Switchable Precision Neural Networks

Luis Guerra[1] Bohan Zhuang[2] Ian Reid[2] Tom Drummond[1]
[1]Monash University [2]The University of Adelaide

Abstract

Instantaneous and on demand accuracy-efficiency trade-off has been recently explored in the context of neural networks slimming. In this paper, we propose a flexible quantization strategy, termed Switchable Precision neural Networks (SP-Nets), to train a shared network capable of operating at multiple quantization levels. At runtime, the network can adjust its precision on the fly according to instant memory, latency, power consumption and accuracy demands. For example, by constraining the network weights to 1-bit with switchable precision activations, our shared network spans from BinaryConnect to Binarized Neural Network, allowing to perform dot-products using only summations or bit operations. In addition, a self-distillation scheme is proposed to increase the performance of the quantized switches. We tested our approach with three different quantizers and demonstrate the performance of SP-Nets against independently trained quantized models in classification accuracy for Tiny ImageNet and ImageNet datasets using ResNet-18 and MobileNet architectures.

1. Introduction

Deep Neural Networks (DNNs) have achieved great success in a wide range of vision tasks, such as image classification [10], semantic segmentation [22] and object detection [29]. However, the large model size and expensive computational complexity remain great obstacles for many applications, especially on some constrained devices with limited memory and computing resources. Network quantization is an active field of research focusing on alleviating such issues. In particular, [9, 15, 28] set the foundations for 1-bit quantization, while [16, 50] for arbitrary bitwidth quantization. Progressive quantization [2, 1, 53, 33], loss aware-quantization [13, 49], improved gradient estimators for non-differentiable functions [21] and RL-aided training [20], have focused on improved training schemes, while mixed precision quantization [36], hardware-aware quantization [37] and architecture search for quantized models [34] have focused on alternatives for standard quantized models. However, these strategies are exclusively focused on improving the performance and efficiency of static networks.

Dynamic routing networks try to provide improvements using an alternative approach. By performing computations conditioned on the inputs, the networks are capable of saving resources by executing just the sufficient amount of operations required to map the input to the desired output. Popular strategies include skipping convolutional layers [39, 6, 41] based on the input data complexity and early classifiers [23].

Our proposed approach taken by slimmable neural networks [44] falls in the category of dynamic networks, but following a different principle, aiming to provide on demand trade-offs, rather than input dependant. To the best of our knowledge, dynamic quantization of DNNs has not been explored in the literature. Slimmable networks provide width (number of channels) *switches* that allow to perform inference utilizing only sections of the network according to on-device demands and resource constraints. Similar to slimmable networks, we attempt to provide the first approach by developing a network whose weights and activations can be quantized at various precision at runtime, permitting instant and adaptive accuracy-efficiency trade-off, which is termed *switchable precision*. In particular, 1-bit DNNs are both an interesting and challenging case of our SP-Net. With binary weights, we can train a shared network ranging from BinaryConnect [9] and Binarized Neural Network [15], where the inner product can be efficiently implemented using summation or bit operations according to on-device resource constraints. Furthermore, our PS-Nets frequently yields higher accuracy than individually trained quantized networks.

However, different precision switches are difficult to optimize as a whole. And we summarize the reasons in two parts. On one hand, during training, batch normalization (BN) layers use the current batch statistic to perform intermediate feature maps normalization, while estimating the global statistics by accumulating a running mean and running variance, which are used as the replacement during testing. However, the batch statistics of each precision switch are different. As a result, the discrepancy of feature mean and variance across different switches leads to inaccurate accumulated statistics of BN layers. To solve this problem, we follow [44] to train a switchable precision network by using independent BN parameters for each switch, named *switchable batch normalization (S-BN)*.

On the other hand, we conjecture that simultaneously optimizing multiple quantized switches will reflect in the loss manifold by progressively quantizing the loss surface, and the higher precision switches will assist the lower precision ones to achieve a less noisy and smoother convergence, potentially leading them to a better minima. Conversely, the network will converge to only minimas which perform well at all the bitwidths potentially harming the performance of some of the switches, particularly the higher precision ones. During training of SP-Nets, the gradients of each switch are combined before running an optimizer step. However, there is no explicit mechanism impeding the individual switches from moving in distinct directions. In order to encourage the different switches to move in approximately the same direction, we propose a self-distillation strategy, where the full precision switch provides a guiding signal for the rest of the switches. Specifically, only the teacher full-precision switch sees the ground-truth while the student low-precision switches are trained by distilling knowledge from the full-precision teacher.

In order to increase the flexibility of our model, in addition to equipping the network with switchable precision representations, we extend our approach to 3 different quantizers and equip them with slimming (width switchable) capability.

Our contributions are summarized as follows:

- We leverage S-BN to train a shared network executable at different bitwidths on weights and activations according to runtime demands.
- We propose a self-distillation scheme to jointly train the full-precision teacher switch and the low-precision student switches. By doing so, the full-precision switch provides a guiding signal to significantly improve the performance of the low-precision switches.
- We investigate the effectiveness of our method on uniform and non-uniform quantization with various quantizers through extensive experiments on image classification.

2. Related Work

Network slimming. Slimmable networks introduced by [44] developed a procedure useful for training a DNN with switchable widths. The motivation is to provide instant and on demand accuracy-efficiency trade-offs. It was further generalized by [43] allowing to efficiently train a network continuously slimmable, executable at any arbitrary width. Moreover, non-uniform slimming was introduced, allowing for layer-wise width selection. The training principle behind network slimming has found applications in the fields of pruning [42], network distillation [47], architecture search [4], adaptive inference [31] and finally, in our work, network quantization.

