Slimmable Neural Networks

Jiahui Yu[1], Linjie Yang[2], Ning Xu[2], Jianchao Yang[3], Thomas Huang[1]

[1]University of Illinois at Urbana-Champaign, [2]Snap Inc., [3]ByteDance Inc.

ICLR 2019

Introduction

- **Deep neural networks are also prevailing in applications on mobile phone, AR, etc.**
  - Requires short response time -> lightweight networks
- **Device performances are various**
  - Models with different size need to be trained individually
    - Big offline table needs to be maintained
    - Switching to a smaller/larger model, the cost of time and data for downloading is not negligible

Table 1: Runtime of MobileNet v1 for image classification on different devices.

	OnePlus 6	Google Pixel	LG Nexus 5	Samsung Galaxy S3	ASUS ZenFone 2
Runtime	24 ms	116 ms	332 ms	553 ms	1507 ms

Given budgets of resources, how to instantly, adaptively and efficiently trade off between accuracy and latency at runtime?

Slimmable Neural Networks

- **Different Neural Network with Shared Weights**

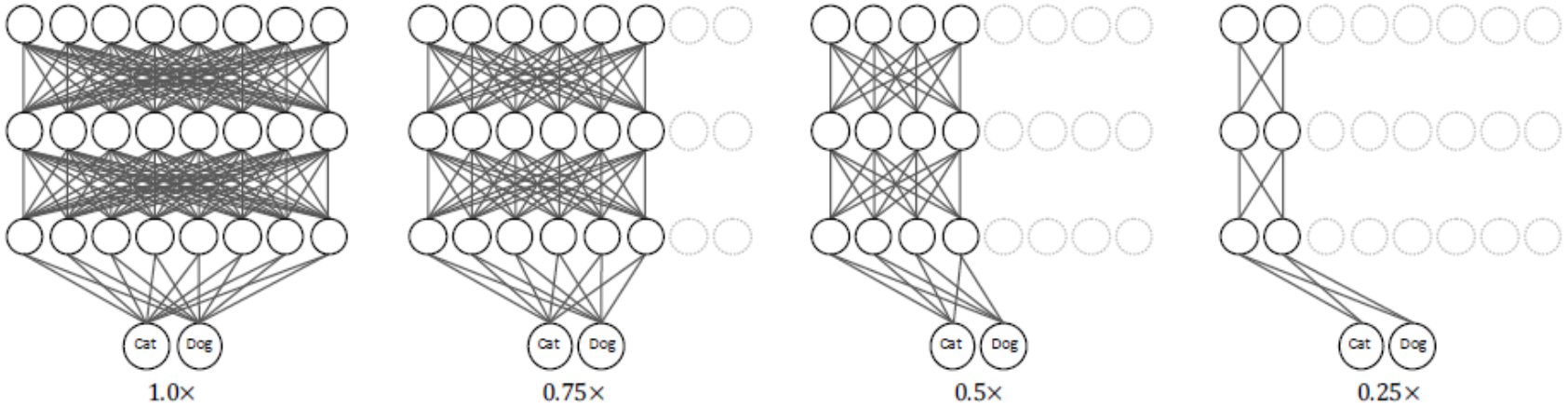


Figure 1: Illustration of slimmable neural networks. The same model can run at different widths (number of active channels), permitting instant and adaptive accuracy-efficiency trade-offs.

How to train?

- **Empirically, naive training approach(weight sharing) leads bad performance**
  - Different widths have different feature mean and variance
- **Progressive training (Tann et al., 2016)**
  - First train a base model A (MobileNet v2 0.35x)
  - Fix it and add extra parameter B to make it an extended model A+B (MobileNet v2 0.5x)
  - Performance is not so good (1000-class ImageNet Top 1 Accuracy)
    - 60.3% -> 61.0% (A -> A+B), while individually trained A+B achieves 65.4% accuracy
- **Solution : Switchable Batch Normalization (switch ≈ width)**

How to Train?

