# NetDistiller: Empowering Tiny Deep Learning via In-Situ Distillation

Shunyao Zhang, *Rice University, Houston, TX, 77005, USA*
Yonggan Fu, *Georgia Institute of Technology, Atlanta, GA, 30332, USA*
Shang Wu, *Rice University, Houston, TX, 77005, USA*
Jyotikrishna Dass, *Rice University, Houston, TX, 77005, USA*
Haoran You, *Georgia Institute of Technology, Atlanta, GA, 30332, USA*
Yingyan (Celine) Lin, *Georgia Institute of Technology, Atlanta, GA, 30332, USA*

*Abstract—Boosting the task accuracy of tiny neural networks (TNNs) has become a fundamental challenge for enabling the deployments of TNNs on edge devices which are constrained by strict limitations in terms of memory, computation, bandwidth, and power supply. To this end, we propose a framework called NetDistiller to boost the achievable accuracy of TNNs by treating them as sub-networks of a weight-sharing teacher constructed by expanding the number of channels of the TNN. Specifically, the target TNN model is jointly trained with the weight-sharing teacher model via (1) gradient surgery to tackle the gradient conflicts between them and (2) uncertainty-aware distillation to mitigate the overfitting of the teacher model. Extensive experiments across diverse tasks validate NetDistiller's effectiveness in boosting TNNs' achievable accuracy over state-of-the-art methods.*

The record-breaking performance recently achieved by neural networks (NNs) has motivated their increasing application in almost every discipline of science and engineering. In parallel, Internet of Things (IoT) connected devices are projected to number 30.9 billion units by 2025 worldwide [1]. It is thus paramount to deploy NN-powered intelligence on numerous IoT devices to harness the data collected at the edge for enabling various on-device intelligent functionalities that can revolutionize human life. This tremendously growing demand has given rise to the field of tiny neural networks (TNNs) which have attracted substantially increasing attention. This is because TNNs enable small and inexpensive edge devices to work directly on local data at a lower power and computing cost, leading to both reduced latency and enhanced privacy as it alleviates or even eliminates the necessity of internet connectivity for sharing and centralizing the data on a cloud server. Nevertheless, the achievable task performance of TNNs is still unsatisfactory due to their limited model capacity that cannot meet the strict resource constraints of edge devices. Hence, improving the task performance of TNNs has become a fundamental challenge for enabling their wide-scale adoption, which is highly desired in numerous real-world edge applications.

To tackle the aforementioned challenge and thus unleash the promise of TNNs at the edge, there has been an increasing research effort towards boosting their achievable task performance. In particular, it has been shown that training TNNs is fundamentally different from training large NNs. For example, the authors of [2] identify that TNNs suffer from under-fitting due to their limited model capacity in contrast to large NNs which exhibit over-fitting. They further note that although adopting data augmentation and regularization techniques improves the ImageNet accuracy achieved by over-parameterized large NNs, e.g., ResNet50, it actually hurts the accuracy of TNNs, e.g., MobileNetV2-Tiny [3], which is 174$\times$ smaller than ResNet50.

Drawing inspiration from prior arts, we hypothesize that augmenting the model capacity (e.g., channels) during training enables TNNs to acquire extra knowledge

