Bag of Tricks for Image Classification with Convolutional Neural Networks

Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, Mu Li

CVPR 2019

Yang Changeun

Introduction

- A large number of refinements, including changes in loss function, preprocessing, and optimization methods, has been proposed in the past years, but has received relatively less attention than model architecture

- In this paper, authors examined a collection of training procedure and model architecture refinements that improve model accuracy but barely change computational complexity

Model	FLOPs	top-1
ResNet-50 [9]	3.9 G	75.3
ResNeXt-50 [27]	4.2 G	77.8
SE-ResNet-50 [12]	3.9 G	76.71
SE-ResNeXt-50 [12]	4.3 G	78.90
DenseNet-201 [13]	4.3 G	77.42
ResNet-50 + tricks (ours)	4.3 G	**79.29**

Paper Outline

1) Baseline training procedure
2) Efficient training
3) Model architecture tweaks for ResNet
4) Training procedure refinements
5) Transfer Learning

1) Baseline Training Procedure

1. Randomly sample an image and decode it into 32-bit floating point raw pixel values in $[0, 255]$.
2. Randomly crop a rectangular region whose aspect ratio is randomly sampled in $[3/4, 4/3]$ and area randomly sampled in $[8\%, 100\%]$, then resize the cropped region into a 224-by-224 square image.
3. Flip horizontally with 0.5 probability.
4. Scale hue, saturation, and brightness with coefficients uniformly drawn from $[0.6, 1.4]$.
5. Add PCA noise with a coefficient sampled from a normal distribution $\mathcal{N}(0, 0.1)$.
6. Normalize RGB channels by subtracting 123.68, 116.779, 103.939 and dividing by 58.393, 57.12, 57.375, respectively.

Algorithm 1 Train a neural network with mini-batch stochastic gradient descent.

initialize(net)
for epoch $= 1, \ldots, K$ **do**
  for batch $= 1, \ldots, \#\text{images}/b$ **do**
    images $\leftarrow$ uniformly random sample b images
    $X, y \leftarrow$ preprocess(images)
    $z \leftarrow$ forward(net, X)
    $\ell \leftarrow$ loss(z, y)
    grad $\leftarrow$ backward(ℓ)
    update(net, grad)
  end for
end for

Model	Baseline		Reference	
	Top-1	Top-5	Top-1	Top-5
ResNet-50 [9]	75.87	92.70	75.3	92.2
Inception-V3 [26]	77.32	93.43	78.8	94.4
MobileNet [11]	69.03	88.71	70.6	-

2) Efficient Training

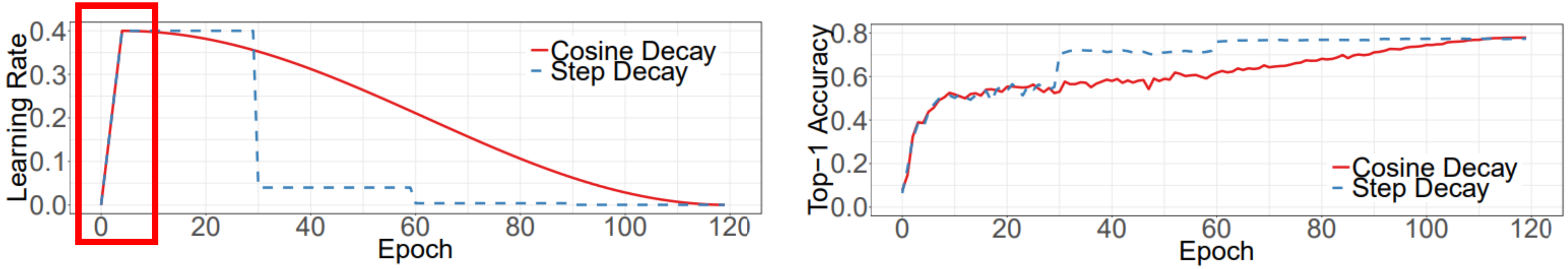
- Mini-batch SGD groups multiple samples to a mini-batch to increase parallelism and decrease communication costs
  - But using large batch size cause low convergence
  - Then How to make the convergence faster?

