Robust Binary Models by Pruning Randomly-Initialized Networks

Chen Liu* [1], Ziqi Zhao* [1], Sabine Süsstrunk[1], Mathieu Salzmann[1]

[1] École Polytechnique Fédérale de Lausanne

March 22, 2022

* Equal Contribution

Overview

Content

Definition (Robustness Problem)

Given a classification model $f(\theta, \mathbf{x}) : \Theta \times \mathbb{R}^M \to \mathbb{R}^K$ parameterized by θ, data points drawn from the distribution $(\mathbf{x}, y) \sim \mathcal{D}$ and loss function $\mathcal{L}$, robustness problem is formulation as follows:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \max_{\mathbf{x}' \in \mathcal{S}_\epsilon(\mathbf{x})} \mathcal{L}(f(\theta, \mathbf{x}'), y) \tag{1}$$

where $\mathcal{S}_\epsilon(\mathbf{x})$ is called the adversarial budget: $\mathcal{S}_\epsilon(\mathbf{x}) = \{\mathbf{x}' | \|\mathbf{x} - \mathbf{x}'\|_\infty \leq \epsilon\}$.

Definition (Robustness Problem)

Given a classification model $f(\theta, \mathbf{x}) : \Theta \times \mathbb{R}^M \to \mathbb{R}^K$ parameterized by θ, data points drawn from the distribution $(\mathbf{x}, y) \sim \mathcal{D}$ and loss function $\mathcal{L}$, robustness problem is formulation as follows:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \max_{\mathbf{x}' \in \mathcal{S}_\epsilon(\mathbf{x})} \mathcal{L}(f(\theta, \mathbf{x}'), y) \tag{1}$$

where $\mathcal{S}_\epsilon(\mathbf{x})$ is called the adversarial budget: $\mathcal{S}_\epsilon(\mathbf{x}) = \{\mathbf{x}' | \|\mathbf{x} - \mathbf{x}'\|_\infty \leq \epsilon\}$.

Adversarial Training (AT): generate optimal x' and then optimize θ on x'.

Slow Convergence. Overfitting. Larger Data Need. Larger Model Capacity Need.

Slow Convergence. Overfitting. Larger Data Need. Larger Model Capacity Need.

Slow Convergence. Overfitting. Larger Data Need. Larger Model Capacity Need.

- For small model, adversarial training first fails to converge while vanilla training still succeed to obtain non-trivial performance.

Slow Convergence. Overfitting. Larger Data Need. Larger Model Capacity Need.

- For small model, adversarial training first fails to converge while vanilla training still succeed to obtain non-trivial performance.
- For big model, performance of vanilla training first saturates with increasing model size while we still see improvement in adversarial training.

Content

Motivation

Incorporation of Model Compression into Adversarial Training

- Related works
  - Adversarial training and model compression (pruning, quantization) at the same time. [1] [2]
  - Pre-train, compression, fine-tune. [3]

[1] Gui, et. al. "Model compression with adversarial robustness: A unified optimization framework". NeurIPS 2019.
[2] Ye, et. al. "Adversarial robustness vs. model compression, or both". ICCV 2019.
[3] Sehwa, et. al. "Hydra: Pruning adversarially robust neural networks". NeurIPS 2019.

Motivation

Incorporation of Model Compression into Adversarial Training

- Related works
  - Adversarial training and model compression (pruning, quantization) at the same time. [1] [2]
  - Pre-train, compression, fine-tune. [3]

Our proposed method: **compression (pruning) itself as a way of training.**

[1] Gui, et. al. "Model compression with adversarial robustness: A unified optimization framework". NeurIPS 2019.
[2] Ye, et. al. "Adversarial robustness vs. model compression, or both". ICCV 2019.
[3] Sehwa, et. al. "Hydra: Pruning adversarially robust neural networks". NeurIPS 2019.

Motivation

Incorporation of Model Compression into Adversarial Training

- Related works
  - Adversarial training and model compression (pruning, quantization) at the same time. [1] [2]
  - Pre-train, compression, fine-tune. [3]

Our proposed method: **compression (pruning) itself as a way of training.**
Adversarial training learns parameters, our method learns model architectures.