XXXX-XXX © 2023 IEEE
Digital Object Identifier 10.1109/XXX.0000.0000000

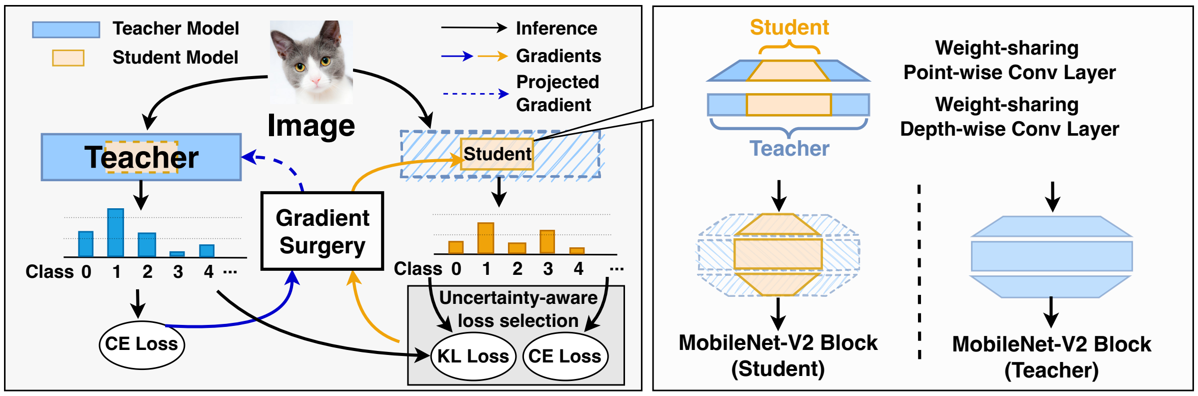

**FIGURE 1.** An overview of *NetDistiller*. The target TNN is a student model serving as a sub-network within a weight-sharing teacher model, constructed by expanding the number of channels of the target TNN. The teacher and student models are trained simultaneously while the teacher model is trained with the ground truth labels and the objective for training the student model is input-adaptively decided between an in-situ distillation mechanism and the ground truth labels based on its output uncertainty. To alleviate the gradient conflict issue observed during the training process, the teacher's gradients are modified via a gradient surgery to remove the conflicting components based on the student's gradients.

and thus achieve improved task accuracy. In vanilla knowledge distillation [4], the knowledge encoded by a large model is transferred to a smaller one by training the small model with the outputs and/or activations of the large model, and thus the small model is able to achieve a higher accuracy by mimicking the behaviors of the large model. In this work, we advocate a new in-situ knowledge distillation scheme that is orthogonal to the vanilla one for further boosting the achievable task performance of TNNs, and make the following contributions:

- We are the first to demonstrate that integrating a weight-sharing supernet with in-situ distillation can serve as an effective training recipe for boosting the achievable task performance of TNNs. Specifically, we propose a framework called *NetDistiller*, which incorporates the target TNN as a student model in a weight-sharing supernet that acts as a teacher model to boost the task performance of the trained TNNs without incurring any inference overhead.
- We identify that vanilla in-situ distillation can cause serious gradient conflicts between the supernet teacher and sub-network student (i.e., the target TNN). Specifically, we find that up to 50% of the student model's weight gradients have negative cosine similarities with those of the teacher model, resulting in poor convergence when accumulated on their shared weights. Furthermore, vanilla in-situ distillation tends to result in overfitting in the teacher model, thus diminishing the effectiveness of our in-situ distillation.
- To alleviate both the two issues identified above, *NetDistiller* proposes to (i) remove the conflicting gradients by projecting the teacher's conflicting gradients onto the normal plane of the student's gradients, and (ii) integrate an uncertainty-aware distillation to dynamically choose the student loss function between the Kullback–Leibler divergence and the cross-entropy loss based on the certainty of the student model output. In this way, *NetDistiller* can unleash the promising effectiveness of in-situ distillation in more favorably training TNNs.
- We perform extensive experimental evaluations and ablation studies to validate the effectiveness of our proposed *NetDistiller* framework for boosting the achievable accuracy of TNNs as compared to the state-of-the-art method, e.g., a 2.3% higher accuracy over *NetAug* [2] when training the MobileNet-V3-w0.35 model on the ImageNet dataset. We understand that *NetDistiller* has opened up a new perspective for boosting the achievable task performance of TNNs and enriched the field of knowledge distillation.

## RELATED WORK

**Efficient / Tiny Neural Networks.** A substantial progress has been made in designing efficient and mobile-friendly neural networks. For example, MobileNets [5] utilize Depthwise Separable Convolutions, showing that a vanilla convolution layer can be replaced with the combination of a depthwise convolution and a pointwise convolution; ShuffleNet utilizes two new operations, i.e., pointwise group convolution and channel shuffle, to greatly reduce NNs' computation cost while maintaining their task accuracy; Compressing off-the-shelf deep NNs with pruning and/or quantization removes model redundancy and reduces the overall complexity, thereby, increasing NNs' inference efficiency. In parallel to the above manually designed efficient networks and compression schemes, automated machine learning has been successfully used via neural architecture search (NAS), such as Efficientnet, FBNet, ProxylessNAS. In contrast to the above techniques, the proposed framework aims to improve the inference accuracy of TNNs via in-situ distillation where the target TNN architecture is used as a sub-network student of a weight-sharing supernet teacher model.

**Knowledge Distillation.** Knowledge Distillation (KD) [6] refers to the idea of transferring “knowledge" acquired by a pre-trained and over-parameterized (teacher) model to a small (student) model that is more suitable for edge deployment. Specifically, the small model usually has insufficient capacity to learn a concise knowledge representation, and KD empowers the student model to learn the exact behaviour of the teacher model by mimicking the teacher's outputs at each level, i.e., ‘soft labels’ (not just the final loss). In this work, we advocate a new in-situ KD scheme, namely, *NetDistiller* for improving the task performance of TNNs. In contrast to vanilla KD where the teacher model is a large NN pretrained on a *different* dataset and the student model is a separate smaller model, *NetDistiller* is an orthogonal approach which incorporates the TNN as a student sub-network *within* a weight-sharing supernet which acts as the teacher model for in-situ distillation on the *same* training set.

