**Computer Vision HW2 Report**

Student ID: b10901091

Name: 李冠儀

**Part 1. (10%)**

**• Plot confusion matrix of two settings. (i.e. Bag of sift and Tiny image) (5%)**

**Ans:**

|  |  |
| --- | --- |
| **Bag of sift** | **Tiny image** |
|  |  |

**• Compare the results/accuracy of both settings and explain the result. (5%)Ans:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | Tiny image | | Bag of sift | |
| Classifier | Nearest neighbor | Random classifier | Nearest neighbor | Random classifier |
| Accuracy | 0.398 | 0.0633 | 0.666 | 0.0673 |

**Explain:**

Classifier: The Nearest Neighbor classifier attempts to find the nearest neighbor, which is more likely to belong to the same class. Therefore, the accuracy of the Nearest Neighbor classifier is much better than that of a random classifier.

Feature: “Bag of sift” extracting local features from images using the Scale-Invariant Feature Transform (SIFT) algorithm, so it can preserve more detailed information than “Tiny image”.

**Part 2. (25%)**

**• Report accuracy of both models on the validation set. (2%)**

**Ans:**

Accuracy of Mynet = 0.8208

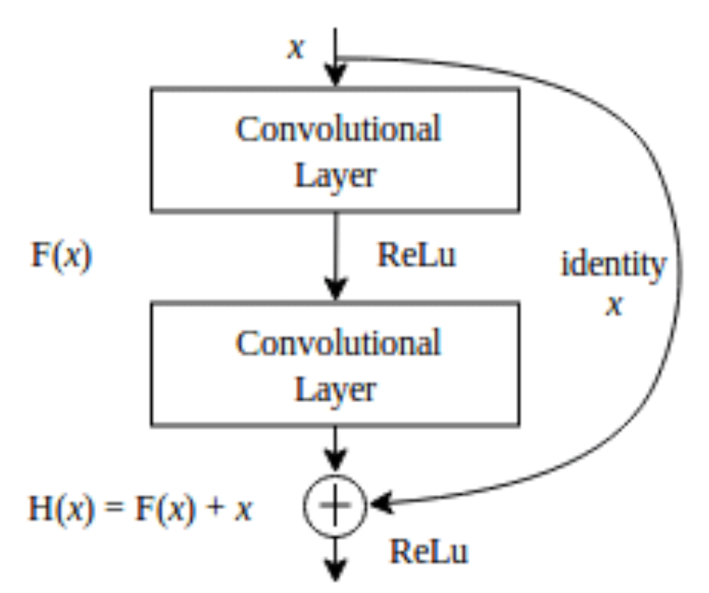
Accuracy of Resnet18 = 0.9060

**• Print the network architecture & number of parameters of both models. What is the main difference between ResNet and other CNN architectures? (5%)**

**Ans:**

Main difference:

The picture below illustrates the simplified architecture of ResNet, there are residual connections so it can solve the problem of vanishing gradients. These residual connections allow the network to bypass certain layers, facilitating the flow of gradients during backpropagation. As a result, even in very deep networks, gradients can propagate effectively, enabling better optimization and training of the model.



Number of parameter:

MyNet :3659402

Resnet18:11173962

Network architecture:

MyNet(

(conv1): Conv2d(3, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(relu1): ReLU()

(pool1): MaxPool2d(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)

(conv2): Conv2d(64, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(relu2): ReLU()

(pool2): MaxPool2d(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)

(conv3): Conv2d(128, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(relu3): ReLU()

(pool3): MaxPool2d(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)

(conv4): Conv2d(256, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(relu4): ReLU()

(pool4): MaxPool2d(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)

(fc1): Linear(in\_features=2048, out\_features=1024, bias=True)

(relu5): ReLU()

(fc2): Linear(in\_features=1024, out\_features=10, bias=True)

)

ResNet18(

(resnet): ResNet(

(conv1): Conv2d(3, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(relu): ReLU(inplace=True)

(maxpool): Identity()

(layer1): Sequential(

(0): BasicBlock(

(conv1): Conv2d(64, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(relu): ReLU(inplace=True)

(conv2): Conv2d(64, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

)

(1): BasicBlock(

(conv1): Conv2d(64, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(relu): ReLU(inplace=True)

(conv2): Conv2d(64, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

)

)

(layer2): Sequential(

(0): BasicBlock(

(conv1): Conv2d(64, 128, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(relu): ReLU(inplace=True)

(conv2): Conv2d(128, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(downsample): Sequential(

(0): Conv2d(64, 128, kernel\_size=(1, 1), stride=(2, 2), bias=False)

(1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

)

)

(1): BasicBlock(

(conv1): Conv2d(128, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(relu): ReLU(inplace=True)

(conv2): Conv2d(128, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

)

)

(layer3): Sequential(

(0): BasicBlock(

(conv1): Conv2d(128, 256, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(relu): ReLU(inplace=True)

(conv2): Conv2d(256, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(downsample): Sequential(

