ICE-Pick: Iterative Cost-Efficient Pruning for DNNs

Wenhao Hu [1] **Perry Gibson** [1] **José Cano** [1]

Abstract

Pruning is one of the main compression methods for Deep Neural Networks (DNNs), where less relevant parameters are removed from a DNN model to reduce its memory footprint. To get better final accuracy, pruning is often performed iteratively with increasing amounts of parameters being removed in each step, and fine-tuning (i.e., additional training epochs) being applied to the remaining parameters. However, this process can be very time-consuming, since the fine-tuning process is applied after every pruning step and calculates gradients for the whole model. Motivated by these overheads, in this paper we propose ICE-Pick, a novel threshold-guided fine-tuning method which freezes less sensitive layers and leverages a custom pruning-aware learning rate scheduler. We evaluate ICE-Pick using ResNet-110, ResNet-152, and MobileNetV2 (all defined for CIFAR-10), and show that it can save up to 87.6% of the pruning time while maintaining accuracy.

1. Introduction

Pruning is a popular method to compress large Deep Neural Network (DNN) models, making them more compact and potentially faster (Deng et al., 2020). Typically, less important parameters are removed from a pre-trained model. The remaining parameters are fine-tuned on the training dataset to compensate for any potential accuracy loss.

To better maintain accuracy, some pruning methods leverage iterative fine-tuning, where an increasing proportion of the model is pruned, with additional training (fine-tuning) being performed after each pruning step. This continues until a target compression ratio is reached. However, this process can be prohibitively expensive. One-shot pruning attempts to mitigate this issue by pruning all of the parameters in a single step, and fine-tuning once (Li et al., 2016). However, this may come with high accuracy penalties.

Motivated by these observations, in this work we propose ICE-Pick (iterative cost-efficient), a simple yet effective accuracy threshold-guided fine-tuning method that leverages a custom pruning-aware learning rate scheduler and layer freezing to reduce training costs, while still maintaining similar accuracy. ICE-Pick accelerates the pruning process by skipping fine-tuning when a given pruning step has a tolerable impact on accuracy. Even when fine-tuning is required, we adapt the concept of layer freezing seen in transfer learning (Liu et al., 2021b). We identify layers whose parameters are less likely to change significantly during fine-tuning and freeze them for cheaper training.

The contributions of this paper include the following:

- We propose and describe our ICE-Pick method, with components including layer freezing, threshold-guided fine-tuning, and a tunable pruning-aware learning rate scheduler.
- We validate the components of ICE-Pick, justifying the use of layer freezing and the benefits of using our pruning-aware learning rate scheduler.
- We evaluate ICE-Pick using ResNet-110, ResNet-152, and MobileNetV2 with the CIFAR-10 dataset, showing that it can save up to 87.6% of the pruning time while maintaining accuracy.

2. Background and Related Work

Pruning methods can be characterized using four key dimensions (Blalock et al., 2020): *Structure*, *Scoring*, *Scheduling*, and *Fine-tuning*. Structure is the level of granularity of the pruning, and is generally divided into two classes: unstructured and structured. In unstructured pruning individual weights are removed, whereas structured pruning removes groups of parameters such as whole filters/channels. In this paper, our ICE-Pick method is formulated to focus on filter pruning. Scoring is how we decide which parameters to prune. For example, a popular approach is to use the L1-norm (Li et al., 2016), where we prune parameters with absolute values closer to zero. Scheduling is how much

[1]School of Computing Science, University of Glasgow, Glasgow, Scotland, United Kingdom. Correspondence to: Wenhao Hu <2692597H@student.gla.ac.uk>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

we prune in each step. For example, we might perform all of the pruning in a single step (one-shot pruning), or gradually increase the amount of pruning in iterative steps. Finally, fine-tuning is how we recover lost accuracy by training the model for a small number of epochs.

As compared to one-shot pruning, iterative pruning can yield better accuracies if we fine-tune after every pruning step (Li et al., 2016). However, the larger the model is, the more expensive fine-tuning can be. Therefore, many methods only fine-tune for 1 or 2 epochs with some accuracy penalty, sometimes training more epochs for the final pruning step (Joo et al., 2021; Luo & Wu, 2017; Luo et al., 2017). Note that even with a low number of epochs the fine-tuning stage can still be prohibitively expensive.