**Network Augmentation.** The authors in [2] propose network augmentation, namely, *NetAug*, to boost the accuracy of tiny deep learning by alleviating the under-fitting issue. Specifically, *NetAug* augments the network (reverse dropout) during training by putting the tiny model as a sub-model into the larger model for auxiliary supervision beyond functioning independently. In contrast, *NetDistiller* provides an alternative scheme to boosting the performance of TNNs via in-situ knowledge distillation. Unlike *NetAug* which samples one augmented network to provide auxiliary supervision that is added to the base supervision at each training step, the TNN in *NetDistiller* is used as a sub-network (student) in a ‘static’ weight-sharing supernet (teacher) constructed by expanding the channels of the target TNN. In the Experimental Results section, we provide comparative study of the proposed *NetDistiller* with the state-of-the-art scheme *NetAug* and find that *NetDistiller* outperforms *NetAug*, e.g., achieves 2.3% higher accuracy when training the MobileNet-V3-w0.35 model on the ImageNet dataset.

## NetDistiller FRAMEWORK

*NetDistiller* is a training recipe for boosting the accuracy of tiny deep learning by incorporating the target TNNs as a student model (sub-network) in a weight-sharing supernet that acts as a teacher model. *NetDistiller* performs in-situ distillation of ‘knowledge’ from a supernet teacher model to the sub-network student model which is our target TNN. In this section, we first describe the construction of the weight-sharing supernet from the TNN followed by practical implementation of in-situ distillation. Then, we describe techniques to resolve the gradient conflicts between the teacher and student models and mitigate the over-fitting issue in the teacher model during the final training stage via uncertainty-aware distillation. Finally, we discuss the training and the inference overheads incurred by *NetDistiller*.

### Enabler 0: Constructing the Weight-Sharing Teacher Model

*NetDistiller* expands the target TNN along the channels to construct a weight-sharing supernet as the teacher model. Thus, the target TNN acts as a sub-network model. The student and teacher models share all the convolution layers weights while maintaining their respective Batch-Normalization layers to take into account the different running statistics (means and variances) in their activation values. As a novel training recipe for the capacity augmentation of a target TNN to alleviate its under-fitting issue and to boost its accuracy, *NetDistiller* constructs a teacher model with $3\times$ times the number of channels as the target TNN (student model). Figure 1 depicts the above-described construction of the weight-sharing teacher model from the TNN.

### Enabler 1: In-Situ Distillation

In-situ knowledge distillation stabilizes the training of the supernet, and improves the performance of sub-networks. TNNs are more likely to get stuck in local minimums due to insufficient capacity, which limits their training and test performance compared to over-parameterized large NNs [2]. *NetDistiller* integrates the target TNN as a sub-network student model in a weight-sharing supernet teacher model constructed by expanding the channels of the target NN. To the best of our knowledge, *NetDistiller* is the first to demonstrate that applying in-situ distillation to a weight-sharing supernet [7] can serve as an effective training recipe for boosting the achievable task performance of TNNs (Figure 1).

In-situ distillation leverages the ‘soft labels’ predicted by the supernet as the training label to supervise the sub-network student model during each training iteration while using ground truth labels for the teacher model. Formally, at training iteration $n$, the supernet parameter $W$ is updated by

$$W^n \leftarrow W^{n-1} + \eta g(W^{n-1}),$$

where $\eta$ is the step size, and

$$g(W^{n-1}) = \nabla_W \Big( \mathcal{L}_{\mathcal{D}}(W) + \mathcal{L}_{stu}\big([W, W_{stu}]; W^{n-1}\big) \Big)\Big|_{W=W^{n-1}} \quad (1)$$

Here, $\mathcal{L}_{\mathcal{D}}(W)$ is the cross-entropy (CE) loss of the supernet teacher on a training dataset $\mathcal{D}$, and $\mathcal{L}_{stu}\big([W, W_{stu}]; W^n\big)$ is the student loss determined by uncertainty-aware distillation (introduced in the Enabler 3 part). Additionally, the distillation process in *NetDistiller* is single-shot, i.e., it is implemented in-situ during training without additional computation and memory cost, unlike two-step vanilla KD where a large model has to be first pre-trained.

