report

March 1, 2023

1 Assignment 1 Report

1.1 Problem 4

1.1.1 Q2:

Hyperparameters:

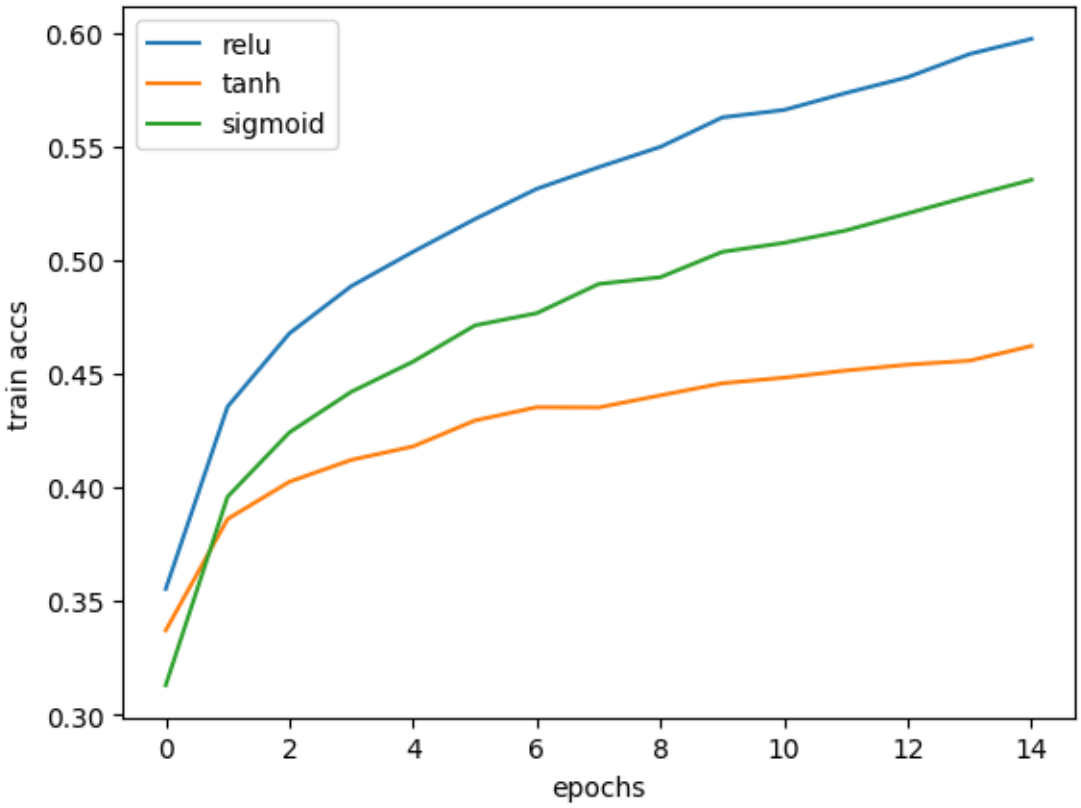
- lr: 1e-3
- epochs: 15
- batch size: 128
- weight decay: 5e-4
- momentum: 0.9