Network quantization. Quantization based methods represent the network weights and/or activations with very low precision, thus yielding highly compact DNN models compared to their floating-point counterparts with considerable memory and computation savings. BNNs [15, 28] propose to constrain both weights and activations to binary values (i.e., $+1$ and -1), where the multiplication-accumulations can be replaced by purely xnor and $\mathrm{popcount}$ operations, which are in general much faster. However, BNNs still suffer from significant accuracy decrease compared with the full precision counterparts. To narrow this accuracy gap, ternary networks [19, 51] and even higher bit fixed-point quantization [50, 48] methods are proposed.

In general, quantization approaches target at tackling two main problems. On one hand, some works target at designing more accurate quantizer to minimize information loss. For the uniform quantizer, works in [7, 17] explicitly parameterize and optimize the upper and/or lower bound of the activation and weights. To reduce the quantization error, non-uniform approaches [26, 46] are proposed to better approximate the data distribution. In particular, LQ-Net [46] proposes to jointly optimize the quantizer and the network parameters. On the other hand, because of the non-differentiable quantizer, some literature focuses on relaxing the discrete optimization problem. A typical approach is to train with regularization [13, 49, 2, 1, 33, 8], where the optimization problem becomes continuous while gradually adjusting the data distribution towards quantized values. Apart from the two challenges, with the popularization of neural architecture search (NAS), Wang *et al.* [38] further propose to employ reinforcement learning to automatically determine the bit-width of each layer without human heuristics.

Knowledge distillation. Knowledge distillation (KD) is a general approach for model compression, where a powerful wide/deep teacher distills knowledge to a narrow/shallow student to improve its performance [12, 30]. In terms of the definition of knowledge to be distilled from the teacher, existing models typically use teacher's class probabilities [12] and/or intermediate features [30, 45]. KD has been widely used in many computer vision tasks [40, 11]. Moreover, there are some works [53, 24, 27] study the combination of KD and quantization, where the full-precision model provides hints to guide the low-precision model training and significantly improves the performance of the low-precision networks. Different from the previous literature, we propose a self-distillation strategy to improve our SP-Net training. Specifically, only the full-precision switch sees the ground-truth while the low-precision switches are learnt by distilling from the full-precision teacher.

3. Preliminaries

The optimization problem of traditional Quantized Neural Networks (QNNs) aims at minimizing an objective function $\mathcal{L}$ given a set of trainable weights W, which take values from C_w, a predefined discrete set typically referred as *codebook*. A common QNN training procedure involves storing a real-valued version of W. During inference, the real W is quantized using a predetermined pointwise quantization function $\mathrm{Quant_w} : \mathbb{R} \mapsto \mathrm{C_w}$. The weights are updated during the optimization process by estimating the gradients w.r.t. the real-valued copy. Additionally, internal network activations can be optionally quantized with their own codebook C_a by $\mathrm{Quant_a} : \mathbb{R} \mapsto \mathrm{C_a}$. Given independent bitwidths $bits_w$ and $bits_a$ for the network weights and activations, $|C_w| = 2^{bits_w}$ and $|C_a| = 2^{bits_a}$. Finally, a quantized convolutional layer with K filters is computed as follows:

$$\mathbf{O}_k = \sum_{i=1}^{I} \mathrm{Quant_a}(\mathbf{A}_{\mathrm{i},:,:}) * \mathrm{Quant_w}(\mathbf{F}_{\mathrm{k,i},:,:}), \qquad (1)$$

where $\mathbf{F}_k \in \mathbb{R}^{I \times h_f \times w_f} \subset W$ is the k-th convolutional filter. I, h_f and w_f denote the number of input channels, height and width of the filters, respectively. $\mathbf{A} \in \mathbb{R}^{I \times h_{in} \times w_{in}}$ and $\mathbf{O}_k \in \mathbb{R}^{h_{out} \times w_{out}}$ denote the input activations and output pre-activations of the filter, where h_{in}, w_{in}, h_{out}, w_{out} represent the height and width of the input and output feature maps, respectively.

In the context of QNNs, multiple quantizers and gradient estimators have been proposed in the literature [3, 21, 50, 5, 25]. For our SP-Net, we stick to common ones, which will be described in Sec. 3.1. Three non-linear quantizers for the activations are implemented. For the weights, the Tanh-based quantizer is used, unless specified otherwise. The motivation to use each quantizer will be explained in each corresponding subsection.

3.1. Straight-Through Estimator (STE) and Base Quantizer

The STE proposed in [3] allows to estimate gradients through the non-differentiable functions round and sign, making the k-bit quantizers employed compatible with the backpropagation algorithm. The STE operates by passing the output gradients unaltered to the input, it is equivalent to the derivative of an $\mathrm{identity}$ mapping on the inputs. Commonly, the gradients for inputs outside of the range $[-1, 1]$ are suppressed and the derivative of the quantization function becomes equivalent to the derivative of the $\mathrm{hardtanh}$ function.

