Bag of Tricks for Image Classification with Convolutional Neural Networks (CVPR2019)

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

Amazon Web Services
{htong,zhiz,hzaws,zhongyue,junyuanx,mli}@amazon.com

Presenter: Nyanye

Introduction

- Much of the recent progress made in image classification research can be credited to training procedure refinements
- changes in
  - data augmentations
  - optimization methods
- examine a collection of such refinements and empirically evaluate their impact on the final model accuracy through ablation study

Baseline

Model

- Widely used implementation of ResNet (ResNet50)

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 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

Training – Large batch training

- Using large batch size, may slow down the training progress. For convex problems, convergence rate decreases as batch size increases.
- for the same number of epochs, training with a large batch size results in a model with degraded validation accuracy compared to the ones trained with smaller batch sizes.
- examine four heuristics that help scale the batch size up for single machine training.

Training – Linear scaling learning rate

- Increasing the batch size does not change the expectation of the stochastic gradient but reduces its variance
- large batch size reduces the noise in the gradient, so we may increase the learning rate to make a larger progress along the opposite of the gradient direction.
- choose 0.1 as the initial learning rate for batch size 256, then when changing to a larger batch size b, we will increase the initial learning rate to $0.1 \times b/256$
- Ex: batch size = 512, initial learning rate = 0.2

Training – Learning rate warmup

- At the beginning of the training, all parameters are typically random values and therefore far away from the final solution
- Using a too large learning rate may result in numerical instability
- In warmup heuristic, we use a small learning rate at the beginning and then switch back to the initial learning rate when the training process is stable
- we will use the first m batches (e.g. 5 data epochs) to warm up, and the initial learning rate is η, then at batch i, $1 \leq i \leq m$, we will set the learning rate to be $i\eta/m$.
- Ex) $m = 1024$ / $i = 512$ / $\eta = 0.1$

$0.1*512/1024 = 0.05$

Training – No bias decay

- weight decay is often applied to all learnable parameters including both weights and bias
- it’s recommended to only apply the regularization to weights to avoid overfitting
- only applies the weight decay to the weights in convolution and fullyconnected layers. Other parameters, including the biases and γ and β in BN layers, are left unregularized.

Training – Low-precision training

- Neural networks are commonly trained with 32-bit floating point (FP32) precision
- New hardware, however, may have enhanced arithmetic logic unit for lower precision data types
- V100 offers 14 TFLOPS in FP32 but over 100 TFLOPS in FP16. As in Table 3, the overall training speed is accelerated by 2 to 3 times after switching from FP32 to FP16 on V100.
- store all parameters and activations in FP16 and use FP16 to compute gradients
- multiplying a scalar to the loss to better align the range of the gradient into FP16

Training

- Switching from FP32 to FP16 at the end of training does not affect the accuracy

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

- using a larger 1024 batch size and FP16 reduces the training time for ResNet-50 from 13.3-min per epoch to 4.4- min per epoch.

Model Tweaks

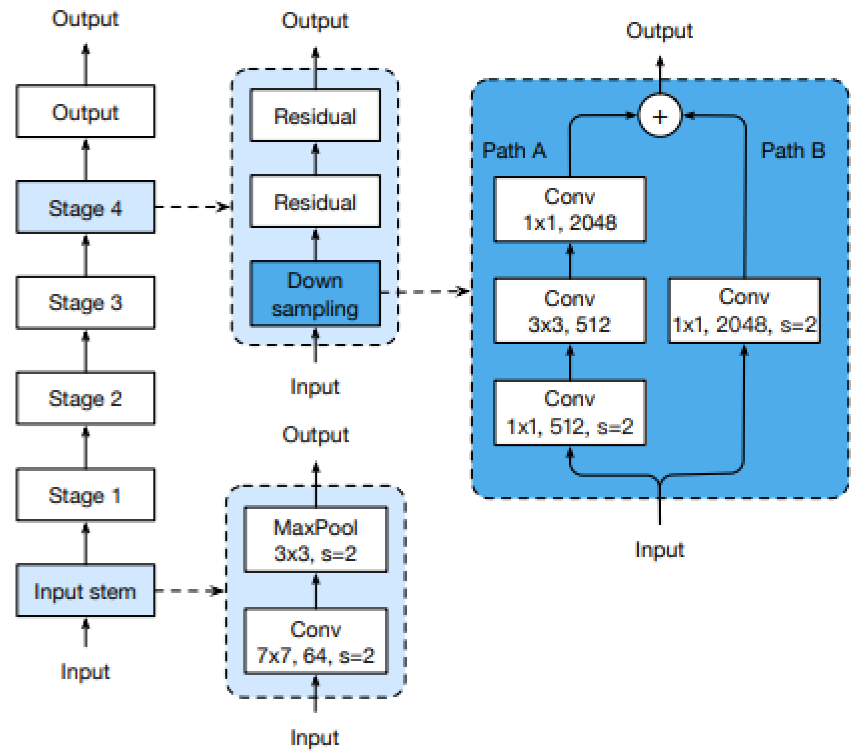


Figure 1: The architecture of ResNet-50. The convolution kernel size, output channel size and stride size (default is 1) are illustrated, similar for pooling layers.

