Visual Recognitionusing Deep Learning

2025 Spring, Homework 1

313551078 吳年茵

Github : `https://github.com/nianyinwu/CV_HW1`

1. Introduction

In this lab, we have to design and train a ResNet-based [1] architecture as the backbone of our model to solve an image classification task with 100 object categories. The total number of model parameters is limited to less than 100 million. The dataset consists of 21,024 RGB images for training and validation and 2,344 for testing.

To improve the model's classification performance for this task, I adopt ResNeXt [2] [3], an advanced variant of ResNet, as the backbone due to its strong feature representation capabilities. I also use pre-trained weights from ImageNet to converge better and learn current training data based on previously learned visual features. Additionally, I experiment with architectural modifications, such as adding extra fully connected layers to the classifier head and introducing the Convolutional Block Attention Module (CBAM) [4], which sequentially applies channel attention followed by spatial attention that is intended to help the model emphasize informative features across both dimensions and enhance its ability to focus on meaningful regions within the input.

2. Method

A. Data Preprocessing

I do some data preprocessing and augmentation techniques to the data using PyTorch's transforms module [5].

The preprocessing process of training data is as follows:

1. Resize the input image to 512×512.
2. Randomly apply one of the following augmentations:
    - Rotation up to 45 degrees.
    - Color jitter with brightness, contrast, and saturation variation 0.2.
    - Horizontal flip with a probability of 0.2.
    - Vertical flip with a probability of 0.2.
3. Convert the image into a PyTorch tensor.
4. Normalize the image using mean=[0.485, 0.456, 0.406] and standard deviation=[0.229, 0.224, 0.225].

The preprocessing process of validation and testing data is as follows:

1. Resize the input image to 512×512.
2. Convert the image to a PyTorch tensor.
3. Normalize the image using mean=[0.485, 0.456, 0.406] and standard deviation=[0.229, 0.224, 0.225].

B. Model Architecture

B.1. Architecture before modification

The ResNet-50 architecture consists of an initial convolution and max pooling layer, followed by four stages (stage 2 - stage 5) with increasing channel dimensions. It ends with a global average pooling and a fully connected layer for classification. Moreover, in this lab, I chose ResNeXt-50 as the model backbone, a ResNet variant. ResNeXt-50 increased the number of convolution output channels in all stages 2 to 5 and uses grouped convolutions to enhance the model's feature representation capability while not increasing model complexity (parameters). In the final stage (stage 6), I only modify the fully connected layer from 1000 output nodes to 100 to match the number of target categories. I also used pre-trained weights from ImageNet to converge better and learn

current training data based on previously learned visual features. The architecture that has not been modified yet is shown in Fig. 1.

Stage	ResNet50	ResNeXt50
1	7x7, 64, stride 2	
2	3x3 max pool, stride 2	
	[1x1, 64 3x3, 64 1x1, 256] x3	[1x1, 128 3x3, 128, C=32 1x1, 256] x3
3	[1x1, 128 3x3, 128 1x1, 512] x4	[1x1, 256 3x3, 256, C=32 1x1, 512] x4
4	[1x1, 256 3x3, 256 1x1, 1024] x6	[1x1, 512 3x3, 512, C=32 1x1, 1024] x6
5	[1x1, 512 3x3, 512 1x1, 2048] x3	[1x1, 1024 3x3, 1024, C=32 1x1, 2048] x3
6	Global average pool 1000-d fc, softmax	Global average pool 1000-d fc, softmax

Fig. 1. Architecture before modification

B.2. Architecture after modification (Additional Experiments)

In this section, I present two parts I want to modify in the ResNeXt-50 model as part of additional experiments. I also provide the motivations behind each modification and the corresponding hypotheses.