Fine-tuning is not a concept exclusive to pruning. In transfer learning (Brock et al., 2017), fine-tuning is also an important step that adapts the models to new data. Similar to pruning, fine-tuning in transfer learning is also very time-consuming (Liu et al., 2021b). To address this problem, some methods use layer freezing to accelerate the fine-tuning process (Brock et al., 2017; Lee et al., 2019; Liu et al., 2021b), where the parameters of some layers are 'frozen' such that they cannot be changed during training. We believe this idea can be adapted and applied to model pruning, due to the similar role of fine-tuning in both tasks.

3. Proposed Method

3.1. Overview

To reduce the fine-tuning time in iterative pruning, we propose ICE-Pick, a technique that combines threshold-guided fine-tuning and layer freezing. The intuition behind ICE-Pick is that instead of fine-tuning the full model on every pruning step, we freeze less sensitive layers and skip retraining when the accuracy reduction is lower than a user-defined threshold. Figure 1 gives an overview of ICE-Pick, showing how the learning rate is adjusted dynamically, with Algorithm 1 describing the steps in more detail.

As shown in Figure 1, our method has 2 main stages. In Stage 1, we freeze the model's less sensitive layers; and in Stage 2, after applying a pruning step, we fine-tune the non-frozen layers adjusting the learning rate dynamically. As well as the user-defined accuracy loss threshold, we also use 'width', which is the number of non-pruned filters in a given layer using the average across the model. Cases ①, ②, and ③ show how we gradually reduce the learning rate as the level of pruning increases, and in case ④ we stop fine-tuning when the accuracy loss is below our threshold.

Algorithm 1 is a more detailed description of the ICE-Pick method. The following subsections discuss the stages of the algorithm, and further motivate our design choices.

Algorithm 1 ICE-Pick Pruning Method

1: **Input:** (1) Layer-wise Pruning operator $P(.)$, (2) Freeze operator $\text{Freeze}(.)$, (3) Accuracy testing operator $\text{Test}(.)$, (4) Parameter calculating operator $\text{Param}(.)$, (5) Fine-tuning operator $\text{FT}(.)$, (6) Model M, (7) Original accuracy acc_{orig}, (8) Accuracy drop threshold θ, (9) Percentage of frozen layers η, (10) Initial learning rate (LR) *base*, (11) Maximum LR change Δ, (12) The level of pruning where the midpoint of the Δ is reached p, (13) The shape control β
2: **Output:** Pruned Model M_P
3: $M_{\text{Frozen}} \leftarrow \text{Freeze}(\text{M}, \eta)$
4: **for** layer $L \in M_{\text{Frozen}}$ **do** ▷ In forward direction
5: $M_{\text{Frozen}} \leftarrow P(M_{\text{Frozen}}, L)$
6: **if** $\text{acc}_{\text{orig}} - \text{Test}(M_{\text{Frozen}}) \geqslant \theta$ **then**
7: $\alpha \leftarrow \frac{\text{Param}(M_{\text{Frozen}})}{\text{Param}(M)}$
8: $max\_lr \leftarrow$ Eq. 1$(\Delta, \alpha, p, \text{base})$;
9: $\text{FT}(M_{\text{Frozen}}, max\_lr)$
10: **end if**
11: **end for**
12: $M_P \leftarrow M_{\text{Frozen}}$
13: **return** M_P

3.2. Layer Freezing

As Stage 1 in Figure 1 and line 3 of Algorithm 1 shows, we adapt the layer freezing technique commonly used in transfer learning (Brock et al., 2017; Lee et al., 2019; Liu et al., 2021b). This is motivated by our observation in Section 4.1 that there are some layers that see smaller gradient shifts than others during fine-tuning, typically layers earlier in the model. Therefore we can skip training them, reducing our fine-tuning costs.

3.3. Pruning and Fine-tuning

Lines 4-11 of Algorithm 1 show the pruning and fine-tuning steps. For each layer, including the frozen ones, we apply filter pruning. If we observe an accuracy drop higher than or equal to our threshold (line 6), then we trigger fine-tuning, otherwise we skip it. The user provides the accuracy threshold, with its value varying depending on the learning task and the user's tolerance for accuracy loss.