The k-bit quantizers described in the next sections share the same base quantization function, $Q(\cdot)$:

$$Q(x) = \begin{cases} \mathrm{sign}(x) & \text{if k} = 1 \\ \frac{1}{2^k-1}\mathrm{round}((2^k-1)x) & \text{if } k > 1 \end{cases}. \qquad (2)$$

During backpropagation, we use the STE:

$$\frac{d\mathcal{L}}{dx} \approx \frac{d\mathcal{L}}{dQ}\mathbb{1}_{|x|\leq 1}. \qquad (3)$$

3.1.1 Tanh-Based Quantizer

[50, 5] proposed to use different quantizers for weights and activations. Weight quantizers approximate the hyperbolic tangent function ($\tanh$), constraining the weights to $[-1, 1]$. However, $\tanh$ is associated with the vanishing gradient problem, thus, it is attractive for activations quantizers to approximate the popular ReLU activation function as described in the next section, constraining the activations to the positive range.

The $\tanh$ is first used to project the input to range $[-1, 1]$ in order to reduce the impact of large values. The quantizer is defined as follows:

$$\mathrm{TanhQuant}(x) = 2 \cdot Q(\frac{\tanh(x)}{2 \cdot \max(|\tanh(x)|)} + \frac{1}{2}) - 1, \quad (4)$$

where $Q(\cdot)$ hereafter is defined in Eq. (2).

It is appealing to train a network with activations switchable down to 1-bit with permissible values $\{-1, 1\}$ since bit operations can be performed given that weights are 1-bit as well. Therefore, when training these type of networks, $\mathrm{TanhQuant}$ quantizer is used on both weights and activations to allow the activations of the intermediate switches to lie in the range $[-1, 1]$. For this same reason, the layers were re-ordered as described in [28] (typical layers ordering is QuantConv→BN →ReLU→QuantConv, while layers reordered are QuantConv→ReLU→BN→QuantConv).

3.1.2 ReLU-Based Quantizer

As mentioned in the previous section, [50, 5] employ a ReLU approximation quantizer for the activations with no layer re-ordering. The method proposed by [50] will be employed here, which consists of simply clipping the activations followed by the base quantizer:

$$\mathrm{ReLUQuant}(x) = Q(\mathrm{clip}(x, 0, 1)). \qquad (5)$$

In particular, [5] proposed both uniform and non-uniform spacing between the codebook elements. They first clipped the activations to the range $[0, v]$, where v is some predetermined value, and the codebook for both cases, uniform and non-uniform, is obtained from the network internal statistics. In our tests with $\mathrm{ReLUQuant}$ quantization, we simply constrain the activations to the range $[0, 1]$ with uniform quantization. In the logaritmic quantizer described in the next section, non-uniform quantization is used.

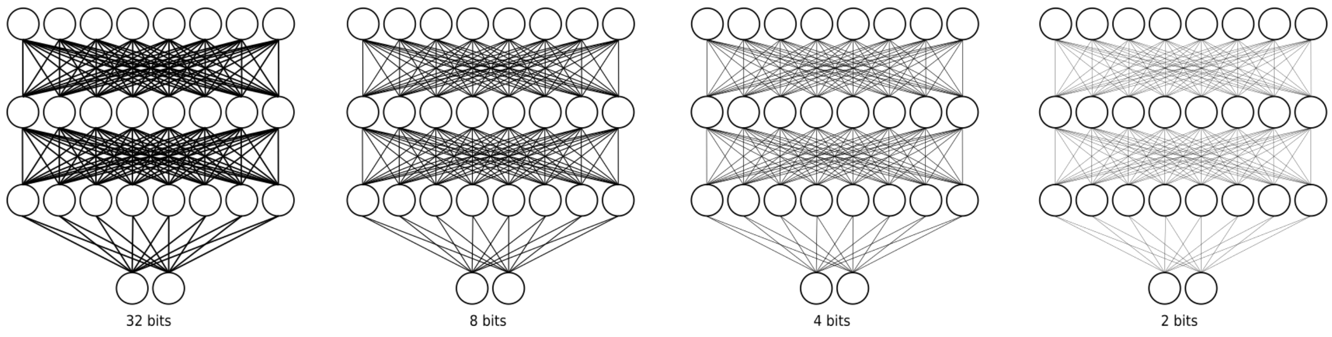

Figure 1: Overview of the proposed approach. In SP-Nets multiple precision switches share a common architecture, making it capable of adjusting the precision of their representations on the fly, granting devices and end-users real-time control over the network performance (thinner connections indicate reduced bitwidth).

3.1.3 Logarithmic Quantizer

In full-precision neural networks, the weights and activations have non-uniform distributions [25]. Taking advantage of that fact, the authors used logarithmic representations, both in weights and activations, achieving higher classification accuracy at the same resolution than uniform quantization schemes at an expense of higher implementation and computation complexity. In the original paper, their $\mathrm{LogQuant}$ layer contains a global Full Scale Range (FSR) parameter which is reported in the paper. Additionally, each layer has its own FSR. In our SP-Net, we use a variation of their $\mathrm{LogQuant}$ layer in order to avoid the FSR parameter. Our modified $\mathrm{LogQuant}$ quantizer is defined as follows:

$$\mathrm{LogQuant}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 2^{\widehat{x}} \cdot x & \text{otherwise} \end{cases}, \quad (6)$$

where

$$\hat{x} = \mathrm{rescale}(Q(\mathrm{normalize}(|\log(x)|))), \quad (7)$$

$$\mathrm{normalize}(x) = (x - \min(x))/(\max(x) - \min(x)), \quad (8)$$

$$\mathrm{rescale}(x) = x \cdot (\max(x) - \min(x)) + \min(x), \quad (9)$$

$$s(x) = \mathrm{sign}(x). \quad (10)$$

Similarly to Sec. 3.1.1, two-sided logarithmic quantization (positive and negative) can be used in order to have activations switchable down to 1-bit. Depending on the choice of activations quantization (one-sided or two-sided), layer re-ordering should be taken into account. In our experiments, logarithm base-2 was used, however, base-$\sqrt{2}$ could provide higher accuracy.

