SNIP: Single-Shot Network Pruning
CS-439 Optimization for Machine Learning – Class Project

Aoyu Gong, Sijia Du, Qiyuan Dong
School of Computer and Communication Sciences, EPFL, Switzerland

***Abstract*—This project focuses on a network pruning algorithm, namely SNIP for single-shot network pruning. We reproduce the experiments of [1] using a similar experimental setup, i.e., evaluate the algorithm on two datasets with seven network architectures given by the reference. To study its robustness, we further introduce one dataset and nine network architectures, provide numerical and visualized results with respect to a wide range of configurations, and compare them with those in [1].**

I. Network Pruning Algorithm

In this section, we briefly introduce SNIP. Let $\mathbf{w} \in \mathbf{R}^M$ denote the parameter vector of a neural network, where M denote the total number of the parameters. To measure the importance of each connection independently of its weight, let $\mathbf{c} \in \{0,1\}^M$ denote the indicator vector representing the connectivity of $\mathbf{w}$. Then, given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, a neural network to be trained, and a sparsity degree κ, the constrained optimization problem can be written as

$$\begin{aligned} &\text{minimize} \quad L(\mathbf{c} \odot \mathbf{w}; \mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} \ell\big(\mathbf{c} \odot \mathbf{w}; (x_i, y_i)\big) \\ &\text{subject to} \quad \|\mathbf{c}\|_0 \leq \kappa, \end{aligned} \tag{1}$$

with variables $\mathbf{c} \in \{0,1\}^M$ and $\mathbf{w} \in \mathbf{R}^M$, where $\odot$ is the element-wise product, $\ell(\cdot)$ is the loss function, $\|\cdot\|_0$ is the ℓ_0-norm, and κ is the number of non-zero weights.

For $j \in \mathcal{M} := \{1, 2, \ldots, M\}$, the values of c_j indicates whether the connection j is active ($c_j = 1$) in the network or pruned ($c_j = 0$). The effect of removing the connection j can be measured by

$$\Delta L_j(\mathbf{w}; \mathcal{D}) = L(\mathbf{1} \odot \mathbf{w}; \mathcal{D}) - L\big((\mathbf{1} - \mathbf{e}_j) \odot \mathbf{w}; \mathcal{D}\big), \tag{2}$$

where $\mathbf{e}_j$ is a vector satisfying $\mathbf{e}_{j,k} = 1$ if $k = j$ and $\mathbf{e}_{j,k} = 0$ otherwise for $k \in \mathcal{M}$, and $\mathbf{1}$ is a vector of ones.

Since computing ΔL_j for every $j \in \mathcal{M}$ is prohibitively expensive and L is not differentiable with respect to c, by relaxing the binary constraint on c, the right-hand side of Eq. (2) can be approximated by

$$\begin{aligned} \Delta L_j(\mathbf{w}; \mathcal{D}) \approx g_j(\mathbf{w}; \mathcal{D}) &= \left.\frac{\partial L}{\partial c_j}(\mathbf{c} \odot \mathbf{w}; \mathcal{D})\right|_{\mathbf{c}=\mathbf{1}} \\ &= \lim_{\delta \to 0} \left.\frac{L(\mathbf{c} \odot \mathbf{w}; \mathcal{D}) - L\big((\mathbf{c} - \delta\mathbf{e}_j) \odot \mathbf{w}; \mathcal{D}\big)}{\delta}\right|_{\mathbf{c}=\mathbf{1}}, \end{aligned}$$

where $g_j(\mathbf{w}; \mathcal{D})$ is the derivative of L with respect to c_j. If the absolute value of $g_j(\mathbf{w}; \mathcal{D})$ is large, the connection c_j has a considerable effect on the loss and, therefore, has to be preserved to learn w_j. Based on this hypothesis, the sensitivity of the connection j is defined as

$$s_j := \frac{g_j(\mathbf{w}; \mathcal{D})}{\sum_{j' \in \mathcal{M}} g_{j'}(\mathbf{w}; \mathcal{D})}.$$

By the sensitivity vector $\mathbf{s}$, $\mathbf{c}$ can be obtained by

$$c_j = \mathbb{1}[s_j - \tilde{s}_\kappa \geq 0],$$

for each $j \in \mathcal{M}$, where $\tilde{s}_\kappa$ is the κ-th largest element in $\mathbf{s}$ and $\mathbb{1}[\cdot]$ is the indicator function. The algorithm for SNIP is formally described in Appendix A. Interested readers are referred to [1] for a detailed discussion.