## Enabler 2: Resolving Gradient conflicts

The gradients from both the student and the teacher models are accumulated on the shared weights. We identify that vanilla in-situ distillation may cause serious gradient conflicts between the supernet teacher and sub-network student (target TNN). Specifically, we find that up to 50% of the student model gradients have a negative cosine-similarity with those of the teacher model. Inspired by the PCGrad [8] which performs gradient surgery for multi-task learning, *NetDistiller* projects the conflicting teacher gradients to the normal plane of student gradients, removing the conflicting components in the teacher gradients and improving the performance of TNNs. Let $\nabla l_{stu}$ and $\nabla l_{tea}$ denote the gradients of the student and the teacher models respectively. We define $\phi$ as the angle between the above two gradients, and $g$ as the final gradient for updating the weights. In order to guarantee the student model training, we project the conflicting teacher's gradient, $proj(\nabla l_{tea})$, when the Cosine-Similarity, $cos(\phi) = \frac{\nabla l_{stu} \cdot \nabla l_{tea}}{||\nabla l_{stu}||\,||\nabla l_{tea}||}$ is negative.

$$g = \nabla l_{stu} + proj(\nabla l_{tea}), \text{ where}$$

$$proj(\nabla l_{tea}) = \begin{cases} \nabla l_{tea} - \frac{\nabla l_{tea}^T \nabla l_{stu}}{||\nabla l_{stu}||^2} \nabla l_{stu}, & \text{if } \cos(\phi) < 0 \\ \nabla l_{tea}, & \text{otherwise} \end{cases} \quad (2)$$

## Enabler 3: Uncertainty-aware Distillation

Unlike the vanilla KD and other uncertainty-aware distillation method (i.e. [9]) using pretrained teacher models, *NetDistiller* simultaneously trains the student and the weight-sharing teacher models via in-situ distillation. We observe that the supernet teacher model suffers from over-fitting at the final training stage. Additionally, [10] advocates that large models have the largest improvement on samples where the small model is most uncertain. As for certain examples, even those where the small model is not particularly accurate, large models are often unable to improve. Thus the teacher model is not always a good teacher during the whole training. We propose uncertainty-aware distillation (UD) to dynamically select the student loss functions between the Kullback–Leibler (KL) divergence and cross entropy (CE) losses based on the certainty of the student model output (see Figure 1). We measure the uncertainty via the entropy of the student outputs. When the entropy of the student output is high (i.e., uncertain), the student is distilled by the weight-sharing teacher (via KL divergence loss), otherwise, the student is trained by the ground truth label (via cross entropy loss).

Let T denote the uncertainty threshold; $\mathcal{L}_{stu}$ denote the student model loss; `KL()` and `CE()` denote the KL loss and the Cross-Entropy loss, respectively; $W$ and $W_{stu}$ denote the teacher and student models; $x$ and $y$ denote the input data and the ground truth labels. We use a variable *uncertainty* to denote the entropy of the student model outputs $W_{stu}(x)$.

$$\mathcal{L}_{stu} = \begin{cases} \mathtt{KL}(W_{stu}(x), W(x)), & uncertainty \geq \mathrm{T} \\ \mathtt{CE}(W_{stu}(x), y), & \text{otherwise} \end{cases} \quad (3)$$

## Analysis of Training and Inference Overhead

In contrast to the two-step distillation process in vanilla KD, *NetDistiller* performs ***one-shot*** in-situ distillation of knowledge from the supernet teacher to the sub-network student model without any additional computation and memory cost. Similar to that of *NetAug*, *NetDistiller* has ***zero*** inference overhead because only the sub-network student model (target TNNs) is used, enabling the deployment of TNNs feasible on resource-constrained edge devices. Despite expanding the target TNN model by 3×, we observe a mere 20% increase in the training time of *NetDistiller* to that of vanilla TNNs. We also evaluated *NetDistiller* on ResNet-50, and found that *NetDistiller* works better with TNNs, because with the increasing of the model size, expanding the model channel significantly increases the requirement of computation resources for training, distilling, and gradient surgery, resulting in an unacceptable training speed decrease.

**TABLE 1.** Benchmark of *NetDistiller* and SOTA method for training TNNs. r160: The input image resolution is 160 × 160. w0.35: The model has 0.35× number of channels than the vanilla one.

| Model | MobileNet-V2-Tiny r144 | MCUNet r176 | MobileNet-V3, r160 w0.35 | ProxylessNAS, r160 w0.35 | ProxylessNAS, r160 w1.0 | MobileNet-V2, r160 w0.35 | MobileNet-V2, r160 w1.0 |
|---|---|---|---|---|---|---|---|
| Params | 0.75M | 0.74M | 2.2M | 1.8M | 4.1M | 1.7M | 3.5M |
| MACs | 23.5M | 81.8M | 19.6M | 35.7M | 164.1M | 30.9M | 154.1M |
| Baseline | 51.7% | 61.5% | 58.1% | 59.1% | 71.2% | 56.3% | 69.7% |
| NetAug [2] | 53.3% | 62.7% | 60.3% | 60.8% | 71.9% | 57.8% | 70.6% |
| In-situ | 54.1% | 62.7% | 62.1% | 60.7% | 71.2% | 58.5% | 71.2% |
| In-situ + PCGrad [8] | 54.5% | 63.4% | 62.3% | 61.3% | 72.5% | 59.0% | 72.0% |
| ***NetDistiller* (ours)** | **54.8%** | **64.2%** | **62.6%** | **61.5%** | **72.8%** | **59.3%** | **72.6%** |