4. Switchable Precision Neural Networks

SP-Nets generalize QNNs, where the learnable weights W are optimized for multiple codebooks $C_w[n]$ and $C_a[n]$ of variable cardinality. The permissible values of the N codebooks are determined by the choice of quantizers $\mathrm{Quant_w}$ and $\mathrm{Quant_a}$, while their cardinality is determined by the bitwidths $bits_w[n]$ and $bits_a[n]$.

DNNs at their current state are not naturally precision switchable as empirically demonstrated in Sec. 6.3. Therefore, we appeal to S-BN, a technique used to train slimmable neural networks, described in detail in Sec. 4.1. Then in Sec. 4.2, we extend our SP-Net to mixed slimmable SP-Net.

4.1. Switchable Batch Normalization (S-BN)

DNNs at their current state are not naturally slimmable [43] nor of switchable precision due to the inconsistent behavior of batch normalization layers during training and inference. During training, BN layers utilize the current batch x_b mean $E_b[x_b]$ and variance $Var_b[x_b]$ to perform intermediate feature maps normalization while accumulating a running mean u and running variance σ^2, which is used as replacement during test.

In a naive SP-Net, the mini-batch local statistics of each quantization switch are different, however, all the switches will contribute to the accumulated global mean u and variance σ^2, thus, enabling the network to train properly, but resulting in inaccurate test inference.

S-BN layers equip regular BN layers with private $u[n]$ and $\sigma^2[n]$ for each of the N switches. The overhead of the additional parameters is negligible since BN parameters account for an insignificant portion of the total amount. It is also negligible in terms of run-time complexity since they involve no additional operations.

Additionally, BN layers count with two learnable parameters β and γ used to provide the layer with the capacity of performing a linear mapping. Unlike u and σ^2, β and γ can be updated by all the switches. Providing with private versions of them is not as crucial, since they allow the network to learn, but it yields an additional accuracy boost [43]. Furthermore, they generate no additional overhead since they can be merged with $u[n]$ and $\sigma^2[n]$ after training.

In Algorithm 1, we illustrate the use of S-BN in one SP-Net training iteration.

Algorithm 1: One SP-Net iteration using S-BN

```
Require: SP-Net $M$. Lists of weights and activations
  bitwidths $bits\_w$, $bits\_a$.
Get mini-batch data $x$ and ground-truth $y$;
for bit_w in bits_w do
    for bit_a in bits_a do
        Switch BN parameters in $M$ for the current
          bitwidths;
        Set $bit\_w$, $bit\_a$ in $M$;
        Forward pass using input $x$;
        Compute loss w.r.t. to $y$;
        Compute gradients using STE;
    end
end
Update weights using accumulated gradients;
```

4.2. Slimmable SP-Net

Network slimming [44] relies on S-BN in order to allow each layer of the network to operate at different widths. Given the current width multiplier $width \in [0, 1]$, a slimmable convolutional layer with K filters computes only the first $\lceil K * width \rceil$ ones.

Network slimming and quantization are complementary techniques and effortlessly work along without technical complications. Therefore, we can train a single shared network with switchable width and precision to increase the flexibility. A single slimmable/SP network can be trained in the cloud and distribute particular switches to different deployment systems based on their specific hardware capabilities, where they can be further slimmed and quantized instantaneously on demand. In Sec. 6.2, we provide a comparison of a slimmable SP-Net with the corresponding individually trained switches.

Although the benefits of a SP-Net in terms of power and speed improvements are evident by performing dot-product operations on quantized vectors, the benefits on memory footprint may not be so apparent, given that the full precision weights must be stored on non-volatile memory at all time, regardless of the active quantization switch. However, during operation, a network clone is stored in the RAM of the processor in order to provide quick access, therefore weight quantization only takes place once every time a new switch is requested and the quantized clone is kept in volatile memory. By quantizing activations, additional RAM savings can be obtained.

5. Self-Distillation

Knowledge distillation is a common strategy used to provide a stronger training signal, typically from a large network to smaller one in a teacher-student scheme. In a similar fashion to knowledge distillation, by simultaneously optimizing multiple quantized switches, implicitly the high precision switches are providing guidance to the noisier low precision updates. However, this behavior is not explicitly encouraged. Therefore, in this section we present a complementary distillation mechanism, denominated *self-distillation*, where only the full-precision switch sees the ground-truth, while the low bitwidth switches try to match the internal representation as well as the output distribution of the full-precision switch. The strategy formulated allows the full-precision switch to guide the optimization process with a significant amount of information flow across all the switches.

