Universally Slimmable Networks and Improved Training Techniques

Jiahui Yu Thomas Huang
University of Illinois at Urbana-Champaign

Abstract

Slimmable networks [25] are a family of neural networks that can instantly adjust the runtime width. The width can be chosen from a predefined widths set to adaptively optimize accuracy-efficiency trade-offs at runtime. In this work, we propose a systematic approach to train universally slimmable networks (US-Nets), extending slimmable networks to execute at arbitrary width, and generalizing to networks both with and without batch normalization layers. We further propose two improved training techniques for US-Nets, named the sandwich rule and inplace distillation, to enhance training process and boost testing accuracy. We show improved performance of universally slimmable MobileNet v1 and MobileNet v2 on ImageNet classification task, compared with individually trained ones and 4-switch slimmable network baselines. We also evaluate the proposed US-Nets and improved training techniques on tasks of image super-resolution and deep reinforcement learning. Extensive ablation experiments on these representative tasks demonstrate the effectiveness of our proposed methods. Our discovery opens up the possibility to directly evaluate FLOPs-Accuracy spectrum of network architectures. Code and models are available at: https://github.com/JiahuiYu/slimmable_networks.

1. Introduction

The ability to run neural network models within latency budget is of paramount importance for applications on mobile phones, augmented reality glasses, self-driving cars, security cameras and many others [22, 10, 15]. Among these applications, many are required to deploy trained models across different devices or hardware versions [25, 9, 13]. However, a single trained network cannot achieve optimal accuracy-efficiency trade-offs across different devices (*e.g.*, face recognition model running on diverse mobile phones). To address the problem, recently slimmable networks [25] were introduced that can switch among different widths at runtime, permitting instant and adaptive accuracy-efficiency trade-offs. The width can be chosen from a predefined widths set, for example $[0.25, 0.5, 0.75, 1.0]\times$, where

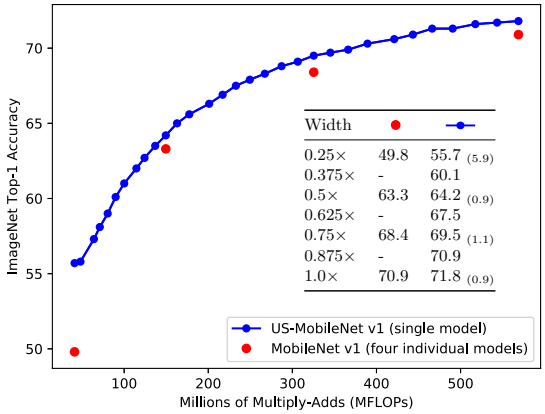

Figure 1. FLOPs-Accuracy spectrum of **single** US-MobileNet v1 model, compared with **four individual** MobileNet v1 models.

$[\cdot]\times$ denotes available widths, and $0.25\times$ represents that the width in all layers is scaled by 0.25 of the full model. To train a slimmable network, switchable batch normalization [25] is proposed that privatizes batch normalization [11] layers for each sub-network. A slimmable network has accuracy similar to that of individually trained ones with the same width [25].

Driven by slimmable networks, a further question arises: *can a single neural network run at arbitrary width*? The question motivates us to rethink the basic form of feature aggregation. In deep neural networks, the value of a single output neuron is an aggregation of all input neurons weighted by learnable coefficients $y = \sum_{i=1}^{n} w_i x_i$, where x is input neuron, y is output neuron, w is learnable coefficient and n is number of input channels. This formulation indicates that each input channel or group of channels can be viewed as a *residual component* [6] to an output neuron. Thus, a wider network should have no worse performance than its slim one (the accuracy of slim one can always be achieved by learning new connections to zeros). In other words, if we consider a single layer, the residual error between full aggregation and partial aggregation decreases and is bounded:

$$|y^n - y^{k+1}| \leq |y^n - y^k| \leq |y^n - y^{k_0}|, \qquad (1)$$

where y^k summarizes the first k channels $y^k = \sum_{i=1}^{k} w_i x_i$, $\forall k \in [k_0, n)$, k_0 is a constant hyper-parameter (for example, $k_0 = \lceil 0.25n \rceil$). The bounded inequality[1] suggests that a slimmable network [25] executable at a discrete widths set can potentially run at any width in between (if properly trained), since the residual error decreases by the increase of width and is bounded. Moreover, the inequality conceptually applies to any deep neural network, regardless of what normalization layers [11, 17] are used. However, as suggested in [25], batch normalization (BN) [11] requires special treatment because of the inconsistency between training and testing.

In this work, we present universally slimmable networks (US-Nets) that can run at any width in a wide range. Three fundamental challenges of training US-Nets are addressed. First, how to deal with neural networks with batch normalization? Second, how to train US-Nets efficiently? Third, compared with training individual networks, what else can we explore in US-Nets to improve overall performance?