- **Batch Normalization (2015)**

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

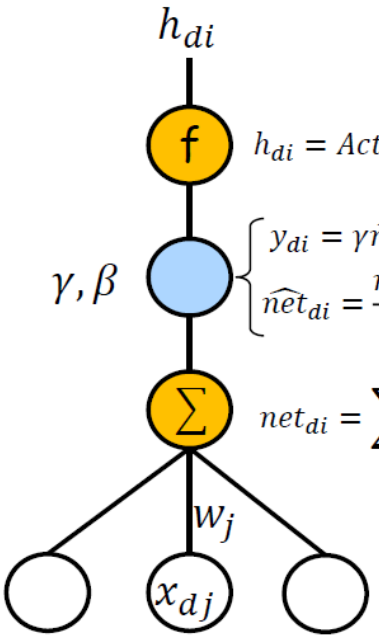
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.



$$net_{di} = \sum_{j=1}^{J} w_j x_{dj} + b$$

$$\mu_B = \frac{1}{D_B}\sum_{d=1}^{D_B} net_{di}$$

$$\sigma_B^2 = \frac{1}{D_B}\sum_{d=1}^{D_B}(net_{di} - \mu)^2$$

Switchable Batch Normalization (S-BN)

- **Four BN variables privatize for different switches**

$$y' = \gamma \frac{y-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta$$

  - μ, σ : Independently accumulate feature during training
  - γ, β : Learnable scale and bias. Act as conditional parameter

- **At runtime, the network can adjust its width on the fly according to on-device benchmarks and resource constraints**

Switchable Batch Normalization (S-BN)

- **Advantages**
  - The number of extra parameters is negligible (less than 1%)

Table 2: Number and percentage of parameters in batch normalization layers.

	MobileNet v1 1.0×	MobileNet v2 1.0×	ShuffleNet 2.0×	ResNet-50 1.0×
Conv and FC	4,210,088 (99.483%)	3,470,760 (99.027%)	5,401,816 (99.102%)	25,503,912 (99.792%)
BatchNorm	21,888 (0.517%)	34,112 (0.973%)	48,960 (0.898%)	53,120 (0.208%)

  - Re-fusing of batch normalization can be done at runtime

Switchable Batch Normalization (S-BN)

Algorithm 1 Training slimmable neural network M.

Require: Define *switchable width list* for slimmable network M, for example, $[0.25, 0.5, 0.75, 1.0]\times$.

Initialize shared convolutions and fully-connected layers for slimmable network M.
Initialize independent batch normalization parameters for each *width* in *switchable width list*.
for $i = 1, ..., n_{iters}$ **do**
    Get next mini-batch of data x and label y.
    Clear gradients of weights, $optimizer.zero\_grad()$.
    for *width* in *switchable width list* **do**
        Switch the batch normalization parameters of current width on network M.
        Execute sub-network at current width, $\hat{y} = M'(x)$.
        Compute loss, $loss = criterion(\hat{y}, y)$.
        Compute gradients, $loss.backward()$.
    end for
    Update weights, $optimizer.step()$.
end for

Experiment

- **1000-class ImageNet**
- **Models**
  - 3 Lightweight networks - MobileNet v1 & v2, ShuffleNet
  - Large Model - ResNet-50
- **Evaluation : Top-1 classification Error**