[1] Gui, et. al. "Model compression with adversarial robustness: A unified optimization framework". NeurIPS 2019.
[2] Ye, et. al. "Adversarial robustness vs. model compression, or both". ICCV 2019.
[3] Sehwa, et. al. "Hydra: Pruning adversarially robust neural networks". NeurIPS 2019.

Lottery Ticket Hypothesis[4]**:** Overparameterized neural networks contain sparse subnetworks that can be trained in isolation to achieve competitive performance. These subnetworks are called *winning tickets*.

[4] Frankle, et. al. "The lottery ticket hypothesis: Finding sparse, trainable neural networks". ICLR 2019.
[5] Ramanujan, et. al. "What's hidden in a randomly weighted neural network". CVPR 2020.

Lottery Ticket Hypothesis[4]**:** Overparameterized neural networks contain sparse subnetworks that can be trained in isolation to achieve competitive performance. These subnetworks are called *winning tickets*.

Strong Lottery Ticket Hypothesis[5]**:** There exist winning tickets with competitive performance even without training.

[4] Frankle, et. al. "The lottery ticket hypothesis: Finding sparse, trainable neural networks". ICLR 2019.
[5] Ramanujan, et. al. "What's hidden in a randomly weighted neural network". CVPR 2020.

Lottery Ticket Hypothesis[4]: Overparameterized neural networks contain sparse subnetworks that can be trained in isolation to achieve competitive performance. These subnetworks are called *winning tickets*.

Strong Lottery Ticket Hypothesis[5]: There exist winning tickets with competitive performance even without training.

Strong lottery ticket hypothesis in the context of adversarial training:
Finding robust models in random-initialized networks.

[4] Frankle, et. al. "The lottery ticket hypothesis: Finding sparse, trainable neural networks". ICLR 2019.
[5] Ramanujan, et. al. "What's hidden in a randomly weighted neural network". CVPR 2020.

Content

Definition (Adversarial Edge-Popup Problem)

Given a classificaiotn model $f(\theta, \mathbf{x}) : \Theta \times \mathbb{R}^M \to \mathbb{R}^K$ with initial parameter θ, data points drawn from the distribution $(\mathbf{x}, y) \sim \mathcal{D}$ and loss function $\mathcal{L}$, we aim to find the optimal subnetwork with mask $\mathbf{m} \in \{0,1\}^{|\Theta|}$ to solve the following problem:

$$\min_{\mathbf{m} \in \{0,1\}^{|\Theta|}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \max_{\mathbf{x}' \in \mathcal{S}_\epsilon(\mathbf{x})} \mathcal{L}(f(\theta \odot \mathbf{m}, \mathbf{x}'), y) \tag{2}$$

where $\mathcal{S}_\epsilon(\mathbf{x})$ is the adversarial budget and $\odot$ the elementwise multiplication.

Definition (Adversarial Edge-Popup Problem)

Given a classificaiotn model $f(\theta, \mathbf{x}) : \Theta \times \mathbb{R}^M \to \mathbb{R}^K$ with initial parameter θ, data points drawn from the distribution $(\mathbf{x}, y) \sim \mathcal{D}$ and loss function $\mathcal{L}$, we aim to find the optimal subnetwork with mask $\mathbf{m} \in \{0,1\}^{|\Theta|}$ to solve the following problem:

$$\min_{\mathbf{m} \in \{0,1\}^{|\Theta|}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \max_{\mathbf{x}' \in \mathcal{S}_\epsilon(\mathbf{x})} \mathcal{L}(f(\theta \odot \mathbf{m}, \mathbf{x}'), y) \tag{2}$$

where $\mathcal{S}_\epsilon(\mathbf{x})$ is the adversarial budget and $\odot$ the elementwise multiplication.

Instead of optimization parameter θ, we optimize the architecture represented by $\mathbf{m}$.