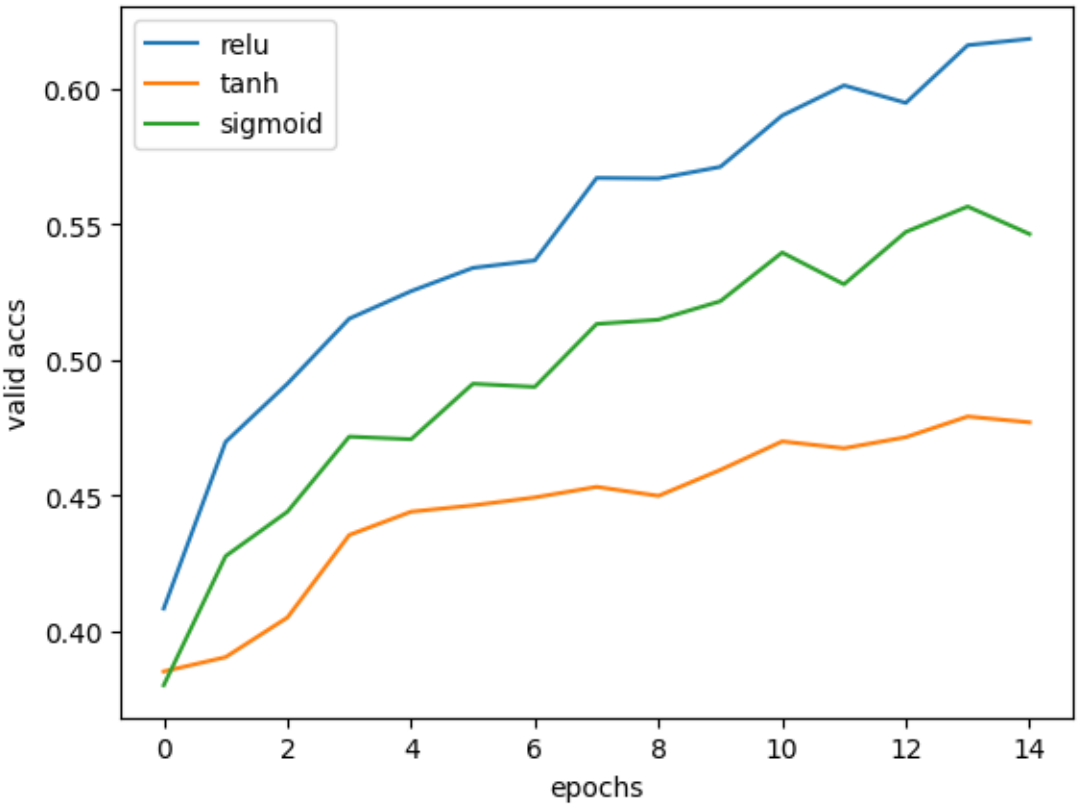
Non-linearity	Test Accuracy
ReLU	0.554885
tanh	0.459948
sigmoid	0.515723

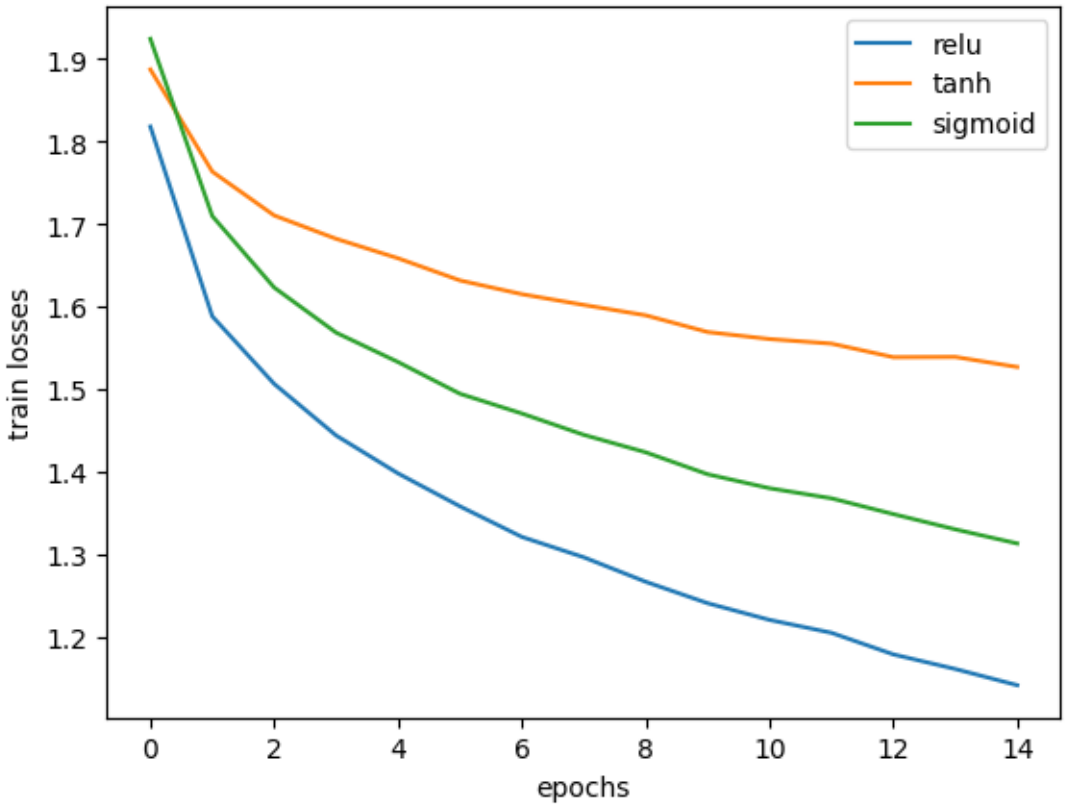
ReLU is the best. This is because ReLU prevents vanishing gradient. Unlike tahn and sigmoid functions, of which the gradient saturates when the model becomes deep, and the model stops learning or learn very slowly.

