Machine Learning Homework 2

Jia-Wei Liao, 309652008
jw.sc09@nycu.edu.tw
Department of Applied Mathematics, NYCU

April 18, 2022

1 Introduction

In this homework, we implement deep learning-based techniques to do CIFA100 image classification. The CIFA100 dataset has 100 classes containing 60,000 images each. We use the EfficientNet [6] and Swin Transformer [4] models, which are the state-of-the-art of the CNN-base and VIT-base models in recent years. Moreover, we attempt to use the different loss function such as cross-entropy loss (CE), mutual channel with cross-entropy loss (MCCE) [1] loss, focal loss (FL) [3], and adaptive focal loss (FLSD53) [5]. After experimenting, the test accuracy of EfficientNet-B4 and Swin Transformer are 89.52% and 93.60 %. Due to the capacity limit, we can only upload the model parameters of EfficientNet-B4. We release the code to Github. It is available at https://github.com/Jia-Wei-Liao/Machine_Learning/tree/main/HW2.

2 Method

2.1 Data pre-processing

2.1.1 Image Normalize

The images are represented as tensors with pixel values ranging from 0 to 255. We apply z-score transform to the pixel values with mean $= (0.485, 0.456, 0.406)$ and std $= (0.229, 0.224, 0.225)$. Since the model's weight is pre-trained on ImageNet, we choose the mean and std as the same as ImageNet.

2.1.2 Data augmentation

To avoid over-fitting and enhance the robustness of a model, we use the following data augmentation techniques:

- **Random rotate 10 degrees**
- **Random flip with horizontal**
- **Random add Gaussian noise:**
  We add the noise with probability by $I(i,j) + \sigma \cdot s$, where $s \sim \mathcal{N}(0,1)$ and I is the grayscale.
- **Auto-augmentation:** We set some policy, which is composed of rotation, sharpness, shear, translation, brightness, etc. Then we randomly choose the policy to transform the image.

2.2 Model architecture

2.2.1 EfficientNet

EfficientNet [6] is proposed in 2019. Scaling up convolution networks is widely used to achieve better accuracy. For example, ResNet [2] scaling up the network's depth (ResNet18 to ResNet152). Wide Residual Network (WRN) [7] scaling up the network's width (increasing the number of output channels). Moreover, scaling up models by image resolution also gives better accuracy. The goal of EfficientNet is to scale up depth/width/resolution simultaneously, as shown in Figure 2. Intuitively, scaling up all of them makes sense. Once the input image becomes larger, the model needs more layers to ensure the receptive fields are large enough and more channels to capture more features. Finally, EfficientNet's accuracy gives a great accuracy, as shown in Figure 1.