1. Linear scaling learning rate
   - Increasing the batch size reduces its variance
   - Less noise in the gradient, so it's okay to increase the learning rate
   - 0.1 as the initial learning rate for batch size 256.

$$lr = 0.1 * \frac{b}{256}, b:\ batch\ size$$

2) Efficient Training

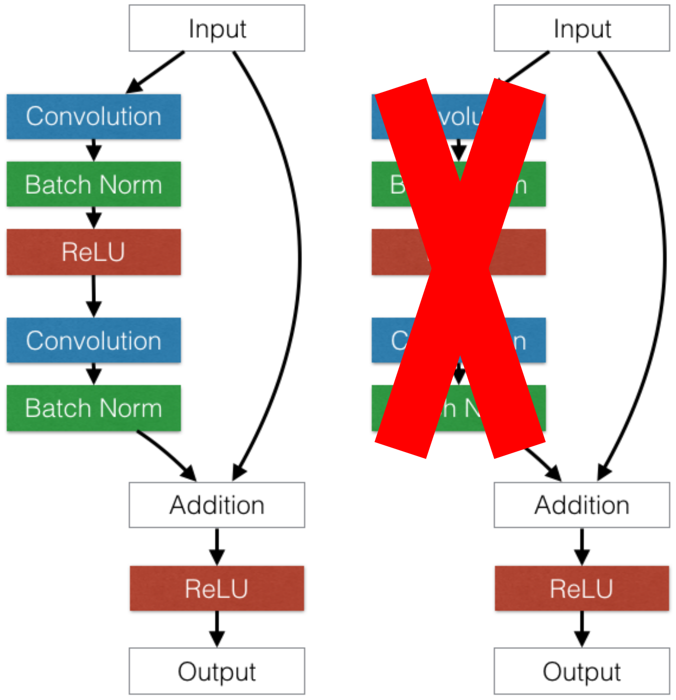
2. Learning rate warmup
    - All parameters are random value and far away from the optimal parameters at the beginning of the training
    - Using a too large learning rate at the beginning may not be stable
    - Increase the learning rate from 0 to the initial learning rate linearly



(a) Learning Rate Schedule (b) Validation Accuracy

2) Efficient Training

3. Zero γ
   - The last layer of the residual block is BN(Batch Normalization) layer
   - Initialize γ = 0 for BN layers in all residual blocks
   - So that all residual blocks just return their inputs
   - The network has less layers and is easier to train at the initial stage



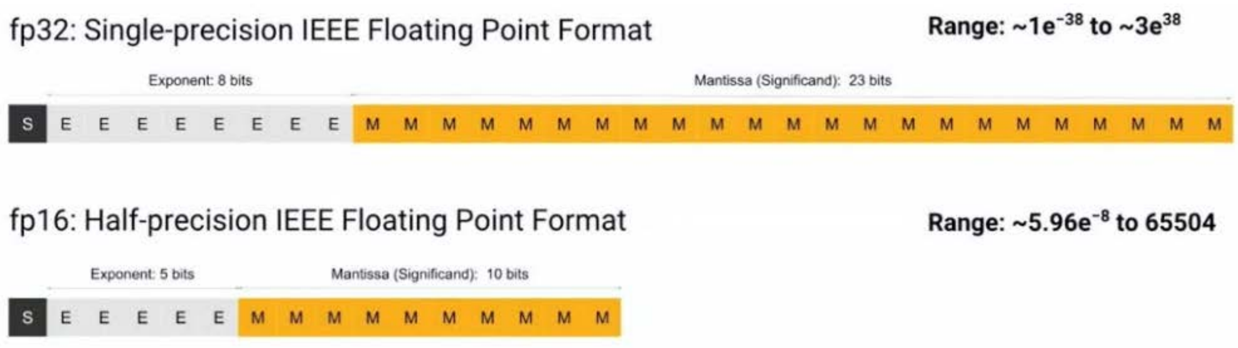
2) Efficient Training

4. No bias decay
   - The weight decay is often applied to all learnable parameters including both weights and bias.

   - Jia et al. recommended to only apply the regularization to weights to avoid overfitting.