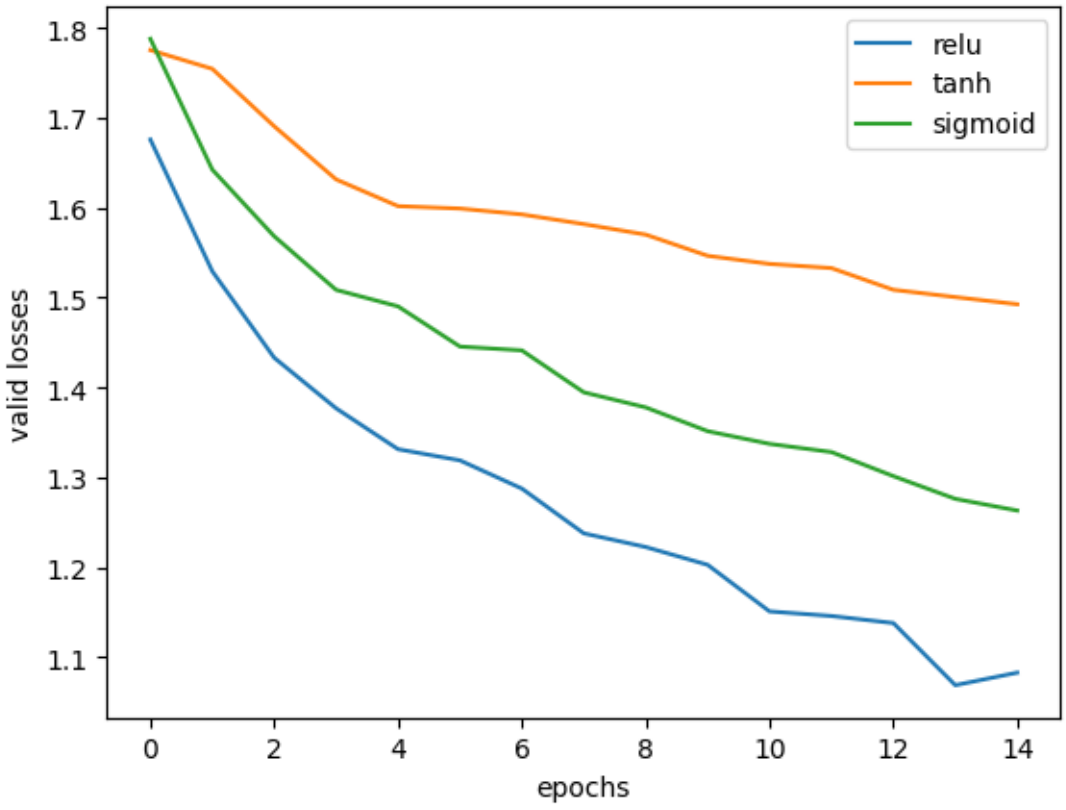
The figures show training accuracies, training loss, validation accuracies and validation loss are shown below.

[2]:
```
from utils import generate_plots

list_of_dirs = ['exp/mlp_default', 'exp/mlp_tanh', 'exp/mlp_sigmoid']
names = ['relu', 'tanh', 'sigmoid']
save_path = 'figs/mlp_exps/'

generate_plots(list_of_dirs, names, save_path)
```


relu
tanh
sigmoid
0.60
0.55
0.50
0.45
0.40
0.35
0.30
train accs
0
2
4
6
8
10
12
14
epochs


relu
tanh
sigmoid
valid accs
0.60
0.55
0.50
0.45
0.40
0
2
4
6
8
10
12
14
epochs


relu
tanh
sigmoid
train losses
epochs
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
0
2
4
6
8
10
12
14