Definition (Adversarial Edge-Popup Problem)

Given a classificaiotn model $f(\theta, \mathbf{x}) : \Theta \times \mathbb{R}^M \to \mathbb{R}^K$ with initial parameter θ, data points drawn from the distribution $(\mathbf{x}, y) \sim \mathcal{D}$ and loss function $\mathcal{L}$, we aim to find the optimal subnetwork with mask $\mathbf{m} \in \{0,1\}^{|\Theta|}$ to solve the following problem:

$$\min_{\mathbf{m} \in \{0,1\}^{|\Theta|}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \max_{\mathbf{x}' \in \mathcal{S}_\epsilon(\mathbf{x})} \mathcal{L}(f(\theta \odot \mathbf{m}, \mathbf{x}'), y) \tag{2}$$

where $\mathcal{S}_\epsilon(\mathbf{x})$ is the adversarial budget and $\odot$ the elementwise multiplication.

Instead of optimization parameter θ, we optimize the architecture represented by $\mathbf{m}$.
Discrete optimization?

$$\min_{\mathbf{m}\in\{0,1\}^{|\Theta|}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \max_{\mathbf{x}'\in\mathcal{S}_\epsilon(\mathbf{x})} \mathcal{L}(f(\theta\odot\mathbf{m},\mathbf{x}'),y)$$

$$\min_{\mathbf{m}\in\{0,1\}^{|\Theta|}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \max_{\mathbf{x}'\in\mathcal{S}_\epsilon(\mathbf{x})} \mathcal{L}(f(\theta \odot \mathbf{m}, \mathbf{x}'), y)$$

Trainable continuous alternative variable 'score' $\mathbf{s}$ to replace $\mathbf{m}$: $\mathbf{m} = M(\mathbf{s})$

- Forward pass: elements with top k scores are assigned 1 in $\mathbf{m}$ and 0 otherwise.
  - The value of k depends on the pruning policy.
- Backward pass: treated as the identity function ('gradient goes through').

$$\min_{\mathbf{m}\in\{0,1\}^{|\Theta|}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \max_{\mathbf{x}'\in\mathcal{S}_\epsilon(\mathbf{x})} \mathcal{L}(f(\theta \odot \mathbf{m}, \mathbf{x}'), y)$$

Trainable continuous alternative variable 'score' $\mathbf{s}$ to replace $\mathbf{m}$: $\mathbf{m} = M(\mathbf{s})$

- Forward pass: elements with top k scores are assigned 1 in $\mathbf{m}$ and 0 otherwise.
  - The value of k depends on the pruning policy.
- Backward pass: treated as the identity function ('gradient goes through').

- Approximation in the backward does not affect adversarial examples generation.
- Can be combined with any variant of adversarial training.
  - Accelerated version, adversarial training with regularization.

Algorithm 1: Edge pop-up algorithm for adversarial robustness.

Input: training set $\mathcal{D}$, batch size B, PGD step size α and iteration number n, adversarial budget $\mathcal{S}_\epsilon$, pruning rate r, mask function M, the optimizer.
Random initialize the model parameters θ and the scores $\mathbf{s}$.
for Sample a mini-batch $\{\mathbf{x}_i, y_i\}_{i=1}^{B} \sim \mathcal{D}$ **do**
  for i = 1, 2, ..., B **do**
    Sample a random noise δ within the adversarial budget $\mathcal{S}_\epsilon$.
    $\mathbf{x}_i^{(0)} = \mathbf{x}_i + \delta$
    for j = 1, 2, ..., n **do**
      $\mathbf{x}_i^{(j)} = \mathbf{x}_i^{(j-1)} + \alpha \nabla_{\mathbf{x}_i^{(j-1)}} \mathcal{L}(f(\theta \odot M(\mathbf{s}, r), \mathbf{x}_i^{(j-1)}), y_i)$
      $\mathbf{x}_i^{(j)} = \mathbf{x}_i + \Pi_{\mathcal{S}_\epsilon}\left(\mathbf{x}_i^{(j)} - \mathbf{x}_i\right)$
    end for
  end for
  Calculate the gradient $\mathbf{g} = \frac{1}{B} \sum_{i=1}^{B} \nabla_{\mathbf{s}} \mathcal{L}(f(\theta \odot M(\mathbf{s}, r), \mathbf{x}_i^{(n)}), y_i)$
  Update the score $\mathbf{s}$ using the optimizer.
end for
Output: the pruning mask $M(\mathbf{s}, r)$.