**TABLE 2.** Ablation study of channel expansion rates on MobileNet-V2-w0.35 (MBV2-w0.35) and MobileNet-V3-w0.35 (MBV3-w0.35). Different teacher sizes in the first row indicate the channel expansion rates. Considering the limited improvement (0.2%) between ×4 and ×3 teachers on MobileNet-V2-w0.35 model and the training efficiency, teacher with ×3 size is selected in *NetDistiller*.

| Teacher Size | Baseline | ×2 | ×3 | ×4 | ×5 |
|---|---|---|---|---|---|
| MobileNet-V2-0.35 | 56.3% | 58.0% | 58.5% | **58.7%** | 58.3% |
| MobileNet-V3-0.35 | 58.1% | 61.3% | **62.1%** | 61.8% | 61.8% |

## EXPERIMENTAL RESULTS

### Experiment Setup

**Models**. We benchmark *NetDistiller* with SOTA TNNs training methods, e.g., *NetAug* [2], and knowledge distillation (KD) [4], on five commonly adopted TNNs, including MobileNet-V2-Tiny, MobileNet-V2 (w0.35 and w1.0), MobileNet-V3, MCUNet (256kb-1mb), and ProxylessNAS (w0.35, w1.0). The w0.35 indicates the models have 0.35 times of channels over the vanilla one (w1.0). Following the model definitions in [2], the channels of w0.35 models are round to products of 8.

**Datasets.** Following [2], we consider the ImageNet dataset with an input resolution of r144, r160, and r176 for different target TNNs. In the external knowledge distillation experiments, the input resolution for the external teacher is r224 to match its pretraining setting. The object detection experiments are trained on PASCAL VOC 2007+2012 datasets and evaluated on PASCAL VOC 2007 eval set with an input resolution of r416.

**Evaluation Metrics.** *NetDistiller* and baseline methods are evaluated in terms of the top-1 accuracy on ImageNet and the average precision at IoU=0.5 (AP50) for the object detection on PASCAL VOC.

**Training Setting.** We train TNNs for 180 epochs using an SGD optimizer with a momentum of 0.9 and an initial learning rate of 0.4 decayed by a cosine learning rate scheduler. We adopt a learning rate warm-up for 5 epochs and the gradients are clipped to 1.0. The label smoothing technique with a factor of 0.1 is adopted when using the ground truth label. For the *NetDistiller* with uncertainty-aware distillation, we use the same training recipe but increase the training epochs to 360. The uncertainty threshold T is set to 3.75 based on the empirical observation of our ablation study. All the ImageNet experiments are run on 8 GPUs with a batch size of 1024. As a recent paper [2] discovered that data augmentation and regularization could be harmful to TNN training, we only utilize standard data augmentations (e.g. random flip, random crop) and disable regularization methods like dropout and drop path. For transfer learning on the object detection task, MobileNet-V2-w0.35 and MobileNet-V3-w0.35 models are connected with a YOLO-v4 head. All the object detection experiments are trained via an SGD optimizer with a momentum of 0.9 and an initial learning rate of 1e-4 decayed by a cosine learning rate scheduler for 100 epochs with a batch size of 8.

**TABLE 3.** Ablation study of gradient surgery on MobileNet-V2-w0.35 and MobileNet-V3-w0.35 models for 360 epochs. We disable gradient surgery and calculate the cosine-similarity between the two gradients (teacher's and student's) of each convolutional layer. The percentage values shown under different epochs reflect the average ratio of the number of layers with negative cosine-similarity (gradient conflicts) w.r.t. the total number of layers in the model.

| Epoch | 1 | 90 | 180 | 270 | 360 |
|---|---|---|---|---|---|
| MobileNet-V2-w0.35 | 51.5% | 40.1% | 37.4% | 39.4% | 38.2% |
| MobileNet-V3-w0.35 | 50.1% | 45.2% | 34.7% | 38.5% | 37.4% |

**TABLE 4.** Ablation study of different uncertainty-aware distillation thresholds on MobileNet-V2-w0.35 and MobileNet-V2-w1.0. The first column is the thresholds. The uncertainty-aware distillation distills the student model if its output entropy (uncertainty) is higher than the threshold and trains the student model with ground truth labels otherwise.

| Model | Uncertainty Threshold 2.5 | 3.75 | 5.0 |
|---|---|---|---|
| MobileNet-V2-w0.35 | 59.1% | **59.3%** | 58.9% |
| MobileNet-V2-w1.0 | 71.9% | **72.6%** | 71.2% |