In US-Net [43] gradients are prevented from flowing from the sub-networks to the largest width switch. Formally, to mimic the outputs, similarly to US-Net, we use the KullbackLeibler divergence as distance measure on the output distributions p_r and p_q of the full-precision switch and the quantized active switch. Let $SG(\cdot)$ denote the stop-gradient function, the output mimic loss is:

$$\mathcal{L}_{out} = D_{KL}(SG(p_r) \| p_q). \tag{11}$$

Additionally, to create the guidance signal, [53] proposes a hint-based training strategy by comparing the intermediate feature maps between the full-precision teacher and the low-precision student. Similarly, let f_r and f_q denote the internal feature maps (*i.e.*, pre-activations) of the full-precision switch and the quantized active switch, the internal representations guidance loss becomes:

$$\mathcal{L}_f = \|f_r - f_q\|_2^2. \tag{12}$$

And in this case, we do not stop the gradients flowing to f_r and quantize f_r.

Finally, the loss for the quantized switches is their combination, while for the full precision switch is simply the regular cross-entropy classification loss:

$$\mathcal{L}_q = \alpha_1 \mathcal{L}_{out} + \alpha_2 \mathcal{L}_f, \tag{13}$$

$$\mathcal{L}_r = \mathcal{L}_{cross-entropy}(p_r, y), \tag{14}$$

where y denotes the ground-truth labels, and α_1 and α_2 are the scaling coefficients to balance $\mathcal{L}_{out}$ and $\mathcal{L}_f$, respectively.

6. Experiments

In this section, we test 2 different achitectures, ResNet [10] and MobileNet [14], for the task of image classification on the ImageNet (ILSVRC-2012) [32] and Tiny ImageNet datasets with 3 different quantizers: Tanh-based quantization [50], ReLU-based quantization [50, 5] and non-uniform Logarithmic quantization [25]. Experiments on ImageNet have been included in the supplementary material.

We implement our SP-Nets using Pytorch. For our ImageNet experiments, we use a regular (*i.e.*, 1 single switch) full-precision pre-trained model, we then replicate the BN parameters across all the S-BN switches and fine-tune the SP-Net model. We use the standard pre-processing and augmentation as reported by [10]. For training with the pre-trained full-precision model, we use Adam [18] optimizer with an initial learning rate of $1e^{-4}$ and decrease it by a factor of 10 at epochs 15 and 20 with a total of 25 epochs. For our Tiny ImageNet experiments, all the networks are learned from scratch using SGD optimizer with initial learning rate of 0.1 and decreased by a factor of 10 every 30 epochs during 100 epochs. As in common practice, the first and last layers are not quantized [28].

Implementation Note. The activations of the full-precision switch, although are not quantized, in some cases they must still be clipped to the same range of the quantized switches, as it can lead the switch to diverge.

6.1. Evaluation on ImageNet

Evaluation on ResNet-18 In Table 1, we compare the Top-1 accuracy for the edge bitwidth cases in both weights and activations for a ResNet-18 architecture. Our SP-net achieved very close accuracy to the independently trained QNNs with the full-precision switch being the most affected, while providing increased efficiency-accuracy flexibility.

Table 1: Top-1 accuracy (%) for ResNet-18 on ImageNet.

Weight	Activation	SP-Net	Independent
2	2	62.4	**62.6**
2	32	65.3	**65.8**
32	2	65.1	**65.3**
32	32	68.1	**69.5**

Evaluation on MobileNet. The MobileNet architecture makes use of the depthwise separable and pointwise convolutions without residual connections drastically reducing the number of parameters without a major decrease in accuracy. However, it is very sensitive to quantization, specially on the activations. [35] re-designs the MobileNet architecture to be more quantization friendly. For our MobileNet SP-Net, we follow their architecture. The re-design involves simple layers replacement and re-ordering. In Table 2, we report the results of different bitwidths for weights and activations using our SP-Net MobileNet and independent networks with the modified architecture. As can be observed, the impact of switchable precisions on MobileNet is higher than on ResNet-18.

Table 2: Top-1 accuracy (%) for MobileNet on ImageNet.

Weight	Activation	SP-Net	Independent
8	8	59.5	**63.5**
8	32	66.0	**69.0**
32	8	59.6	**63.6**
32	32	66.1	**69.7**

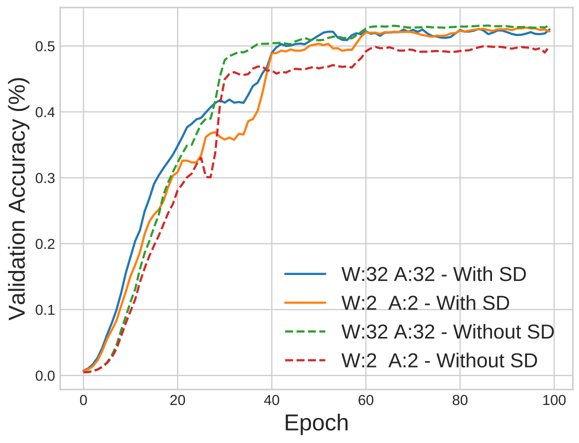

Figure 2: Top-1 accuracy for SP-Net ResNet-18 with and without self-distillation (SD) on Tiny ImageNet

6.2. Evaluation on Tiny ImageNet

The Tiny ImageNet dataset is a downsampled ImageNet version (64×64) with 100K images for training and 10K for validation, spread across 200 classes. For the experiments on Tiny ImageNet, we used the same ResNet-18 architecture as that for ImageNet except for the first layer, whose filter size is $3 \times 3 \times 64$ with $stride = 1$ and $padding = 1$.

Table 3: Top-1 accuracy (%) for a 1-bit SP-Net ResNet-18 on Tiny ImageNet.