Two pruning strategies for pruning rate r in existing works:

- Global pruning: top $(1-r)|\Theta|$ elements with the largest scores.
- Fixed pruning rate: prune parameters of each layer with the same proportion.

Two pruning strategies for pruning rate r in existing works:

- Global pruning: top $(1-r)|\Theta|$ elements with the largest scores.
- Fixed pruning rate: prune parameters of each layer with the same proportion.

However... these do not achieve ideal performance in the context of adversarial training.

Consider a neural network with n_1, n_2, ..., n_{L-1}, n_L parameters originally, and m_1, m_2, ..., m_{L-1}, m_L parameters after pruning. The pruning rate $1 - \frac{m_1+m_2+...+m_L}{n_1+n_2+...+n_L} = r$ is fixed.

Consider a neural network with n_1, n_2, ..., n_{L-1}, n_L parameters originally, and m_1, m_2, ..., m_{L-1}, m_L parameters after pruning. The pruning rate $1 - \frac{m_1+m_2+...+m_L}{n_1+n_2+...+n_L} = r$ is fixed.

Fixed pruning rate

$$\frac{m_1}{n_1} = \frac{m_2}{n_2} = ... = \frac{m_L}{n_L}$$

Consider a neural network with n_1, n_2, ..., n_{L-1}, n_L parameters originally, and m_1, m_2, ..., m_{L-1}, m_L parameters after pruning. The pruning rate $1 - \frac{m_1+m_2+...+m_L}{n_1+n_2+...+n_L} = r$ is fixed.

Fixed pruning rate

$$\frac{m_1}{n_1} = \frac{m_2}{n_2} = ... = \frac{m_L}{n_L}$$

Maximal of $\Pi_{i=1}^{L} \binom{n_i}{m_i}$

Consider a neural network with n_1, n_2, ..., n_{L-1}, n_L parameters originally, and m_1, m_2, ..., m_{L-1}, m_L parameters after pruning. The pruning rate $1 - \frac{m_1+m_2+...+m_L}{n_1+n_2+...+n_L} = r$ is fixed.

Fixed pruning rate

$$\frac{m_1}{n_1} = \frac{m_2}{n_2} = ... = \frac{m_L}{n_L}$$

Maximal of $\Pi_{i=1}^{L} \binom{n_i}{m_i}$

Maximize the search space for $\mathbf{m}$.

Consider a neural network with n_1, n_2, ..., n_{L-1}, n_L parameters originally, and m_1, m_2, ..., m_{L-1}, m_L parameters after pruning. The pruning rate $1 - \frac{m_1+m_2+...+m_L}{n_1+n_2+...+n_L} = r$ is fixed.

Fixed pruning rate

$$\frac{m_1}{n_1} = \frac{m_2}{n_2} = ... = \frac{m_L}{n_L}$$

Maximal of $\Pi_{i=1}^{L} \binom{n_i}{m_i}$

Maximize the search space for $\mathbf{m}$.

Maximize # input-output paths.

Consider a neural network with n_1, n_2, ..., n_{L-1}, n_L parameters originally, and m_1, m_2, ..., m_{L-1}, m_L parameters after pruning. The pruning rate $1 - \frac{m_1+m_2+...+m_L}{n_1+n_2+...+n_L} = r$ is fixed.

Fixed pruning rate

$$\frac{m_1}{n_1} = \frac{m_2}{n_2} = ... = \frac{m_L}{n_L}$$

Maximal of $\Pi_{i=1}^{L} \binom{n_i}{m_i}$

Maximize the search space for $\mathbf{m}$.

Maximal of $\Pi_{i=1}^{L} m_i$

Maximize # input-output paths.