If we trigger fine-tuning, we want to minimize training time by converging on a higher accuracy more quickly. Dynamic learning rates are widely used to adapt the changing learning conditions (Li et al., 2019; Liu et al., 2019). It has been shown that narrower models (i.e., smaller width, with fewer filters per layer) have a narrower loss landscape (Li et al., 2018), which may require lower maximum learning rates. When we prune, we also shrink the loss landscape. Therefore, inspired by S-Cyc (Liu et al., 2021a), we design a pruning-aware learning rate scheduler where the maximum learning rate is determined by the level of pruning, defined by the average width. Our maximum learning rate is defined by:

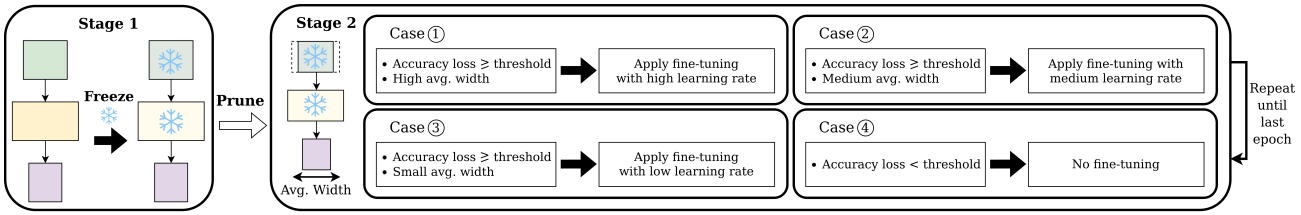
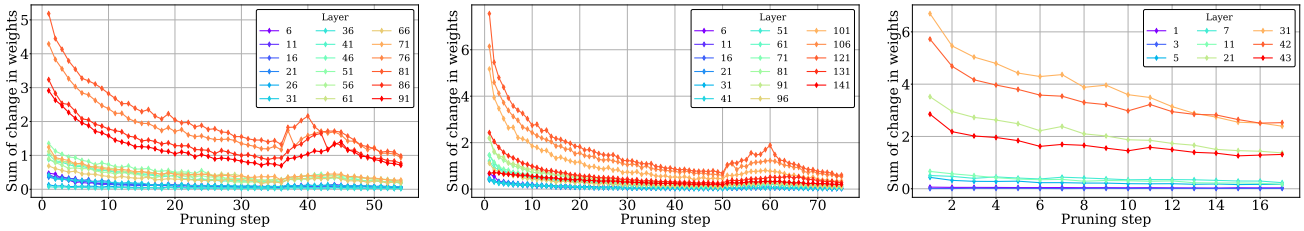


Figure 1: Overview of ICE-Pick. First, less sensitive layers are frozen (Stage 1), then for each layer we prune (the dotted lines are the edges of the pruned parts) and fine-tune the model (Stage 2). The learning rate is adjusted dynamically, and fine-tuning for a given step is halted if our accuracy loss is low.



(a) ResNet-110 (b) ResNet-152 (c) MobileNetV2

Figure 2: Weights L1 shift delta after full DNN fine-tuning during iterative pruning; the learning rate is 0.001.

$$max\_learning\_rate = base - \frac{\Delta}{1 + (\frac{\alpha}{2 \times (1-p) - \alpha})^{\beta}} \quad (1)$$

where α is the proportion of unpruned parameters, and the hyper-parameters are: $base$, the initial learning rate; Δ, the maximum change in the learning rate during pruning, with range (0, base); p, the level of pruning where the midpoint of Δ is reached, with range (0, 50%]; and β, which controls the shape of the curve, with range (0, $+\infty$). By tuning these hyper-parameters, we can flexibly formulate different types of decreasing learning schedules for different situations. Table 1 shows the comparison in total number of fine-tunings triggered to achieve comparable performance between the fixed learning rate and our ICE-Pick learning rate scheduler.

4. Evaluation

In this section, we validate ICE-Pick and demonstrate its applicability in reducing the costs of pruning while maintaining accuracy. In Section 4.1 we show how some DNN layers change much less than others during fine-tuning, justifying the use of layer freezing. Section 4.2 discusses the performance of ICE-Pick under varying parameter configurations. In Section 4.3 we validate the use of our custom ICE-Pick learning rate scheduler. Section 4.4 discusses the potential impact of ICE-Pick on future pruning works.