Batch normalization [11] has been one of the most important components in deep learning. During training, it normalizes feature with mean and variance of current mini-batch, while in inference, moving averaged statistics of training are used instead. This inconsistency leads to failure of training slimmable networks, as shown in [25]. The switchable batch normalization [25] (we address the version of shared scale and bias by default, the version of private scale and bias will be discussed in Section 6) is then introduced. However, it is not practical for training US-Nets for two reasons. First, accumulating independent BN statistics of all sub-networks in a US-Net during training is computationally intensive and inefficient. Second, if in each iteration we only update some sampled sub-networks, then these BN statistics are insufficiently accumulated thus inaccurate, leading to much worse accuracy in our experiments. To properly address the problem, we adapt the batch normalization with a simple modification. The modification is to calculate BN statistics of all widths after training. The weights of US-Nets are fixed after training, thus all BN statistics can be computed in parallel on cluster servers. More importantly, we find that a *randomly sampled subset* of training images, as few as 1 mini-batch (1024 images), already produces accurate estimation. Thus calculating BN post-statistics can be very fast. We note that to be more general, we intentionally avoid modifying the formulation of BN or proposing new normalization.

Next we propose an improved training algorithm for US-Nets motivated by the bounded inequality in Equation 1. To train a US-Net, a natural solution is to accumulate or average losses sampled from different widths. For example, in each training iteration we randomly sample n widths in the range of $[0.25, 1.0]\times$. Taking a step further, we should notice that in a US-Net, performances at all widths are bounded by performance of the model at smallest width (*e.g.*, $0.25\times$) and largest width (*e.g.*, $1.0\times$). In other words, optimizing performance lower bound and upper bound can implicitly optimize the model at all widths. Thus, instead of sampling n widths randomly, in each training iteration we train the model at smallest width, largest width and (n-2) randomly sampled widths. We employ this rule (named *the sandwich rule*) to train US-Nets and show better convergence behavior and overall performance.

Further we propose *inplace distillation* that transfers knowledge inside a single US-Net from full-network to sub-networks *inplace* in each training iteration. The idea is motivated by two-step knowledge distilling [7] where a large model is trained first, then its learned knowledge is transferred to a small model by training with predicted soft-targets. In US-Nets, by *the sandwich rule* we train the model at largest width, smallest width and other randomly sampled widths all together in each iteration. Remarkably, this training scheme naturally supports inplace knowledge transferring: we can directly use the predicted label of the model at the largest width as the training label for other widths, while for the largest width we use ground truth. It can be implemented inplace in training without additional computation and memory cost. Importantly, the proposed *inplace distillation* is general and we find it works well not only for image classification, but also on tasks of image super-resolution and deep reinforcement learning.

We apply the proposed methods to train universally slimmable networks on representative tasks with representative networks (both with and without BN, and both residual and non-residual networks). We show that trained US-Nets perform similarly or even better than individually trained models. Extensive ablation studies on *the sandwich rule* and *inplace distillation* demonstrate the effectiveness of our proposed methods. Our contributions are summarized as follows:

1. For the first time we are able to train a single neural network executable at arbitrary width, using a simple and general approach.
2. We further propose two improved training techniques in the context of US-Nets to enhance training process and boost testing accuracy.
3. We present experiments and ablation studies on image classification, image super-resolution and deep reinforcement learning.
4. We further intensively study the US-Nets with regard to (1) width lower bound k_0, (2) width divisor d, (3) number of sampled widths per training iteration n, and (4) size of subset for BN post-statistics s.

[1]The analysis is based on a single hidden layer. Future research on theoretical analysis of deep neural networks with nonlinear activation may fully reveal why or why not universally slimmable networks exist.

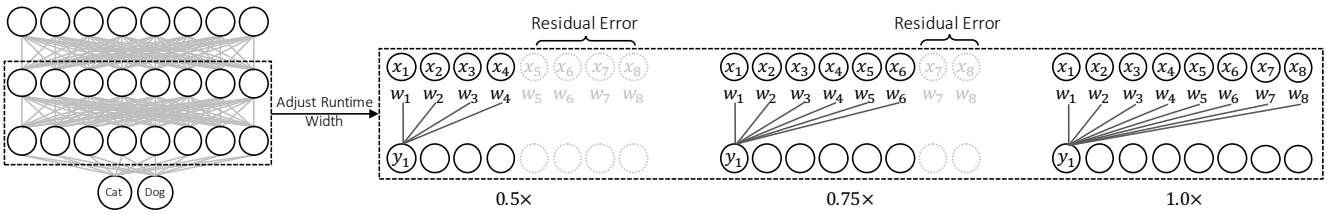

Figure 2. Illustration of a network executing at different widths. We specifically consider an output neuron y_1 in a layer (right, zoomed).

5. We further show that our method can also be applied to train nonuniform US-Nets where each layer can adjust its own width ratio, instead of a global width ratio uniformly applied on all layers.

6. Our discovery opens up the possibility to many related fields, for examples, network comparison in terms of FLOPs-Accuracy spectrum (Figure 1), and one-shot architecture search for number of channels [24].

2. Related Work

Slimmable Networks. Yu *et al*. [25] present the initial approach to train a single neural network executable at different widths, permitting instant and adaptive accuracy-efficiency trade-offs at runtime. The width can be chosen from a predefined widths set. The major obstacle of training slimmable networks is addressed: accumulating different numbers of channels results in different feature mean and variance. This discrepancy across different sub-networks leads to inaccurate statistics of shared Batch Normalization layers [11]. Switchable batch normalization is proposed that employs independent batch normalization for different sub-networks in a slimmable network. On tasks of image recognition (*i.e*., classification, detection and segmentation), slimmable networks achieve accuracy similar to that of individually trained models [25].