Consider a neural network with n_1, n_2, ..., n_{L-1}, n_L parameters originally, and m_1, m_2, ..., m_{L-1}, m_L parameters after pruning. The pruning rate $1 - \frac{m_1+m_2+...+m_L}{n_1+n_2+...+n_L} = r$ is fixed.

Fixed pruning rate

$$\frac{m_1}{n_1} = \frac{m_2}{n_2} = ... = \frac{m_L}{n_L}$$

Maximal of $\Pi_{i=1}^{L} \binom{n_i}{m_i}$

Maximize the search space for $\mathbf{m}$.

Fixed parameter number

$$m_1 = m_2 = ... = m_L$$

Maximal of $\Pi_{i=1}^{L} m_i$

Maximize # input-output paths.

Consider a neural network with n_1, n_2, ..., n_{L-1}, n_L parameters originally, and m_1, m_2, ..., m_{L-1}, m_L parameters after pruning. The pruning rate $1 - \frac{m_1+m_2+...+m_L}{n_1+n_2+...+n_L} = r$ is fixed.

Fixed pruning rate

$$\frac{m_1}{n_1} = \frac{m_2}{n_2} = ... = \frac{m_L}{n_L}$$

Maximal of $\Pi_{i=1}^{L} \binom{n_i}{m_i}$

Maximize the search space for **m**.

Poor performance when r is big.

Fixed parameter number

$$m_1 = m_2 = ... = m_L$$

Maximal of $\Pi_{i=1}^{L} m_i$

Maximize # input-output paths.

Poor performance when r is small.

Adaptive pruning strategy: $\frac{m_1}{n_1^p} = \frac{m_2}{n_2^p} = ... = \frac{m_L}{n_L^p}$

- Degrade to fixed pruning rate and fixed parameter number with $p = 1$ and $p = 0$.
- Larger $r \rightarrow$ smaller p; smaller $r \rightarrow$ larger p.

Methodology

Binary Initialization

- The scale of initializaiton is voided by normalization layers.

- The scale of initializaiton is voided by normalization layers.
- Each linear layer should be followed by a normalization layer, including the last layer.

- The scale of initializaiton is voided by normalization layers.
- Each linear layer should be followed by a normalization layer, including the last layer.

Two advantages of models including the last normalization layer.

- The scale of initializaiton is voided by normalization layers.
- Each linear layer should be followed by a normalization layer, including the last layer.

Two advantages of models including the last normalization layer.

- Arbitrary initialization scale of random parameters.
  - Binary initialization: save $1/2$ and $1/4$ computational time in forward and backward.

- The scale of initializaiton is voided by normalization layers.
- Each linear layer should be followed by a normalization layer, including the last layer.

Two advantages of models including the last normalization layer.

- Arbitrary initialization scale of random parameters.
  - Binary initialization: save $1/2$ and $1/4$ computational time in forward and backward.
- Insensitive to the initialization scale of $\mathbf{m}$.

Content

Pruning rate r and p in adaptive pruning strategy.

Prune Strategy	$r = 0.5$	$r = 0.8$	$r = 0.9$	$r = 0.95$	$r = 0.99$	$r = 0.995$	$r = 0.998$
$p = 0.0$	2.16	6.86	23.01	41.61	44.60	40.70	**34.97**
$p = 0.1$	4.35	15.03	28.12	42.65	**44.88**	**40.97**	33.09
$p = 0.2$	8.01	19.21	27.99	43.72	42.92	40.52	32.99
$p = 0.5$	9.21	32.70	42.84	43.62	42.45	40.55	30.08
$p = 0.8$	28.90	41.51	**43.64**	**43.88**	39.12	33.61	28.07
$p = 0.9$	39.09	41.71	43.07	42.28	38.68	33.89	17.43
$p = 1.0$	**42.85**	**43.23**	42.13	41.12	34.57	26.67	20.56

Table: Robust accuracy (in %) on the CIFAR10 test set under different pruning rates r and values of p in *adaptive pruning*. The best result for each pruning rate is marked in bold.

Advantages with the last batch normalization layer.
Assume the score **s** is initialized based on $\mathcal{N}(0, a^2)$.