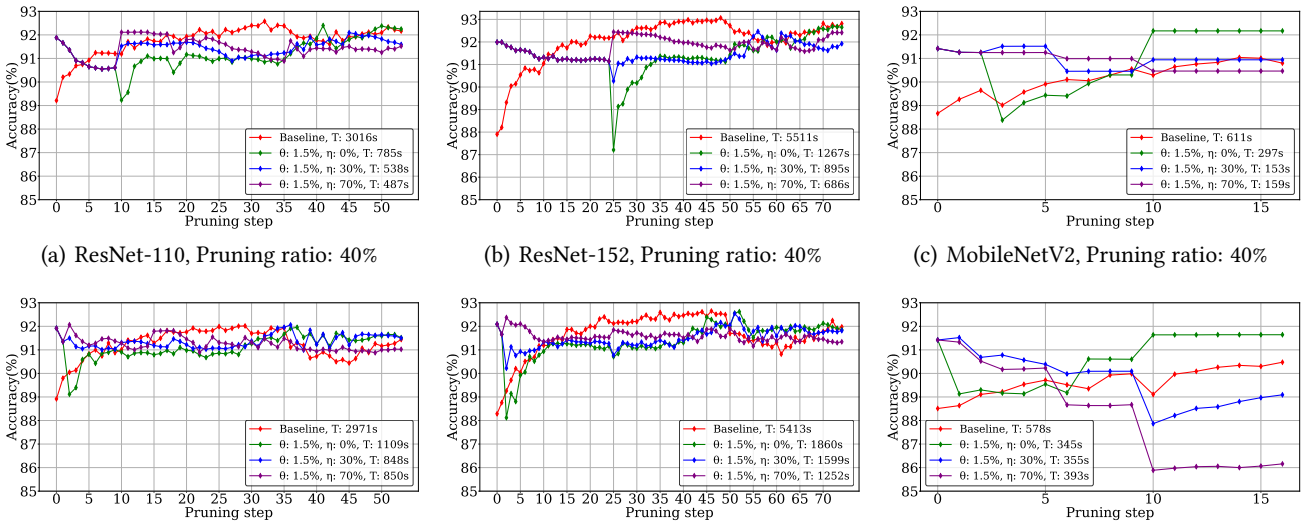
In our evaluation, we run three DNN models: ResNet-110 and ResNet-152 (He et al., 2016), and MobileNetV2 (Howard et al., 2017), all defined on the CIFAR-10 dataset (Krizhevsky et al., 2009). We evaluate them on an NVIDIA TITAN RTX GPU, taking the mean value of 10 evaluations for each experiment. We use filter-pruning with L1-norm scoring, and prune one block per step for our scheduling, where a block is a sequence of consecutive layers. For fine-tuning, we use SGD (Ruder, 2016) enhanced by knowledge distillation (Hinton et al., 2015), with at most one epoch for each step, a momentum of 0.9, a weight decay of 1e−4, and a batch size of 128 for train and test.

4.1. Validation of Layer Freezing

Figure 2 shows how the weights of different layers change as the amount of fine-tuning increases. We observe that across our three DNN models, earlier layers tend to see smaller shifts, and the ordering of the layers is generally maintained. This justifies both freezing less sensitive layers (determined by observing weight shifts from one pruning step), and applying the freezing step only once.

4.2. Parameter Perturbation

We compare the impact on accuracy and overall pruning time for different parameter combinations and pruning ratios. For our baseline, we prune with a fixed learning rate of 0.001 and do not exploit any of the features of ICE-Pick such as freezing and an accuracy drop threshold. For ICE-Pick, we prune using varying freezing ratios and an accuracy drop threshold of 1.5%. We observe that higher freezing ratios give higher reductions in overall time. For example, at a 40% pruning level (Figures 3(a)-3(c)), we re-



(a) ResNet-110, Pruning ratio: 40%
(b) ResNet-152, Pruning ratio: 40%
(c) MobileNetV2, Pruning ratio: 40%
(d) ResNet-110, Pruning ratio: 60%
(e) ResNet-152, Pruning ratio: 60%
(f) MobileNetV2, Pruning ratio: 60%

Figure 3: Impact of varying parameter configurations, in terms of accuracy as pruning increases. The freezing ratios η are 0%, 30%, 70%, with the legends reporting the total time required. θ is the value of the accuracy drop threshold.