First, I add two fully connected layers in the final stage (stage 6) to enhance the model's ability to learn feature representations from the training data.(Fig. 2) After adding each fully connected layer, I add the Batch Normalization and Dropout to avoid overfitting. In addition, I use SiLU [6] as the activation function after each Batch Normalization layer to introduce non-linearity since it is characteristic of smooth and non-monotonic behavior and allows preserving small gradients that help the model learn more expressive and complex feature mappings, especially in deeper architectures.

```
(fc): Sequential(
  (0): Linear(in_features=2048, out_features=1024, bias=True)
  (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (2): SiLU()
  (3): Dropout(p=0.3, inplace=False)
  (4): Linear(in_features=1024, out_features=512, bias=True)
  (5): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (6): SiLU()
  (7): Dropout(p=0.3, inplace=False)
  (8): Linear(in_features=512, out_features=100, bias=True)
)
```

Fig. 2. Added fully connected layer

Second, I add the Convolutional Block Attention Module (CBAM) after stage 5. Since CBAM sequentially applies channel attention followed by spatial attention (Fig. 3),

it helps the model learn informative refined features along both the channel and spatial dimensions, enhancing its ability to focus on meaningful regions within the input to improve classification performance.

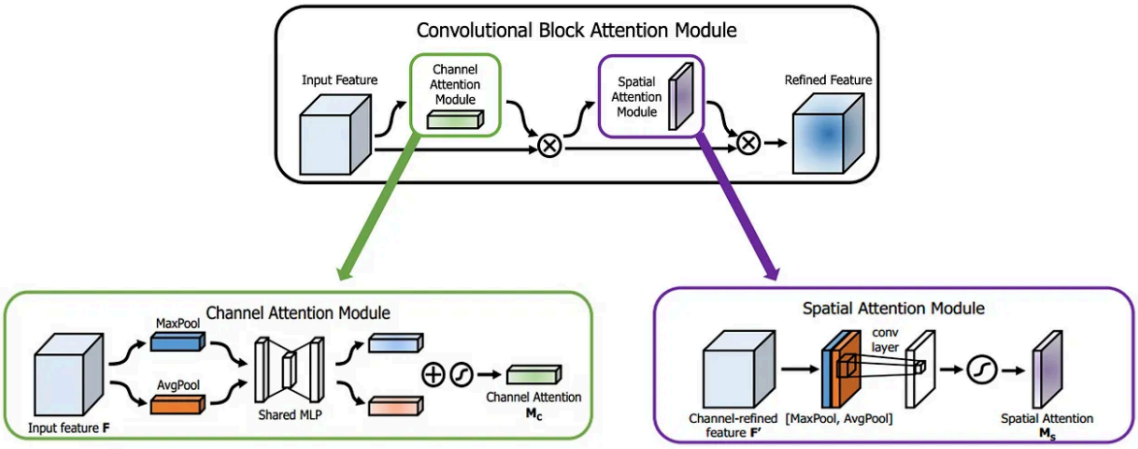


Fig. 3. CBAM module from [7]

C. Hyperparameters Setting

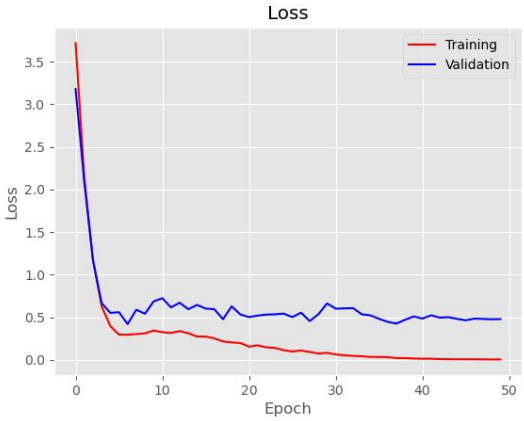
In the experiments, I use the following hyperparameter settings to train the model:

- Learning rate: 1e-4
- Batch size: 16
- Number of epochs: 50
- Dropout rate: 0.3
- Optimizer: Adam
- Loss function: Cross-Entropy Loss
- Learning rate scheduler: ReduceLROnPlateau with factor:0.1, patience:3 (dynamically reduces learning rate when validation loss stops improving)