1.1.2 Q3:

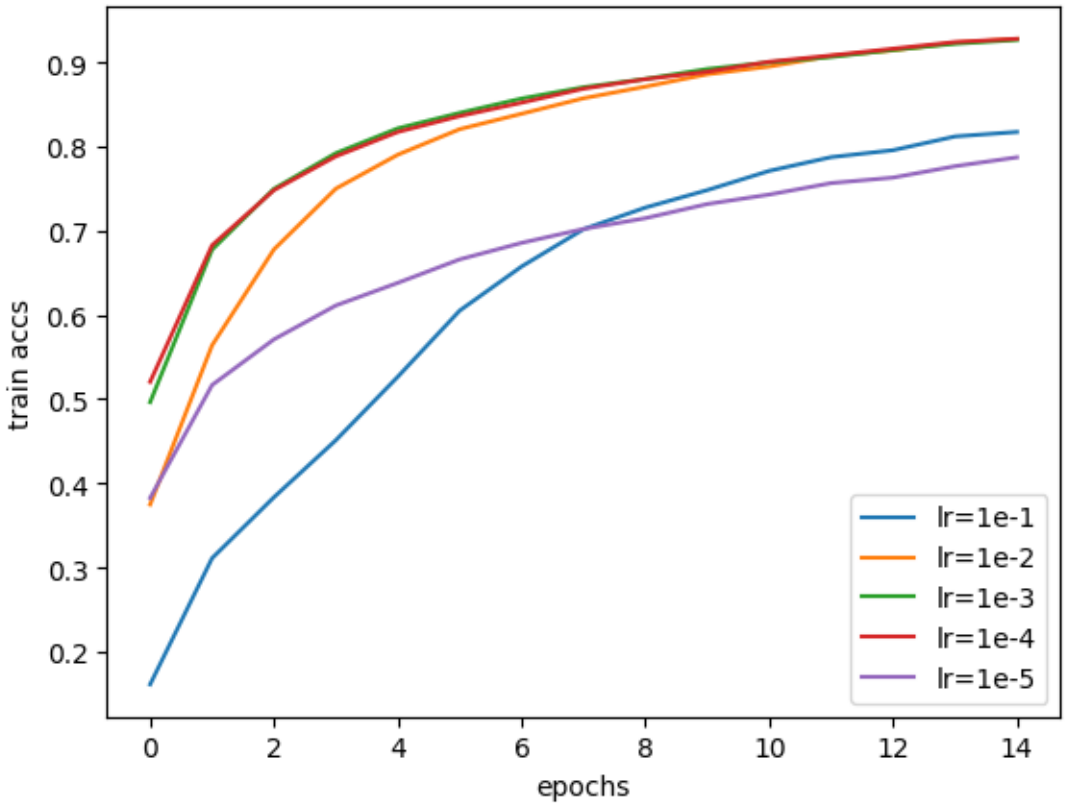
Hyperparameters other than learning rate: - epochs: 15 - batch size: 128 - weight decay: 5e-4 - momentum: 0.9

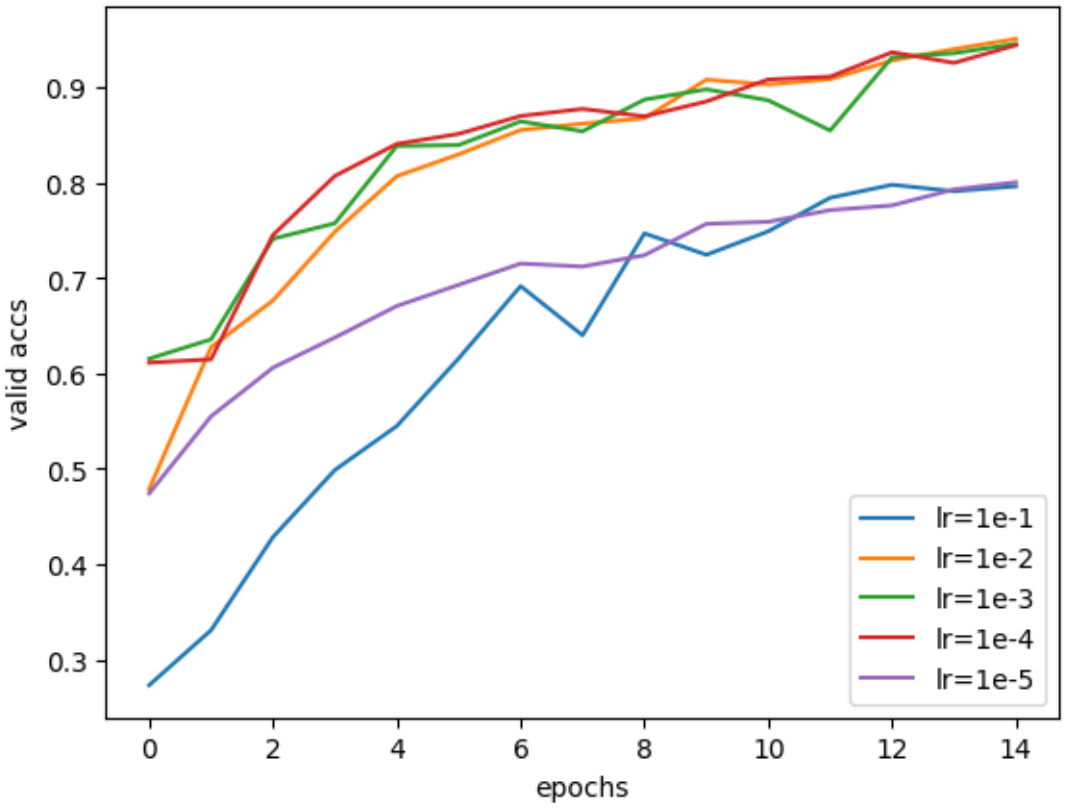
Learning rate	Test Accuracy
1e-1	0.759592
1e-2	0.872033
1e-3	0.884394
1e-4	0.846518
1e-5	0.725870

