Tiny Updater: Towards Efficient Neural Network-Driven Software Updating

Linfeng Zhang, Kaisheng Ma*
Institute for Interdisciplinary Information Sciences, Tsinghua University
zhanglinfeng1997@outlook.com, kaisheng@mail.tsinghua.edu.cn

Abstract

Significant advancements have been accomplished with deep neural networks in diverse visual tasks, which have substantially elevated their deployment in edge device software. However, during the update of neural network-based software, users are required to download all the parameters of the neural network anew, which harms the user experience. Motivated by previous progress in model compression, we propose a novel training methodology named Tiny Updater to address this issue. Specifically, by adopting the variant of pruning and knowledge distillation methods, Tiny Updater can update the neural network-based software by only downloading a few parameters (10%∼20%) instead of all the parameters in the neural network. Experiments on eleven datasets of three tasks, including image classification, image-to-image translation, and video recognition have demonstrated its effectiveness. Codes have been released in https://github.com/ArchipLab-LinfengZhang/TinyUpdater.

1. Introduction

With the availability of large-scale datasets [15, 16, 51] and high-performance computing platforms, deep neural networks have achieved remarkable achievements in various visual tasks such as image classification [18, 28, 75, 80], segmentation [53, 62], and object detection [50, 67, 68, 76]. Encouraged by their impressive performance, numerous software developers have effectively integrated neural networks into their software products and deployed them on edge devices such as mobile phones and tablet computers.

Typically, the development roadmap for a neural network-based software follows the paradigm illustrate in Figure 1(a). Initially, users install the software by downloading all the neural network parameters from a cloud platform. Subsequently, as the software interacts with its users, an abundance of new training data and requirements can be collected. Then, software developers may retrain the neural network with the gathered data and update their software to enhance its performance. However, during the retraining phase, all the neural network parameters are typically changed compared to their values before updating. Consequently, users are compelled to download all the parameters of the neural network again from the cloud platform, seriously impairing their experience. While recent research has explored text prompts and adapter layers to fine-tune a large-scale pre-trained model at a low training cost [52, 79], there have been no prior efforts to reduce the download cost during model updating, which has a more direct impact on users with edge devices.

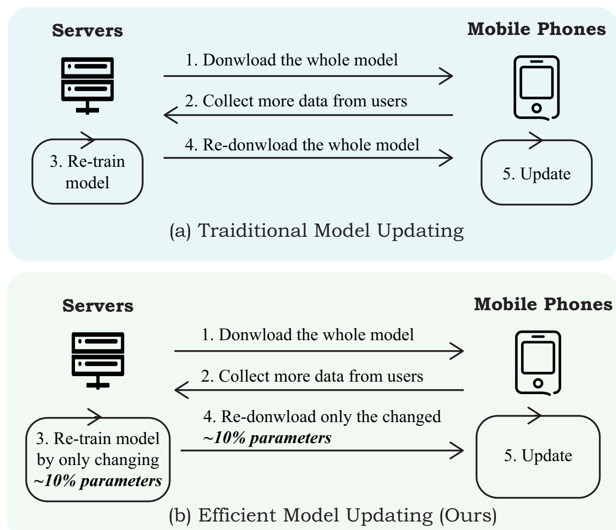


Figure 1. Comparison between (a) the traditional model updating scheme and (b) the proposed efficient model updating. In efficient model updating, only around 10% parameters in the neural network are actually changed and thus users only require to download these changed parameters for updating.

This paper proposes the challenge of *efficient model updating* with the objective of reducing the download overhead of neural network-based software during updating. As

*Corresponding author

depicted in Figure 1(b), during the retraining phase, efficient model updating introduces an additional constraint limiting the change of only a small subset (*e.g.* 10%) of the neural network parameters compared to the pre-updating model. Consequently, users are only required to download a few parameters that have actually changed instead of all the parameters in the neural network. In general, efficient model updating raises two questions: *how to find the optimal parameters that should be changed in the neural network*, and *how to achieve comparable accuracy with the fully-updated model* (*i.e.,* the model which has all the parameters changed during updating).

To tackle this challenge, we propose a novel neural network training framework named Tiny Updater. Motivated by previous works in model compression, Tiny Updater is composed of the variants of two typical model compression techniques - neural network pruning and knowledge distillation. Firstly, to determine which channels or layers in neural networks should be modified during updating, Tiny Updater applies the pruning technique. As shown in Figure 2, it iteratively calculates the L_1-norm distance between the pre-updating model and the post-updating model parameters. The channels with smaller distances are deemed unnecessary for updating and are pruned to their original values before updating. Conversely, the channels with larger distances are considered essential for updating, and thus they are actually changed. Secondly, during the retraining period, we propose to improve the performance of the partially-updated model by distilling the knowledge from a fully-updated teacher model. The partially-updated student model is trained to give predictions that are similar to those of the fully-updated teacher model by optimizing the knowledge distillation loss. This ensures that the partially-updated model achieves comparable performance with the fully-updated model.

Extensive experiments have been conducted on eleven datasets and three different tasks, including, image classification, video classification, and image-to-image translation, using seven different neural networks. Experimental results demonstrate that Tiny Updater can update the model by changing only 10% to 20% of the parameters with minimal performance degradation. In summary, the main contribution of this paper can be outlined as follows.

- To the best of our knowledge, we first propose the challenge of efficient model updating to reduce the communication cost of downloading neural network parameters to the edge devices for software updating.
- To tackle this challenge, we propose Tiny Updater, which prunes the fully-updated model to the pre-updating model and distills the knowledge from the fully-updated model to the partially-updated model.
- Extensive experiments have been conducted on three tasks with eleven datasets, including image classification, image-to-image translation and video recognition. Experimental results demonstrate that Tiny Updater can reduce the training overhead by 80%∼90% with almost no performance degradation.

2. Related Work

2.1. Neural Network Pruning

Neural network pruning is one of the most effective methods for deep neural network compression by deleting the unimportant neurons [42, 47], filters [29, 38], channels [30, 90] and layers [8, 69]. LeCun *et al.* and Hassibi *et al.* first propose to prune neural networks depending on the Hessian matrix [27, 41]. Recently, Han *et al.* propose deep compression, which finds the important connection in neural networks with its absolute value and then iteratively prunes them [24]. Liu *et al.* train a meta-network to generate the weights of neural networks and then apply it to search the best-pruned architecture [58]. Ding *et al.* propose to apply a LSTM to learn the hierarchical characteristics of deep neural networks and then generates the corresponding pruning scheme [17]. Liu *et al.* propose joint multi-dimension pruning to prune the channels, layers and resolutions at the same time [59]. Frankle *et al.* propose the lottery ticket hypothesis and find that a standard pruning technique naturally uncovers subnetworks whose initializations made them capable of training effectively [21]. Besides classification, neural network pruning has also be utilized in image-to-image translation [37, 42], unconditional image generation [56], object detection [22, 66], video retrieval [10] and pre-trained language models [14].

