. It is a surprising result that a subnet with 2% fixed random weights still succeeds in obtaining nearly 50% accuracy on a task with 100 categories. It shows that the structure in networks already provides valuable information for classification.

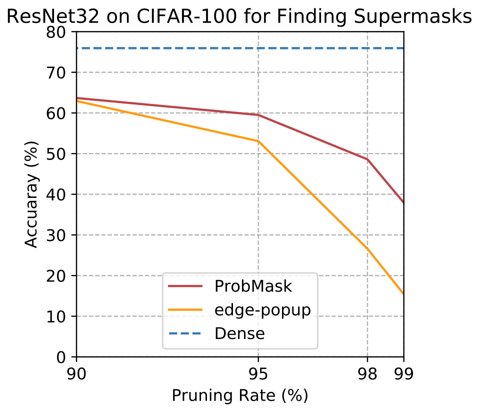


Figure 6: ProbMask can find a supermask with just 2% remaining weights and nearly 50% accuracy on CIFAR-100. Weights are fixed at initialization state.

Table 5: ResNet32 on CIFAR-100 for finding supernets

Dataset	CIFAR-100			
Ratio	90%	95%	98%	99%
ResNet32	75.94	-	-	-
edge-popup	62.94	53.07	26.6	15.39
ProbMask	**63.67**	**59.51**	**48.59**	**37.91**

A.3 Proof for equation 3

Proof. The PDF (probability density function) of $\mathrm{Gumbel}(\mu, 1)$ is

$$f(z;\mu) = e^{-(z-\mu)-e^{-(z-\mu)}}. \tag{8}$$

The CDF (cumulative distribution function) of $\mathrm{Gumbel}(\mu, 1)$ is

$$F(z;\mu) = e^{-e^{-(z-\mu)}}. \tag{9}$$

We just need to prove that

$$\forall i,\ P(\log(s_i) - \log(1-s_i) + g_{1,i} - g_{2,i} \geq 0) = s_i. \tag{10}$$

$g_{1,i}$ and $g_{2,i}$ are two $\mathrm{Gumbel}(0,1)$ random variables sampled for s_i. The probability is taken with respect to $g_{1,i}$ and $g_{2,i}$. s_i can be seen as a constant in the following proof.

Let $z_1 = \log(s_i) + g_{1,i}, z_2 = \log(1-s_i) + g_{2,i}$. Then $z_1 \sim \mathrm{Gumbel}(\log(s_i), 1)$, $z_2 \sim \mathrm{Gumbel}(\log(1-s_i), 1)$.

$$
\begin{aligned}
&P(\log(s_i) - \log(1-s_i) + g_{1,i} - g_{2,i} \geq 0) && (11)\\
=&P(z_2 \leq z_1) && (12)\\
=&\int_{-\infty}^{+\infty}\int_{-\infty}^{z_1} f(z_2;\log(1-s_i))f(z_1;\log(s_i))dz_2dz_1 && (13)\\
=&\int_{-\infty}^{+\infty} F(z_1;\log(1-s_i))f(z_1;\log(s_i))dz_1 && (14)\\
=&\int_{-\infty}^{+\infty} e^{-e^{-(z_1-\log(1-s_i))}} \cdot e^{-(z_1-\log s_i)-e^{-(z_1-\log s_i)}}dz_1 && (15)\\
=&\int_{-\infty}^{+\infty} e^{-e^{-z_1}(1-s_i)-z_1+\log s_i - e^{-z_1}s_i}dz_1 && (16)
\end{aligned}
$$

$$=s_i \int_{-\infty}^{+\infty} e^{-e^{-z_1}-z_1} dz_1 \tag{17}$$

$$=s_i \tag{18}$$

$\int_{-\infty}^{+\infty} e^{-e^{-z_1}-z_1} dz_1$ is the integral of a Gumbel(0,1) random variable. □