Knowledge Distilling. The idea of knowledge distilling [7] is to transfer the learned knowledge from a pre-trained network to a new one by training it with predicted features, soft-targets or both. It has many applications in computer vision, network compression, reinforcement learning and sequence learning problems [2, 4, 12, 14, 16]. FitNet [16] proposes to train a thinner network using both outputs and intermediate representations learned by the teacher network as hints. Net2Net [4] proposes to transfer the knowledge from a pretrained network to new deeper or wider one for accelerating training. Actor-Mimic [14] trains a single policy network to behave in multiple tasks with guidance of many teacher networks. Knowledge distillation is also effectively applied to word-level prediction for neural machine translation [12].

3. Universally Slimmable Networks

3.1. Rethinking Feature Aggregation

Deep neural networks are composed of layers where each layer is made of neurons. As the fundamental element of deep learning, a neuron performs weighted sum of all input neurons as its value, propagating layer by layer to make final predictions. An example is shown in Figure 2. The output neuron y is computed as:

$$y = \sum_{i=1}^{n} w_i x_i, \tag{2}$$

where n is the number of input neurons (or channels in convolutional networks), $x = \{x_1, x_2, ..., x_n\}$ is input neurons, $w = \{w_1, w_2, ..., w_n\}$ is learnable coefficient, y is a single output neuron. This process is also known as *feature aggregation*: each input neuron is responsible for detecting a particular feature, and the output neuron aggregates all input features with learnable transformations.

The number of channels in a network is usually a manually picked hyper-parameter (*e.g*., 128, 256, ..., 2048). It plays a significant role in the accuracy and efficiency of deep models: wider networks normally have better accuracy with sacrifice of runtime efficiency. To provide the flexibility, many architecture engineering works [8, 18, 26] individually train their proposed networks with different width multipliers: a global hyper-parameter to slim a network uniformly at each layer.

We aim to train a single network that can directly run at arbitrary width. It motivates us to rethink the basic form of feature aggregation in deep neural networks. As shown in Figure 2, feature aggregation can be explicitly interpreted in the framework of *channel-wise residual learning* [6], where each input channel or group of channels can be viewed as a *residual component* [6] for the output neuron. Thus, a wider network should have no worse performance than its slim one (the accuracy of slim one can always be achieved by learning new connections to zeros). In other words, the residual error δ between fully aggregated feature y^n and partially aggregated feature y^k decreases and is bounded:

$$0 \leq \delta_{k+1} \leq \delta_k \leq \delta_{k_0}, \delta_k = |y^n - y^k|, \tag{3}$$

where y^k summarizes the first k channels $y^k = \sum_{i=1}^{k} w_i x_i$,

$\forall k \in [k_0, n)$, k_0 is a constant hyper-parameter (for example, $k_0 = \lceil 0.25n \rceil$).

The bounded inequality in Equation 3 provides clues about several speculations: (1) Slimmable network [25] executable at a discrete widths set can potentially run at any width in between (if properly trained). In other words, a single neural network may execute at any width in a wide range for k from k_0 to n, since the residual error of each feature is bounded by δ_{k_0}, and decreases by increase of width k. (2) Conceptually the bounded inequality applies to any deep neural network, regardless of what normalization layers (*e.g.*, batch normalization [11] and weight normalization [17]) are used. Thus, in the following sections we mainly explore how to train a single neural network executable at arbitrary width. These networks are named as *universally slimmable networks*, or simply *US-Nets*.

3.2. Post-Statistics of Batch Normalization

However, as suggested in [25], batch normalization [11] requires special treatment because of the inconsistency between training and testing. During training, features in each layer are normalized with mean and variance of the current mini-batch feature values x_B:

$$\hat{x}_B = \gamma \frac{x_B - E_B[x_B]}{\sqrt{Var_B[x_B] + \epsilon}} + \beta, \tag{4}$$

where ϵ is a small value (e.g. 10^{-5}) to avoid zero-division, γ and β are learnable scale and bias. The values of feature mean and variance are then updated to global statistics as moving averages:

$$\begin{aligned} \mu_t &= m\mu_{t-1} + (1-m)E_B[x_B], \\ \sigma_t^2 &= m\sigma_{t-1}^2 + (1-m)Var_B[x_B], \end{aligned} \tag{5}$$

where m is the momentum (*e.g.*, 0.9), and t is the index of training iteration. We denote $\mu = \mu_T, \sigma^2 = \sigma_T^2$, assuming the network is trained for T iterations totally. During inference, these global statistics are used instead:

$$\hat{x}_{test} = \gamma^* \frac{x_{test} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta^*, \tag{6}$$

where γ^* and β^* are the optimized scale and bias. Note that after training, the Equation 6 can be reformulated as a simple linear transformation:

$$\hat{x}_{test} = \gamma' x_{test} + \beta', \gamma' = \frac{\gamma^*}{\sqrt{\sigma^2 + \epsilon}}, \beta' = \beta^* - \gamma'\mu, \tag{7}$$

and usually γ' and β' can be further fused into its previous convolution layer.