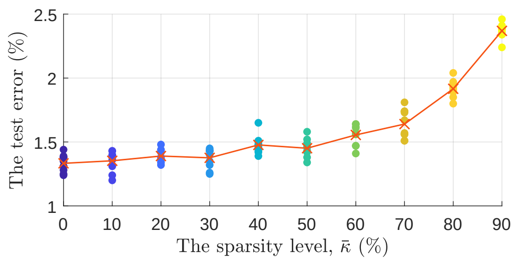
II. Experiments

In this section, we evaluate SNIP on MNIST [2], KMNIST [3], and CIFAR-10 [4] with various network architecture. For simplicity, we use MNISTs to denote both MNIST and KMNIST. The experiment setup is introduced as follows.

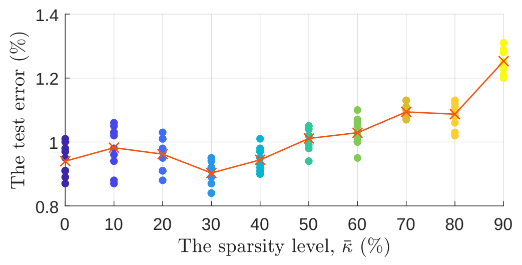
Experiment setup: The sparsity level is defined as $\bar{\kappa} := \frac{M-\kappa}{M} \times 100\%$. To effectively utilize GPU resources, batch sizes for pruning and training are set to 500 for MNISTs and 512 for CIFAR-10, unless stated otherwise. We first prune models and then train them using SGD with momentum of 0.9. The learning rate is set to 0.1 initially and decayed by 0.1 every 5000 iterations for MNISTs and 7500 iterations for CIFAR-10. The maximum numbers of iterations are set to 30000 for MNISTs and 37500 for CIFAR-10. Training images are split into a training set and a validation set at the ratio of 9 to 1. As in [1], SNIP requires no other schedules and hyperparameters. For the training set, translation up to 4 pixels and random horizontal flip are applied. The code can be found here: https://github.com/aygong/O4MLproj.

A. Comparisons under Varied Sparsity

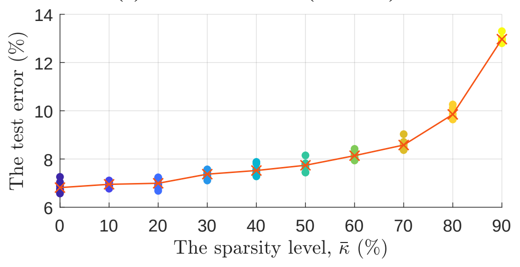
We first test SNIP on two standard neural networks, i.e., LeNet-300-100 and LeNet-5-Caffe, with MNISTs. The former consists of three fully connected layers with 2.67×10^5 parameters. The latter consists of two convolutional layers with 4.31×10^5 parameters. Fig. 1 presents the test error as a function of the sparsity level $\bar{\kappa}$, where $\bar{\kappa} = 0$ refers to the reference network. We run the experiment 10 times for each value of $\bar{\kappa}$ by changing weight initialization and random seeds.



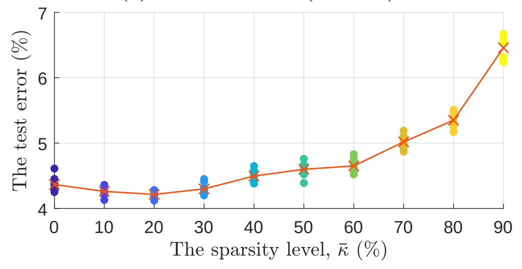
(a) LeNet-300-100 (MNIST).