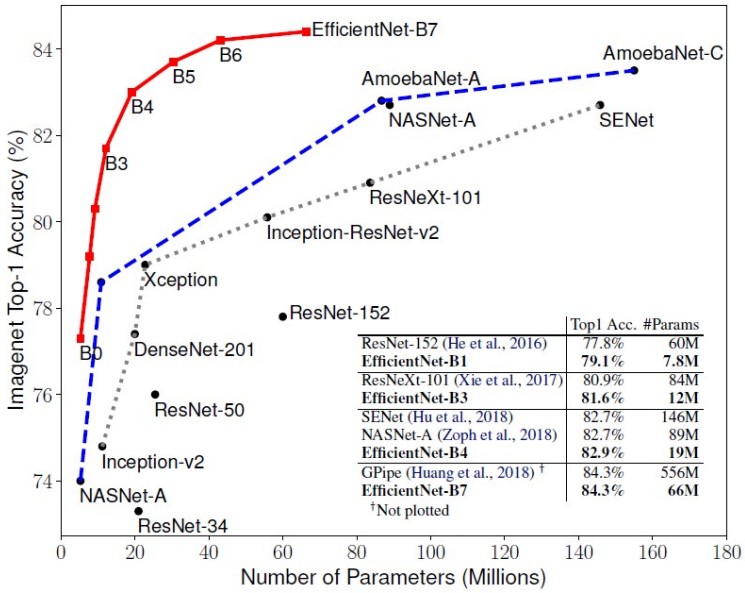

Figure 1: Model size vs. ImageNet accuracy

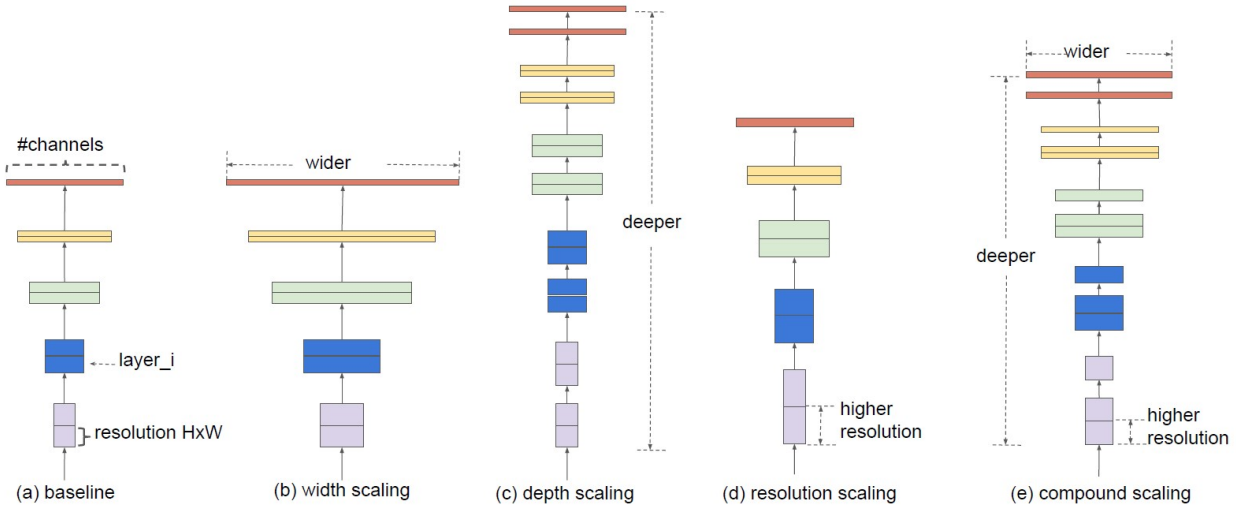

Figure 2: Model scaling

In the following, we show the code of the EfficientNet-B4. Since the weights are pre-trained on ImageNet and the input size is $224 \times 224 \times 3$, we use upsampling layer before inputting the EfficientNet-B4. On the other, We take out the feature map and let the subsequent MC loss use it.

```
class EfficientNet_b4(nn.Module):
    def __init__(self, num_classes=219):
        super(EfficientNet_b4, self).__init__()
        model = efficientnet_b4(pretrained=True)
        model.classifier[1] = torch.nn.Linear(
            model.classifier[1].in_features, num_classes)
        model_child = list(model.children())
        self.feature_extr = nn.Sequential(*model_child[:-2])
        self.avgpool = model_child[-2]
        self.classifier = model_child[-1]
        self.upsample = nn.Upsample(size=(224, 224))


    def forward(self, inputs, use_mc_loss=False, *args, **kwargs):
        x = self.upsample(inputs)
        feature = self.feature_extr(x)
        x = self.avgpool(feature)
        x = x.view(x.size(0), -1)
        outputs = self.classifier(x)

        if use_mc_loss:
            return outputs, feature

        else:
            return outputs
```

2.2.2 Swin Transformer

Swin Transformer [4] is proposed in 2021, which achieved the best paper award of ICCV 2021. It uses the window-based multi-head self-attention (W-MSA) to reduce linear time complexity. To reinforce the connectivity of different windows, it proposes a window shift mechanism. In Figure 3, we show the architecture of Swin Transformer. There are four stages in the Swin Transformer, and each stage is composed of a patch merging layer and two successive Swin Transformer blocks. The Swin Transformer blocks are the feature extractor, which output size maintains input size. The patch merging layer uses the inverse pixel shuffle to attain the dimension reduction. Its role is as a maximum poling in the CNN model. In Figure 4, we show the Swin Transformer block, which is composed of layer normalization (LN), (shift) window-based multi-head self-attention ((S)W-MSA), and multilayer perceptron (MLP). Moreover, it also has the residual mechanism after (S)W-MSA and MLP layers.