   - The biases and γ and β in BN layers are left unregularized.

2) Efficient Training



5. Low-precision training
   - Neural networks are commonly trained with 32-bit floating point(FP32) precision
   - New hardwares(ex. Nvidia V100) are much faster in FP16
   - But a reduced precision has a narrow range and it can disturb the training progress
   - Mickikevicius et al. proposed mixed training
     - FP16 for computing gradients, FP32 for parameter updating

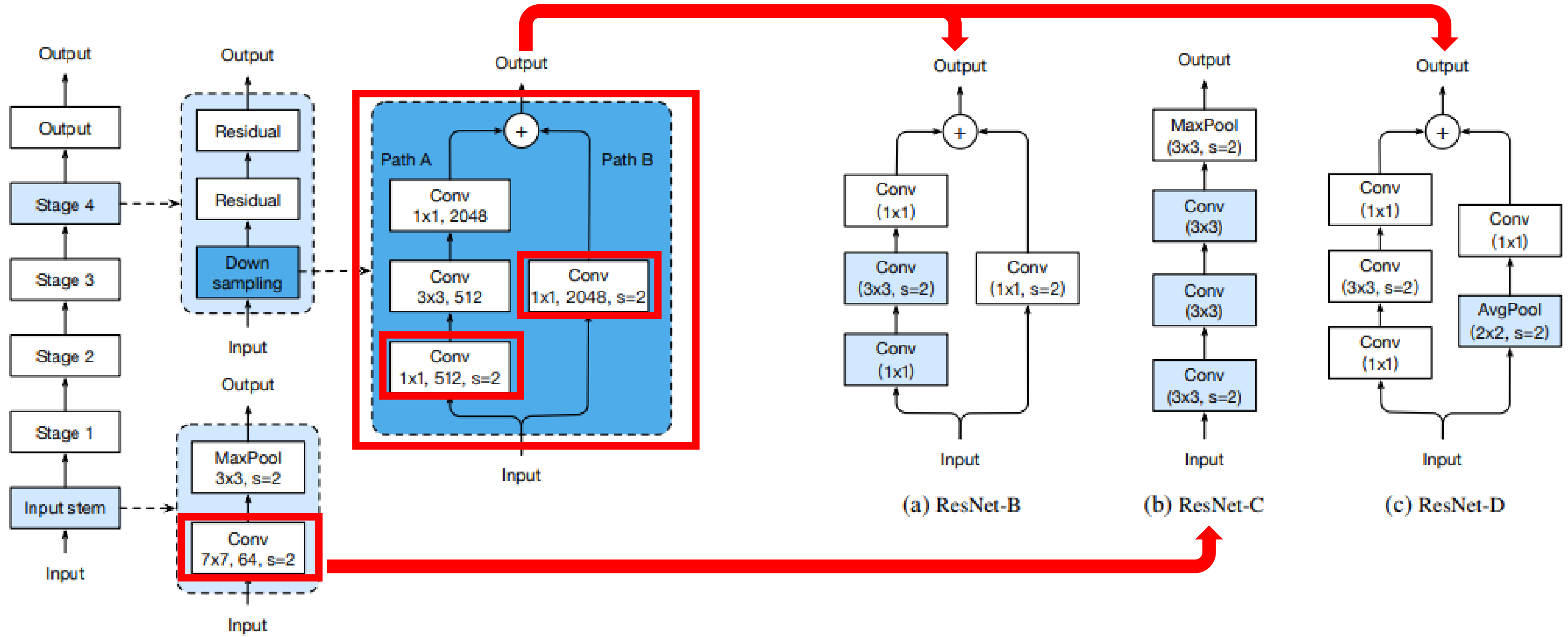
2) Efficient Training

Model	FP16, BS=1024 Efficient			FP32, BS=256 Baseline		
	Time/epoch	Top-1	Top-5	Time/epoch	Top-1	Top-5
ResNet-50	**4.4 min**	**76.21**	**92.97**	13.3 min	75.87	92.70
Inception-V3	**8 min**	**77.50**	**93.60**	19.8 min	77.32	93.43
MobileNet	**3.7 min**	**71.90**	**90.47**	6.2 min	69.03	88.71