Weight	Activation	SP-Net	Independent
1	1	**43.5**	40.0
1	3	**48.8**	46.3
1	8	**48.8**	47.0
1	32	49.0	**49.1**

Comparison of Different Quantizers.

As explained in Sec. 3.1, different quantizers can be used in different scenarios. $\mathrm{ReLUQuant}$ quantizer is used in activations in general. When it is desirable to have activations quatizable to 1-bit, $\mathrm{TanhQuant}$ quantizer is used. $\mathrm{LogQuant}$ quantizer is chosen when higher accuracy is desired with the same bitwidth, at an expense of higher complexity.

In Table 4, 4 quantizer configurations were tested for SP-Nets with multiple weight and activation switches . In our results, Tanh-based configuration obtained the lowest accuracy across all the switches and ReLU-based configura-

Table 4: Comparison of different quantizers on a SP-Net for ResNet-18 on Tiny ImageNet. The networks were slimmed with a factor of 0.25 to magnify the effect of the quantizers. [1] ReLU-based: TanhQuant quantizer on weights, ReLUQuant on activations. [2] Tanh-based: TanhQuant quantizer on both weights and activations *with layer re-ordering*. [3]Logarithmic 1: TanhQuant quantizer on weights and LogQuant quantizer on activations. [4]Logarithmic 2: LogQuant quantizer on both on weights and activations.

Width	Bitwidth		Quantizer Accuracy(%)			
	Weight	Activation	ReLU-based[1]	Tanh-based[2]	Logarithmic 1[3]	Logarithmic 2[4]
0.25	2	2	**32.5**	29.9	29.1	28.9
0.25	2	4	**34.5**	32.7	34.0	33.9
0.25	2	8	**34.8**	32.8	34.1	34.1
0.25	2	32	34.3	32.8	34.4	**35.5**
0.25	4	2	**34.1**	33.7	33.3	32.5
0.25	4	4	**37.4**	36.8	37.3	36.7
0.25	4	8	37.4	36.6	**37.5**	37.0
0.25	4	32	37.5	36.6	37.6	**37.7**
0.25	8	2	**34.4**	33.0	33.9	32.6
0.25	8	4	**37.3**	36.5	37.2	36.4
0.25	8	8	37.3	36.9	**37.4**	36.9
0.25	8	32	37.5	37.0	37.6	**37.7**
0.25	32	2	**36.3**	33.6	33.5	32.8
0.25	32	4	**37.3**	36.5	37.2	36.8
0.25	32	8	37.2	37.0	**37.4**	36.9
0.25	32	32	**37.7**	37.0	37.5	37.5

tion frequently obtained the best results. Logarithmic based configurations ocasionally performed better, particularly at higher activations bitwidths. Logarithmic based configurations were expected to produce the best results, however our FSR-free modified version and logarithm base choice may have harmed their performance. All the networks were slimmed by a factor of 0.25 to magnify the effect of the quantizers.

1-bit SP-Net. We trained a network with a single bit per weight and switchable precision activations. Our network spans from the popular BinaryConnect, where dot products are computed using only summation, to Binarized Neural Networks (BNNs), where activations are quantized to 1-bit and dot products are computed using xnor and popcount operations.

Training a network with precision switchable down to 1-bit per activation is particularly challenging since it causes a significant decrease in performance across all the bitwidths when using a ReLU-based quantization. Therefore, networks with activations switchable to 1-bit are trained with Tanh-based quantization with layer re-ordering. Weights are trained by simply using the sign function with STE.

The results in Table 3 show that our 1-bit SP-Net surpasses the independently trained counterparts by a large margin when the activations are in extremely low-precision.

Slimmable SP-Net. Our results for slimmable SP-Net in Table 5 demonstrate the flexibility of our network, however they show no clear accuracy advantage over independently trained networks. We tested widths of 0.25, 0.5 and 1.0, with slimmable SP-Net and independently trained networks constantly outperforming each other. However, it is worth noting that our slimmable SP-Net network can simultaneously switch width and bitwidth on demand.

Table 5: Top-1 accuracy (%) for a slimmable SP-Net ResNet-18 on Tiny ImageNet.

Width	Weight	Activation	Slim/SP-Net	Independent
0.25	2	2	**30.3**	26.5
0.25	2	32	**36.5**	34.2
0.25	32	2	**35.6**	32.8
0.25	32	32	**39.7**	37.8
0.5	2	2	40.9	**41.3**
0.5	2	32	44.8	**49.7**
0.5	32	2	**46.1**	45.8
0.5	32	32	49.0	**52.8**
1.0	2	2	**47.8**	47.4
1.0	2	32	50.5	**54.4**
1.0	32	2	**51.8**	50.7
1.0	32	32	53.0	**56.4**

Self-Distillation. Our self-distillation method in Sec. 5 proved to be very effective, by improving the accuracy across all the switches, as can be observed in Table 6. Additionally it can be observed that the validation performance of intermediate switches is superior to the performance of the full-precision one, however we credit this to over-fitting, since the training accuracy is higher for the full precision switches. We also compared the performance of the distillation losses, confirming our feature maps distillation loss $\mathcal{L}_f$ is indeed benefiting the overall performance of the network. The value of $\mathcal{L}_f$ is expected to be several orders of magnitude larger than $\mathcal{L}_{out}$, since it is dictated by the number of layers and the size of the intermediate feature maps. We set the hyper-parameters $\alpha_1 = 1$ and $\alpha_2 = 1e^{-7}$ to bring $\mathcal{L}_f$ to the same order of magnitude with $\mathcal{L}_{out}$. In Figure 2, we plot the accuracy curves during training of our SP-Net with and without self-distillation.