In this paper, Tiny Updater introduces neural network pruning methods to find the parameters which are actually valuable for model updating. Its main difference compared with previous pruning methods is that Tiny Updater prunes the parameters to their value before updating instead of zero, which makes it easier to be solved than traditional pruning. Our experiments also show that Tiny Updater can be utilized on the pruned models.

2.2. Knowledge Distillation

Knowledge distillation, also known as student-teacher learning, has become one of the most popular deep learning techniques in various domains [6, 49, 72, 85, 86]. It firstly trains an over-parameterized teacher model and then distills teacher knowledge to a lightweight student. By training the student to mimic the prediction results of the teacher, the student can inherit the knowledge learned by the teacher. Thus, it can achieve better performance than traditional training. The concept of knowledge distillation is first proposed by Bucilua *et al.* [7] to compress ensemble

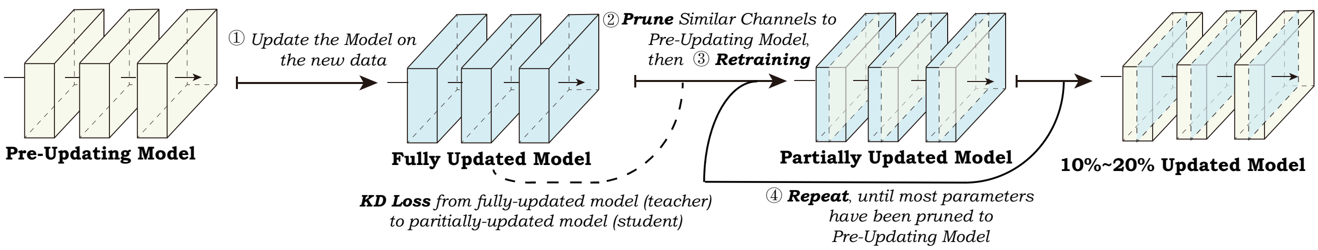


Figure 2. The details of Tiny Updater. (1) Tiny Updater re-trains the model with the new data collected from users (named as fully-updated models) by changing all the parameters. (2) Secondly, Tiny Updater finds the channels in the fully-updated models which have similar value to the pre-updating model, and prunes these channels to their value before updating (the obtained models are named as partially-updated models). (3) Then, Tiny Updater re-trains the unpruned weights. During re-training, the fully-updated model is utilized as the teacher model for the partially-updated model with knowledge distillation (KD) loss to improve its performance. (4) Tiny Updater repeats step-2 and step-3 until most channels have been pruned to their value before pruning. (5) Finally, users only require to download very few parameters which are actually changed for updating.

of neural networks, and then extended by Hinton *et al.* with a temperature hyper-parameter in the softmax function [32]. Following their success, abundant methods have been proposed to distill teacher knowledge in their backbone features [42, 70], spatial attention [20, 84], task-oriented information [87], pixel-wise relation [20,44,49,86], sample-wise relation [63, 65, 78], prediction residual [43] and so on.

Previous knowledge distillation methods usually transfer teacher knowledge in the categorical probability distribution. Then, abundant methods have been introduced to distill the knowledge in the teacher feature [70] and its variants, such as attention [84, 86], relational information [63, 78], self-supervised knowledge [81] and task-oriented information [87]. Besides distilling knowledge from a large teacher to a tiny student, there have also been fruitful knowledge distillation methods that distill knowledge from deeper layers to shallow layers [88, 89], from all the channels to partial channels [83], from ensemble classifiers to single classifier [40], from multiple frames to few frames [5], from RGB images to RGB-D images [23]. In this paper, we introduce knowledge distillation by distilling the knowledge from a teacher model, which has all the parameters changed during updating to a student model, which is only updated by changing a few parameters.

2.3. Incremental Learning

Inspired by the observation that human beings can incrementally learn knowledge about new tasks and categories without forgetting the old knowledge, incremental learning (*a.k.a.* lifelong learning) is proposed to give the neural networks the similar ability. Elastic weight consolidation (EWC) is proposed to maintain the knowledge of old tasks and categories by first estimating the importance of each neuron with Bays estimation or Fisher information matrix and then training them with a consolidation constraint [39, 55, 77]. Then, memory aware synapse [1] is proposed to compute the importance score of neural network parameters in an unsupervised and online manner, which is also similar to the Hebbian learning in biological systems [31]. Besides these consolidation-based methods, knowledge distillation methods have also been utilized in incremental learning to maintain the old knowledge by learning its response to new tasks [13, 48, 92]. Besides image classification, abundant recent research has applied incremental learning to the other visual tasks, such as object detection [11, 26], action recognition [60] and image-to-image translation [74].

Both the proposed efficient model updating and incremental learning expect the neural network to maintain its knowledge of old parameters. Their main difference is that: **(a)** Incremental learning does not limit the change in old parameters strictly, while efficient model updating has a direct constraint on the number of changed parameters to reduce the download overhead for edge devices. **(b)** Incremental learning usually assumes that the data for old knowledge is not available while efficient model updating can still access data, which is more practical to industrial applications. **(c)** The target of incremental learning is to build human-like artificial intelligence which can incrementally learn various tasks, while the target of Efficient Model Updating is to reduce the number of parameters updated when new data for the same task is collected.

2.4. Efficient Pretrained Model Finetuning

Large-scale pretrained models recently have shown powerful representation ability in both natural language processing and computer vision. However, since these pretrained models usually have billions of parameters, directly finetuning them on downstream tasks usually suffers from massive training overhead, making them impractical in real-world applications. Recently, prompts methods have been proposed to address this problem by designing specific input templates for the specific downstream tasks [4,25,45,52, 54,79,91]. Moreover, adapter methods have also been introduced to finetune several additional trainable layers instead of the whole pre-trained model [3, 33]. These methods are firstly proposed for pre-trained language models and then extended to language-vision multi-modal models and vision models [12, 36, 82]. Both these efficient finetuning methods and the proposed efficient model updating aim to freeze the parameters of neural networks. Their main difference is that: **(a)** Efficient finetuning is usually applied to the models which are firstly pre-trained on large-scale datasets to learn task-unbiased knowledge and then finetuned for specific downstream tasks. Instead, the pre-updating models in Efficient Model Updating are trained with very few training samples (*e.g.* only 20% in our experiments) and then updated with the new data collected from users. **(b)** The target of efficient finetuning is to reduce the *training overhead* of pre-trained models and improve their performance in downstream tasks. In contrast, the target of Efficient Model Updating is to reduce the *download overhead* during software updating. **(c)** Besides, adapter-based methods can be considered as a special case in Tiny Updater where all the parameters of backbone layers are frozen and all the adapter layers are updated. Our experiments also demonstrate that Tiny Updater achieves clearly better performance than directly applying efficient model updating methods. Please refer to Appendix A for a more detailed comparison among the proposed efficient model updating, incremental learning, and efficient pretrained model finetuning.