In slimmable networks, accumulating different numbers of channels results in different feature means and variances, which further leads to inaccurate statistics of shared BN [25]. Yu *et al.* introduced switchable batch normalization that privatizes γ, β, μ, σ^2 of BN for each sub-network. Although parameter γ, β can be merged after training (Equation 7), slimmable networks with shared γ and β have close performance [25].

Regarding universally slimmable networks, however, switchable batch normalization [25] is not practical for two reasons. First, accumulating independent BN statistics of all sub-networks in a US-Net during training is computationally intensive and inefficient. For example, assuming an $n-channel$ layer can adjust its width from $\lceil 0.25n \rceil$ to n, totally there are $(n - \lceil 0.25n \rceil)$ sub-networks to evaluate and $\lceil 0.25n \rceil + (\lceil 0.25n \rceil + 1) + ... + n = \mathcal{O}(n^2)$ variables of BN statistics to update in each training iteration. Second, if in each iteration we only update some sampled sub-networks, then these BN statistics are insufficiently accumulated thus inaccurate, leading to much worse accuracy in our experiments.

To this end, we adapt the batch normalization with a simple modification that can properly address the problem. The modification is to calculate BN statistics of all widths after training. Trainable parameters of US-Nets are fixed, thus all BN statistics can be computed in parallel on cluster servers. After training, we can calculate BN statistics over training samples, either as moving averages in Equation 5 or exact averages as follows:

$$\begin{aligned} m &= (t-1)/t, \\ \mu_t &= m\mu_{t-1} + (1-m)E_B[x_B], \\ \sigma_t^2 &= m\sigma_{t-1}^2 + (1-m)Var_B[x_B]. \end{aligned} \tag{8}$$

Our experiments show that exact averages have slightly better performance than moving averages.

In practice, we find it is not necessary to accumulate BN statistics over all training samples: a randomly sampled subset (*e.g.*, $1k$ images) already produces accurate estimations. With this option, calculating post-statistics of BN can be extremely fast (by default we calculate over all training samples). In experiments, we will compare the accuracy for different sample sizes. Moreover, in research or development, it is important to track the validation accuracy of a model as it trains. Although it is not supported with post-statistics of BN, we can use a simple engineering trick in training US-Nets: always tracking BN statistics of the model at largest and smallest width during training.

4. Improved Training Techniques

In this section, we describe our training algorithm for US-Nets from bottom to top. We first introduce motivations and details of *the sandwich rule* and *inplace distillation*, and then present the overall algorithm for training universally slimmable networks.

4.1. The Sandwich Rule

To train a US-Net, a natural solution is to accumulate or average losses sampled from different sub-networks. For example, in each training iteration we randomly sample n widths in the range of $[0.25, 1.0]\times$ and apply gradients back-propagated from accumulated loss. Taking a step further, the bounded inequality in Equation 3 tells that in a US-Net, performances at all widths are bounded by performance of the model at smallest width $0.25\times$ and largest width $1.0\times$. In other words, optimizing performance lower bound and upper bound can implicitly optimize all sub-networks in a US-Net. Thus, we propose *the sandwich rule* that in each iteration we train the model at smallest width, largest width and $(n-2)$ random widths, instead of n random widths. We employ this rule and show better convergence behavior and overall performance in experiments.

The sandwich rule brings two additional benefits. First, as mentioned in Section 3.2, by training smallest width and largest width, we can explicitly track the validation accuracy of a model as it trains, which also indicates the performance lower bound and upper bound of a US-Net. Second, training the largest width is also important and necessary for our next training technique: *inplace distillation*.

4.2. Inplace Distillation

The essential idea behind *inplace distillation* is to transfer knowledge inside a single US-Net from full-network to sub-networks inplace in each training iteration. It is motivated by two-step knowledge distilling [7] where a large model is trained first, then its learned knowledge is transferred to a small model by training with predicted class soft-probabilities. In US-Nets, by *the sandwich rule* we train the model at largest width, smallest width and other randomly sampled widths all together in each iteration. Remarkably, this training scheme naturally supports inplace knowledge distillation: we can directly use the predicted label of the model at the largest width as the training label for other widths, while for the largest width we use ground truth.

The proposed *inplace distillation* is simple, efficient, and general. In contrast to two-step knowledge distillation [7], *inplace distillation* is single-shot: it can be implemented inplace in training without additional computation or memory cost. And it is generally applicable to all our tasks including image classification, image super-resolution and deep reinforcement learning. For image classification, we use predicted soft-probabilities by largest width with cross entropy as objective function. In image super-resolution, predicted high-resolution patches are used as labels with either ℓ_1 or ℓ_2 as training objective. For deep reinforcement learning we take proximal policy optimization algorithm (Actor-Critic) [19] as an example. To distill, we run the policy predicted by the model at largest width as roll-outs for training other widths.