(0): Conv2d(128, 256, kernel\_size=(1, 1), stride=(2, 2), bias=False)

(1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

)

)

(1): BasicBlock(

(conv1): Conv2d(256, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(relu): ReLU(inplace=True)

(conv2): Conv2d(256, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

)

)

(layer4): Sequential(

(0): BasicBlock(

(conv1): Conv2d(256, 512, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(relu): ReLU(inplace=True)

(conv2): Conv2d(512, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(downsample): Sequential(

(0): Conv2d(256, 512, kernel\_size=(1, 1), stride=(2, 2), bias=False)

(1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

)

)

(1): BasicBlock(

(conv1): Conv2d(512, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(relu): ReLU(inplace=True)

(conv2): Conv2d(512, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

)

)

(avgpool): AdaptiveAvgPool2d(output\_size=1)

(fc): Linear(in\_features=512, out\_features=10, bias=True)

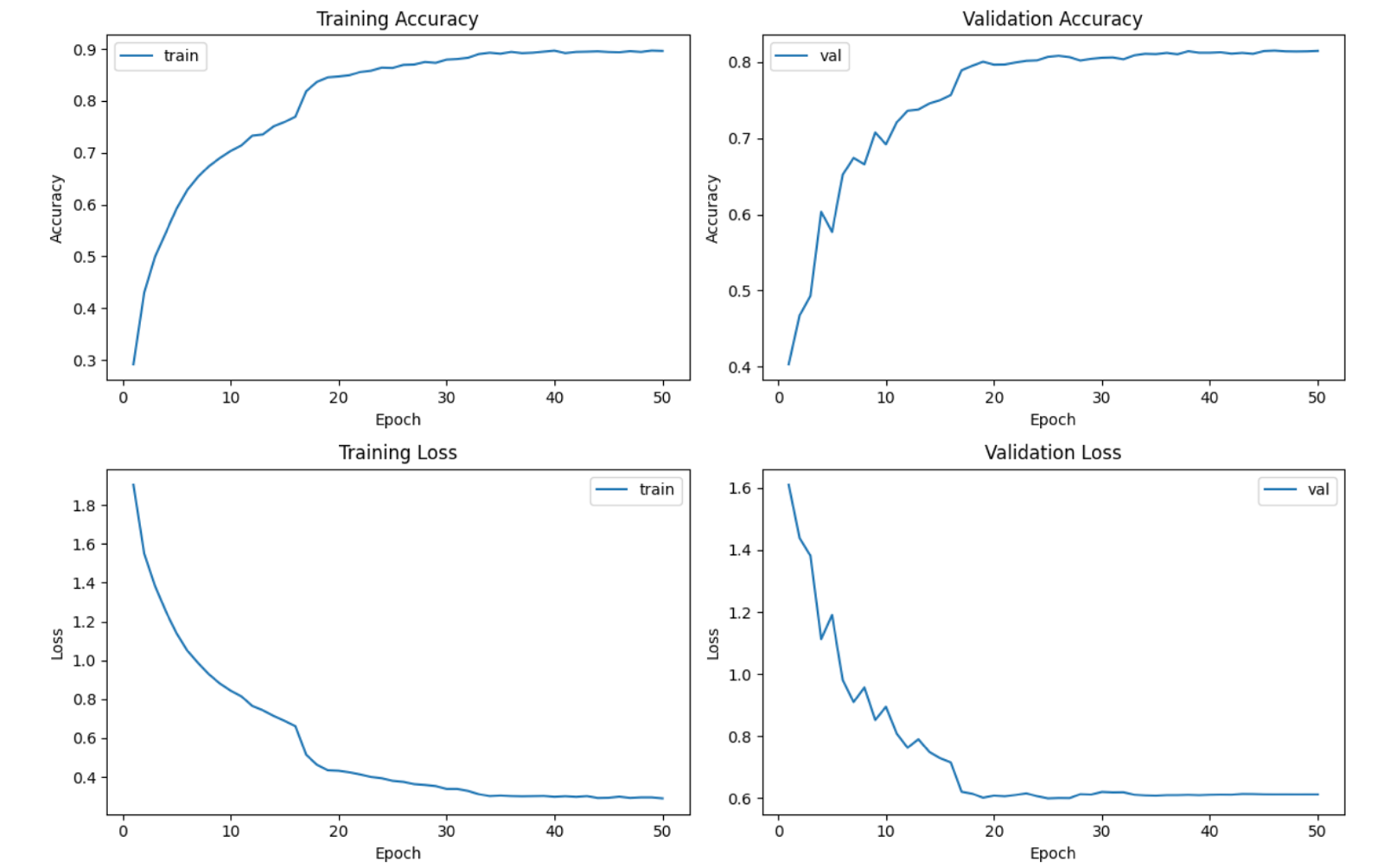
)

)