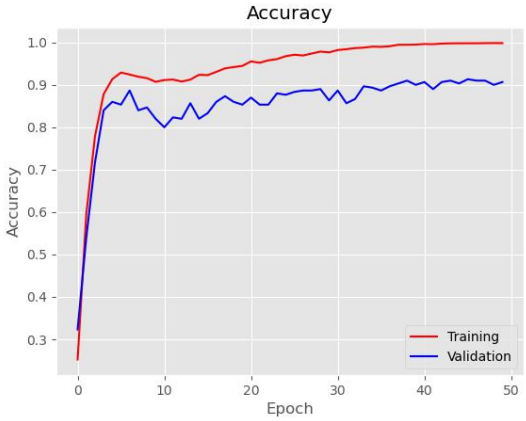
3. Experiment Results

To explore better performance, I conducted four different model settings: (1) ResNeXt-50 (fully connected layer only has 100 nodes) (2) ResNeXt-50 and added two fully connected layers in the classifier head (3) ResNeXt-50 added the CBAM module after stage 5. (4) ResNeXt-50 with both addition layer and module.

1. ResNeXt-50 (baseline)

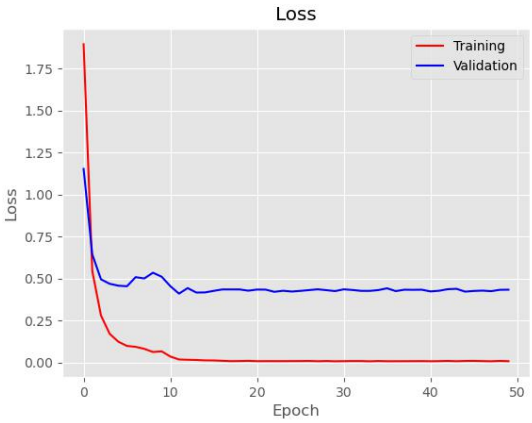


(a) Loss Curve

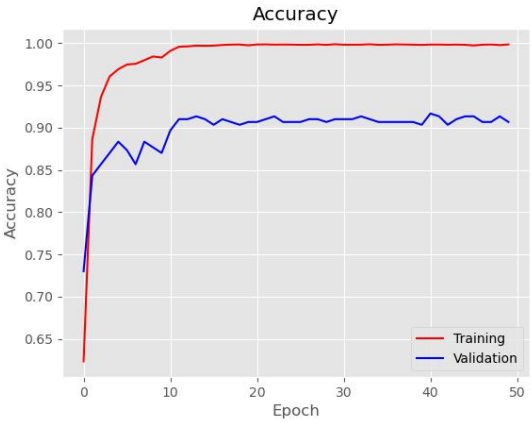


(b) Accuracy Curve

2. ResNeXt-50 + two fully connect layers

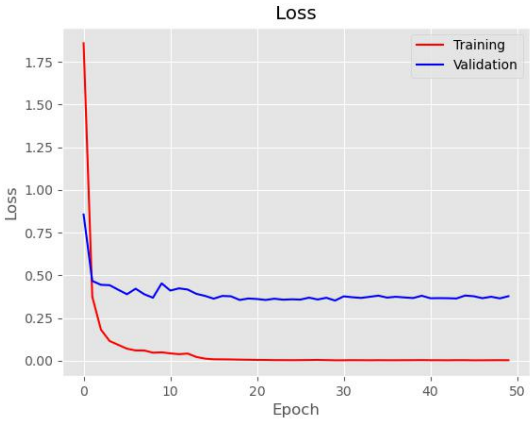


(a) Loss Curve

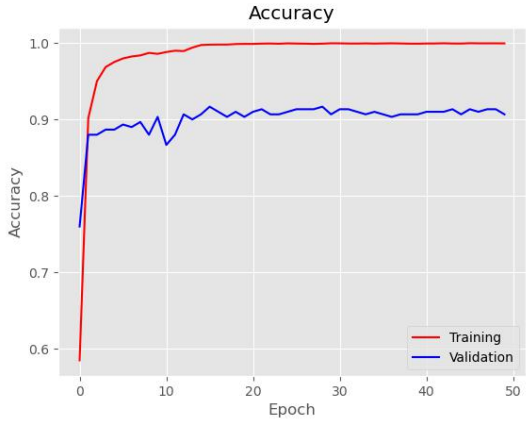


(b) Accuracy Curve

3. ResNeXt-50 + CBAM module

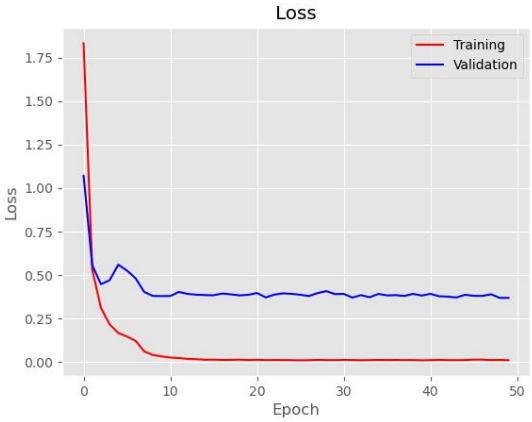


(a) Loss Curve

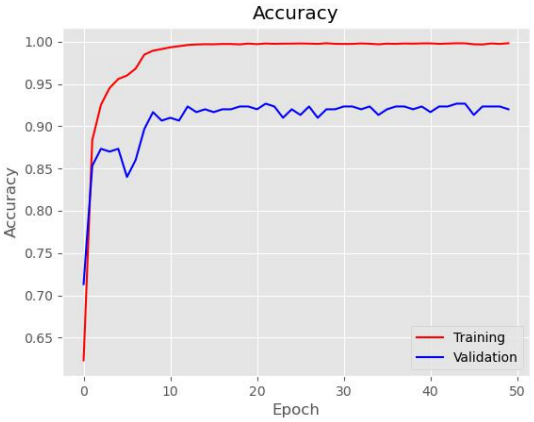


(b) Accuracy Curve

4. ResNeXt-50 + two fully connect layers + CBAM module



(a) Loss Curve



(b) Accuracy Curve

Compared to the results of the four different model configurations, the first setting, ResNeXt-50 (baseline), which only modified the fully connected layer to 100 nodes, can achieve high training accuracy. However, the validation accuracy is only **91%**, and the loss curve shows apparent fluctuation; thus, the model has an overfitting problem in this setting. As a result of the second setting that only added two fully connected layers to the classifier head, the validation accuracy **exceeded 92%**, and the validation loss became smoother and dropped below 0.5. This indicates that the model was able to learn more fine-grained features for classification, and the follow-up Batch Normalization and Dropout after each fully connected layer also helped prevent overfitting and stabilized the training process. As shown in the results of the third setting, it has a minor loss compared to the second. I initially thought the third setting might improve accuracy since the CBAM module can help the model focus on the most informative regions. However, the accuracy shows it was slightly smaller than the additional fully connected layers (**approximately 92%**), but the results were still better than the baseline. Last,

combining additional layers and modules has the best results in validation accuracy; it can **reach over 93%**, and validation loss is similar to that of the third setting. This indicates that the two components can offer complementary benefits: CBAM enhances attention and feature selection, while the added FC layers strengthen the learning capacity of the classifier. Together, they allow the model to capture meaningful patterns in the training data better while maintaining strong generalization on unseen data. Thus, I chose the ResNeXt-50 with two fully connected layers and the CBAM module as the final model for this task.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[2] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks, 2017.

[3] https://pytorch.org/vision/main/models/resnext.html.

[4] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module, 2018.

[5] http://pytorch.org/vision/main/transforms.html.

[6] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, 2017.

[7] https://medium.com/@andy6804tw/通道空間注意力-cbam-805deb9ebe1c.