(b) LeNet-5-Caffe (MNIST).



(c) LeNet-300-100 (KMNIST).



(d) LeNet-5-Caffe (KMNIST).

Figure 1: The test error as a function of the sparsity level $\bar{\kappa}$, where $\bar{\kappa} = 0$ refers to the reference network.

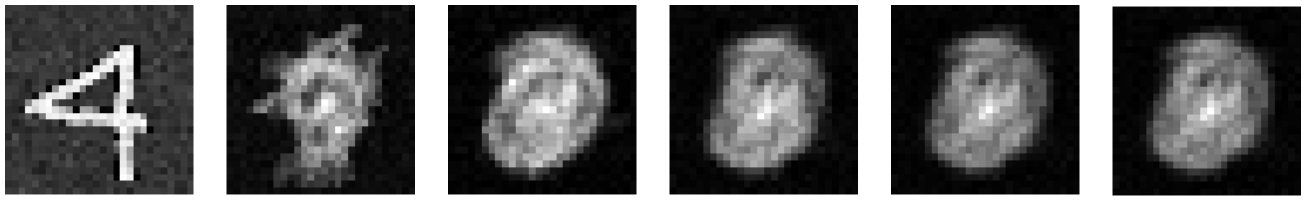
We see from Fig. 1 that the performance of the pruned networks is similar to the reference network when $\bar{\kappa}$ is low. However, there exists visible loss when $\bar{\kappa}$ is high, especially from $\bar{\kappa} = 70$ to 90: 0.73% loss in Fig. 1a, 0.16% loss in Fig. 1b, 4.39% loss in Fig. 1c, and 1.44% loss in Fig. 1d. This loss for MNIST is larger than [1] but still negligible, while this loss for KMNIST is not trivial. As illustrated in Figs. 1a and 1c, after speeding up training for MNIST, the pruned networks get performance similar to or better than [1] when $\bar{\kappa}$ is low, but worse performance when $\bar{\kappa}$ is high. Moreover, the test errors for KMNIST are greater than those for MNIST and increase more rapidly as $\bar{\kappa}$ becomes higher. This indicates that SNIP performs worse as the batch sizes increase and with more complex datasets. This phenomenon will be confirmed again in the following sections.

B. *Comparisons with Multiple Network Architectures*

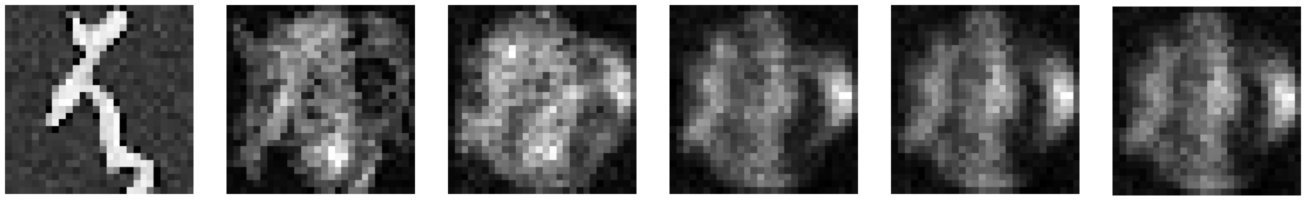
We test SNIP on CIFAR-10 with multiple network architectures, including ones provided and newly introduced.