In practice, it is important to stop gradients of label tensor predicted by the largest width, which means that the loss of a sub-network will never back-propagate through the computation graph of the full-network. Also, the predicted label is directly computed in training mode if it has batch normalization. It works well and saves additional forward cost of inference mode. We tried to combine both ground truth label and predicted label as training label for sub-networks, using either constant balance of two losses or decaying balance, but the results are worse.

4.3. Training Universally Slimmable Networks

Equipped with *the sandwich rule* and *inplace distillation*, the overall algorithm for training US-Nets is revealed in Algorithm 1. For simplicity, calculating post-statistics of BN using Equation 8 is not included. It is noteworthy that: (1) The algorithm is general for different tasks and networks. (2) The GPU memory cost is the same as training individual networks thus we can use the same batch size. (3) In all our experiments, same hyper-parameters are applied. (4) It is relatively simple to implement and we show PyTorch-Style pseudo code as an example in Algorithm 1.

Algorithm 1 Training universally slimmable network M.

Require: Define *width range*, for example, $[0.25, 1.0]\times$.
Require: Define n as number of sampled widths per training iteration, for example, $n = 4$.

Initialize training settings of shared network M.
for $t = 1, ..., T_{iters}$ **do**
  Get next mini-batch of data x and label y.
  Clear gradients, $optimizer.zero\_grad()$.
  Execute full-network, $y' = M(x)$.
  Compute loss, $loss = criterion(y', y)$.
  Accumulate gradients, $loss.backward()$.
  Stop gradients of y' as label, $y' = y'.detach()$.
  Randomly sample $(n-2)$ widths, as *width samples*.
  Add smallest width to *width samples*.
  for *width* in *width samples* **do**
    Execute sub-network at *width*, $\hat{y} = M'(x)$.
    Compute loss, $loss = criterion(\hat{y}, y')$.
    Accumulate gradients, $loss.backward()$.
  end for
  Update weights, $optimizer.step()$.
end for

5. Experiments

In this section, we first present experiments on tasks of ImageNet classification, image super-resolution and deep reinforcement learning. Next we provide extensive ablation studies regarding *the sandwich rule* and *inplace distillation*. We further study US-Nets with regard to size of samples for

BN post-statistics s, width lower bound k_0, width divisor d and number of sampled widths per training iteration n. In all tables and figures, we use I-Net to denote individually trained models at different widths, S-Net to denote 4-switch slimmable networks [25] and US-Net to denote our proposed universally slimmable networks.

5.1. Main Results

ImageNet Classification. We experiment with the ImageNet [5] classification dataset with 1000 classes. Two representative mobile network architectures, MobileNet v1 [8] and MobileNet v2 [18], are evaluated. Note that MobileNet v1 is a non-residual network, while MobileNet v2 is a residual network.

Table 1. Results (top-1 error) on ImageNet classification of I-Net [8, 18], S-Net [25] and US-Net, given same width configurations and FLOPs.

Network	Width	FLOPs	I-Net	S-Net	US-Net
MobileNet v1	1.0×	569M	29.1	28.5 (0.6)	28.2 (0.9)
	0.75×	317M	31.6	30.5 (1.1)	30.5 (1.1)
	0.5×	150M	36.7	35.2 (1.5)	35.8 (0.9)
	0.25×	41M	50.2	46.9 (3.3)	44.3 (5.9)
	AVG	269M	36.9	35.3 (1.6)	**34.7 (2.2)**
MobileNet v2	1.0×	301M	28.2	29.5 (-1.3)	28.5 (-0.3)
	0.75×	209M	30.2	31.1 (-0.9)	30.3 (-0.1)
	0.5×	97M	34.6	35.6 (-1.0)	35.0 (-0.4)
	0.35×	59M	39.7	40.3 (-0.6)	37.8 (1.9)
	AVG	167M	33.2	34.1 (-0.9)	**32.9 (0.3)**

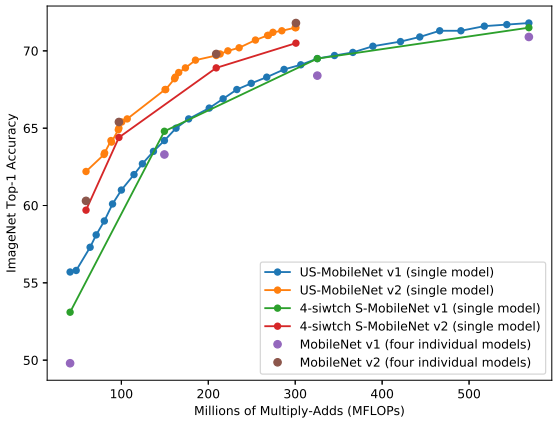

Figure 3. FLOPs-Accuracy spectrum of US-MobileNet v1 and US-MobileNet v2, compared with I-Net [8, 18] and S-Net [25].

We use default training and testing settings in [8, 18] except: (1) We only train US-Nets for 250 epochs instead of 480 epochs for fast experimentation. (2) We use stochastic gradient descent as the optimizer instead of the *RMSProp*. (3) We decrease learning rate linearly from 0.5 to 0 with batch size 1024 on 8 GPUs. We always report results with the model of final training epoch. To be fair, we use $n = 4$ for training US-Nets following Algorithm 1.