3. Methodology

3.1. Efficient Model Updating

Given a training dataset $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$, the software developers firstly train a deep neural network $\mathcal{F}$ with parameter Θ for their software. After deploying $\mathcal{F}$ on the edge devices, abundant new training samples $\{(x_{n+1}, y_{n+1}), ..., (x_{n+m}, y_{n+m})\}$ can be collected from users and the dataset can be extended as $\mathcal{D}^+ = \mathcal{D} \cup \{(x_{n+1}, y_{n+1}), ..., (x_{n+m}, y_{n+m})\}$. Then, software developers can re-train $\mathcal{F}$ on $\mathcal{D}^+$ to obtain better performance, whose parameters can be denoted as Φ. Usually, during this re-training step, all the values in θ can be totally different from that in Φ. Thus, the users have to download all the parameters of Φ to update the software, which harms the user experience and limits the frequently updating of neural network-driven software. In such kind of paradigm, we name the model before updating as the pre-updating model, the model after updating as the post-updating model, the model with all the parameters updated as the fully-updated model, and the model with partial parameters updated as the partially updated model. The target of efficient model updating is to obtain a partially updated model which has most of the parameters unchanged while achieving similar performance to the fully-updated model.

Algorithm 1 The proposed Tiny Updater.

Input: Dataset $\mathcal{D}^+ = \{(x_1, y_1), ..., (x_{n+m}, y_{n+m})\}$, the pre-updating model $\mathcal{F}$ with parameter Θ, an expected ratio of updated parameters τ.
Output: The partially-updated model $\mathcal{F}$ with parameter Φ.
`// Update the model with` $\mathcal{D}^+$.
Initialize the partially-updated model $\mathcal{F}$ with parameter Φ.
while $\mathcal{F}_\Phi$ is not converged **do:**
  Sample a batch of data $\mathcal{X}$ from $\mathcal{D}^+$
  Compute $\hat{y} := \mathcal{F}_\Phi(\mathcal{X})$.
  Compute the task loss between y and $\hat{y}$.
  Back propagate gradients and update Φ.
`// Pruning, and re-training with KD loss.`
Initialize the parameters of the teacher $\mathcal{T} := \Phi$, an index set of pruned weights as $\mathcal{I} = \{\}$.
while $\frac{Card(\mathcal{I})}{Card(\Theta)} < 1 - \tau$ **do:**
  Compute the L_1-norm distance between Φ and θ.
  Append the indices of channels which have relatively smaller distance to $\mathcal{I}$.
  $\Phi[\mathcal{I}] := \Theta[\mathcal{I}]$. `// Prune` Φ `to` Θ.
  while $\mathcal{F}_\Phi$ is not converged **do:**
    Sample a batch of data $\mathcal{X}$ from $\mathcal{D}^+$
    Compute $\hat{y} := \mathcal{F}_\Phi(\mathcal{X})$ and $y_t := \mathcal{F}_\mathcal{T}(\mathcal{X})$.
    Compute the task loss between y and $\hat{y}$, and KD loss between y and y_t.
    Back propagate gradients and update the parameters of Φ which are not in $\mathcal{I}$.
Return The partially-updated model $\mathcal{F}$ with parameter Φ.

3.2. Tiny Updater

Fruitful previous works in model compression have successfully proven that even a very tiny neural network can have powerful representation ability, which motivates us to propose to learn the knowledge in the collected training data $\mathcal{D}^+$ with only a few parameters instead of all the parameters. The optimization objective of Tiny Updater can be formulated as

$$\arg\min_{\Phi} \frac{1}{n+m} \sum_{i=1}^{n+m} \mathcal{L}_{\text{task}}(x_i, y_i) \quad \text{subject to } \frac{|\Theta - \Phi|_0}{\text{Card}(\Theta)} < \tau \tag{1}$$

where $\mathcal{L}_{\text{task}}$ indicates the original task-specific loss function, such as cross-entropy loss for image classification. $|\cdot|_0$ indicates the L_0-norm, which measures the number of non-zero elements in a tensor. $\text{Card}(\cdot)$ denotes the cardinality, which describes the number of parameters in a tensor. τ is a ratio threshold that determines how many parameters should be changed during updating. It is observed that when θ in Equation (1) becomes zero, the optimization objective is similar to another deep learning technique - network prun-

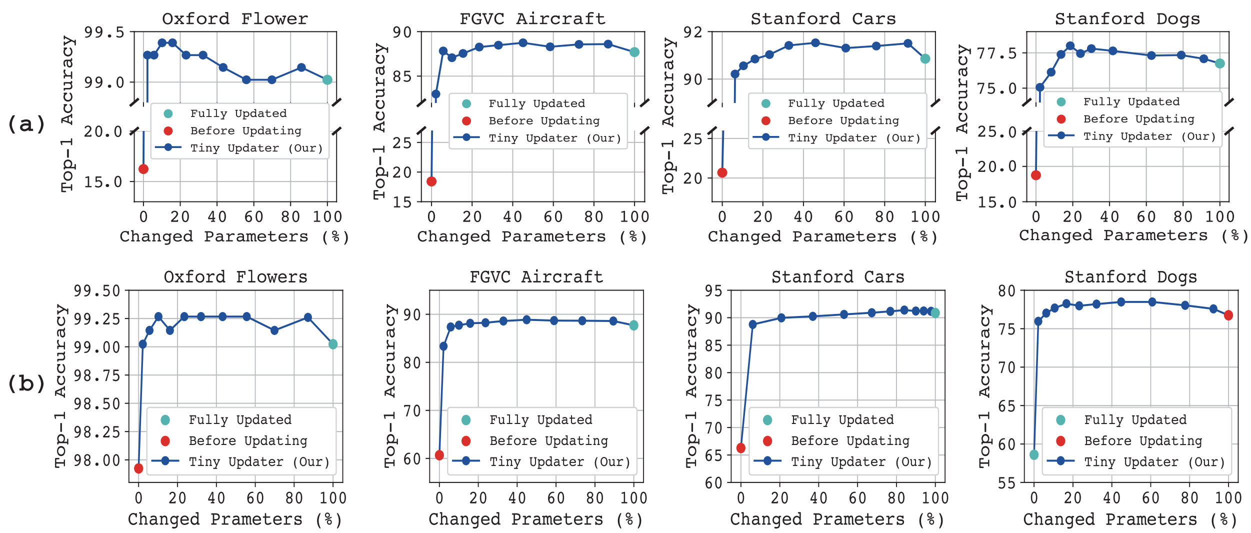


Figure 3. Experiments on four fine-grained image classification datasets with ResNet50. The pre-updating and the post-updating models are trained with 25% and 100% training data, respectively. In subfigure (a), the newly collected images come from all categories uniformly. In subfigure (b), the newly collected images come from only the categories that are not available before updating.

