Part1

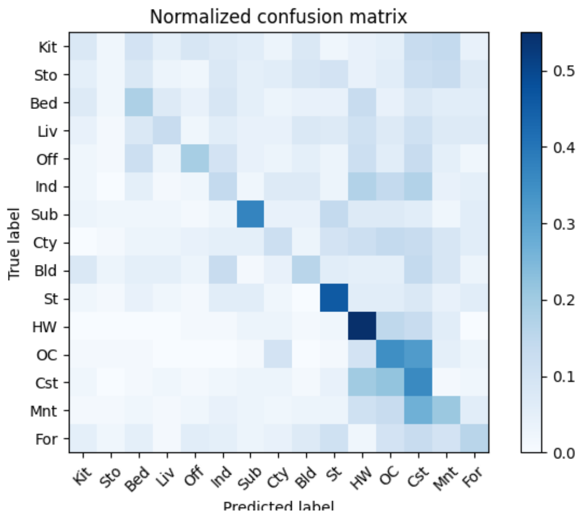
1. Accuracy of two setting

Tiny image (K = 1): 0.232

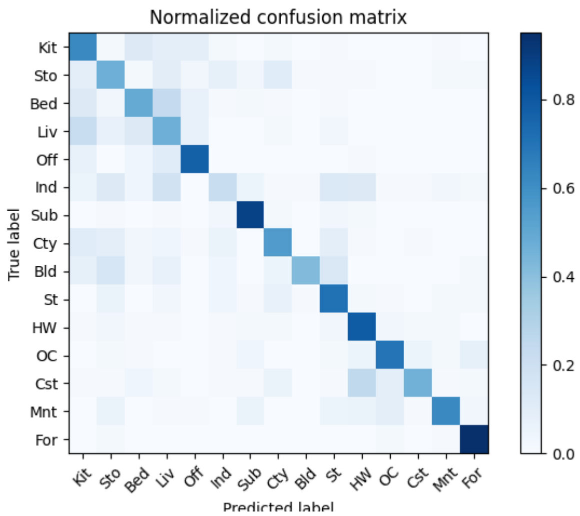
Bag of sift (K = 10, vocab size = 100, sift step in build_vocabulary = 16, sift step in get_bag_of_sifts = 3): 0.608

2. Confusion matrix

Tiny image



Bag of sift



3. Compare

Different K of KNN

K\setting	Tiny image	Bag of sift	K\setting	Tiny image	Bag of sift
1	0.232	0.5333	8	0.22	0.584
2	0.2007	0.5206	10	0.226	0.608
4	0.216	0.558	12	0.2287	0.5987
6	0.2147	0.5687	14	0.2273	0.5913

對於 tiny image 來說 K 為 1 時結果最好，因為只是把圖片縮小而已，如果 K 過多反而會

讓一些與目標不相關的特徵但相似的點被算進來；而 bag of sift 則是在 K 為 10 結果最好，認為是因為 sift 可以抽取到局部的特徵，bag of sift 則是去看這張有多少訓練資料的局部特徵，因此在 K 增加可以給予更多的資訊，但如果 K 太大結果會下降，可能是因為與

Different distance function in KNN

Distance function\setting	Tiny image	Bag of sift
Euclidean	0.2327	0.5346
Cosine	0.2327	0.5453
standardized Euclidean	0.232	0.608

對於 tiny image 來說，距離函式影響沒那麼大；而對 bag of sift 來說，standardized euclidean 結果最好，因為是使用 histogram 所以每個 feature 數量是不平均的，直接用 euclidean，會對於比較大的那一個 feature 的權重比較重，因此對每一個 feature 做標準化才比較能代表真正的距離。

Part2

1. The network architectures & number of parameters

conv

```
ConvNet(
  (cnn): Sequential(
    (0): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Linear(in_features=256, out_features=120, bias=True)
    (1): ReLU()
    (2): Linear(in_features=120, out_features=84, bias=True)
    (3): ReLU()
    (4): Linear(in_features=84, out_features=10, bias=True)
  )
)
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 6, 24, 24]             156
              ReLU-2            [-1, 6, 24, 24]               0
         MaxPool2d-3            [-1, 6, 12, 12]               0
            Conv2d-4             [-1, 16, 8, 8]           2,416
              ReLU-5             [-1, 16, 8, 8]               0
         MaxPool2d-6             [-1, 16, 4, 4]               0
            Linear-7                  [-1, 120]          30,840
              ReLU-8                  [-1, 120]               0
            Linear-9                   [-1, 84]          10,164
             ReLU-10                   [-1, 84]               0
           Linear-11                   [-1, 10]             850
================================================================
Total params: 44,426
Trainable params: 44,426
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.08
Params size (MB): 0.17
Estimated Total Size (MB): 0.25
----------------------------------------------------------------
```