**TABLE 5.** Combination of *NetDistiller* and External knowledge distillation. KD: Distill the target TNN with an external teacher (ImageNet pretrained ResNet-50). *NetDistiller* w/o UD: Uncertainty-aware distillation is turned off in the external KD experiments. NetDistiller+KD: The external teacher distills both *NetDistiller* teacher and student models.

| Model | Baseline | KD | NetDistiller w/o UD | NetDistiller+KD |
|---|---|---|---|---|
| MobileNet-V2-Tiny, r144 | 51.7% | 53.7% (+2.0%) | 55.5% (+3.8%) | **56.1% (+4.4%)** |
| MobileNet-V2-w0.35, r160 | 56.3% | 58.4% (+2.1%) | 59.0% (+2.7%) | **59.5% (+3.2%)** |
| MobileNet-V3-w0.35, r160 | 58.1% | 61.6% (+3.5%) | 62.3% (+4.2%) | **62.5% (+4.4%)** |
| ProxylessNAS-w0.35, r160 | 59.1% | 60.8% (+1.7%) | 61.3% (+2.2%) | **61.9% (+2.8%)** |

## Benchmark with SOTA Methods for Training TNNs

As shown in Table 1, TNNs trained via in-situ distillation only can achieve similar or outperform the SOTA TNNs training method, *NetAug* [2]. Though suffering form the gradient conflicts problem, in-situ distillation improves up to 4.0% of accuracy compared with the baselines on MobileNet-V3-w0.35, and up to 1.8% higher than the SOTA method. After the PCGrad [8] is applied to alleviate the gradient conflicts problem, the TNNs accuracy improve about 0.5%, which also confirms gradient conflict is an inherent problem when training TNNs via in-situ distillation. To further improve TNNs performance, we combine the in-situ distillation with PCGrad and the uncertainty-aware distillation in the proposed *NetDistiller*.

## Ablation Studies of *NetDistiller*

**Channel expansion rates of the teacher model.** Since *NetDistiller* expands the channels of TNNs to create a weight-sharing supernet as the teacher model, the channel expansion rate of the teacher model could impact the effectiveness of the in-situ distillation mechanism, considering that a thin teacher will have limited capacity and the information of an extremely wide teacher cannot be effectively inherited. In order to identify a proper channel expansion rate, we evaluate *NetDistiller* with $\times 2$, $\times 3$, $\times 4$, and $\times 5$ channel expansion rates on top of two TNNs, MobileNet-V2-w0.35 and MobileNet-V3-w0.35. As shown in Table 2, we can observe that (1) all four teacher models can boost TNNs' accuracy, indicating the general effectiveness of *NetDistiller*, and (2) MobileNet-V2-w0.35/MobileNet-V3-w0.35 achieve the best accuracy under a channel expansion rate of $\times 4/\times 3$, respectively. To save the training overhead introduced by the expanded teacher, we adopt a channel expansion rate of $\times 3$ by default in *NetDistiller*.

**Visualizing the gradient conflicts.** Due to the weight-sharing mechanism, jointly training the student and the teacher will accumulate gradients to the same weights, inevitably resulting in gradient conflicts. To validate whether gradient conflicts happen in different training stages, we measure the ratio of the numbers of the layers with negative cosine-similarity averaged over the validation set w.r.t. the total number of the layers in the model along the training process. The results in Table 3 indicate that the teacher model and the student model indeed suffer from gradient conflicts with up to 51.5% layers having conflict gradients. To tackle this, adopting the gradient surgery results in an accuracy improvement of 0.5% and 0.2% on MobileNet-V2-w0.35 and MobileNet-V3-w0.35, respectively, according to Table 1.

**Deciding the uncertainty threshold of uncertainty-aware distillation.** Our proposed uncertainty-aware distillation mechanism adaptively decides the objective for the student model between in-situ distillation and the cross entropy over ground-truth labels based on the uncertainty of student model outputs. To decide the uncertainty threshold, we validate MobileNet-V2-w0.35 and MobileNet-V2-w1.0 models with uncertainty thresholds of 5.0, 3.75, and 2.5, considering the entropy of ImageNet models is in the range of $[1.5, 10]$ when adopting a label smoothing factor of 0.1. As shown in Table 4, both two MobileNet-V2 models achieve their best accuracy under an uncertainty threshold of 3.75 with an accuracy improvement of 0.3% and 0.6%, respectively, compared to *NetDistiller* w/o uncertainty-aware distillation. Without bells and whistles, we adopt 3.75 by default when enabling uncertainty-aware distillation.

