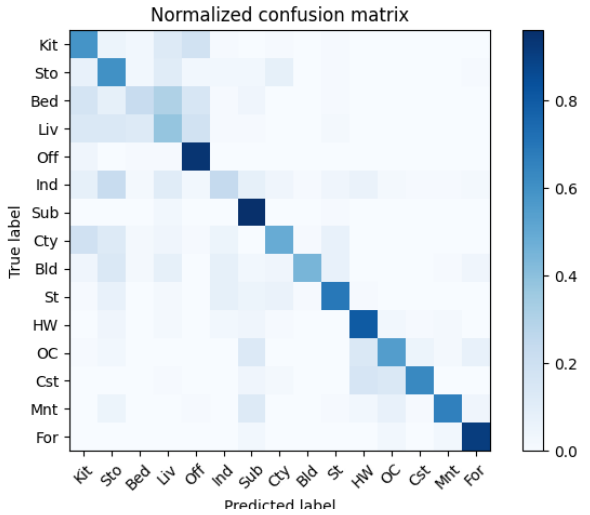
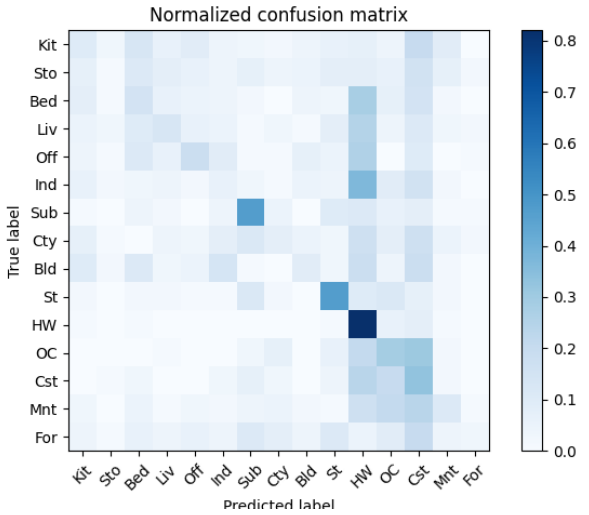
Computer Vision HW2 Report

Student ID: D11921B09
Name: 徐子程

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:

The accuracy of both settings

Bag of sift	tiny image
0.6080	0.2213

The accuracy of the Bag of sift methods is much better than the tiny image method. The result can be observed from the confusion matrix of both methods since the blocks in the diagonal are darker than another one.

The Bag of sift method is based on feature extraction. While the tiny image method only does the data argumentation. Thus the Bag of sift gives better accuracy.

In the nearest neighbor classify, the choice of k in kNN is also an important factor that affects the accuracy. Considering the accuracy in both methods, set k=5 for the final version.

Method	k=1	k=2	k=3	k=4	k=5	k=6
Bag of sift	0.5860	0.5653	0.5860	0.5927	0.608	0.6053
tiny image	0.23066	0.2273	0.2240	0.2220	0.2213	0.2187

Moreover, different metrics in computing the distance between each pair of features in the two images also influence the accuracy. Finally, the metric "cityblock" setting was applied for better evaluation performance and accuracy.

Part 2. (25%)

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

Ans:

MyNet	**ResNet18** (modified)
0.8698	0.8714

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

Ans:

MyNet
MyNet((network): Sequential((0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (1): ReLU() (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (4): ReLU() (5): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False) (7): Dropout(p=0.2, inplace=False) (8): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (9): ReLU() (10): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (11): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (12): ReLU() (13): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (14): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False) (15): Dropout(p=0.3, inplace=False) (16): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))

```
    (17): ReLU()
    (18): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (19): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU()
    (21): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (22): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (23): Dropout(p=0.4, inplace=False)
    (24): Flatten()
    (25): Linear(in_features=4096, out_features=512, bias=True)
    (26): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (27): Dropout(p=0.2, inplace=False)
    (28): ReLU()
    (29): Linear(in_features=512, out_features=10, bias=True)
  )
)
```

Number of parameters = 3231626

ResNet18 (modified)

```
ResNet18(
  (conv1): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (layer0): Sequential(
    (0): BasicBlock(
      (conv1): Sequential(
        (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU()
      )
      (conv2): Sequential(
        (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (relu): ReLU()
    )
    (1): BasicBlock(
      (conv1): Sequential(
        (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU()
      )
      (conv2): Sequential(
        (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (relu): ReLU()
    )
  )
  (layer1): Sequential(
```

```
    (0): BasicBlock(
      (conv1): Sequential(
        (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU()
      )
      (conv2): Sequential(
        (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (1): 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))
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (relu): ReLU()
    )
    (1): BasicBlock(
      (conv1): Sequential(
        (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU()
      )
      (conv2): Sequential(
        (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (relu): ReLU()
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Sequential(
        (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU()
```

```
        )
        (conv2): Sequential(
          (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (1): 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))
          (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
        (relu): ReLU()
      )
      (1): BasicBlock(
        (conv1): Sequential(
          (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU()
        )
        (conv2): Sequential(
          (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
        (relu): ReLU()
      )
    )
    (layer3): Sequential(
      (0): BasicBlock(
        (conv1): Sequential(
          (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU()
        )
        (conv2): Sequential(
          (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
```