Heuristic	BS=256		BS=1024	
	Top-1	Top-5	Top-1	Top-5
Linear scaling	75.87	92.70	75.17	92.54
+ LR warmup	76.03	92.81	75.93	92.84
+ Zero γ	76.19	93.03	76.37	92.96
+ No bias decay	76.16	92.97	76.03	92.86
+ FP16	76.15	93.09	76.21	92.97

?

3) Model Tweaks



- ResNet-B : 1x1 Conv. with stride=2 loses data -> put stride=2 on 3x3 Conv. instead
- ResNet-C : 7x7 Conv. takes too much computational cost -> use three 3x3 Conv. instead
- ResNet-D : Replace 1x1 Conv. with stride=2 in Path B with Average Pooling

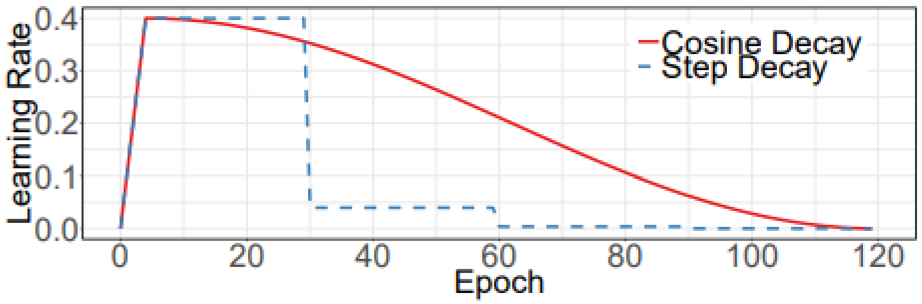
3) Model Tweaks

- ResNet-50-D improves ResNet-50 by 1%

- Four models have the same model size
  - ResNet-50-D is only 3% slower in training throughput compared to ResNet-50

Model	#params	FLOPs	Top-1	Top-5
ResNet-50	25 M	**3.8 G**	76.21	92.97
ResNet-50-B	25 M	4.1 G	76.66	93.28
ResNet-50-C	25 M	4.3 G	76.87	93.48
ResNet-50-D	25 M	4.3 G	**77.16**	**93.52**

4) Training Refinements



(a) Learning Rate Schedule

1. Cosine Learning Rate Decay

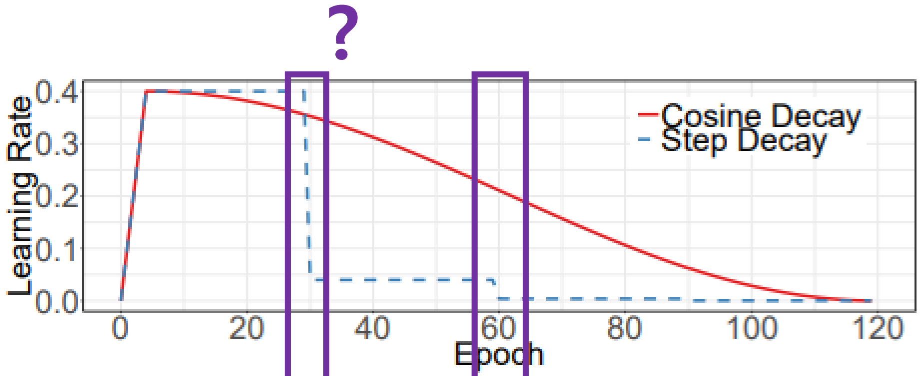
- He et al. decreased the learning rate at 0.1 for every 30 epochs
  - => Step Decay
- Loshchilov et al. proposed cosine decay: decreasing the learning rate from the initial value to 0 by following cosine function