- **AlexNet-s, b:** The same architectures as in [1].
- **VGG-C, D, like:** The same architectures as in [1].
- **ResNet-18, 34:** The same architectures as in [5] includes shortcut connections making the networks with considerably increased depth easier to be optimized.
- **SqueezeNet-vanilla, bypass:** The architectures similar to [6]. They have $50\times$ fewer parameters than AlexNet and maintains similar performance. SqueezeNet-bypass is SqueezeNet-vanilla with simple bypass connections between some Fire modules. For the two architectures, we remove dropout layers and use max pooling layers with 'VALID' padding.
- **GoogLeNet (Inception v1):** The architecture similar to [7] consists of Inception modules with dimensionality reduction. We remove auxiliary outputs and use batch normalization before the ReLU activation function.
- **DenseNet-121, 169, 201, 264:** The same architectures as in [8] have dense connectivity that improves flow of information and gradients throughout the network.

The pruning results of SNIP on various network architectures are presented in Appendix B. We see from Table I that the convolutional networks achieve performance similar to [1] at $\bar{\kappa} = 0$, but worse performance at high $\bar{\kappa}$. The loss (i.e. Δ) of all these networks is about 2%, confirming the phenomenon in Section II-A again. We further see that the performance of the squeeze and inception networks becomes much worse than the convolutional networks as $\bar{\kappa}$ increases. The reason is that, for SqueezeNet, it already has the fewest parameters and, after pruning, its preserved connections are too few to maintain similar accuracy. Although other model compression techniques are used in [6] and the compressed model has equivalent accuracy, the sparsity level is only set to 33% which is much smaller than here. For GoogLeNet, the effect of the pruned connections may be amplified after aggregation at the end of each Inception module and leads to degraded performance. Moreover, it is shown that SNIP achieves negligible or no loss in accuracy ($< 1.2\%$) on the residual and dense networks. On ResNet-34, DenseNet-201, and -264, the performance of the networks is even better at $\bar{\kappa} = 95\%$. This is because, by pruning connections with low sensitivity, the training of important connections is sped up. This result is also observed in [1]. Overall, SNIP performs well on most of the networks, especially the ones which have



(a) $|\mathcal{D}^b| = 1$. (b) $|\mathcal{D}^b| = 10$. (c) $|\mathcal{D}^b| = 100$. (d) $|\mathcal{D}^b| = 1000$. (e) $|\mathcal{D}^b| = 10000$. (f) Training set.



(g) $|\mathcal{D}^b| = 1$. (h) $|\mathcal{D}^b| = 10$. (i) $|\mathcal{D}^b| = 100$. (j) $|\mathcal{D}^b| = 1000$. (k) $|\mathcal{D}^b| = 10000$. (l) Training set.

Figure 2: The visualizations of pruned parameters of the first layer in LeNet-300-100 with different $|\mathcal{D}^b|$. Figs. 2a to 2f are the results with MNIST and Figs. 2g to 2l are the results with KMNIST. The test errors for Figs. 2a to 2f are 2.73%, 2.39%, 2.23%, 2.30%, 2.24%, 2.25% and those for Figs. 2g to 2l are 14.51%, 13.35%, 12.78%, 12.68%, 12.67%, 12.73%.

a large number of parameters and shortcut connections.

C. Understanding Which Connections Are Pruned

To illustrate which connections are actually pruned, we consider the first layer in LeNet-300-100, represented by the parameter matrix $\mathbf{w}_1 \in \mathbf{R}^{784\times 300}$. Let $\mathbf{c}_1 \in \{0,1\}^{784\times 300}$ denote its corresponding indicator matrix. We visualize $\mathbf{c}_1$ by averaging it across columns and reshaping it to a matrix of 28×28. Here, a batch of 100 training images from the same class in MNISTs (i.e. digits for MNIST and hiragana characters for KMNIST) is sampled for pruning. The results are illustrated in Appendix C. In Fig. 3, bright pixels mean parameters connected to them have high sensitivity and are preserved. We see from Fig. 3a that the parameters linked to the digits survive while the majority to the background is pruned. Moreover, the pixels connected with the important parameters seem to reconstruct the images. Similar results are observed in Fig. 3b at low $\bar{\kappa}$. However, when $\bar{\kappa}$ is high, the parameters linked to the hiragana characters are also largely pruned. For instance, it is hard to identify ‘き’ (the character in the second column) at $\bar{\kappa} = 90$. This confirms the phenomenon in Section II-A, i.e., SNIP performs worse with more complex datasets when $\bar{\kappa}$ is high.