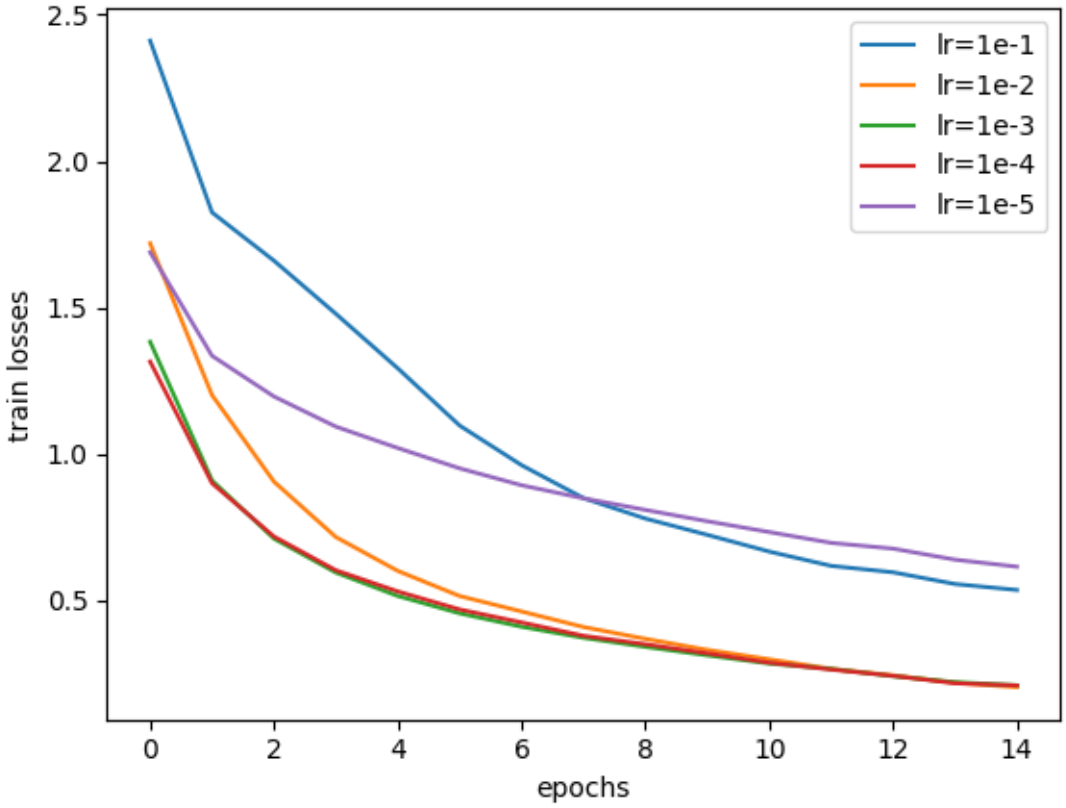
1e-3 is the best learning rate. 1e-1 is too large. It could have overshot the minimum. 1e-5 is too small. It could have stayed at a local minimum.