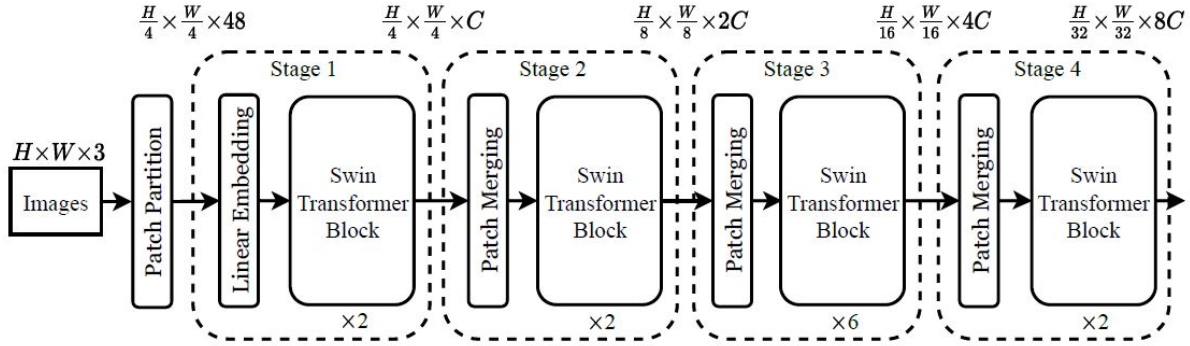

Figure 3: Architecture of Swin Transformer

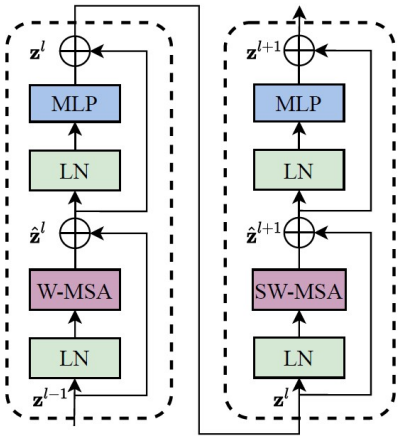

Figure 4: Two successive Swin Transformer block

2.3 Loss function

The most popular loss function in classification tasks is Cross-Entropy, and there are many loss functions modified based on it. In the following, we introduce them in detail.

2.3.1 Cross-Entropy Loss (CE)

Let y be the probability of prediction, y^{gt} be the one-hot label and C be the number of category. Cross-Entropy Loss is defined by

$$\mathcal{L}_{CE}(y, y^{gt}) = -\sum_{i=1}^{C} y^{gt} \log y_i$$

2.3.2 Mutual Channel Loss (MC)

Mutual Channel Loss is proposed in February 2020, which solves the fine-grained image classification problem. This method can be applied directly to the ResNet50 model. In this homework, we use the EfficientNet-b4. As shown in Figure 5, the only thing we need to do is access the layer before the average pooling layer. MC-loss attempt to represent each class by several feature channels. For example, there are 100 classes in total in this task, so there are 8 (9) channels to represent each of the first 179 (last 49) classes, respectively. Mutual Channel Loss is composed of two main components:

- Force all feature channels belonging to the same class to be discriminative.
- Force the feature channels in the same class to discover different discriminative regions.

$$\mathcal{L}_{MC} = \mathcal{L}_{CE} + \mu(\mathcal{L}_{dis} - \lambda\mathcal{L}_{div})$$

where μ, λ are hyper-parameters.

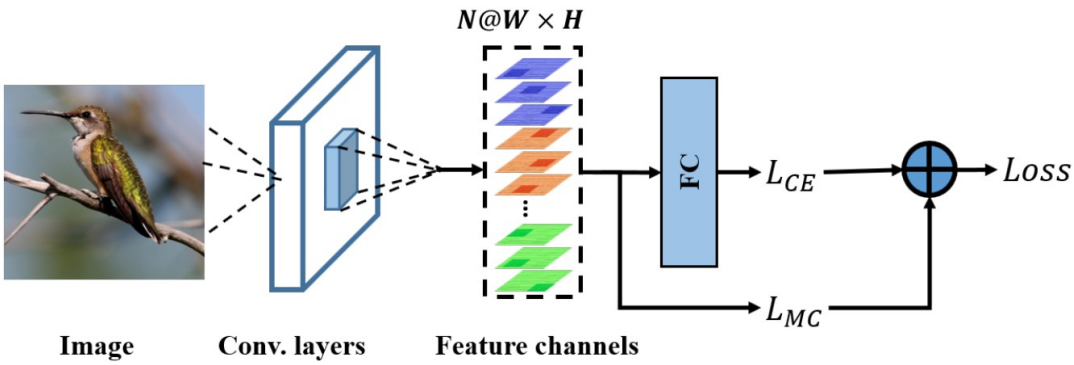

Figure 5: MC loss framework

2.3.3 Focal Loss (FL)

Focal Loss was proposed in 2018, which is used to solve the data imbalance problem. Since the candidate object in a picture has most of the proportions of the background rather than the foreground, there will be an imbalance in calculating the loss. The Focal Loss is down-weighted for the easy example. The hard example can be trained as much as possible during the training process and ignored those easy examples. So it can solve the imbalanced category problem. Let y, y^{gt} be the predicted category and ground truth category. Focal Loss is defined as

$$\mathcal{L}_{FL}(y, y^{gt}) = -(1 - p^t)^\gamma \log p_t$$

where $p_t = \begin{cases} y, & \text{if } y^{gt} = 1 \\ 1 - y, & \text{otherwise} \end{cases}$ and $\gamma \geq 0$ is called focusing parameter.