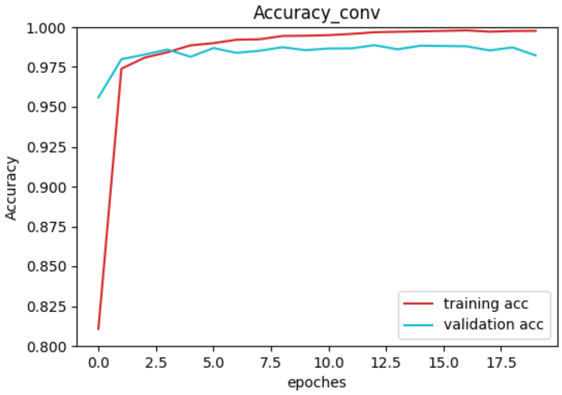
Mynet

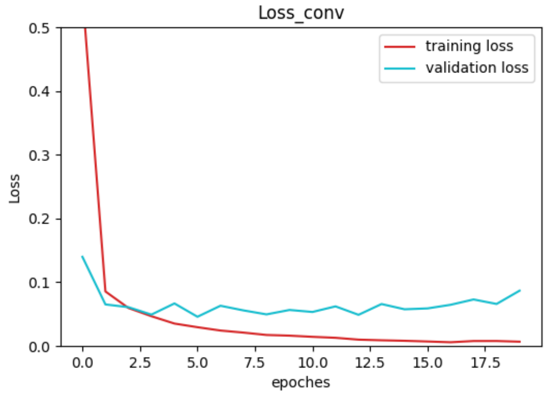
```
MyNet(
  (cnn): Sequential(
    (0): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Dropout(p=0.1, inplace=False)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): ReLU()
    (8): Dropout(p=0.1, inplace=False)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (12): ReLU()
    (13): Dropout(p=0.1, inplace=False)
    (14): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (16): ReLU()
    (17): Dropout(p=0.1, inplace=False)
    (18): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Linear(in_features=576, out_features=120, bias=True)
    (1): ReLU()
    (2): Linear(in_features=120, out_features=84, bias=True)
    (3): ReLU()
    (4): Linear(in_features=84, out_features=10, bias=True)
  )
)
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 16, 28, 28]             160
       BatchNorm2d-2           [-1, 16, 28, 28]              32
              ReLU-3           [-1, 16, 28, 28]               0
           Dropout-4           [-1, 16, 28, 28]               0
         MaxPool2d-5           [-1, 16, 14, 14]               0
            Conv2d-6           [-1, 32, 14, 14]           4,640
       BatchNorm2d-7           [-1, 32, 14, 14]              64
              ReLU-8           [-1, 32, 14, 14]               0
           Dropout-9           [-1, 32, 14, 14]               0
        MaxPool2d-10             [-1, 32, 7, 7]               0
           Conv2d-11             [-1, 64, 7, 7]          18,496
      BatchNorm2d-12             [-1, 64, 7, 7]             128
             ReLU-13             [-1, 64, 7, 7]               0
          Dropout-14             [-1, 64, 7, 7]               0
           Conv2d-15             [-1, 64, 7, 7]          36,928
      BatchNorm2d-16             [-1, 64, 7, 7]             128
             ReLU-17             [-1, 64, 7, 7]               0
          Dropout-18             [-1, 64, 7, 7]               0
        MaxPool2d-19             [-1, 64, 3, 3]               0
           Linear-20                  [-1, 120]          69,240
             ReLU-21                  [-1, 120]               0
           Linear-22                   [-1, 84]          10,164
             ReLU-23                   [-1, 84]               0
           Linear-24                   [-1, 10]             850
================================================================
Total params: 140,830
Trainable params: 140,830
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.81
Params size (MB): 0.54
Estimated Total Size (MB): 1.35
----------------------------------------------------------------
```

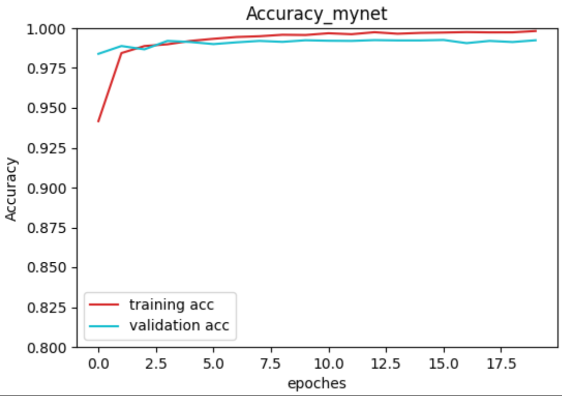
2. The learning curve

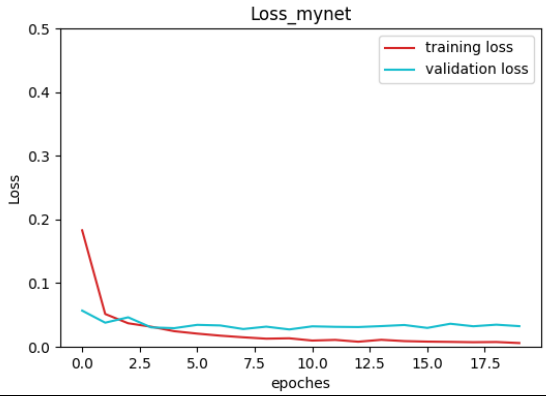
conv





mynet





3. Compare

從 learning curve 來看，可以看到 mynet 比較快收斂，認為是因為加了 batchnorn 使得 error surface 變得更平滑，較容易訓練。

另外 convolution layer 也增加，讓 model 可以看到更遠的資訊。

由於參數變多，所以使用 dropout 減緩 overfitting 的發生。

最後結果如下，在 validation set 上 mynet 比起 conv accuracy 高了 0.004。

Best validation accuracy

conv: 0.989

mynet: 0.993