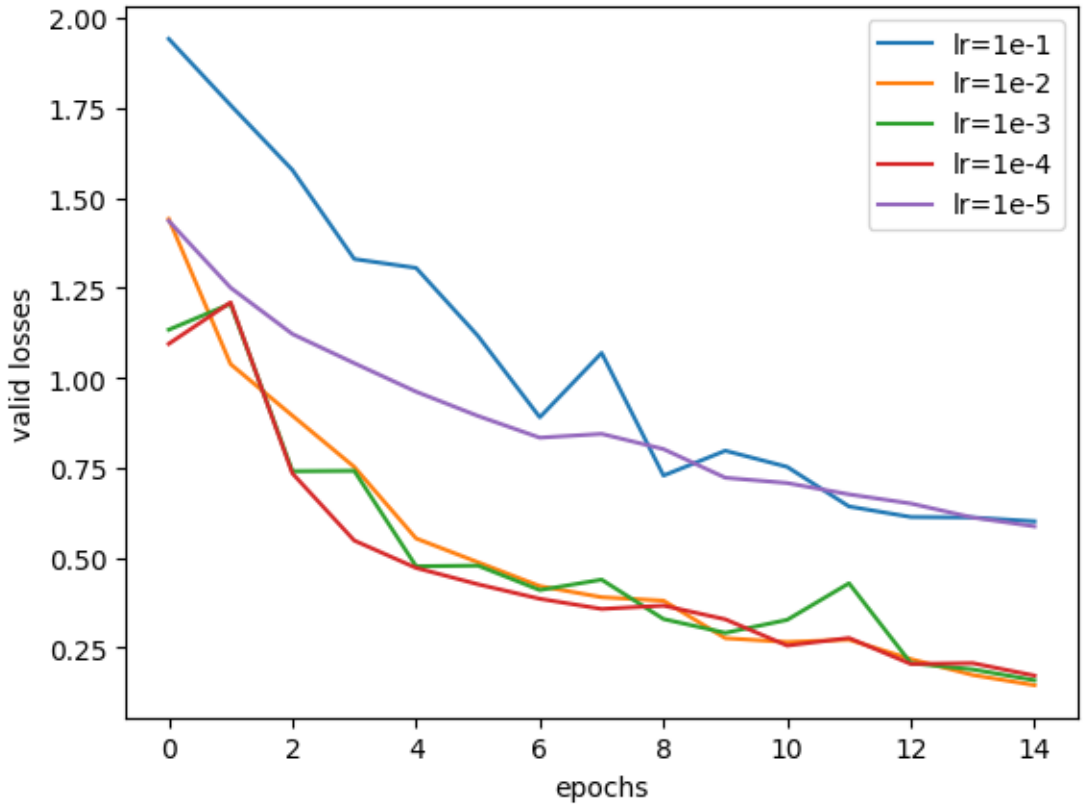
[2]:
```
from utils import generate_plots

list_of_dirs = ['exp/resnet1', 'exp/resnet2', 'exp/resnet3', 'exp/resnet4', 'exp/
 ↪resnet5',]
names = ['lr=1e-1', 'lr=1e-2', 'lr=1e-3', 'lr=1e-4', 'lr=1e-5',]
```

```
save_path = 'figs/resnet18_exps'

generate_plots(list_of_dirs, names, save_path)
```




valid accs
epochs
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
2
4
6
8
10
12
14
lr=1e-1
lr=1e-2
lr=1e-3
lr=1e-4
lr=1e-5


lr=1e-1
lr=1e-2
lr=1e-3
lr=1e-4
lr=1e-5
train losses
epochs
2.5
2.0
1.5
1.0
0.5
0
2
4
6
8
10
12
14



1.1.3 Q4

Hyperparameters: - lr: 1e-3 - epochs: 15 - batch size: 128 - weight decay: 5e-4 - momentum: 0.9

patch size	number of model parameters	Average training time(s)	Test Accuracy
2	2,376,458	43.053162	0.741
4	2,188,298	13.214978	0.756
8	2,175,818	8.241076	0.738
16	2,310,938	8.447986	0.642
32	2,897,678	9.384463	0.584