We first show numerical results in Table 1. Compared with individual models and 4-switch slimmable networks [25], US-Nets have better classification accuracy on average. In Figure 3, we show FLOPs-Accuracy spectrum of US-MobileNet v1 at widths of $[.25 : .025 : 1.0]\times$ and US-MobileNet v2 at widths of $[.35 : .025 : 1.0]\times$.

Image Super-Resolution. We experiment with DIV2K dataset [21] which contains 800 training and 100 validation 2K-resolution images, on the task of bicubic $\times 2$ image super-resolution. The network WDSR [23] is evaluated. Note that WDSR network has no batch normalization layer [11], instead weight normalization [17] is used, which requires no further modification in US-Nets. We first individually train two models at width $n = 32$ and width $n = 64$ with 8 residual blocks. We then train US-Nets that can execute at any width in $[32, 64]$, either with or without proposed *inplace distillation* in Section 4.2.

The results are shown in Figure 4. US-WDSR have slightly worse performance than individually trained models (but only 0.01 lower PSNR). The US-WDSR trained without *inplace distillation* has slightly worse performance. It is noteworthy that we use default hyper-parameters optimized for individual models, which may not be optimal for our slimmable models (e.g., learning rate, initialization, weight decay, etc).

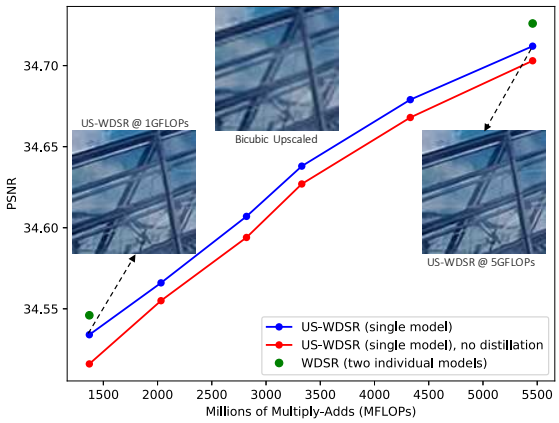

Figure 4. FLOPs-PSNR spectrum of US-WDSR and super-resolved high-resolution images under different computations. FLOPs are calculated using input size 48×48.

Deep Reinforcement Learning. We experiment with Atari game *BreakoutNoFrameskip-v4* [3] using Actor-Critic proximal policy optimization algorithm [19]. Following baseline models [19], we stack three convolutions with base channel number as 32, 64 , 32, kernel size as 8, 4, 3, stride as 4, 2, 1, and a fully-connected layer with 512 output fea-

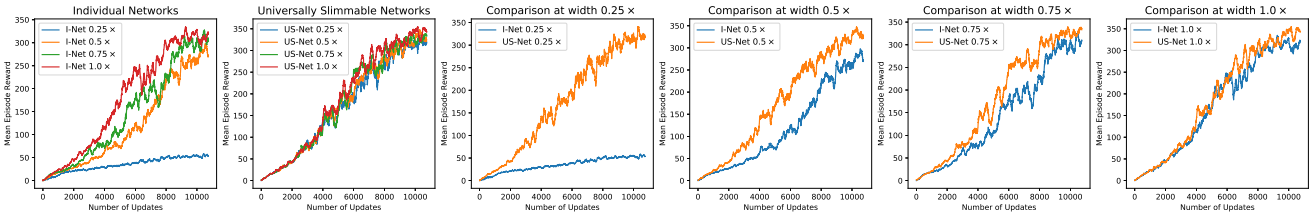

Figure 5. Mean Episode Reward with US-Net and I-Net based on actor-critic style PPO [19]. Curves are not smoothed.

Table 2. Results on ImageNet classification with different **width sampling rules** during training. We denote min as smallest width, max as largest width, $random$ as randomly sampled widths.

Sampling Rule	0.25×	0.5×	0.75×	1.0×	AVG
3 $random$	55.9	35.8	31.0	30.1	38.20
min+2 $random$	46.2	37.2	32.2	31.3	36.73
max+2 $random$	58.4	37.0	31.1	28.3	38.70
min+1 $random$+max	46.6	38.6	32.4	28.2	**36.45**

Table 3. Performance comparison (top-1 error) of different methods for calculating **post-statistics of batch normalization**. We use either moving (Equation 5) or exact (Equation 8) averages.

Size of Samples	Average	0.25×	0.5×	0.75×	1.0×
1.28M	Moving	44.4	35.8	30.6	28.2
1.28M	Exact	**44.3**	35.8	**30.5**	28.2
1k	Exact	44.4	35.8	30.6	28.2
2k	Exact	**44.3**	35.8	**30.5**	28.2

tures. The output is shared for both actor (with an additional fully-connected layer to number of actions) and critic (with an additional fully-connected layer to 1). Note that the network has no batch normalization layer.