$$\eta_t = \frac{1}{2}\left(1 + \cos\left(\frac{t\pi}{T}\right)\right)\eta,$$

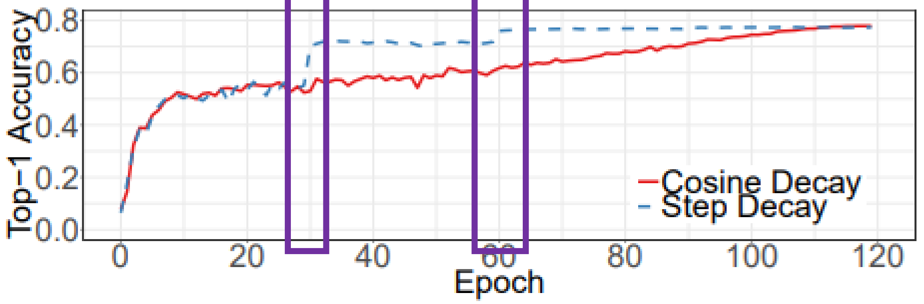
- T : total number of batches(the warmup stage is ignored)
- t : current batch, η_t : learning rate
- η : the initial learning rate

4) Training Refinements

1. Cosine Learning Rate Decay



(a) Learning Rate Schedule



(b) Validation Accuracy

4) Training Refinements

2. Label Smoothing

- When the number of labels is *K*, the predicted score for class *i* is *z_i*, the softmax *q_i* is

$$q_i = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)}.$$

- And its loss: cross entropy *l(p,q)* is

$$\ell(p, q) = -\sum_{i=1}^{K} \cancelto{p_i}{q_i} \log \cancelto{q_i}{p_i}$$

When *p* is a truth distribution

4) Training Refinements

2. Label Smoothing

- If *p* is one hot encoding, the cross entropy is

$$\ell(p,q) = -\log \overset{q\_y}{\cancel{p_y}} = -z_y + \log\left(\sum_{i=1}^{K} \exp(z_i)\right)$$

- And the optimal solution is *z*_y=inf*, which can leads to overfitting

- In order to prevent this, change the true probability *p_i* to

$$\overset{p\_i}{\cancel{q_i}} = \begin{cases} 1-\varepsilon & \text{if } i = y, \\ \varepsilon/(K-1) & \text{otherwise,} \end{cases}$$

ε=0.1, K=1000

4) Training Refinements

2. Label Smoothing

- Then, the optimal solution becomes

$$z_i^* = \begin{cases} \log((K-1)(1-\varepsilon)/\varepsilon) + \alpha & \text{if } i = y, \\ \alpha & \text{otherwise,} \end{cases}$$

Where α is arbitrary real number

- This encourages a finite output from the fully-connected layer and can generalize better

4) Training Refinements

3. Knowledge Distillation

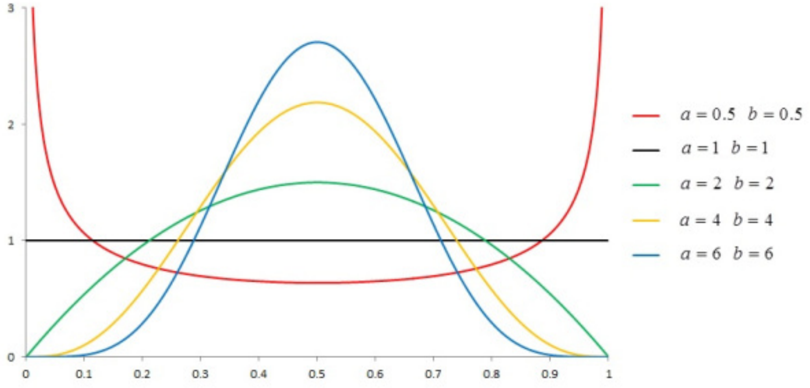
- A teacher model helps train the current model(student model)
  - ex) ResNet-152: teacher model, ResNet-50: student model