ing. Thus, we propose to solve this problem by introducing iterative pruning methods. Besides, knowledge distillation is also introduced to improve the model performance with only a few parameters changed during updating with the teacher from a model with all the parameters changed. Please note that Tiny Updater is a framework that applies pruning and knowledge distillation methods to tackle the challenge of efficient model updating and it does not propose any new pruning and knowledge distillation method.

The details of Tiny Updater are shown in Algorithm 1. With the extended dataset $\mathcal{D}^{+}$, it first trains a fully-updated model $\mathcal{F}_{\Phi}$ without any constraints and copies its parameters to another neural network $\mathcal{F}_{\mathcal{T}}$, which is considered as the teacher network. Then, channel pruning is utilized to find the parameters in Φ (*e.g.* channels in convolutional layers) whose value is not significantly changed compared with the pre-updating model Θ. Then, these parameters are reset to their value in Θ and are not further trained in the later re-training. During the re-training period, the partially-updated model is trained with not only the origin task loss but also the knowledge distillation loss from the teacher of the fully-updated model $\mathcal{F}_{\mathcal{T}}$. Please note that the proposed Tiny Updater is a framework that employs pruning and knowledge distillation to tackle the challenge of efficient model updating and it does not depend on any specific pruning and knowledge distillation methods.

4. Experiment

4.1. Image Classification

Experiment Setting Our method has mainly been evaluated on seven image classification datasets, including CIFAR10, CIFAR100, and ImageNet for general image classification, Stanford Car, Oxford Flower, Stanford Dog, FGVC Aircraft for fine-grained image classification. Please refer to Appendix B for the details of these datasets. Note that on the fine-grained image classification datasets, models are initialized with backbone weights pre-trained on ImageNet. In ImageNet experiments, we adopt the basic training policy from PyTorch [64] with task-oriented feature distillation [87]. On the other datasets, each model is trained by 200 epochs with the naive logit and feature knowledge distillation loss [32, 70].

Table 1. The performance of Tiny Updater on ImageNet. Models are firstly trained with 25% data and then updated with all the data. "Updated Ratio" indicates the ratio of updated parameters.

Model	Updated Ratio (%)	Top-1 Accuracy	Top-5 Accuracy
ResNet50	00.00	68.86 $_{-6.44}$	88.74 $_{-3.46}$
	21.54	74.76 $_{-0.54}$	91.70 $_{-0.50}$
	42.42	75.12 $_{-0.18}$	91.93 $_{-0.27}$
	65.69	76.25 $_{+0.95}$	93.14 $_{-0.94}$
	100.0	75.30	92.20
MobileNetv2	00.00	62.23$_{-3.08}$	84.25$_{-2.29}$
	24.31	64.54$_{-0.77}$	85.53$_{-1.01}$
	31.66	65.23$_{-0.08}$	86.42$_{-0.12}$
	66.55	66.30$_{+0.99}$	87.19$_{+0.65}$
	87.60	66.86$_{+1.55}$	87.50$_{+0.96}$
	100.0	65.31	86.54
Swin Transformer	00.00	72.04$_{-9.26}$	89.75$_{-5.85}$
	18.56	79.91$_{-1.39}$	94.33$_{-1.27}$
	36.31	80.87$_{-0.43}$	94.53$_{-1.07}$
	80.26	81.73$_{+0.43}$	95.74$_{+0.14}$
	100.0	81.30	95.60

Main Results The performance of Tiny Updater on ImageNet, CIFAR10&100, and four fine-grained image classifi-

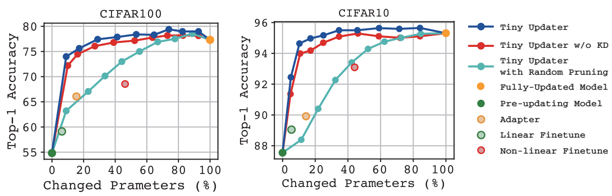


Figure 4. Experiments on CIFAR with ResNet50. The pre-updating models and fully-updated models are trained with 25% and 100% training data, respectively. Tiny Updater with random pruning indicates pruning the randomly selected channels.

cation are shown in Table 1, Figure 4, Figure 3, respectively. It is observed that: (i) Tiny Updater achieves consistent effectiveness on all seven datasets. By changing around 20% parameters during updating, the neural networks trained with Tiny Updater achieve a similar performance with fully-updated models on all these datasets. (ii) On ImageNet, Tiny Updater achieves significant performance on the regular convolutional network such as ResNet [28], the efficient convolutional networks such as MobileNetv2 [71], and Swin Transformer [57]. (iii) With the proposed Tiny Updater, when a large ratio of parameters is changed, it even leads to higher performance than the fully-updated model. For instance, on Oxford Flowers datasets in Figure 3, the 84% updated model trained with Tiny Updater has 0.25% higher accuracy than the fully-updated model. We suggest this accuracy benefit is caused by the knowledge distillation loss in Tiny Updater. As pointed out by some previous research [2, 61], even if the student and the teacher have similar performance, knowledge distillation can still lead to consistent accuracy benefits.

Ablation Study Tiny Updater is mainly composed of two modules - pruning and knowledge distillation. Ablation studies on CIFAR10 and CIFAR100 are shown in Figure 4. It is observed that: (i) Compared with Tiny Updater with random pruning, Tiny Updater with L_1-norm based pruning leads to consistently higher accuracy (around 6%) especially when only a low ratio (0$\sim$50%) of parameters are changed, indicating L_1-norm is an effective metric to find which channels are actually important for model updating. (ii) Significant accuracy boosts can be observed by applying knowledge distillation during re-training. For instance, on CIFAR100, when around 10% and 20% parameters are changed, 1.79% and 1.15% accuracy boost can be observed with knowledge distillation, respectively. These observations indicate that the pruning and knowledge distillation in Tiny Updater have their individual effectiveness.