Let $c_{k,1}$ and $c_{k,0}$ represent the numbers of preserved and pruned parameters of the k-th layer in LeNet-300-100 for $k \in \{1,2,3\}$. The ratios of $c_{k,1}$ and $c_{k,0}$ to $c_{k,0} + c_{k,1}$ as a function of the sparsity level $\bar{\kappa}$ are shown in Fig. 4. We see that $\frac{c_{k,0}}{c_{k,0}+c_{k,1}}$ increases as $\bar{\kappa}$ becomes higher and decreases from low layers to high layers. The ratios for MNIST are similar to those for KMNIST in all the cases. As the area of the digits is smaller than the hiragana characters, after being pruned similar ratios of parameters, the visualizations of the digits are clearer than the characters as in Fig. 3.

D. Effects of Batch Sizes

Recall that SNIP uses a batch of training images $\mathcal{D}^b$ to compute the sensitivity vector $\mathbf{s}$. To study the effect of the batch sizes, we first change the batch size for pruning and set that to 500 for training. The visualizations are presented in Fig. 2. Every test error is obtained from 10 independent runs. We set $\bar{\kappa} = 90$ in all the cases. It is shown that, for $|\mathcal{D}^b| = 1$, the sampled images are ‘4’ for MNIST and ‘は’ for KMNIST. For both datasets, as $|\mathcal{D}^b|$ increases, the preserved parameters are closer to those pruned with the training sets. Besides, there is a nontrivial decrease in the test errors from $|\mathcal{D}^b| = 1$ to 100 and almost no decrease as $|\mathcal{D}^b|$ continues to increase. Such observations indicate that SNIP can prune the network effectively only with a small batch size in single shot (i.e. prune once before training). We further change the batch sizes for pruning and training at $\bar{\kappa} = 10, 50, 90$. The results are presented in Appendix D. We see from Table II that SNIP achieves the best performance when $|\mathcal{D}^b| = 100$ confirming the results in Section II-A again.

III. Conclusion

To test the performance of SNIP, experiments of [1] have been reproduced using a similar experimental setup. A new dataset KMNIST and nine network architectures, including residual, squeeze, inception, and dense networks, have been introduced for investigating the robustness of the algorithm. Numerical results and visualizations have shown that SNIP performs well in most cases, while its performance becomes worse on more complex datasets with larger batch sizes. The interpretation of the results has been provided.

Acknowledgements

The authors thank the TML and MLO groups for their careful guidance.

References

[1] N. Lee, T. Ajanthan, and P. H. S. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," in *Proc. ICLR*, 2019.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep learning for classical japanese literature," *arXiv preprint arXiv:1812.01718*, 2018.

[4] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016, pp. 770–778.

[6] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with $50x$ fewer parameters and < 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE CVPR*, 2015, pp. 1–9.

[8] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE CVPR*, 2017, pp. 4700–4708.

Appendix A. The Algorithm

The algorithm for SNIP is formally described as follows.