**Benchmark and integrate with knowledge distillation.** One natural baseline for *NetDistiller* is standard knowledge distillation. Based on recent observations that large gaps between the teacher and student models may cause inferior knowledge distillation performances [11], we hypothesize that (a) our proposed in-situ distillation is a better mechanism to enable TNN training as compared to knowledge distillation, and (b) knowledge distillation is orthogonal to our method and can be applied in parallel. To validate this, we distill the knowledge of an ImageNet pretrained ResNet-50 for both *NetDistiller*'s teacher and student models, which is dubbed an external distillation mechanism to distinguish from our in-situ distillation, and benchmark

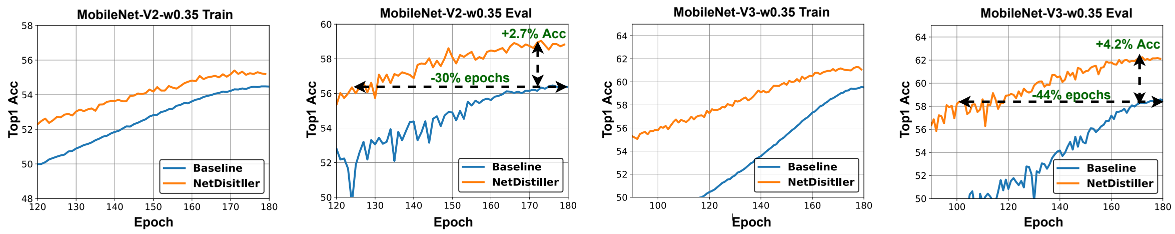

**FIGURE 2.** Visualizing the training process of *NetDistiller* and baselines for MobileNet-V2-w0.35 and MobileNet-V3-w0.35 models. *NetDistiller* boosts both training and evaluation accuracy for the TNNs, showing the huge potential that *NetDistiller* are able to empower TNNs and help TNNs alleviate the under-fitting issue.

with (1) vanilla *NetDistiller* and (2) standard knowledge distillation. As shown in Table 5, we can observe that (1) vanilla *NetDistiller* can outperform standard knowledge distillation, e.g., an accuracy improvement of 3.8% and 4.% on MobileNet-V2-Tiny and MobileNet-V3-w0.35, respectively, validating our hypothesis (a), and (2) knowledge distillation is orthogonal to *NetDistiller* and applying knowledge distillation on top of *NetDistiller* can lead to an accuracy improvement of 4.4% on MobileNet-V2-Tiny and MobileNet-V3-w0.35, respectively, validating our hypothesis (b).

## Visualization of Training Trajectories

We visualize the training curves of MobileNet-V2-w0.35 and MobileNet-V3-w0.35 trained by *NetDistiller* for 180 epochs, as well as the corresponding standard training baseline, in Figure 2. We can observe that both *NetDistiller*'s training and evaluation accuracy are consistently higher than the corresponding baselines under the same training epoch during the whole training process, e.g., *NetDistiller* achieves a 2.7% and 4.2% accuracy improvement over the baselines for MobileNet-V2-w0.35 and MobileNet-V3-w0.35, respectively. In addition, to achieve a comparable accuracy, *NetDistiller* requires less training epochs, e.g., a 44% training epochs saving as annotated in Figure 2.

## Transfer Learning Study on Object Detection

In order to validate the generality of the representations learned by *NetDistiller*, we transfer *NetDistiller*'s trained MobileNet-V2-w0.35 and MobileNet-V3-w0.35 to a downstream object detection task and benchmark with those standardly pretrained models (w/ and w/o KD) on ImageNet. In particular, the final pooling and linear layers in MobileNet-V2-w0.35 and MobileNet-V3-w0.35 are replaced with the YOLO-v4 object detection head. As shown in Table 6, *NetDistiller* consistently wins a better transferability with a 1.9% / 1.8% higher AP on MobileNet-V2-w0.35 / MobileNet-V3-w0.35 as compared to standard training baselines. Note that although pretrained features on classification tasks may not be necessarily useful for downstream tasks [12], [2], which is also echoed with our results of KD pretrained MobileNet-V3-w0.35 in Table 6, *NetDistiller* is still able to improve the achievable AP by up to 1.9% on both models, which also indicates that our method is generally applicable across different tasks and datasets.