Table 6: Top-1 accuracy (%) for a SP-Net network with and without self-distillation for ResNet-18 on Tiny ImageNet.

Weight	Activation	Regular	L_{out}	$L_{out} + L_f$
2	2	50.1	53.0	**53.3**
2	3	50.5	53.8	**53.9**
2	32	51.2	**54.2**	53.8
3	2	50.8	53.4	**53.9**
3	3	51.5	54.0	**54.4**
3	32	52.3	54.1	**54.5**
32	2	51.4	52.6	**53.3**
32	3	52.1	53.4	**53.9**
32	32	52.9	**52.9**	52.8

6.3. Ablation Studies

Non-Switchable Batch Normalization. We replaced our S-BN layers with standard BN ones to verify and demonstrate that privatizing all BN layers for each switch is essential to successfully perform inference. Figure 3 shows the validation plots for a SP-Net network with 4 switches with and without S-BN. The switches of the network with S-BN learn at different rates, with the high precision ones learning faster and achieving higher accuracy than the low precision ones, while the accuracy of the switches without S-BN quickly stagnates and does not recover.

Impact of Multiple Switches. The impact of increasing the number of switches was investigated. In Table 7, we present the results of SP-Net with 4 and 16 switches slimmed with a factor of 0.5. The first remark that we notice is that with increasing number of switches, the initial learning rate should be lowered. For the network with 16 switches, we set the initial learning rate to 0.03. The accuracy obtained uses less switches is higher, we conjecture is due to the additional constraints, however it indicates an interesting direction of research.

Table 7: Top-1 accuracy (%) of SP-Nets with 4 and 16 switches slimmed with a factor of 0.5.

		Weights bitwidth				Weights bitwidth			
		2	4	8	32	2	4	8	32
Activations bitwidht	2	**41.3**	-	-	**43.8**	37.7	41.3	41.4	41.4
	4	-	-	-	-	38.6	42.3	42.9	42.4
	8	-	-	-	-	38.4	42.5	42.7	42.5
	32	**43.3**	-	-	**45.2**	39.6	42.7	43.2	43.1

Switchable Precision in Weights vs Activations. [52] found that QNNs activations are more sensitive to quantization than weights. Our observations on a SP-Net MobileNet confirm their findings. However, for our SP-Net ResNet-18, we observe that weights and activations are about equally sensitive. For example in Table 4, by freezing the weights to 2-bit and varying the bitwidths of activations in $\{2, 4, 8, 32\}$, we can observe that the accuracy gap is 1.8% with top accuracy of 34.3% for the ReLU-based quantizer, and gap of 6.6% with top accuracy of 35.5% for Logarithmic 2 quantizer. Similarly, by freezing the activations to 2-bit and varying the bitwidths of weights, the gap for ReLU-based quantizer is 3.8% with top accuracy of 36.3%, and for Logarithmic 2 the gap is 3.9% with top accuracy of 32.8%.

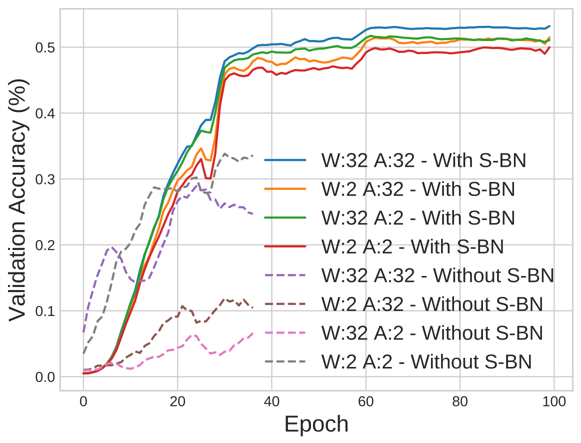

Figure 3: Top-1 accuracy for SP-Net ResNet-18 with and without switchable batch normalization (S-BN) on Tiny ImageNet.

Additionally, in Figure 3, we can observe that the switch "W:2 A:32" learns slower than "W:32 A:2", but towards the end of training, it catches up.

7. Conclusion and Future Work

In this paper we have proposed a DNN capable of operating at variable precisions on demand. With this approach, we grant devices and end-users real-time control over the performance of the DNNs powering inference algorithms. Our approach is lightweight and does not require altering the model, making it compatible with connectivity-free and limited memory devices. An additional virtue of our proposed network lies in ability to train a single network that can be distributed to different devices based on their capabilities. We have demonstrated the flexibility of our method with multiple quantization functions and a slimming complementary strategy. Moreover, we have proposed a training procedure to increase the accuracy of our network across the available precisions. Finally, we performed multiple ablation studies to analyse the performance of our approach in different scenarios.

Following the spirit of Universally Slimmable networks, a continuously quantizable SP-Net would be an interesting research direction as well as mixed precision SP-Net, where each layer is quantized independently on-demand. This method could be used to find the optimal layer-wise bitwidth for weights and activations.

8. Acknowledgements

This work was supported by the Australian Research Council through the ARC Centre of Excellence for Robotic Vision (project number CE1401000016).