We first individually train the model at different widths of $[0.25, 0.5, 0.75, 1.0]\times$. Then a US-Net is trained with *inplace distillation* following Section 4.2 and Algorithm 1. The performances are shown in Figure 5. From left to right, we show individually trained models, universally slimmable models (four corresponding widths are shown for comparison), and performance comparison between I-Net and US-Net at widths of $[0.25, 0.5, 0.75, 1.0]\times$. The curves show that the US-Net consistently outperforms four individually trained networks in the task of deep reinforcement learning.

We note that we include the Atari game example mainly to illustrate that our slimmable training is also applicable to CNNs for RL. We believe it is important because in more challenging RL solutions, for example *AlphaGo* [20] and *AlphaStar* [1], the inference latency and adaptive computation ability will be critical.

5.2. Ablation Study

The Sandwich Rule. We study the effectiveness of *the sandwich rule* by ablation experiments. We train four models of US-MobileNet v1 with $n = 3$ using different width sampling rules: n randomly sampled widths, $(n-1)$ randomly sampled widths plus the smallest width, $(n-1)$ randomly sampled widths plus the largest width, and $(n-2)$ randomly sampled widths plus both the smallest and largest width. Results are shown in Table 2. The US-Net trained with *the sandwich rule* has better performance on average, with good accuracy at both smallest width and largest width. Moreover, training the model at smallest width is more important than training the model at largest width as shown in the 2nd row and 3rd row of Table 2, which suggests the importance of width lower bound k_0. *Inplace distillation* is not used in all these experiments since it is not applicable to width sampling rules excluding largest width.

Inplace Distillation. Next we study the effectiveness of proposed *inplace distillation* mainly on ImageNet classification. The results of image super-resolution (both with and without *inplace distillation*) and deep reinforcement learning (with *inplace distillation*) are already shown in Figure 4 and Figure 5. We use the same settings to train two US-MobileNet v1 models either with or without *inplace distillation*, and show the comparison in Figure 6. *Inplace distillation* significantly improves overall performance at no cost. We suppose it could be an essential component for training slimmable networks.

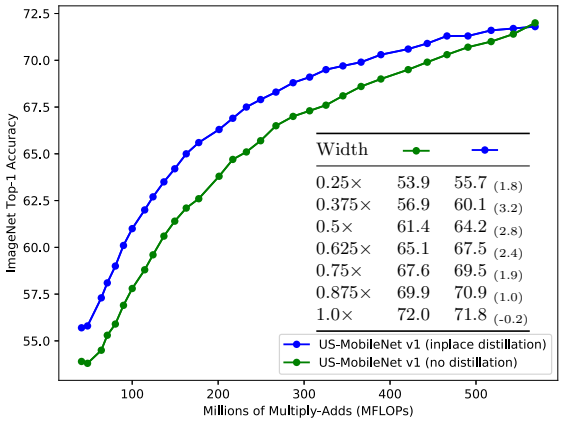

Figure 6. FLOPs-Accuracy spectrum of two US-MobileNet v1 models trained either with or without *inplace distillation*.

Post-Statistics of Batch Normalization. We further study post-statistics for batch normalization in US-Nets. We update BN statistics after training US-MobileNet v1 when all weights are fixed. We then compute BN statistics using four methods: moving average over entire training set, exact average over entire training set, exact average over randomly sampled $1k$ training subset, and exact average over

randomly sampled $2k$ training subset. Table 3 shows that exact averaging has slightly better performance and a small subset produces equally accurate BN statistics. It indicates that calculating post-statistics of BN can be very fast.

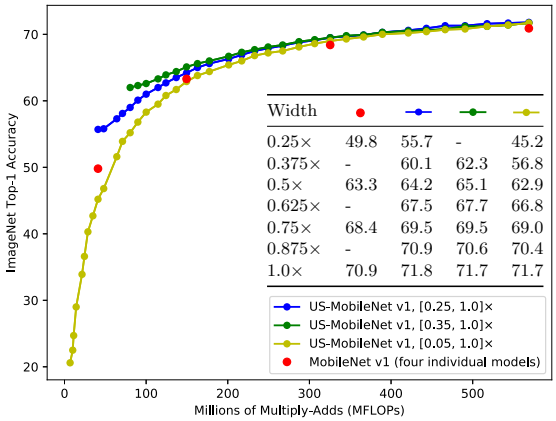

Figure 7. FLOPs-Accuracy spectrum of three US-MobileNet v1 models with different **width lower bounds**.

Width Lower Bound k_0. Width lower bound k_0 is of central importance in the bounded Equation 3. Although it is usually enough to adjust a model between width $0.25\times$ and $1.0\times$, we are interested in how the width lower bound affects overall performance. We train three US-MobileNet v1 models with different width lower bounds k_0 as $0.25\times$, $0.35\times$, $0.05\times$ and show results in Figure 7. It reveals that the performance of a US-Net is grounded on its width lower bound, as suggested in our analysis in Section 3.1.

