Deep Learning: Coding Project 2

Yuan Ying, 2020010930

March 25, 2022

1 Model Details

The basic model is EfficientNet-b0, a mobile-size baseline proposed by Mingxing Tan et al. It can achieve equivalent classification accuracy to ResNet-50 and DenseNet-169 while using far less than a half number of parameters and FLOPs.

The main building block is mobile inverted bottleneck MBConv, to which squeeze-and-excitation optimization is also added. MBConv decomposes the regular Conv+BN+ReLU to DepthwiseConv+BN+ReLU+ PointwiseConv+BN+ReLU.

The model can be divided to the following stages:

1. Conv3x3 - 1 layer
2. MBConv1, k3x3 - 1 layer
3. MBConv6, k3x3 - 2 layers
4. MBConv6, k5x5 - 2 layers
5. MBConv6, k3x3 - 3 layers
6. MBConv6, k5x5 - 3 layers
7. MBConv6, k5x5 - 4 layers
8. MBConv6, k3x3 - 1 layer
9. Conv1x1 & Pooling & FC - 1 layer

2 Hyperparameters

Batch size: 32
Optimizer: Adam
Learning rate: Start from 1e-3, use CosineAnnealingLR with T_max=300, η_min=0
Weight decay: 0
Epoch number: 300

3 Training techniques

3.1 Data augmentation

To learn more robust features in the dataset, we use the data augmentation technique. For each training data, we first randomly choose from RandomResizedCrop, RandomAffine and RandomHorizontalFlip, and apply it to the picture. Then we apply RandomGrayscale with p=0.1. In this way we can boost the validation accuracy by approximately 2%.

3.2 Learning Rate Scheduler

Compared with MultiStepLR, CosineAnnealingLR is more stable and requires less manual finetuning. So we choose CosineAnnealingLR, whose updating policy is

$$\alpha(t) = \eta_{min} + 0.5(\alpha_0 - \eta_{min}) \left(1 + \cos\left(\frac{t}{T_{max}}\pi\right)\right), \tag{1}$$

where t is the current epoch. In this case, learning rate changes periodically with period $2T_{max}$. Using scheduler we can boost the validation accuracy by $1\% \sim 2\%$.

3.3 Quantization

Since the size of EfficientNet-b0 is 32M, which is still larger than the required size 20M, we use quantization method to save the parameters in linear layers in float16 type rather than the original float32 type. In this way we can achieve the required size while maintaining a rather good accuracy.

4 Training curve

The training curves are shown in Figure 1, 2, 3. Our model achieves 91.2% accuracy on validation dataset and 97.5% accuracy on training dataset after 500 epochs of training.

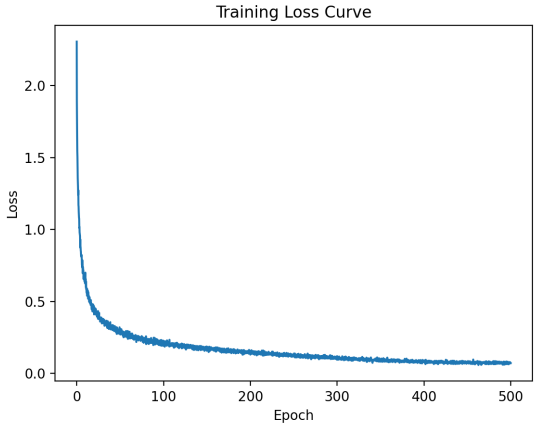

Figure 1: Training Loss Curve

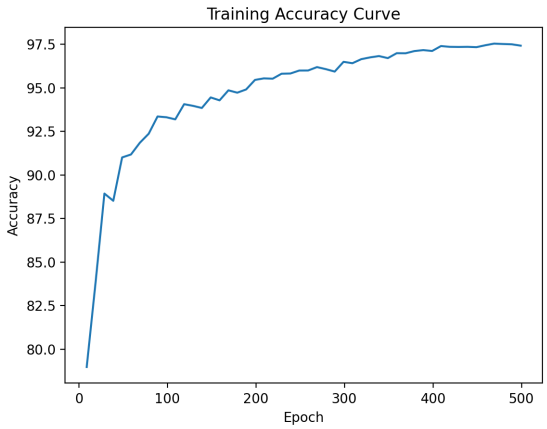

Figure 2: Training Accuracy Curve (per 10 epochs)

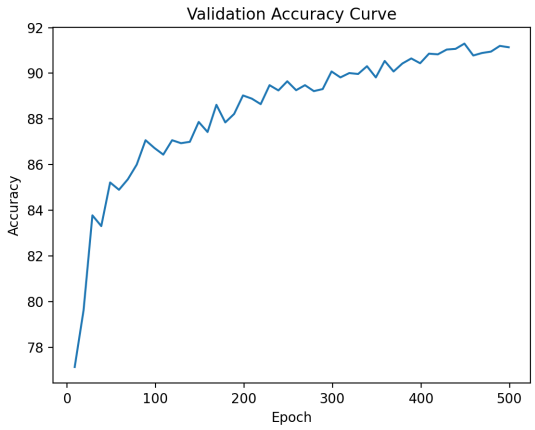

Figure 3: Validation Accuracy Curve (per 10 epochs)

5 Additional Ablation Studies

5.1 Other models

Other than EfficientNet, we also experiment on other models, such as ResNet18, DenseNet121, HCGNet, etc. Plain training achieves 83% with ResNet18, 76% with HCGNet, 82.3% with DenseNet121. Compared with them, plain training achieves a better 85.43% with EfficientNet-b0.

5.2 Hyperparameter tuning

Batch size: Based on EfficientNet-b0 and no other tricks, our training achieves 85.34% with batch size 16, 85.43% with batch size 32, and 84.56% with batch size 64. Since larger batch size typically learns more robust features and trains faster, we use 32 as our batch size.

Learning rate: Based on EfficientNet-b0 and batch size 32, we compare MultiStepLR with CosineAnnealingLR. We analyze the learning curve without learning rate scheduler, and purposefully set milestone

as (70, 90) and $\gamma = 0.4$ in MultiStepLR. As for CosineAnnealingLR, we simply set T_{max} as the number of epochs. Training with MultiStepLR achieves 86.36% while training with CosineAnnealingLR achieves 86.85% accuracy. Since CosineAnnealingLR performs better and needs no manual tuning on milestone, we use it as our learning rate scheduler.

Weight decay: Since the current model doesn't seem to overfit and applying weight decay avoids it from learning as much as before, we choose not to use weight decay.

5.3 Data augmentation

Based on EfficientNet-b0 and batch size 32, we add data augmentation to test its effect. With RandomResizedCrop, RandomAffine, RandomHorizontalFlip and RandomGreyScale, the training achieves 86.85%, boosting accuracy by approximately 2%.

5.4 Other tries

We also consider the idea that first downsample the input data by a half, giving faster training with ResNet18 and getting an opaque understanding of the images, and then train the model with high-resolution images. Due to limited time and computational resources, we have not experimented on this idea yet.