Table 1: Comparison between fixed and ICE-Pick learning rates (LR). The pruning ratios are 60% and 40% for the ResNets and MobileNetV2 respectively and the accuracy drop threshold is 2.5%. For ICE-Pick, $base$ is 0.001, $\Delta = 7.5\text{e}{-4}$, $p = 0.12$, and $\beta = 50$. #Ft is the total number of fine-tunings triggered in 10 experiments.

Model	Freeze	LR	Accuracy	#Ft
ResNet-110	30%	0.001	90.38%	132
	30%	ICE-Pick	**90.67%**	**102**
	70%	0.001	90.22%	133
	70%	ICE-Pick	**90.55%**	**116**
ResNet-152	30%	0.001	90.35%	196
	30%	ICE-Pick	**91.35%**	**124**
	70%	0.001	90.34%	131
	70%	ICE-Pick	**90.61%**	**112**
MobileNetV2	30%	0.001	**89.56%**	21
	30%	ICE-Pick	89.26%	**20**
	70%	0.001	**90.1%**	27
	70%	ICE-Pick	89.74%	**24**

duce the time required by up to 83.9%, 87.6%, and 74.0% for ResNet-110, ResNet-152, and MobileNetV2 respectively.

For higher pruning ratios, we observe a lower average reduction in pruning time, since we need to fine-tune more, due to higher accuracy drops. At a pruning ratio of 60% (Figures 3(d)-3(f)), we save up to 71.4%, 76.9%, and 32.0% for ResNet-110, ResNet-152, and MobileNetV2 respectively.

For varying accuracy drop thresholds, we do not include these results for brevity. However, comparing thresholds of 0.5%, 1.5%, and 2.5%, as expected, higher thresholds reduced the time required while still reasonably maintaining accuracies. This shows that ICE-Pick can keep the final accuracy even with bigger accuracy drop thresholds.

4.3. Learning Rate Schedule Validation

To validate ICE-Pick's learning rate scheduler, we compare it against a fixed learning rate. Both approaches still use layer-freezing and the accuracy drop threshold to optimize fine-tuning. In Table 1 we see that our scheduler triggers significantly fewer fine-tunings for 2 out of 3 models.

4.4. Discussion

In our experiments, we demonstrate how ICE-Pick works, showing that we can maintain accuracy while reducing fine-tuning time significantly, up to 87.5%. ICE-Pick has tunable hyper-parameters, for which we explored the trade-offs. For larger models or datasets, these trade-offs may vary, but we will explore in future work.

5. Conclusion

Iteratively pruning DNN models can be very time-consuming. In this paper, we propose ICE-Pick, a simple yet effective threshold-guided method to accelerate the fine-tuning procedure in iterative pruning. ICE-Pick can save up to 87.5% of the pruning time while maintaining similar final accuracies. In future work, we will consider ICE-Pick in a broader range of models and datasets, as well as other pruning methods such as unstructured pruning.

References

Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., and Guttag, J. What is the State of Neural Network Pruning? In Dhillon, I., Papailiopoulos, D., and Sze, V. (eds.), *Proceedings of Machine Learning and Systems*, volume 2, pp. 129–146, 2020.

Brock, A., Lim, T., Ritchie, J. M., and Weston, N. Freezeout: Accelerate training by progressively freezing layers. *arXiv preprint arXiv:1706.04983*, 2017.

Deng, L., Li, G., Han, S., Shi, L., and Xie, Y. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4): 485–532, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network, 2015.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861 [cs]*, April 2017.

Joo, D., Yi, E., Baek, S., and Kim, J. Linearly replaceable filters for deep network channel pruning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9): 8021–8029, May 2021.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Lee, J., Tang, R., and Lin, J. What would elsa do? freezing layers during transformer fine-tuning. *arXiv preprint arXiv:1911.03090*, 2019.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Li, Y., Wei, C., and Ma, T. Towards explaining the regularization effect of initial large learning rate in training neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

Liu, S., Tan, C. M. J., and Motani, M. S-cyc: A learning rate schedule for iterative pruning of relu-based networks, 2021a.

Liu, Y., Agarwal, S., and Venkataraman, S. Autofreeze: Automatically freezing model blocks to accelerate fine-tuning. *arXiv preprint arXiv:2102.01386*, 2021b.

Luo, J.-H. and Wu, J. An entropy-based pruning method for cnn compression. *ArXiv*, abs/1706.05791, 2017.

Luo, J.-H., Wu, J., and Lin, W. 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.

Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.