Categorical Incremental Updating Figure 3(b) shows the effectiveness of Tiny Updater in the categorical incremental settings, where the pre-updating models are trained with 25% training data belonging to 25% categories. During the updating period, the model is further trained with data of all the categories. It is observed that Tiny Updater can still achieve significant performance by only changing around 10% parameters, indicating that Tiny Updater is also effective on the categorical incremental updating setting.

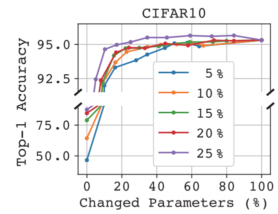


Figure 5. Experiments on CIFAR10 with the pre-updating models trained with different ratios of data.

Influence from Pre-updating Model In previous experiments, the pre-updating models are trained with 25% data. With less training data, the pre-updating models tend to have less representative ability, and thus the performance of Tiny Updater tends to be reduced. In this subsection, we study how the performance of the pre-updating model influence Tiny Updater on CIFAR10 in Figure 3(a). It is observed that when most of the parameters (>50%) are changed during updating, the accuracy gap between different pre-updating models is not significant (<0.2%). In contrast, when only a few parameters (10%$\sim$20%) are changed, the ratio of data for training pre-updating models has a more significant influence. For example, compared with the pre-updating model trained with 25% data, the pre-updating model trained with only 5% data leads to around 2.5% accuracy loss when 20% parameters are changed during updating, indicating that the performance of the pre-updating model has a direct influence to Tiny Updater. Specifically, when the pre-updating model is not trained with any data, it can be considered as a model with all parameters initialized with zero matrices, and thus Tiny Updater in this case degenerates to the common neural network pruning problem.

Multi-step Model Updating Previous experiments mainly show the result of one-step updating from using 25% to 100% training data. In this subsection, we show the performance of Tiny Updater in a multi-step updating, where the model is first trained with 5% data and then updated with 10%, 15%, 20% 25%, and 100% training data, successively. As shown in Table 4, there is almost no accuracy loss when around 20% parameters are changed for each update. When there are around 10% parameters changed for each update, accuracy loss becomes significant with the increment on the update iterations (from -0.44% to -2.33%). This observation indicates that the accuracy loss tends to be accumulated in multi-step updating when an extremely low ratio of parameters is changed for updating.

Table 2. Experiments of multi-step updating on CIFAR10. Models are firstly trained with 5% training data and then updated by five times with 10%, 15%, 20%, 25%, and 100% training data by changing all, 10% and 20% parameters.

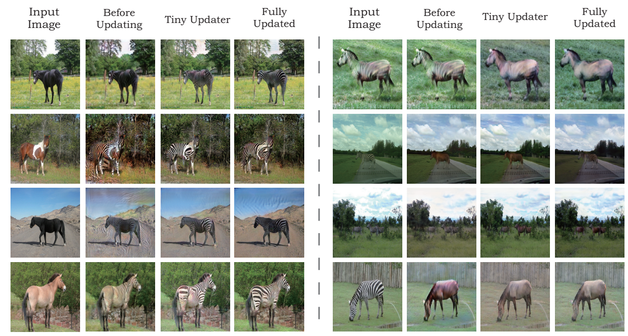
Params. Ratio \ Data Ratio	5%	10%	15%	20%	25%	100%
Fully-Updated	46.5	64.2	78.8	84.6	87.8	95.3
10% Params. Updated	46.5	63.8	78.3	83.2	85.9	92.9
20% Params. Updated	46.5	64.3	79.0	84.2	87.5	94.8

Tiny Updater on Pruned Models In this subsection, we study whether the proposed Tiny Updater can be utilized on a pruned model. Concretely, we first train a ResNet18 model with 25% training data on CIFAR10, which achieves 87.43% top-1 accuracy. Then, we apply the L_1-norm pruning technique to prune 53% neurons at the expense of 0.37% accuracy loss (87.43%→87.06%). Thirdly, the proposed Tiny Updater is applied to update this model with 100% training data, which achieves 94.22% at the expense of changing 24.3% parameters (87.06%→94.22%). In contrast, a pruned fully-updated model with the same pruning ratio achieves 94.87% accuracy, which is only 0.65% higher than the pruned model with Tiny Updater (87.06%→94.87%). These observations indicate that the proposed Tiny Updater can be utilized on pruned models.

Comparison with Finetuning and Adapter Finetuning and adapter-based methods are two well-known efficient training methods and the parameters changed in these two methods are also much fewer than global finetuning. Figure 4 gives the comparison between Tiny Updater and these two methods on CIFAR10, CIFAR100 with ResNet18. It is observed that when the same number of parameters are changed, Tiny Updater outperforms these two methods by a clear margin, indicating that Tiny Updater is more effective than directly applying previous methods.

4.2. Image-to-Image Translation with GANs

Experiment Setting We mainly evaluate Tiny Updater on image-to-image translation with CycleGAN for Horse→Zebra translation, and Pix2Pix for Edge→Shoe translation. CycleGAN is a typical unpaired image-to-image translation model, which has a ResNet-based generator trained with GAN loss and cycle consistency [94]. Pix2Pix is a typical paired image-to-image translation model, which has a U-Net architecture generator trained with conditional GAN loss and the L_1-norm loss [35]. Horse→Zebra is an unpaired dataset that translates natural images of horses to zebras and vice versa. It consists of 1,187 horse images and 1,474 zebra [93]. Edge→Shoe is a paired data that translates the edges of shoes to their corresponding natural images [34]. *Fréchet Inception Distance (FID)*, which measures the distance between the feature distribution of the real and the generated images, is utilized as the metric for both datasets. A lower FID indicates the synthetic images have better quality.



(a) Horse to Zebra (b) Zebra to Horse

Figure 6. Qualitative analysis on Horse→Zebra and Zebra→Horse with CycleGAN. The partially-updated model trained with Tiny Updater has around 20% parameters changed. The pre-updating models are trained with 25% data.

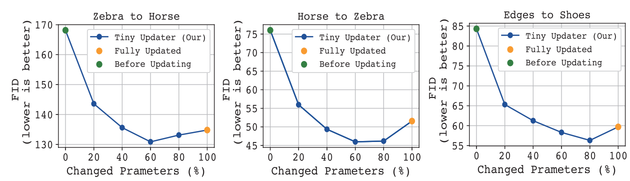


Figure 7. Experiments of image-to-image translation on Horse→Zebra and Zebra→Horse with CycleGAN, and Edge→Shoe with Pix2Pix. In all these experiments, the pre-updating models are trained with 25% training data.