Algorithm 1 SNIP: Single-shot Network Pruning based on Connection Sensitivity

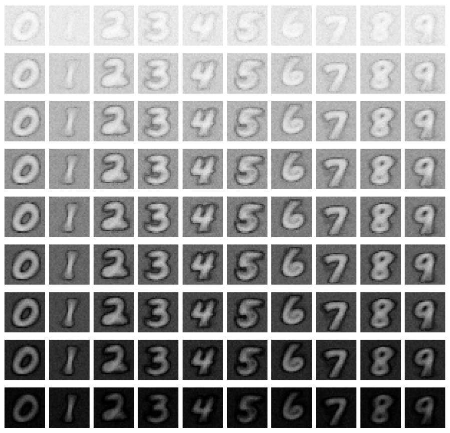
```
Require: A training set $\mathcal{D}$, sparsity degree $\kappa$, and loss function $\ell(\cdot)$    ▷ Refer to the problem (1).
Ensure: $\|\mathbf{w}\|_0 \leq \kappa$
1: $\mathbf{w} \leftarrow$ VarianceScalingInitialization
2: $\mathcal{D}^b = \{(x_i, y_i)\}_{i=1}^b \sim \mathcal{D}$    ▷ Sample a batch of training images for pruning.
3: $\mathbf{s} \leftarrow \frac{g_j(\mathbf{w};\mathcal{D}^b)}{\sum_{j' \in \mathcal{M}} g_{j'}(\mathbf{w};\mathcal{D}^b)}, \forall j \in \mathcal{M}$    ▷ Compute the sensitivity of each connection.
4: $\tilde{\mathbf{s}} \leftarrow$ SortDescending($\mathbf{s}$)
5: $\mathbf{c} \leftarrow \mathbb{1}[s_j - \tilde{s}_\kappa \geq 0], \forall j \in \mathcal{M}$    ▷ Pruning (i.e. preserve connections with top-$\kappa$ largest sensitivity).
6: $\mathbf{w}^* \leftarrow \arg\min_{\mathbf{w} \in \mathbf{R}^M} L(\mathbf{c} \odot \mathbf{w}; \mathcal{D})$    ▷ Standard training.
7: $\mathbf{w}^* \leftarrow \mathbf{c} \odot \mathbf{w}^*$
```

Appendix B. The Pruning Results on Multiple Network Architectures

Architecture	Model	Sparsity (%)	# Parameters	Error (%)	Δ
Convolutional	AlexNet-s	90.0	$5.07 \times 10^6 \rightarrow 5.07 \times 10^5$	$14.42 \rightarrow 16.66$	$+2.24$
	AlexNet-b	90.0	$8.49 \times 10^6 \rightarrow 8.49 \times 10^5$	$14.52 \rightarrow 16.68$	$+2.16$
	VGG-C	95.0	$1.05 \times 10^7 \rightarrow 5.26 \times 10^5$	$7.56 \rightarrow 9.64$	$+2.08$
	VGG-D	95.0	$1.52 \times 10^7 \rightarrow 7.62 \times 10^5$	$7.18 \rightarrow 9.24$	$+2.06$
	VGG-like	97.0	$1.50 \times 10^7 \rightarrow 4.49 \times 10^5$	$7.35 \rightarrow 9.73$	$+2.38$
Residual	ResNet-18	95.0	$1.10 \times 10^7 \rightarrow 5.50 \times 10^5$	$8.20 \rightarrow 8.45$	$+0.25$
	ResNet-34	95.0	$2.11 \times 10^7 \rightarrow 1.06 \times 10^6$	$12.68 \rightarrow 8.26$	$\mathbf{-4.42}$
Squeeze	SqueezeNet-vanilla	95.0	$7.41 \times 10^5 \rightarrow 3.70 \times 10^4$	$14.49 \rightarrow 20.95$	$+6.46$
	SqueezeNet-bypass	95.0	$7.41 \times 10^5 \rightarrow 3.70 \times 10^4$	$13.53 \rightarrow 18.88$	$+5.35$
Inception	GoogLeNet	95.0	$2.75 \times 10^6 \rightarrow 1.38 \times 10^5$	$14.11 \rightarrow 19.23$	$+5.12$
Dense	DenseNet-121	95.0	$4.40 \times 10^6 \rightarrow 2.20 \times 10^5$	$14.51 \rightarrow 14.75$	$+0.24$
	DenseNet-169	95.0	$7.84 \times 10^6 \rightarrow 3.92 \times 10^5$	$12.97 \rightarrow 14.12$	$+1.15$
	DenseNet-201	95.0	$1.11 \times 10^7 \rightarrow 5.55 \times 10^5$	$18.36 \rightarrow 14.50$	$\mathbf{-3.86}$
	DenseNet-264	95.0	$1.87 \times 10^7 \rightarrow 9.33 \times 10^5$	$16.79 \rightarrow 13.79$	$\mathbf{-3.00}$

Table I: The pruning results of SNIP on multiple network architectures (before $\rightarrow$ after).