Experiment

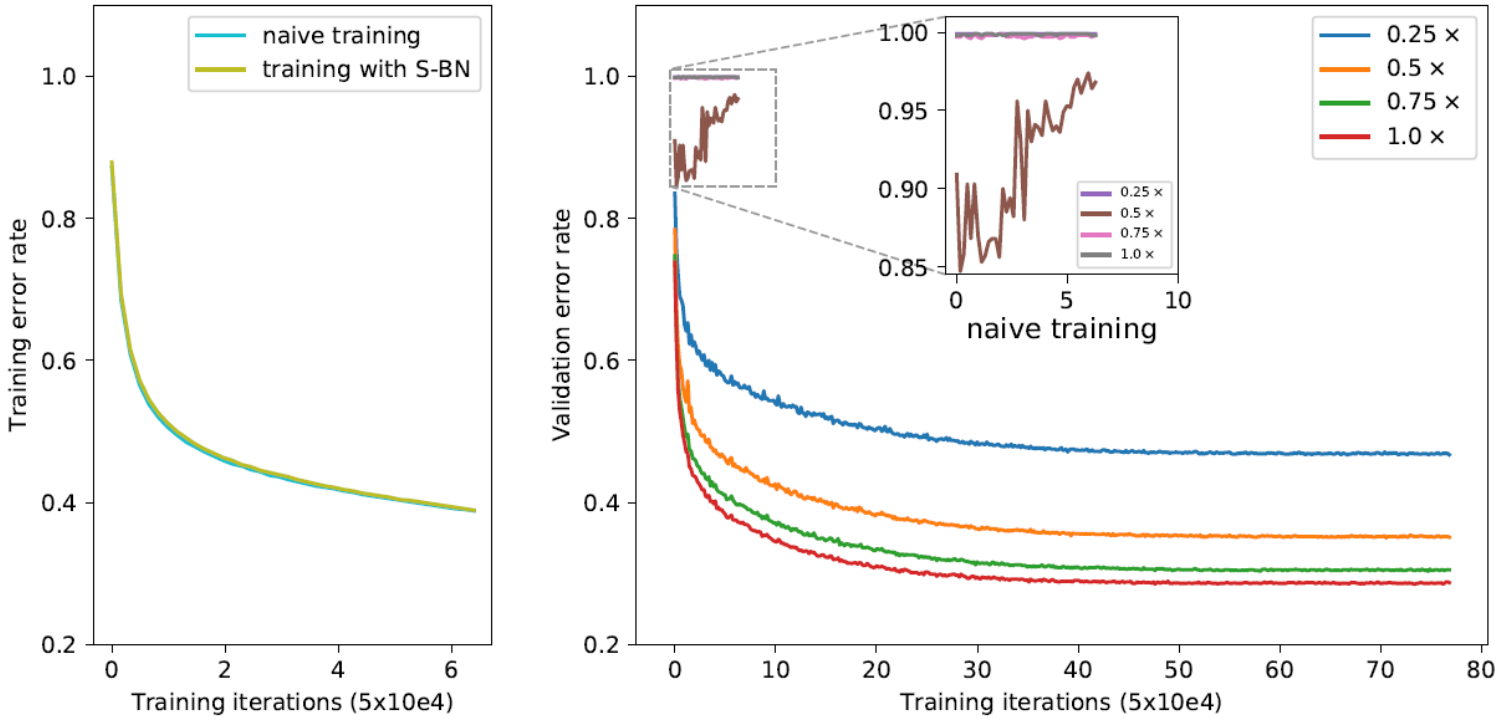


Figure 2: Training and validation curves of slimmable networks. Left shows the training error of the largest switch. Right shows testing errors on validation set with different switches. For naive approach, the training is stable (left) but testing error is high (right, zoomed). Slimmable networks trained with *S-BN* have stable and rank-preserved testing accuracy across all training iterations.

Experiment

Table 3: Results of ImageNet classification. We show top-1 error rates of individually trained networks and slimmable networks given same width configurations and FLOPs. We use S- to indicate slimmable models, † to denote our reproduced result.