Main Results Experiments of Tiny Updater on image-to-image translation are shown in Figure 7. It is observed that on the three image-to-image translation tasks, Tiny Updater is able to improve model performance by 5∼10 FID when around 60% parameters are changed during updating. Besides, Tiny Updater can update the models by changing 40% parameters with no performance loss or by changing only 20% parameters with a slight FID increment. Qualitative analysis in Figure 6 shows that the pre-updating models can not transform the whole body of horses into the zebras, or remove all the stripes of zebras. In contrast, with Tiny Updater, the partially-updated model with only 20% parameters changed can address this problem and achieve comparable performance with the fully-updated models, indicating that Tiny Updater enables the neural network to be successfully updated by changing only a few parameters.

4.3. Video Recognition

Experiment Setting We evaluate Tiny Updater in video recognition datasets including UCF-101 [73] and Diving-48 [46] with video recognition models including SlowOnly [19] and Inception3D [9]. *UCF-101* is an ac-

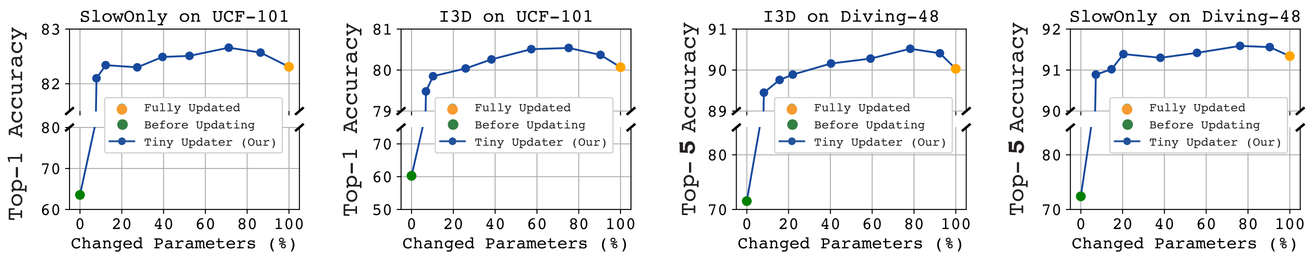


Figure 8. Experimental results of Tiny Updater of video recognition on UCF-101 and Diving-48 with SlowOnly and Inception3D (I3D). The pre-updating models are trained with 25% training data.

tion recognition dataset with 101 action classes over 13,000 video clips [73]. *Diving-48* is a fine-grained video dataset of competitive diving. It has around 18,000 video clips belonging to 48 dive sequences [9]. Both top-1 and top-5 accuracy are reported.

Main Results Figure 8 shows the performance of Tiny Updater on video recognition. It is observed that on both SlowOnly and Inception3D (I3D), UCF-101 and Diving-48, the model with only 20% parameters changed trained by Tiny Updater achieves comparable and even higher performance than the fully-updated models. Moreover, the model with only 10% parameters changed during updating leads to only around 0.5% accuracy loss.

5. Discussion

5.1. Choice of Pruning Granularity

Existing network pruning methods can be mainly divided into regular pruning (*e.g.* channel-wise pruning) and irregular pruning (*e.g.* element-wise pruning). Usually, irregular pruning can achieve a higher compression rate, while regular pruning is more friendly to hardware. In this paper, we prefer to use regular pruning in Tiny Updater and suggest that using irregular pruning may cause the problem of index overhead and memory access overhead.

Index Overhead When Tiny Updater is utilized for updating neural network-based software, users should download not only the parameters which are changed during updating but also an index file that records the corresponding position of each changed parameter in the neural network. Given a neural network with N parameters, when irregular pruning is utilized, for each updated parameter, $\log_2 N$ bits storage space is required to record which layer, filter, and position in the kernel this parameter should be. Thus, an ignorable additional downloading overhead is caused. In contrast, the index overhead for channel-wise pruning is $\log_2 \frac{N}{C \times K^2}$ bits, which is small enough to be almost ignorable. Here C and K indicate the channel number and the kernel size, respectively.

Memory Access Overhead In Tiny Updater, irregular pruning can result in the updated parameters being distributed across various neural layers, channels, and filters, leading to their storage in different data blocks of the file system. Consequently, when updating the model with the downloaded parameters, almost all data blocks of the edge device need to undergo a writing operation, resulting in a significant memory access overhead. In contrast, regular pruning ensures that the updated weights are channel-wise and, therefore, can be stored in the same data blocks. Consequently, only these data blocks for the updated parameters require memory access, potentially reducing the updating overhead on edge devices.

6. Conclusion

This paper introduces the challenge of efficient model updating to reduce the download overhead during neural network-based software updating. To this end, we propose Tiny Updater, which aims to change only a small subset of the parameters during model retraining, thereby reducing the number of parameters that need to be downloaded by users. Tiny Updater comprises an iterative pruning technique that prunes the parameters of the fully-updated models to the pre-updating models and a knowledge distillation technique that treats the fully-updated models as teachers and the partially-updated models as students.

Extensive experiments were conducted on eleven datasets covering general image classification, fine-grained image classification, video recognition, paired image-to-image translation, and unpaired image-to-image translation. The results demonstrate that Tiny Updater can update neural network-based software by changing only 10% to 20% of the parameters with almost no loss in accuracy. In addition, the experimental results show the effectiveness of Tiny Updater in various scenarios, including categorical updating, multi-step updating, and pruning. We believe that this paper will encourage further research to address the challenge of efficient model updating on edge devices.

Table 3. Comparison among incremental learning, efficient finetuning, and the proposed efficient model updating.

Problem	Target	Pre / Post Tasks	Usage of Training Data	Constraints on Parameters
Incremental Learning	The human-like ability of incrementally learning new tasks without forgetting old tasks.	Different.	A small ratio of data is used for training pre-updating models. More data is used for updating. Data for pre-training data can not be accessed during updating (learning new tasks).	No direct constraints. Reducing the number of changed parameters sometimes lead to positive influence.
Efficient Finetuning	Finetuning pre-trained models for downstream tasks with low training overhead, and achieving better downstream performance.	Different.	Pre-updating models are pretrained on large-scale datasets, and then finetuned on much less data for downstream tasks. Usually do not access data for pre-training during finetuning.	No direct constraints. Do not finetune the whole model usually leads to no changes in pre-trained weights. Additional parameters in adapter layers and prompts are required.
Efficient Model Updating	Update models with new data by changing only a few parameters to reduce communicat -ion overhead.	Same.	A small ratio of data is used for pre-training models, and these data can still be accessed during updating.	Having direct constraints, the changed parameters should be as few as possible.