Why the focal loss can address data imbalance?

We believe that the data which rarely appears would get a lower probability of ground class. On the contrary, the frequently appearing data would highly support ground class likelihood. We focus on the blue line and purple link in Figure 6. The blue line is cross-entropy loss which is a particular case of the focal loss. The purple line is focal loss with $\gamma = 2$. Compared with the two lines, if we set 0.2, 0.6 to the probability of ground truth class, the blue line would get 1.6, 0.5 of the loss, and the purple line would get 1.05, 0.05 of the loss. We discover the differences in the purple line are more significant by the following equation

$$\frac{1.6}{0.5} = 3.2 < 21 = \frac{1.05}{0.05}$$

This shows the focal loss can highlight the data which perform worst, and it doesn't have to work hard on good performance data.

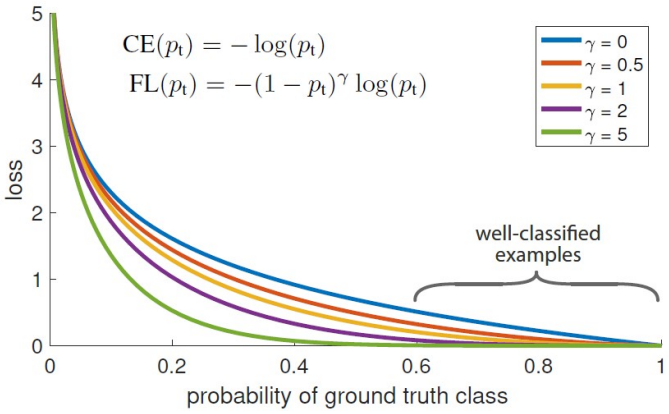

Figure 6: Focal Loss

2.4 Optimization

We use AdamW as the optimizer. It is a stochastic optimization method that modifies Adam's typical implementation of weight decay by decoupling weight decay from the gradient update. We give the algorithm like the following:

Algorithm AdamW

Input: $f(\theta)$ (objective), γ (lr), $\beta_1, \beta_2, \theta_0, \varepsilon, \lambda$ (weight decay)
Initial: $m_0 \leftarrow 0$ (first moment), $v_0 \leftarrow 0$ (second moment),
for $t = 1$ to ... **do**
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  $\theta_t \leftarrow \theta_{t-1} - \gamma\lambda\theta_{t-1}$
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
  $\hat{m}_t \leftarrow \dfrac{m_t}{1-\beta_1^t}$
  $\hat{v}_t \leftarrow \dfrac{v_t}{1-\beta_2^t}$
  $\theta_t \leftarrow \theta_{t-1} - \gamma \dfrac{\hat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}}$
end for
return θ_t

2.5 Learning Rate Scheduler

To further improve results and let the model converge to the global minimum, we adjust the learning rate by exponent decay. It represent by the following:

$$\text{Learning Rate} = (\text{Initial Learning Rate}) \times (\text{Decay Factor})^{\frac{\text{Epoch}}{\text{Decay Period}}}$$

3 Experiments results

All the experiments are set to a learning rate decay factor of 0.8, learning decay period of 3, and weight decay of $1e-4$. In Figure 7, 8, we show the curve of test accuracy of different loss functions and different models.