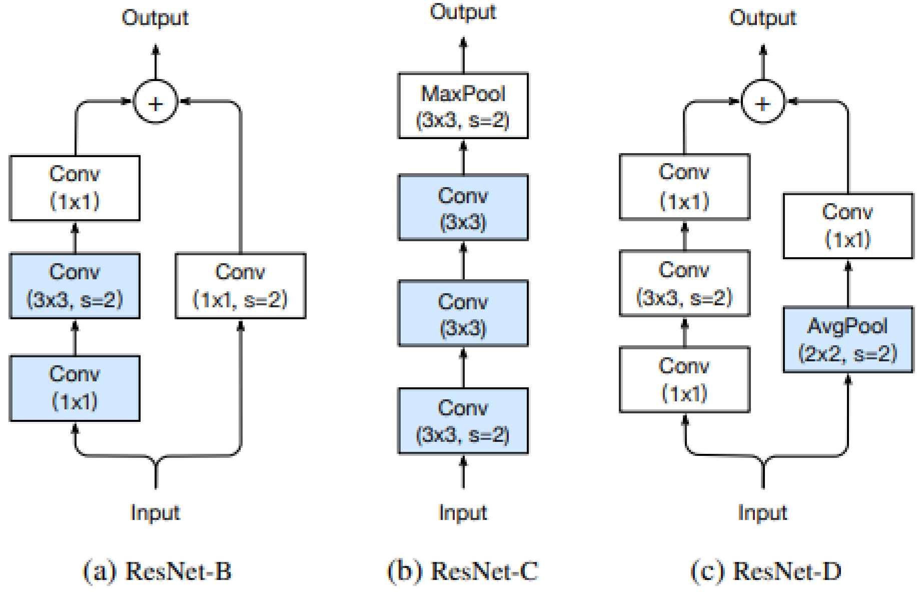


Figure 2: Three ResNet tweaks. ResNet-B modifies the downsampling block of Resnet. ResNet-C further modifies the input stem. On top of that, ResNet-D again modifies the downsampling block.

Model Tweaks

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**

Table 5: Compare ResNet-50 with three model tweaks on model size, FLOPs and ImageNet validation accuracy.

Training Refinement

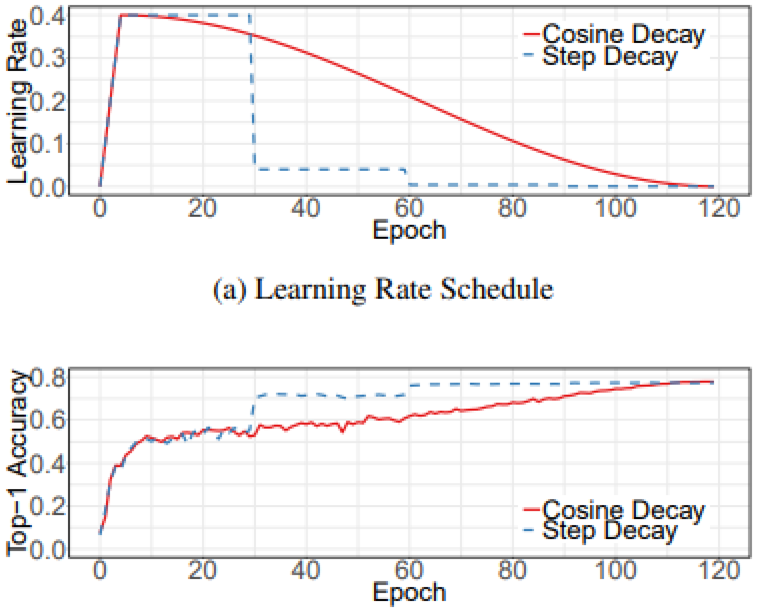
cosine learning rate decay

- Learning rate adjustment is crucial to the training
- Loshchilov et al. [18] propose a cosine annealing strategy. An simplified version is decreasing the learning rate from the initial value to 0 by following the cosine function.
- Assume the total number of batches is T (the warmup stage is ignored), then at batch t, the learning rate η_t is computed as

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

Training Refinement

cosine learning rate decay



(a) Learning Rate Schedule

(b) Validation Accuracy

Figure 3: Visualization of learning rate schedules with warm-up. Top: cosine and step schedules for batch size 1024. Bottom: Top-1 validation accuracy curve with regard to the two schedules.