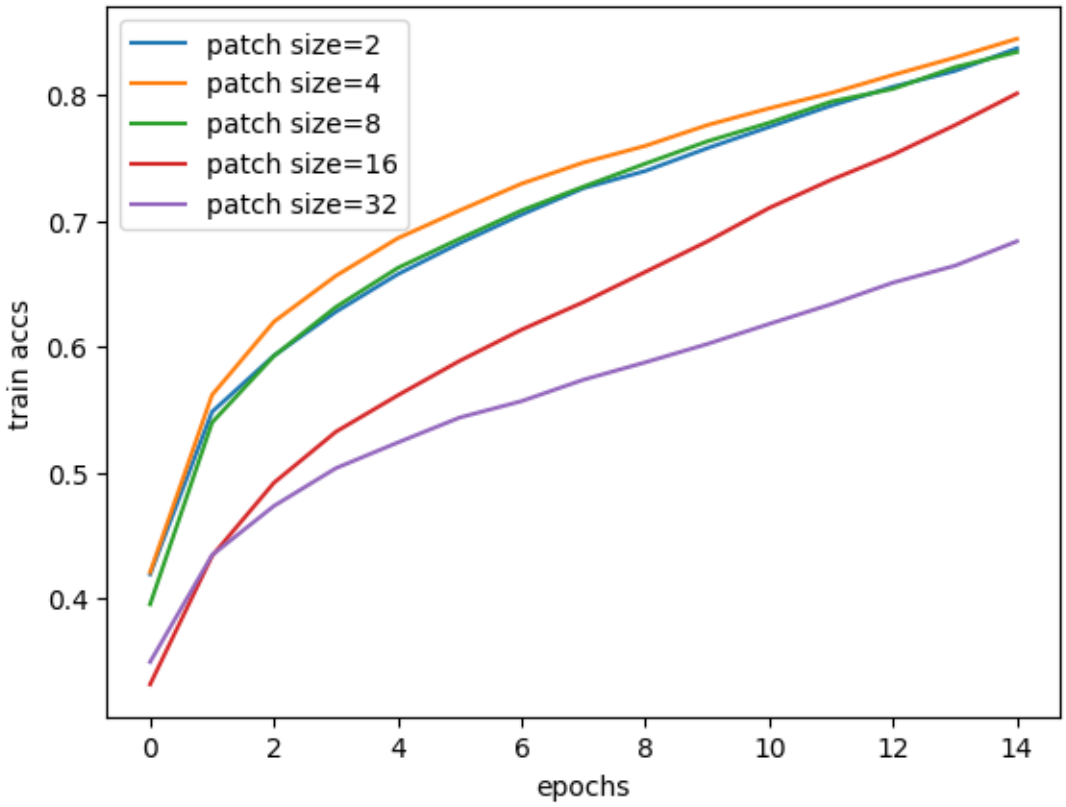
The smaller the patch size the better the model performance. The best patch size is 4, yielding a test accuracy of 75.6. The test accuracy decreases as the patch size gets too small or too large. The test accuracy is the lowest when the patch size is 32, only 58.4%.

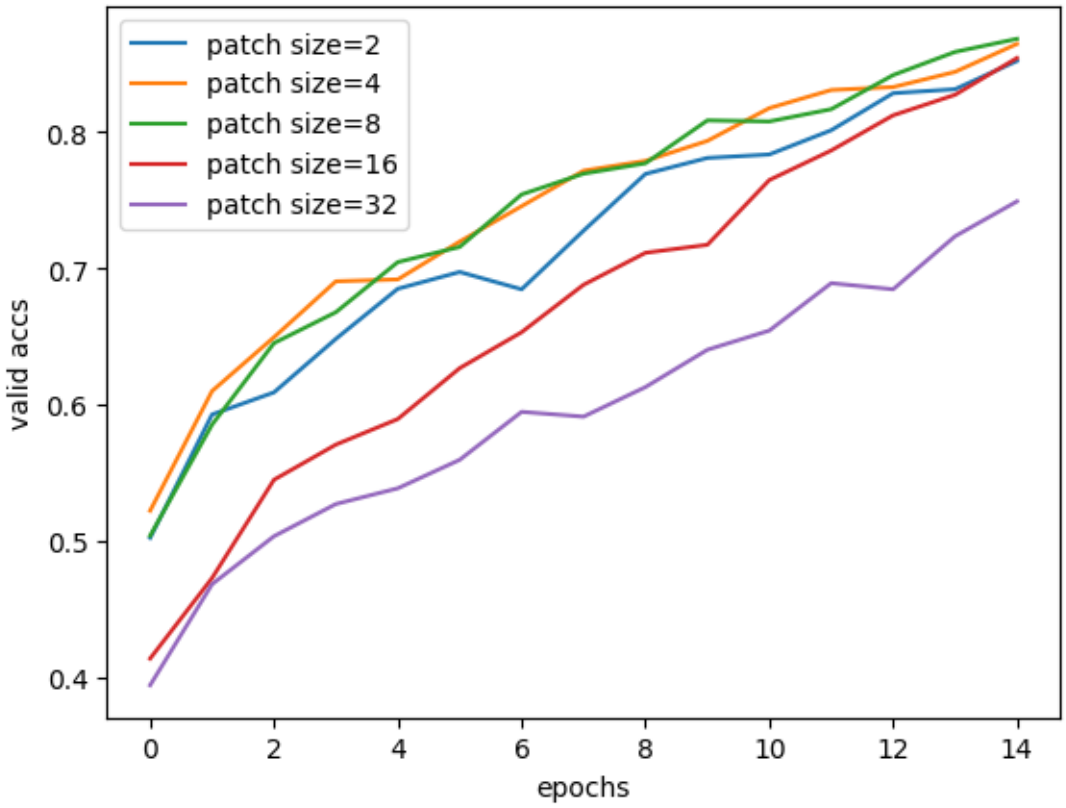
A patch size of 32 takes the most number of parameters and a patch size of 8 takes the least number of parameters. A patch size of 2 takes the longest time to train and a patch size of 8 takes the shortest time to train.