**TABLE 6.** Transfer learning to object detection on PASCAL VOC 2007+2012 datasets of MobileNet-V2-w0.35 (MBV2) and MobileNet-V3-w0.35 (MBV3) models with *NetDistiller* pretrained weights on ImageNet. Both of the two models are connected with YOLO-v4 detection head. The experiments are measured by Average Percision at IoU=0.5 (AP50).

| Model | Baseline AP50 | KD AP50 | *NetDistiller* AP50 |
|---|---|---|---|
| MobileNet-v2-w0.35 | 60.4% | 61.1% | **62.3%** |
| MobileNet-v3-w0.35 | 63.6% | 62.8% | **65.2%** |

## CONCLUSION

It is highly desired to deploy TNNs on resource-constrained platforms due to their superior model efficiency, whereas their achievable accuracy is often bottlenecked by the limited model capacity. To bridge this gap, we propose *NetDistiller* that incorporates a weight-sharing supernet teacher model constructed by expanding the channel capacity of tiny model. Extensive experiments and ablation studies demonstrate that *NetDistiller* boosts the achievable accuracy of tiny neural networks compared to *NetAug* and vanilla knowledge distillation method, thereby, empowering tiny neural networks to overcome under-fitting issues resulting from insufficient capacity.

## Acknowledgement

The work is supported by the National Science Foundation (NSF) through the EPCN program (Award number: 1934767) and by CoCoSys, one of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

## REFERENCES

1. Lionel Sujay Vailshery. Iot and non-iot connections worldwide 2010-2025. https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/. Accessed: 2021-03-08.
2. Han Cai, Chuang Gan, Ji Lin, and song han. Network augmentation for tiny deep learning. In International Conference on Learning Representations, 2022.
3. Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, Song Han, et al. Mcunet: Tiny deep learning on iot devices. Advances in Neural Information Processing Systems, 33:11711–11722, 2020.
4. Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2(7), 2015.
5. Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.
6. Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, Advances in NIPS, volume 27. Curran Associates, Inc., 2014.
7. Dilin Wang, Chengyue Gong, Meng Li, Qiang Liu, and Vikas Chandra. Alphanet: Improved training of supernets with alpha-divergence. In International Conference on Machine Learning, pages 10760–10771. PMLR, 2021.
8. Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. Advances in NIPS, 33:5824–5836, 2020.
9. Youcai Zhang, Zhonghao Lan, Yuchen Dai, Fangao Zeng, Yan Bai, Jie Chang, and Yichen Wei. Prime-aware adaptive distillation. In ECCV, pages 658–674. Springer, 2020.
10. Taman Narayan, Heinrich Jiang, Sen Zhao, and Sanjiv Kumar. Predicting on the edge: Identifying where a larger model does better. arXiv preprint arXiv:2202.07652, 2022.
11. Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In Proceedings of the IEEE/CVF ICCV, pages 4794–4802, 2019.
12. Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In Proceedings of the IEEE/CVF ICCV, pages 4918–4927, 2019.

**Shunyao Zhang** is a Ph.D. student at Rice University, Houston, USA. He received his master's degree in Electrical and Computer Engineering from Carnegie Mellon University, Pittsburgh, USA. His research interests are tiny ML and adversarial robustness. Contact him at sz74@rice.edu.

**Yonggan Fu** is a Ph.D. student at Georgia Institute of Technology. Before that, he obtained his Bachelor's degree from the School of The Gifted Young at the University of Science and Technology of China. His research focus and passion is to develop efficient and robust AI algorithms and co-design the corresponding hardware accelerators toward a triple-win in accuracy, efficiency, and robustness. Contact him at yfu314@gatech.edu.

**Shang Wu** is a master's student at RICE University where he majored in Electrical and Computer Engineering. He received his bachelor's degree in computer science at George Washington University. His research interests are in efficient ML, robust ML and generative AI. Contact him at sw99@rice.edu.

**Jyotikrishna Dass** is a Research Scientist at Rice University and manages the activities at Rice Data to Knowledge Program. His research interests are in distributed and parallel machine-learning systems for efficient edge computing. Previously, he was a postdoc at Dr. Yingyan Lin's lab. Dr. Dass received his Ph.D. in Computer Engineering from Texas A&M University. Contact: jdass@rice.edu

**Haoran You** is currently a Ph.D. student in the CS Department of Georgia Insitute of Technology. He received his bachelor's degree in the advanced class at Huazhong University of Science and Technology and his master's degree at Rice University. He is pursuing his doctoral degree in machine learning and computer architecture realm. His research interests include but are not limited to resource-constrained machine learning, computer vision, deep learning, and algorithm/accelerator co-design. Contact: haoran.you@gatech.edu

**Yingyan (Celine) Lin** is currently an Associate Professor in the School of Computer Science at Georgia Institute of Technology. She leads the Efficient and Intelligent Computing (EIC) Lab, which focuses on developing efficient machine learning systems via cross-layer innovations from algorithm to architecture down to chip design, aiming to promote green AI and enable ubiquitous machine learning powered intelligence. She received a Ph.D. degree in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign, and was an assistant professor at Rice University from 2017 to 2022. She is the corresponding au-

thor of this article. Contact her at celine.lin@gatech.edu