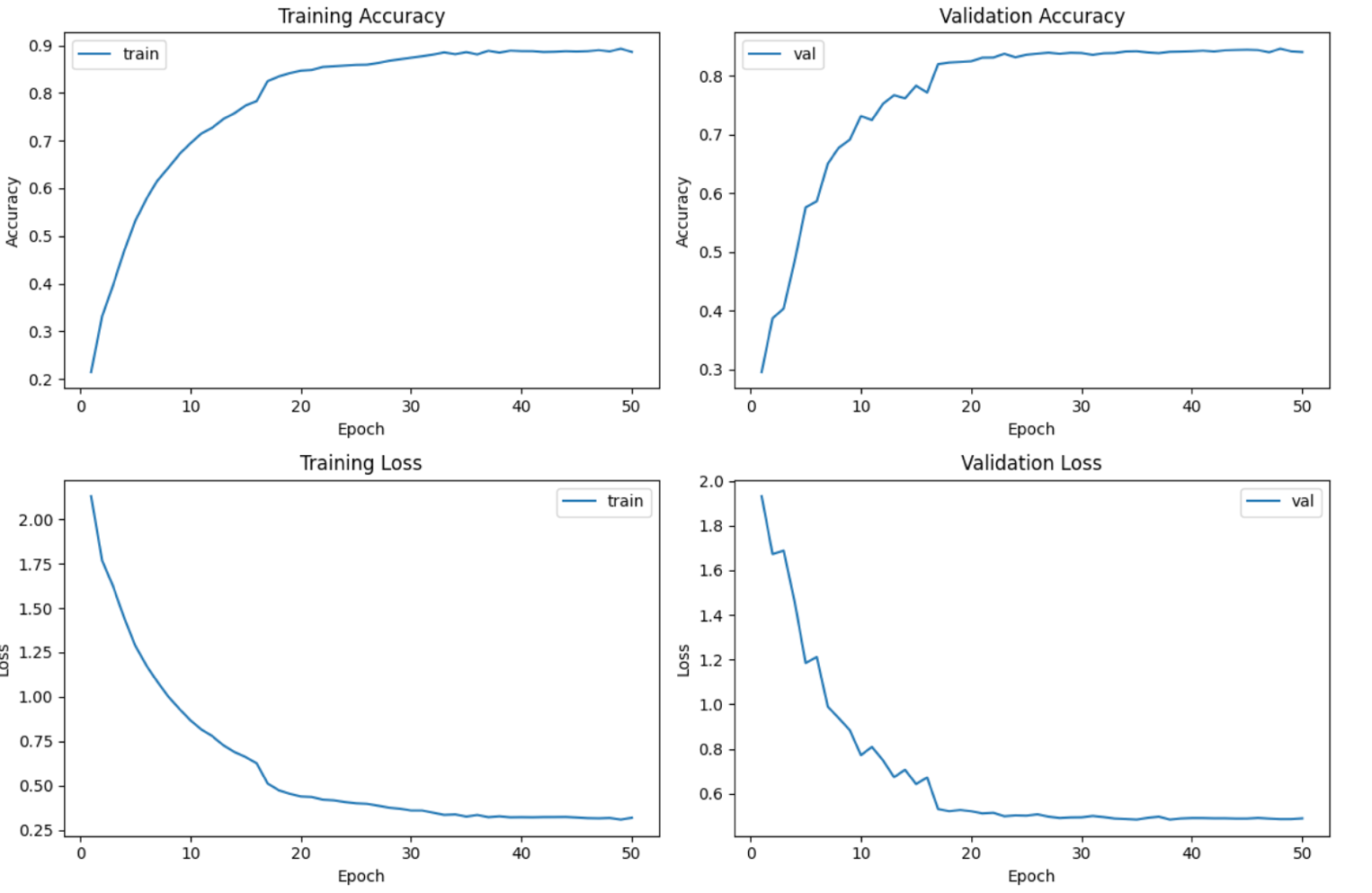
**• Plot four learning curves (loss & accuracy) of the training process (train/validation) for both models. Total 8 plots. (8%)**

**Ans:**

MyNet

****

Resnet18 model

****

**• Briefly describe what method do you apply on your best model? (e.g. data augmentation, model architecture, loss function, etc) (10%)**

**Ans:**

1. Data Augmentation

I apply rotation, cropping, and horizontal flipping to the dataset. As a result, the model is trained with different images in each epoch, and the accuracy much increase.

1. Model architecture: Max pool = Identity()

In resnet18, I remove the max pooling layer by making the input and output of the max pooling layer equal. This preserves features and improves the accuracy of the model.

After that, the validation accuracy increase from 0.78 to 0.84.

1. Adam vs. SGD

I experimented with different optimization algorithms and found that SGD yields higher accuracy compared to Adam. Therefore, I chose to use SGD in my model.

1. Reduce the kernel size and stride

The original kernel size (7x7) is too big. Hence, I reduce the kernel size, stride of the first convolution layer. After some trial, I decide to set kernel size to 3x3, it is the best setup.