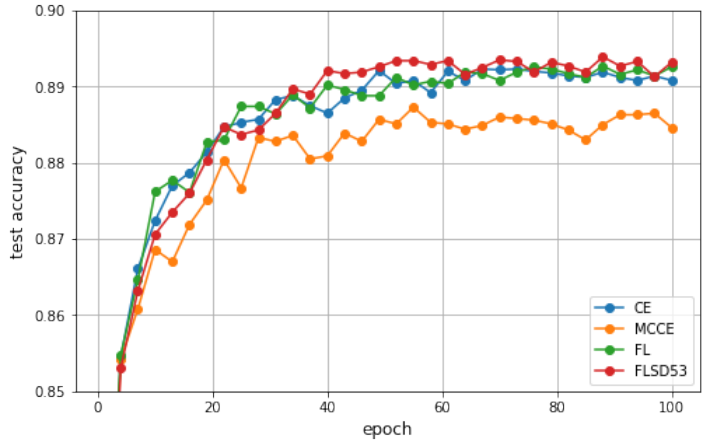

Figure 7: Comparing the curve of accuracy with different loss function

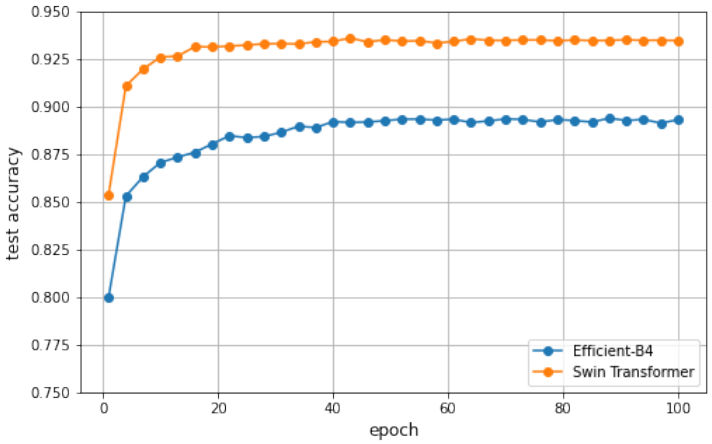

Figure 8: Comparing the curve of accuracy with different model

In Table 1, we show the test accuracy of our experiments with a different model, batch size, loss function, and learning rate.

model	batch size	epoch	upsampling	loss function	learning rate	accuracy
EfficientNet-B4	32	200	224	CE	$1e-3$	88.57%
EfficientNet-B4	64	200	224	CE	$1e-3$	89.34%
EfficientNet-B4	32	200	224	MC	$1e-3$	88.37%
EfficientNet-B4	64	200	224	MC	$1e-3$	88.73%
EfficientNet-B4	32	200	224	FL	$1e-3$	88.24%
EfficientNet-B4	64	200	224	FL	$1e-3$	89.35%
EfficientNet-B4	32	200	224	FLSD53	$1e-3$	88.90%
EfficientNet-B4	64	200	224	FLSD53	$1e-3$	**89.52%**
Swin-B	64	200	384	FL	$3e-5$	93.56%
Swin-B	64	200	384	FLSD53	$3e-5$	**93.60%**

Table 1: Test accuracy

4 Conclusion

In this homework, we implement the EfficientNet-b4 and Swin-Transformer model. We use AdamW and the learning rate scheduler for the optimization part to let our model converge to a better location. To increase the data diversity, we use random flip, random rotation, random adding noise, and auto-augmentation, which is helpful to enlarge our dataset. Moreover, we attempt to use many loss functions, such as cross-entropy loss (CE), the mutual channel with cross-entropy loss (MCCE) loss, focal loss (FL), and adaptive focal loss (FLSD53), which are powerful in the classification task. Finally, we have a test accuracy of 89.35% and 93.60 % of EfficientNet-B4 and Swin-B. Due to the capacity limit, we choose the EfficientNet-B4 result to hand in this homework.

5 Future work

We have also tried the ranger optimizer, which combines the RAdam and Look ahead mechanism. Unfortunately, the result was not great. We got the test accuracy of only about 80%. We think some hyperparameters we choose are inappropriate, so we hope to use them successfully in the future.

6 Code on GitHub

Since We have too much code to fit in the report, we provide the link below. If you want to use the checkpoint to do the testing, you must use the torch 1.10.2 version, otherwise it will get error message.

Auto-augmentation	link
Swin Transformer	link
Mutual Channel Loss	link
Focal Loss	link
Ranger Optimizer	link

Table 2: Link of our code

References

[1] Dongliang Chang, Yifeng Ding, Jiyang Xie, Ayan Kumar Bhunia, Xiaoxu Li, Zhanyu Ma, Ming Wu, Jun Guo, and Yi-Zhe Song. The devil is in the channels: Mutual-channel loss for fine-grained image classification. *IEEE Transactions on Image Processing*, 29:4683–4695, 2020.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[3] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[4] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, October 2021.

[5] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating deep neural networks using focal loss. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15288–15299. Curran Associates, Inc., 2020.

[6] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.

[7] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016.