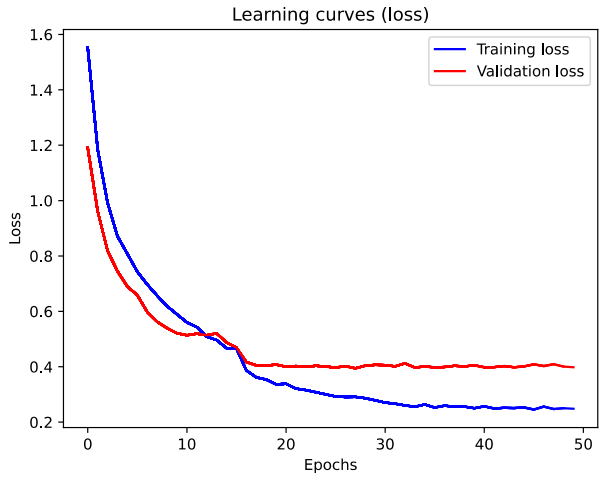
```
        (downsample): Sequential(
          (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2))
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
        (relu): ReLU()
      )
      (1): BasicBlock(
        (conv1): Sequential(
          (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU()
        )
        (conv2): Sequential(
          (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
        (relu): ReLU()
      )
    )
    (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
    (fc): Linear(in_features=512, out_features=10, bias=True)
)
```

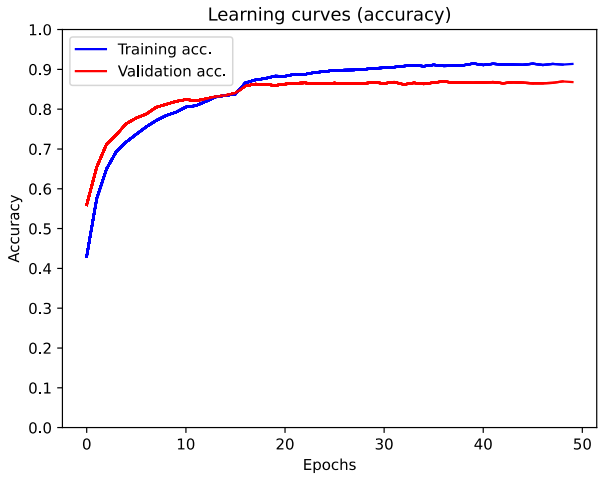
Number of parameters = 11174858

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

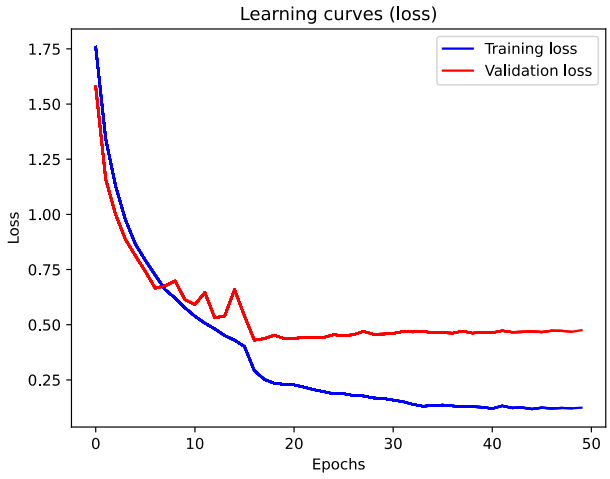
Ans:

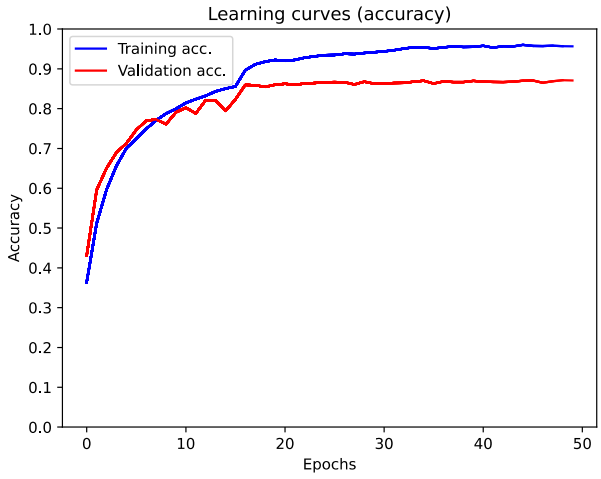
MyNet





ResNet18 (modified)





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

Ans:

I applied the following data argumentation methods:

```
transforms.RandomHorizontalFlip(p=0.5),
transforms.RandomAffine(degrees = 5, translate = (0.1, 0.1)),
```

In MyNet, two conv2D filters with maxpooling layer build a basic block. There are three layers of basic block followed by two layers of full-connected network in this model. To speed up training, the batch normalization is placed in two conv2D filters. To avoid overfitting, the dropout layer is inserted between each block.

In ResNet18 architecture, I modified the kernel size in the first convolution 2D layer from 7x7 to 3x3, which may be more suitable for the images in the CIFAR-10 dataset (32x32 pixel). The configuration of other layers is kept the same as the original ResNet18.