- Distillation loss is used to penalize the difference between the softmax outputs from the teacher model and the student model

$$\underbrace{\boxed{\ell(p, \mathrm{softmax}(z))}}_{\text{original loss}} + \underbrace{\boxed{T}}_{\text{hyper parameter}}^2 \underbrace{\boxed{\ell(\mathrm{softmax}(r/T), \mathrm{softmax}(z/T))}}_{\text{distillation loss}},$$

- T = 20

4) Training Refinements



4. Mixup Training

- Two random examples (x_i, y_i) and (x_j, y_j) are chosen and a new example is made by a weighted linear interpolation

$$
\begin{aligned}
\hat{x} &= \lambda x_i + (1-\lambda) x_j, \\
\hat{y} &= \lambda y_i + (1-\lambda) y_j,
\end{aligned}
$$

[1.0, 0.0] [0.0, 1.0] [0.7, 0.3]

- λ : a random number drawn from the Beta(α, α) distribution (α=0.2)
- In mixup training, only the new examples are used
- Mixup training asks for a longer training progress to converge better

4) Training Refinements

- They trained ResNet, Inception-V3 and MobileNet with these refinements
- By stacking cosine decay, label smoothing and mixup, those models were improved steadily

Refinements	ResNet-50-D		Inception-V3		MobileNet	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
Efficient	77.16	93.52	77.50	93.60	71.90	90.53
+ cosine decay	77.91	93.81	78.19	94.06	72.83	91.00
+ label smoothing	78.31	94.09	78.40	94.13	72.93	91.14
+ distill w/o mixup	78.67	94.36	78.26	94.01	71.97	90.89
+ mixup w/o distill	79.15	94.58	**78.77**	**94.39**	**73.28**	**91.30**
+ distill w/ mixup	**79.29**	**94.63**	78.34	94.16	72.51	91.02

4) Training Refinements

- These tricks are also used to train a ResNet-50-D on MIT Places365 dataset

- The refinements improve the top-5 accuracy consistently on both the validation and test set

Model	Val Top-1 Acc	Val Top-5 Acc	Test Top-1 Acc	Test Top-5 Acc
ResNet-50-D Efficient	56.34	86.87	57.18	87.28
ResNet-50-D Best	**56.70**	**87.33**	**57.63**	**87.82**

5) Transfer Learning

1. Object Detection

- VGG-19 base model in Faster-RCNN is replaced with pretrained model
- Keep other settings the same so the gain is solely from the base model

Refinement	Top-1	mAP
B-standard	76.14	77.54
D-efficient	77.16	78.30
+ cosine	77.91	79.23
+ smooth	78.34	80.71
+ distill w/o mixup	78.67	80.96
+ mixup w/o distill	79.16	81.10
+ distill w/ mixup	79.29	**81.33**

5) Transfer Learning

2. Semantic Segmentation
   - the base network of FCN is replaced
   - Cosine learning rate schedule improves the accuracy
   - But other refinements provide suboptimal results
   - Soften labels by label smoothing, distillation and mixup blurred pixel-level information and degraded overall accuracy?

Refinement	Top-1	PixAcc	mIoU
B-standard	76.14	78.08	37.05
D-efficient	77.16	78.88	38.88
+ cosine	77.91	**79.25**	**39.33**
+ smooth	78.34	78.64	38.75
+ distill w/o mixup	78.67	78.97	38.90
+ mixup w/o distill	79.16	78.47	37.99
+ mixup w/ distill	79.29	78.72	38.40

Conclusion

- They survey a dozen tricks to train deep convolutional neural networks to improve model accuracy
- Results on ResNet-50, Inception-V3 and MobileNet indicate that these tricks improve model accuracy consistently
- These improved pre-trained models show strong advantages in transfer learning