Model	Value of a in score initialization			
	0.001	0.01	0.1	1
no LBN	33.08	39.96	**41.01**	31.04
LBN	**45.06**	44.88	44.63	44.41

Table: Robust accuracy (in %) on the CIFAR10 test set for models with and without the last batch normalization layer (LBN) under different values of a for score **s** initialization. The best results are marked in bold.

Advantages with the last batch normalization layer.

Prune Strategy	Signed KC		Binary	
	no LBN	LBN	no LBN	LBN
$p = 0.0$	39.38	42.83	40.94	44.65
$p = 0.1$	39.62	45.01	**41.01**	**45.06**
$p = 0.2$	36.66	**45.04**	37.85	41.58
$p = 0.5$	**39.98**	42.64	40.61	39.95
$p = 0.8$	37.96	41.71	35.15	38.95
$p = 0.9$	34.75	40.14	35.64	35.81
$p = 1.0$	36.88	39.32	30.02	30.62

Table: Robust accuracy (in %) on the CIFAR10 test set with the *Signed Kaiming Constant* (Signed KC) and the binary initialization. We include models both with and without the last batch normalization layer (LBN). The best results are marked in bold.

Experiment

Comparison with Existing Works

Method	Architecture	Pruning Strategy	Pruning Rate	CIFAR10		CIFAR100	
				FP	Binary	FP	Binary
AT	ResNet34	-	-	43.26	40.34	36.63	26.49
AT	ResNet34-LBN	-	-	42.39	39.58	35.15	32.98
HYDRA	ResNet34	$p = 0.1$	0.99	42.73	29.28	33.00	23.60
HYDRA	ResNet34	$p = 1.0$	0.99	40.51	26.40	31.09	18.24
HYDRA	ResNet34-LBN	$p = 0.1$	0.99	40.55	33.99	13.63	24.69
HYDRA	ResNet34-LBN	$p = 1.0$	0.99	32.93	26.23	29.96	17.75
ATMC	ResNet34	Global	0.99	34.14	25.62	25.10	11.09
ATMC	ResNet34	$p = 0.1$	0.99	34.58	24.65	25.37	11.04
ATMC	ResNet34	$p = 1.0$	0.99	30.50	20.21	22.28	2.53
ATMC	ResNet34-LBN	Global	0.99	33.55	19.01	23.16	15.73
ATMC	ResNet34-LBN	$p = 0.1$	0.99	31.61	22.88	25.16	17.33
ATMC	ResNet34-LBN	$p = 1.0$	0.99	27.88	13.22	22.12	9.55
Ours	ResNet34-LBN	$p = 0.1$	0.99	-	**45.06**	-	**34.83**
Ours	ResNet34-LBN	$p = 1.0$	0.99	-	34.57	-	26.32
Ours (Fast)	ResNet34-LBN	$p = 0.1$	0.99	-	40.77	-	34.45
Ours (Fast)	ResNet34-LBN	$p = 1.0$	0.99	-	29.68	-	24.97

Table: Robust accuracy (in %) on the CIFAR10 and CIFAR100 for AT, HYDRA, ATMC and our proposed method. The best results for full precision (FP) models are underlined; the best results for binary models are marked in bold.