Individual Networks			**Slimmable Networks**			FLOPs
Name	Params	Top-1 Err.	Name	Params	Top-1 Err.	
MobileNet v1 1.0×	4.2M	29.1	S-MobileNet v1 [0.25, 0.5, 0.75, 1.0]×	4.3M	28.5 (0.6)	569M
MobileNet v1 0.75×	2.6M	31.6			30.5 (1.1)	317M
MobileNet v1 0.5×	1.3M	36.7			35.2 (1.5)	150M
MobileNet v1 0.25×	0.5M	50.2			46.9 (3.3)	41M
MobileNet v2 1.0×	3.5M	28.2	S-MobileNet v2 [0.35, 0.5, 0.75, 1.0]×	3.6M	29.5 (-1.3)	301M
MobileNet v2 0.75×	2.6M	30.2			31.1 (-0.9)	209M
MobileNet v2 0.5×	2.0M	34.6			35.6 (-1.0)	97M
MobileNet v2 0.35×	1.7M	39.7			40.3 (-0.6)	59M
ShuffleNet 2.0×	5.4M	26.3	S-ShuffleNet [0.5, 1.0, 2.0]×	5.5M	28.7 (-2.4)	524M
ShuffleNet 1.0×	1.8M	32.6			34.5 (-0.9)	138M
ShuffleNet 0.5×	0.7M	43.2			42.7 (0.5)	38M
ResNet-50 1.0×	25.5M	23.9	S-ResNet-50 [0.25, 0.5, 0.75, 1.0]×	25.6M	24.0 (-0.1)	4.1G
ResNet-50 0.75×†	14.7M	25.3			25.1 (0.2)	2.3G
ResNet-50 0.5×†	6.9M	28.0			27.9 (0.1)	1.1G
ResNet-50 0.25×†	2.0M	36.2			35.0 (1.2)	278M

Experiment

- **More switches in slimable networks**
  - Slimable network with more switches have similar performance
  - Demonstrating the scalability of proposed approach

Table 4: Top-1 error rates on ImageNet classification with individually trained networks, 4-switch S-MobileNet v1 [0.25, 0.5, 0.75, 1.0]× and 8-switch S-MobileNet v1 [0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 1.0]×.

	0.25×	0.35×	0.45×	0.5×	0.55×	0.65×	0.75×	0.85×	1.0×
Individual	50.2	-	-	36.7	-	-	31.6	-	29.1
4-switch	46.9 (3.3)	-	-	35.2 (1.5)	-	-	30.5 (1.1)	-	28.5 (0.6)
8-switch	47.6 (2.6)	41.1	36.6	-	33.8	31.4	30.2 (1.4)	29.2	28.4 (0.7)

Experiment

- **Object detection, Instance segmentation, Keypoint detection**
  - Faster R-CNN, Mask-RCNN, Keypoints R-CNN

Type	Individual Networks			Slimmable Networks		
	Box AP	Mask AP	Kps AP	Box AP	Mask AP	Kps AP
Faster 1.0×	36.4	-	-	36.8 (0.4)	-	-
Faster 0.75×	34.7	-	-	36.1 (1.4)	-	-
Faster 0.5×	32.7	-	-	34.0 (1.3)	-	-
Faster 0.25×	27.5	-	-	29.6 (2.1)	-	-
Mask 1.0×	37.3	34.2	-	37.4 (0.1)	34.9 (0.7)	-
Mask 0.75×	35.6	32.9	-	36.7 (1.1)	34.3 (1.4)	-
Mask 0.5×	33.4	30.9	-	34.7 (1.5)	32.6 (1.7)	-
Mask 0.25×	28.2	26.6	-	30.2 (2.0)	28.6 (2.0)	-
Kps 1.0×	50.5	-	61.3	52.8 (2.3)	-	63.9 (2.6)
Kps 0.75×	49.6	-	60.5	52.7 (3.1)	-	63.6 (3.1)
Kps 0.5×	48.5	-	59.8	51.6 (3.1)	-	62.6 (2.8)
Kps 0.25×	45.4	-	56.7	48.2 (2.8)	-	59.5 (2.8)

-> Probability due to implicit distillation and richer supervision

Conclusion

- **Switchable Batch Normalization**
- **Simple and general method to train a single neural network executable at different widths**
- **Provides practical approach to adjust model for different computation resource**