Appendix C.
The Visualizations of Pruned Parameters


(a) MNIST.

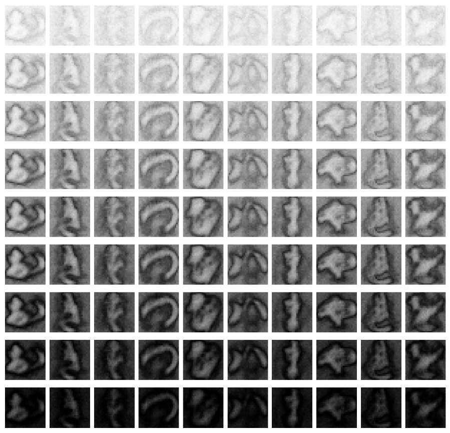
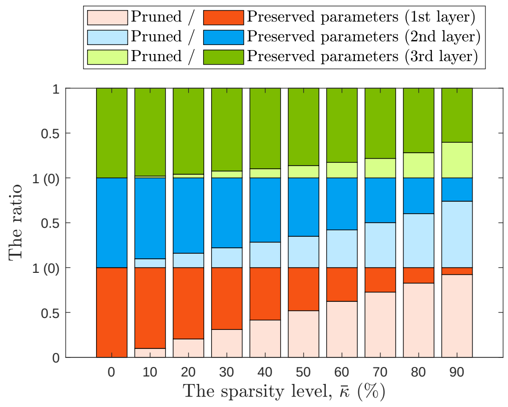
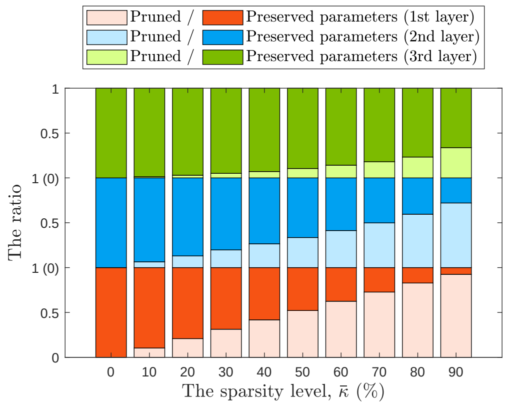

(b) KMNIST.

Figure 3: The visualizations of pruned parameters of the first layer in LeNet-300-100, where each column in Figs. 3a and 3b is the results for one class with different $\bar{\kappa}$, from 10 (top) to 90 (bottom). The batch size for pruning is set to 100.



(a) MNIST.



(b) KMNIST.

Figure 4: The ratios of $c_{k,0}$ and $c_{k,1}$ to $c_{k,0} + c_{k,1}$ for $k \in \{1, 2, 3\}$ as a function of the sparsity level $\bar{\kappa}$. The batches for pruning are sampled from the entire training sets and their size is set to 100.

Appendix D.
The Pruning Results Using Different Batch Sizes

Dataset	Sparsity (%)	Error (%)			
–	–	$\lvert\mathcal{D}^b\rvert = 50$	$\lvert\mathcal{D}^b\rvert = 100$	$\lvert\mathcal{D}^b\rvert = 500$	$\lvert\mathcal{D}^b\rvert = 1000$
MNIST	10.0	1.43	1.26	1.35	1.50
	50.0	1.40	1.32	1.47	1.67
	90.0	1.99	2.09	2.30	2.62
KMNIST	10.0	7.64	6.65	7.09	8.44
	50.0	7.88	7.05	7.62	8.44
	90.0	12.08	11.87	12.56	13.75

Table II: The pruning results of SNIP on LeNet-300-100 using different batch sizes for pruning and training when $\bar{\kappa} = 10, 50, 90$. The product of the batch size and the maximum number of iterations are set to 1.5×10^7.