Experiment

Comparison with Existing Works

Method	Architecture	Pruning Strategy	Pruning Rate	CIFAR10		CIFAR100	
				FP	Binary	FP	Binary
AT	ResNet34	-	-	80.99	74.37	61.48	47.87
AT	ResNet34-LBN	-	-	80.96	74.17	57.73	60.08
HYDRA	ResNet34	$p = 0.1$	0.99	75.31	62.09	55.92	45.96
HYDRA	ResNet34	$p = 1.0$	0.99	73.89	59.69	54.88	37.84
HYDRA	ResNet34-LBN	$p = 0.1$	0.99	74.37	68.43	53.65	46.09
HYDRA	ResNet34-LBN	$p = 1.0$	0.99	66.59	57.15	26.64	37.16
ATMC	ResNet34	Global	0.99	81.85	72.97	57.15	36.39
ATMC	ResNet34	$p = 0.1$	0.99	81.37	73.34	59.99	32.68
ATMC	ResNet34	$p = 1.0$	0.99	74.33	57.01	54.06	5.19
ATMC	ResNet34-LBN	Global	0.99	80.72	68.25	55.69	40.51
ATMC	ResNet34-LBN	$p = 0.1$	0.99	76.39	51.31	57.34	43.97
ATMC	ResNet34-LBN	$p = 1.0$	0.99	69.83	19.26	49.69	28.78
Ours	ResNet34-LBN	$p = 0.1$	0.99	-	76.59	-	60.16
Ours	ResNet34-LBN	$p = 1.0$	0.99	-	65.70	-	47.21
Ours (Fast)	ResNet34-LBN	$p = 0.1$	0.99	-	**81.63**	-	**63.73**
Ours (Fast)	ResNet34-LBN	$p = 1.0$	0.99	-	70.72	-	50.98

Table: Clean accuracy (in %) on the CIFAR10 and CIFAR100 for AT, HYDRA, ATMC and our proposed method. The best results for full precision (FP) models are underlined; the best results for binary models are marked in bold.

Experiments

Structures of Pruned Subset Networks

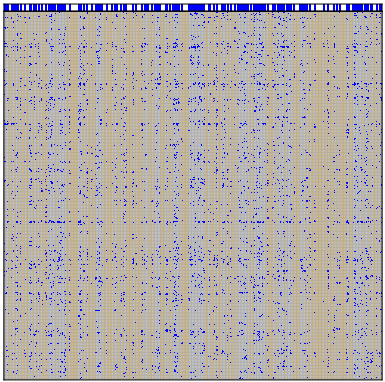

Figure: Mask visualization of the weight of a random convolutional layer in our model. The parameters retained is highlighted as blue dots. The dimension of the convolutional kernel is (r_{out}, r_{in}, 3, 3). We reshape this kernel in rectangle of shape ($r_{out} \times 3$, $r_{in} \times 3$). Channels with no remaining weight are colored orange. The top bar indicates whether the channel is empty (white) or not (blue).

Experiments

Structures of Pruned Subset Networks

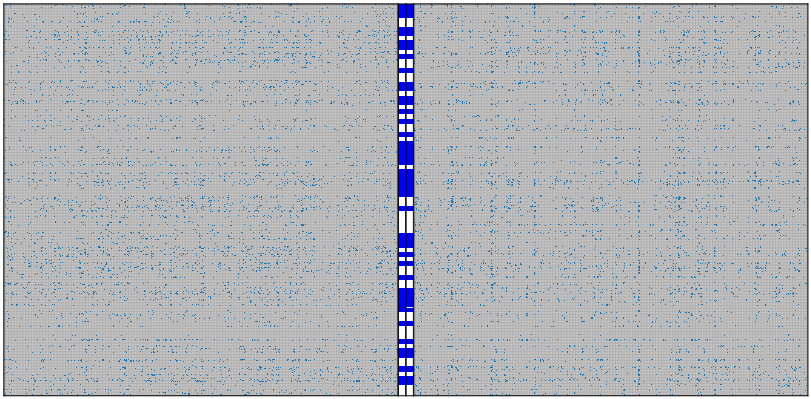

Figure: Distribution of weights in two consecutive layers. In layer1 (left), the masks are reshaped into ($r_{out} \times 3$, $r_{in} \times 3$) while masks in layer2 (right) are reshaped into ($r_{in} \times 3$, $r_{out} \times 3$). The output channels totally pruned in layer1 and the input channels totally pruned in layer2 are highlighted as the white bars in the middle.

Content

Conclusion

- We extend 'Strong Lottery Hypothesis' to robust learning.
- We propose 'adaptive pruning strategy' to improve the performance.
- We adapt the algorithm to binary networks to accelerate computation.

Unsolved challenges and potential future works.

- Automatic pruning strategy.
- Structured pruning on randomly-initialized network.

Thank You!