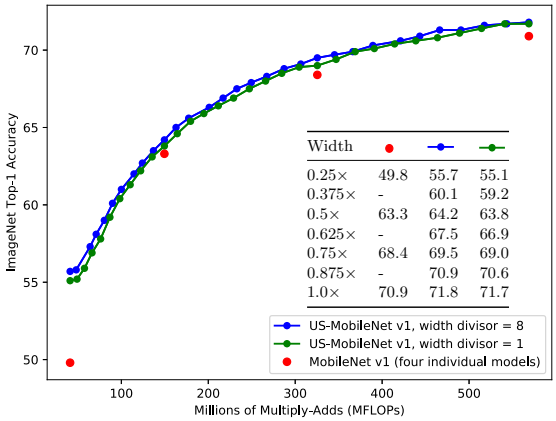

Figure 8. FLOPs-Accuracy spectrum of two US-MobileNet v1 models with different **width divisors**.

Width Divisor d. Width divisor is introduced in MobileNets [8, 18] to floor the channel number approximately as $\lfloor nr/d \rfloor * d$, where n is base channel number, r is width multiplier, d is width divisor[2]. To exactly match FLOPs of MobileNets and have a fair comparison, by default we follow MobileNets and set width divisor $d = 8$. This results in the minimal adjustable channel number as 8 instead of 1, and slightly benefits overall performance, as shown in Figure 8. In practice, with $d = 8$ the US-Nets already provide enough adjustable widths. Also in many hardware systems, matrix multiplication with size dividable by $d = 8, 16, ...$, may be as fast as a smaller size due to alignment of processing unit (*e.g.*, warp size in GPU is 32).

[2]Details are in hyperlink TensorFlow Models (PDF required).

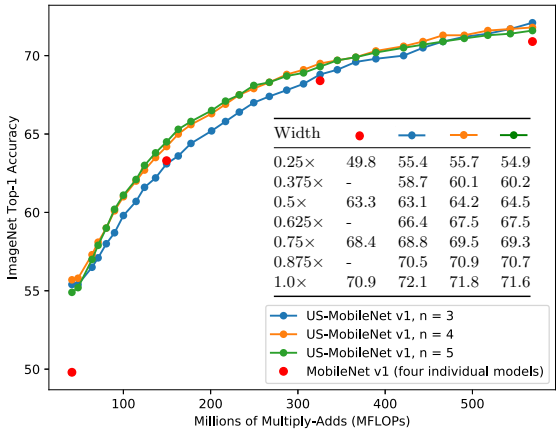

Figure 9. FLOPs-Accuracy spectrum of two US-MobileNet v1 trained with different **numbers of sampled widths per iteration**.

Number of Sampled Widths Per Iteration n. Finally we study the number of sampled widths per training iteration. It is important because larger n leads to more training time. We train three US-MobileNet v1 models with n equal to 3, 4 or 5. Figure 9 shows that the model trained with $n = 4$ has better performance than the one with $n = 3$, while $n = 4$ and $n = 5$ achieve very similar performances. By default, in all our experiments we use $n = 4$.

6. Discussion

We mainly discuss three topics in this section, with detailed results shown in the supplementary materials.

First, for all trained US-Nets so far, the width ratio is uniformly applied to all layers. Can we train a nonuniform US-Net where each layer can independently adjust its own ratio using our proposed methods? This requirement is especially important for related tasks like network slimming. Our answer is YES and we show a simple demonstration on how the nonuniform US-Net can help in network slimming.

Second, perhaps the question is naive, but are deep neural networks naturally slimmable? The answer is NO, a naively trained model fails to run at different widths even if their BN statistics are calibrated.

Third, in slimmable networks [25], private scale and bias are used as conditional parameters for each sub-network, which brings performance gain slightly. In US-Nets, by default we share scale and bias. We also propose an option that mimics conditional parameters: averaging the output by the number of input channels.

References

[1] Alphastar: Mastering the real-time strategy game starcraft ii. 7

[2] Hessam Bagherinezhad, Maxwell Horton, Mohammad Rastegari, and Ali Farhadi. Label refinery: Improving imagenet classification through label progression. *arXiv preprint arXiv:1805.02641*, 2018. 3

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. 6

[4] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015. 3

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009. 6

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 3

[7] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2, 3, 5

[8] 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. 3, 6, 8

[9] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017. 1

[10] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 304–320, 2018. 1

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 1, 2, 3, 4, 6

[12] Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016. 3

[13] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. *arXiv preprint arXiv:1701.00299*, 2017. 1

[14] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015. 3

[15] 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

[16] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 3

[17] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016. 2, 4, 6

[18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *arXiv preprint arXiv:1801.04381*, 2018. 3, 6, 8

[19] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 5, 6, 7

[20] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017. 7

[21] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, Lei Zhang, Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, Kyoung Mu Lee, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1110–1121. IEEE, 2017. 6

[22] Linjie Yang, Yanran Wang, Xuehan Xiong, Jianchao Yang, and Aggelos K Katsaggelos. Efficient video object segmentation via network modulation. *arXiv preprint arXiv:1802.01218*, 2018. 1

[23] Jiahui Yu, Yuchen Fan, Jianchao Yang, Ning Xu, Xinchao Wang, and Thomas S Huang. Wide activation for efficient and accurate image super-resolution. *arXiv preprint arXiv:1808.08718*, 2018. 6

[24] Jiahui Yu and Thomas Huang. Network slimming by slimmable networks: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*, 2019. 3

[25] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018. 1, 2, 3, 4, 6, 8

[26] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017. 3