Training Refinement

Label smoothing

```
If label_smoothing is nonzero
smooth the labels towards 1 / num_classes:
new_onehot_labels = onehot_labels * (1 - label_smoothing) + label_smoothing / num_classes
```

```
tf.losses.softmax_cross_entropy(
    onehot_labels,
    logits,
    weights=1.0,
    label_smoothing=0,
    scope=None,
    loss_collection=tf.GraphKeys.LOSSES,
    reduction=Reduction.SUM_BY_NONZERO_WEIGHTS
)
```

Training Refinement

Knowledge distillation

- Teacher Model = ResNet-152
- Student Model = ResNet-50
- add a distillation loss to penalize the difference between the softmax outputs from the teacher model and the learner model
- z and r are outputs of the last fully-connected layer of the student model and the teacher model

negative cross entropy loss $\ell(p, \mathrm{softmax}(z))$ to measure the difference between p and z, here we use the same loss again for the distillation. Therefore, the loss is changed to

$$\ell(p, \mathrm{softmax}(z)) + T^2 \ell(\mathrm{softmax}(r/T), \mathrm{softmax}(z/T)), \tag{6}$$

where T is the temperature hyper-parameter to make the softmax outputs smoother thus distill the knowledge of label distribution from teacher's prediction.

Training Refinement

Mixup training

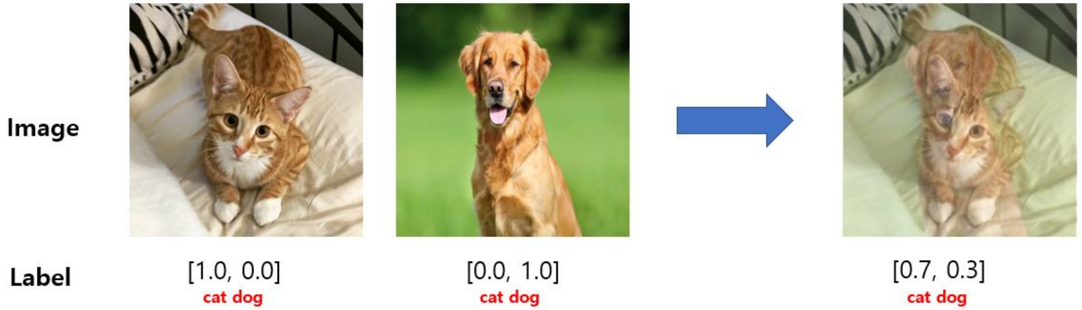
- randomly sample two examples (xi , yi) and (xj , yj). Then we form a new example by a weighted linear interpolation of these two examples:

$$\hat{x} = \lambda x_i + (1 - \lambda) x_j, \tag{7}$$
$$\hat{y} = \lambda y_i + (1 - \lambda) y_j, \tag{8}$$

where $\lambda \in [0, 1]$ is a random number drawn from the $\textbf{Beta}(\alpha, \alpha)$ distribution. In mixup training, we only use the new example $(\hat{x}, \hat{y})$.

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

where $\lambda \in [0, 1]$ is a random number



Training Refinement

Experiment results

- Teacher model = ResNett-152-D
- Label smoothing ε = 0.1
- Model distillation T = 20
- Mixup α = 0.2

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

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**

Table 7: Results on both the validation set and the test set of MIT Places 365 dataset. Prediction are generated as stated in Section 2.1. ResNet-50-D Efficient refers to ResNet-50-D trained with settings from Section 3, and ResNet-50-D Best further incorporate cosine scheduling, label smoothing and mixup.

Training Refinement

Experiment results on Transfer Learning (Object Detection)

- Teacher model = ResNet-152-D
- Label smoothing ε = 0.1
- Model distillation T = 20
- Mixup α = 0.2

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**

Table 8: Faster-RCNN performance with various pre-trained base networks evaluated on Pascal VOC.

Training Refinement

Experiment results on Transfer Learning (Semantic Segmentation)

- Teacher model = ResNet-152-D
- Label smoothing $\varepsilon = 0.1$
- Model distillation $T = 20$
- Mixup $\alpha = 0.2$

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

Table 9: FCN performance with various base networks evaluated on ADE20K.

Conclusion

Bag of Tricks for Image Classification with Convolutional Neural Networks

- These tricks introduce minor modifications to the model architecture, data preprocessing, loss function, and learning rate schedule
- tricks improve model accuracy consistently
- stacking all of them together leads to a significantly higher accuracy
- strong advantages in transfer learning, which improve both object detection and semantic segmentation