A.4 Proof for Theorem 1

Proof. The projection from $\boldsymbol{z}$ to set C can be formulated in the following optimization problem:

$$\min_{\boldsymbol{s}\in\mathbb{R}^n} \frac{1}{2}\|\boldsymbol{s}-\boldsymbol{z}\|^2,$$
$$s.t.\mathbf{1}\top \boldsymbol{s} \leq K \text{ and } 0 \leq \boldsymbol{s}_i \leq 1.$$

Then we solve the problem with Lagrangian multiplier method.

$$L(\boldsymbol{s}, v) = \frac{1}{2}\|\boldsymbol{s}-\boldsymbol{z}\|^2 + v(\mathbf{1}^\top \boldsymbol{s} - K) \tag{19}$$

$$= \frac{1}{2}\|\boldsymbol{s}-(\boldsymbol{z}-v\mathbf{1})\|^2 + v(\mathbf{1}^\top \boldsymbol{z} - K) - \frac{n}{2}v^2, v \geq 0. \tag{20}$$

with the implicit constraint $0 \leq \boldsymbol{s}_i \leq 1$. Minimize the problem with respect to $\boldsymbol{s}$, we have

$$\tilde{\boldsymbol{s}} = \mathbf{1}_{\boldsymbol{z}-v\mathbf{1}\geq 1} + (\boldsymbol{z}-v\mathbf{1})_{1>\boldsymbol{z}-v\mathbf{1}>0} \tag{21}$$

We have

$$g(v) = L(\tilde{\boldsymbol{s}}, v) = \frac{1}{2}\|[\boldsymbol{z}-v\mathbf{1}]_- + [\boldsymbol{z}-(v+1)\mathbf{1}]_+\|^2 + v(\mathbf{1}^\top \boldsymbol{z} - s) - \frac{n}{2}v^2 \tag{22}$$

$$= \frac{1}{2}\|[\boldsymbol{z}-v\mathbf{1}]_-\|^2 + \frac{1}{2}\|[\boldsymbol{z}-(v+1)\mathbf{1}]_+\|^2 + v(\mathbf{1}^\top \boldsymbol{z} - s) - \frac{n}{2}v^2, v \geq 0. \tag{23}$$

$$g'(v) = \mathbf{1}^\top[v\mathbf{1}-\boldsymbol{z}]_+ + \mathbf{1}^\top[(v+1)\mathbf{1}-\boldsymbol{z}]_- + (1^T\boldsymbol{z} - s) - nv \tag{24}$$

$$= \mathbf{1}^\top \min(1, \max(0, \boldsymbol{z}-v\mathbf{1})) - K, v \geq 0. \tag{25}$$

It is easy to verify that $g'(v)$ is a monotone decreasing function with respect to v and we can use a bisection method solve the equation $g'(v) = 0$ with solution v_1^*. Then we get that $g(v)$ increases in the range of $(-\infty, v_1^*]$ and decreases in the range of $[v_1^*, +\infty)$. The maximum of g(v) is achieved at 0 if $v_1^* \leq 0$ and v_1^* if $v_1^* > 0$. Then we set $v_2^* = max(0, v_1^*)$. Finally we have

$$\boldsymbol{s}^* = \mathbf{1}_{\boldsymbol{z}-v_2^*\mathbf{1}\geq 1} + (\boldsymbol{z}-v_2^*\mathbf{1})_{1>\boldsymbol{z}-v_2^*\mathbf{1}>0} \tag{26}$$

$$= \min(1, \max(0, \boldsymbol{z}-v_2^*\mathbf{1})). \tag{27}$$

□

A.5 Comparison with previous stochastic methods

We notice a few existing stochastic methods aiming to optimize the expected loss. Louizos et al. (2017) proposes an method based on reparameterization to the expected loss minimization and but is reported to fail to work on ImageNet (Gale et al., 2019). Louizos et al. (2017) cannot learn a deterministic mask and thus lead to much difference between the training and test-time versions of models. Srinivas et al. (2017) solves the expected loss minimization using straight-through estimator (Bengio et al., 2013) and ensures the convergence of probabilities to 0 or 1 through two regularizers. However it includes much bias produced by ignoring the Heaviside in the likelihood during the gradient evaluation (Louizos et al., 2017).