7. Acknowledgements

This research was partially supported by National Key R&D Program of China (2022YFB2804103), Key Research and Development Program of Shaanxi (2021ZDLGY01-05), Tsinghua University Dushi Program, National Natural Science Foundation of China (20211710187), and Tsinghua University Talent Program.

A. Detailed Comparison

A detailed comparison among incremental learning, efficient finetuning and the proposed efficient model updating is shown in Table 3.

B. Detailed Experimental Setting

CIFAR10/100 are two datasets for low resolution general image classification with 50,000 images and 1,000 images in the training and validating set, respectively. ImageNet is a large-scale dataset for general image classification with images belonging to 1,000 categories. Stanford Car is a dataset for fine-grained image classification with car images belonging to 197 classes. Oxford Flower is a dataset for fine-grained image classification which consists of 102 different categories of flowers common to the UK. Stanford Dog is a dataset for fine-grained image classification which contains 20,580 dog images of 120 breeds of dogs from around the world. FGVC Aircraft is a dataset for fine-grained image classificaiton which contains 10,000 images of aircrafts spanning 100 aircraft models.

C. Experiments on More Datasets and Tasks

Experimental results on more kinds of tasks and datasets are shown in Table 4 and Table 5. It is observed that our method leads to consistent good performance on cross-domain image classification, object detection and image super-resolution. Besides, our method outperforms previous efficient finetuning methods on ImageNet.

Table 4. Comparison between our method with previous efficient finetuning methods on ImageNet with Swin Transformer. Models are firstly trained with 25% data and then updated with all the data.

Updating Method	Ratio of Updated Params.(%)	Top-1 Acc (%)
Pre-updating Model	0.0%	72.04
Global Finetuning	100.0%	81.30
Linear Finetuning	1.16	72.23
Non-linear Finetuning^2	34.6	75.53
Non-linear Finetuning^3	66.3	78.80
Adapter^1	3.5	72.40
Adapter^2	17.2	74.39
Visual Prompt Tuning	$\leq$ 1%	72.32
Ours^1	36.31	80.87
Ours^2	18.56	79.91

Table 5. Experimental results on more tasks.

Task	Dataset	Model	Updated Params (%)	Acc. / mAP / PSNR
Cross Domain Image Classification	Office-31	ResNet50	0.00	92.3
			18.7	98.3
			10.9	98.0
			100.0	98.5
Object Detection	MS COCO	Faster RCNN	0.00	30.5
			27.3	36.2
			14.6	35.6
			100.0	36.4
Single Image Super-Resolution	DIV2K	ESRGAN	0.00	33.90
			22.4	34.60
			13.0	34.56
			100.0	34.61

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018. 3

[2] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *CoRR*, abs/2012.09816, 2020. 6

[3] Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. Simple, scalable adaptation for neural machine translation. *arXiv preprint arXiv:1909.08478*, 2019. 4

[4] Eyal Ben-David, Nadav Oved, and Roi Reichart. Pada: A prompt-based autoregressive approach for adaptation to unseen domains. *arXiv preprint arXiv:2102.12206*, 2021. 3

[5] Shweta Bhardwaj, Mukundhan Srinivasan, and Mitesh M Khapra. Efficient video classification using fewer frames. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 354–363, 2019. 3

[6] Cunling Bian, Wei Feng, Liang Wan, and Song Wang. Structural knowledge distillation for efficient skeleton-based action recognition. *IEEE Transactions on Image Processing*, 30:2963–2976, 2021. 2

[7] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM, 2006. 2

[8] Efe Camci, Manas Gupta, Min Wu, and Jie Lin. Qlp: Deep q-learning for pruning deep neural networks. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2022. 2

[9] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 7, 8

[10] Hyun Sung Chang, Sanghoon Sull, and Sang Uk Lee. Efficient video indexing scheme for content-based retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1269–1279, 1999. 2

[11] Li Chen, Chunyan Yu, and Lvcai Chen. A new knowledge distillation for incremental object detection. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2019. 3

[12] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022. 4

[13] Ali Cheraghian, Shafin Rahman, Pengfei Fang, Soumava Kumar Roy, Lars Petersson, and Mehrtash Harandi. Semantic-aware knowledge distillation for few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2534–2543, 2021. 3

[14] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 2

[15] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 248–255, 2009. 1

[17] Guiguang Ding, Shuo Zhang, Zizhou Jia, Jing Zhong, and Jungong Han. Where to prune: Using lstm to guide data-dependent soft pruning. *IEEE Transactions on Image Processing*, 30:293–304, 2021. 2

[18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[19] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019. 7

[20] Yingchao Feng, Xian Sun, Wenhui Diao, Jihao Li, and Xin Gao. Double similarity distillation for semantic image segmentation. *IEEE Transactions on Image Processing*, 30:5363–5376, 2021. 3

[21] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018. 2

[22] Sharath Girish, Shishira R Maiya, Kamal Gupta, Hao Chen, Larry S Davis, and Abhinav Shrivastava. The lottery ticket hypothesis for object recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 762–771, 2021. 2

[23] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. In *CVPR*, pages 2827–2836, 2016. 3

[24] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*, 2016. 2

[25] Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. Ptr: Prompt tuning with rules for text classification. *arXiv preprint arXiv:2105.11259*, 2021. 3

[26] Yu Hao, Yanwei Fu, Yu-Gang Jiang, and Qi Tian. An end-to-end architecture for class-incremental object detection with knowledge distillation. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2019. 3

[27] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pages 164–171, 1993. 2

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceed-*

ings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016. 1, 6

[29] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018. 2

[30] 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*, pages 1389–1397, 2017. 2

[31] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005. 3

[32] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPS*, 2014. 3, 5

[33] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019. 4

[34] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Pix2pix datasets, http://efrosgans.eecs.berkeley.edu/pix2pix/datasets/. 7

[35] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 7

[36] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2022. 4

[37] Qing Jin, Jian Ren, Oliver J Woodford, Jiazhuo Wang, Geng Yuan, Yanzhi Wang, and Sergey Tulyakov. Teachers do more than teach: Compressing image-to-image models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13600–13611, 2021. 2

[38] Hyeong-Ju Kang. Accelerator-aware pruning for convolutional neural networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(7):2093–2103, 2020. 2