References

[1] Thalaiyasingam Ajanthan, Puneet K Dokania, Richard Hartley, and Philip HS Torr. Proximal mean-field for neural network quantization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4871–4880, 2019. 1, 2

[2] Yu Bai, Yu-Xiang Wang, and Edo Liberty. Proxquant: Quantized neural networks via proximal operators. In *Proc. Int. Conf. Learn. Repren.*, 2019. 1, 2

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 3

[4] Han Cai, Chuang Gan, and Song Han. Once for all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 2

[5] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 5918–5926, 2017. 3, 6

[6] Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*, 2017. 1

[7] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018. 2

[8] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Learning low precision deep neural networks through regularization. *arXiv preprint arXiv:1809.00095*, 2018. 2

[9] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 3123–3131, 2015. 1

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 770–778, 2016. 1, 5, 6

[11] Tong He, Chunhua Shen, Zhi Tian, Dong Gong, Changming Sun, and Youliang Yan. Knowledge adaptation for efficient semantic segmentation. *arXiv preprint arXiv:1903.04688*, 2019. 2

[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *Proc. Adv. Neural Inf. Process. Syst. Workshops*, 2014. 2

[13] Lu Hou and James T Kwok. Loss-aware weight quantization of deep networks. In *Proc. Int. Conf. Learn. Repren.*, 2018. 1, 2

[14] 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. 5

[15] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 4107–4115, 2016. 1, 2

[16] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061*, 2016. 1

[17] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 4350–4359, 2019. 2

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learn. Repren.*, 2015. 6

[19] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016. 2

[20] Yifan Liu, Bohan Zhuang, Chunhua Shen, Hao Chen, and Wei Yin. Training compact neural networks via auxiliary overparameterization. *arXiv preprint arXiv:1909.02214*, 2019. 1

[21] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proc. Eur. Conf. Comp. Vis.*, 2018. 1, 3

[22] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3431–3440, 2015. 1

[23] Mason McGill and Pietro Perona. Deciding how to decide: Dynamic routing in artificial neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2363–2372. JMLR. org, 2017. 1

[24] Asit Mishra and Debbie Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In *Proc. Int. Conf. Learn. Repren.*, 2018. 2

[25] Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016. 3, 4, 6

[26] Eunhyeok Park, Junwhan Ahn, and Sungjoo Yoo. Weighted-entropy-based quantization for deep neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 5456–5464, 2017. 2

[27] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. In *Proc. Int. Conf. Learn. Repren.*, 2018. 2

[28] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proc. Eur. Conf. Comp. Vis.*, pages 525–542, 2016. 1, 2, 3, 6

[29] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 779–788, 2016. 1

[30] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *Proc. Int. Conf. Learn. Repren.*, 2015. 2

[31] Adrià Ruiz and Jakob Verbeek. Adaptative inference cost with convolutional neural mixture models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1872–1881, 2019. 2

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Comp. Vis.*, 115(3):211–252, 2015. 6

[33] Charbel Sakr, Jungwook Choi, Zhuo Wang, Kailash Gopalakrishnan, and Naresh Shanbhag. True gradient-based training of deep binary activated neural networks via continuous binarization. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2346–2350. IEEE, 2018. 1, 2

[34] Mingzhu Shen, Kai Han, Chunjing Xu, and Yunhe Wang. Searching for accurate binary neural architectures. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 1

[35] Tao Sheng, Chen Feng, Shaojie Zhuo, Xiaopeng Zhang, Liang Shen, and Mickey Aleksic. A quantization-friendly separable convolution for mobilenets. In *2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*, pages 14–18. IEEE, 2018. 6

[36] Frederick Tung and Greg Mori. Clip-q: Deep network compression learning by in-parallel pruning-quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7873–7882, 2018. 1

[37] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019. 1

[38] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 8612–8620, 2019. 2

[39] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proc. Eur. Conf. Comp. Vis.*, pages 409–424, 2018. 1

[40] Yi Wei, Xinyu Pan, Hongwei Qin, Wanli Ouyang, and Junjie Yan. Quantization mimic: Towards very tiny cnn for object detection. In *Proc. Eur. Conf. Comp. Vis.*, pages 267–283, 2018. 2

[41] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 8817–8826, 2018. 1

[42] Jiahui Yu and Thomas Huang. Network slimming by slimmable networks: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*, 2019. 2

[43] Jiahui Yu and Thomas Huang. Universally slimmable networks and improved training techniques. *arXiv preprint arXiv:1903.05134*, 2019. 2, 4, 5

[44] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018. 1, 2, 5

[45] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *Proc. Int. Conf. Learn. Repren.*, 2017. 2

[46] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proc. Eur. Conf. Comp. Vis.*, 2018. 2

[47] 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. *arXiv preprint arXiv:1905.08094*, 2019. 2

[48] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *Proc. Int. Conf. Learn. Repren.*, 2017. 2

[49] Aojun Zhou, Anbang Yao, Kuan Wang, and Yurong Chen. Explicit loss-error-aware quantization for low-bit deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9426–9435, 2018. 1, 2

[50] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 1, 2, 3, 6

[51] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *Proc. Int. Conf. Learn. Repren.*, 2017. 2

[52] Shilin Zhu, Xin Dong, and Hao Su. Binary ensemble neural network: More bits per network or more networks per bit? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4923–4932, 2019. 8

[53] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Towards effective low-bitwidth convolutional neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. 1, 2, 5