[39] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 3

[40] Xu Lan, Xiatian Zhu, and Shaogang Gong. Knowledge distillation by on-the-fly native ensemble. *arXiv preprint arXiv:1806.04606*, 2018. 3

[41] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990. 2

[42] Muyang Li, Ji Lin, Yaoyao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. Gan compression: Efficient architectures for interactive conditional gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5284–5294, 2020. 2, 3

[43] Xuewei Li, Songyuan Li, Bourahla Omar, Fei Wu, and Xi Li. Reskd: Residual-guided knowledge distillation. *IEEE Transactions on Image Processing*, 30:4735–4746, 2021. 3

[44] Xiaojie Li, Jianlong Wu, Hongyu Fang, Yue Liao, Fei Wang, and Chen Qian. Local correlation consistency for knowledge distillation. In *European Conference on Computer Vision*, pages 18–33. Springer, 2020. 3

[45] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021. 3

[46] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 513–528, 2018. 7

[47] Yun Li, Zechun Liu, Weiqun Wu, Haotian Yao, Xiangyu Zhang, Chi Zhang, and Baoqun Yin. Weight-dependent gates for network pruning. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2022. 2

[48] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 3

[49] Zeqi Li, Ruowei Jiang, and Parham Aarabi. Semantic relation preserving knowledge distillation for image-to-image translation. In *European Conference on Computer Vision*, pages 648–663. Springer, 2020. 2, 3

[50] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 1

[51] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1

[52] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021. 1, 3

[53] Weide Liu, Guosheng Lin, Tianyi Zhang, and Zichuan Liu. Guided co-segmentation network for fast video object segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(4):1607–1617, 2021. 1

[54] Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021. 3

[55] Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2262–2268. IEEE, 2018. 3

[56] Yuchen Liu, Zhixin Shu, Yijun Li, Zhe Lin, Federico Perazzi, and Sun-Yuan Kung. Content-aware gan compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12156–12166, 2021. 2

[57] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In

Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 10012–10022, 2021. 6

[58] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 2

[59] Zechun Liu, Xiangyu Zhang, Zhiqiang Shen, Yichen Wei, Kwang-Ting Cheng, and Jian Sun. Joint multi-dimension pruning via numerical gradient update. *IEEE Transactions on Image Processing*, 30:8034–8045, 2021. 2

[60] Rashid Minhas, Abdul Adeel Mohammed, and Q. M. Jonathan Wu. Incremental learning in human action recognition based on snippets. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(11):1529–1541, 2012. 3

[61] Hossein Mobahi, Mehrdad Farajtabar, and Peter L Bartlett. Self-distillation amplifies regularization in hilbert space. *arXiv preprint arXiv:2002.05715*, 2020. 6

[62] Diego Ortego, Juan C. Sanmiguel, and José M. Martínez. Hierarchical improvement of foreground segmentation masks in background subtraction. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(6):1645–1658, 2019. 1

[63] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, 2019. 3

[64] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 5

[65] Baoyun Peng, Xiao Jin, Jiaheng Liu, Dongsheng Li, Yichao Wu, Yu Liu, Shunfeng Zhou, and Zhaoning Zhang. Correlation congruence for knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5007–5016, 2019. 3

[66] Zheng Qin, Zeming Li, Zhaoning Zhang, Yiping Bao, Gang Yu, Yuxing Peng, and Jian Sun. Thundernet: Towards real-time generic object detection on mobile devices. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 2

[67] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016. 1

[68] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[69] Youngmin Ro and Jin Young Choi. Layer-wise pruning and auto-tuning of layer-wise learning rates in fine-tuning of deep networks. *arXiv preprint arXiv:2002.06048*, 2020. 2

[70] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *International Conference on Learning Representations*, 2015. 3, 5

[71] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 6

[72] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019. 2

[73] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 7, 8

[74] Daniel Stanley Tan, Yong-Xiang Lin, and Kai-Lung Hua. Incremental learning of multi-domain image-to-image translations. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(4):1526–1539, 2021. 3

[75] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 1

[76] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020. 1

[77] Brian Thompson, Jeremy Gwinnup, Huda Khayrallah, Kevin Duh, and Philipp Koehn. Overcoming catastrophic forgetting during domain adaptation of neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2062–2068, 2019. 3

[78] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1365–1374, 2019. 3

[79] Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Guihong Cao, Daxin Jiang, Ming Zhou, et al. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*, 2020. 1, 3

[80] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5987–5995, 2017. 1

[81] Guodong Xu, Ziwei Liu, Xiaoxiao Li, and Chen Change Loy. Knowledge distillation meets self-supervision. In *European Conference on Computer Vision*, pages 588–604. Springer, 2020. 3

[82] Yuan Yao, Ao Zhang, Zhengyan Zhang, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. Cpt: Colorful prompt tuning for pre-trained vision-language models. *arXiv preprint arXiv:2109.11797*, 2021. 4

[83] Jiahui Yu and Thomas S. Huang. Universally slimmable networks and improved training techniques. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 3

[84] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*, 2017. 3

[85] Linfeng Zhang, Xin Chen, Xiaobing Tu, Pengfei Wan, Ning Xu, and Kaisheng Ma. Wavelet knowledge distillation: Towards efficient image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12464–12474, June 2022. 2

[86] Linfeng Zhang and Ma Kaisheng. Improve object detection with feature-based knowledge distillation: Towards accurate and efficient detectors. In *International Conference on Learning Representations*, 2021. 2, 3

[87] Linfeng Zhang, Yukang Shi, Zuoqiang Shi, Kaisheng Ma, and Chenglong Bao. Task-oriented feature distillation. In *NeurIPS*, 2020. 3, 5

[88] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *arXiv preprint:1905.08094*, 2019. 3

[89] Linfeng Zhang, Zhanhong Tan, Jiebo Song, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Scan: A scalable neural networks framework towards compact and efficient models. *ArXiv*, abs/1906.03951, 2019. 3

[90] Mingxin Zhao, Junbo Peng, Shuangming Yu, Liyuan Liu, and Nanjian Wu. Exploring structural sparsity in cnn via selective penalty. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(3):1658–1666, 2022. 2

[91] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. *arXiv preprint arXiv:2203.05557*, 2022. 3

[92] Peng Zhou, Long Mai, Jianming Zhang, Ning Xu, Zuxuan Wu, and Larry S Davis. M2kd: Multi-model and multi-level knowledge distillation for incremental learning. *arXiv preprint arXiv:1904.01769*, 2019. 3

[93] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Pix2pix datasets, `http://efrosgans.eecs.berkeley.edu/cyclegan/datasets`. 7

[94] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 7