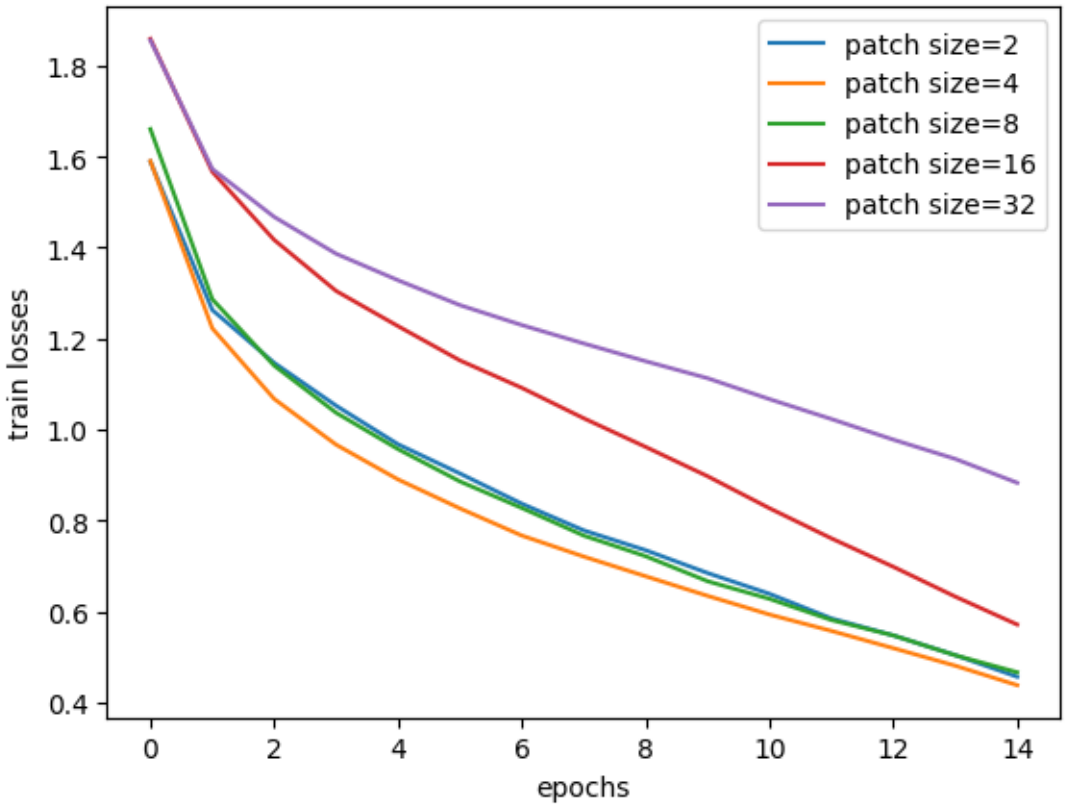
For patch sizes 2, 4 and 8, the number of model parameters decrease as patch size increases. The average training time also decreases as the patch size increases. This is because the larger the patch size, the fewer the number of tokens are fed into token-mixing MLPs, so fewer weights parameters are needed. Training time decreases because fewer multiplications are needed in MLPs as the patch size decreases.

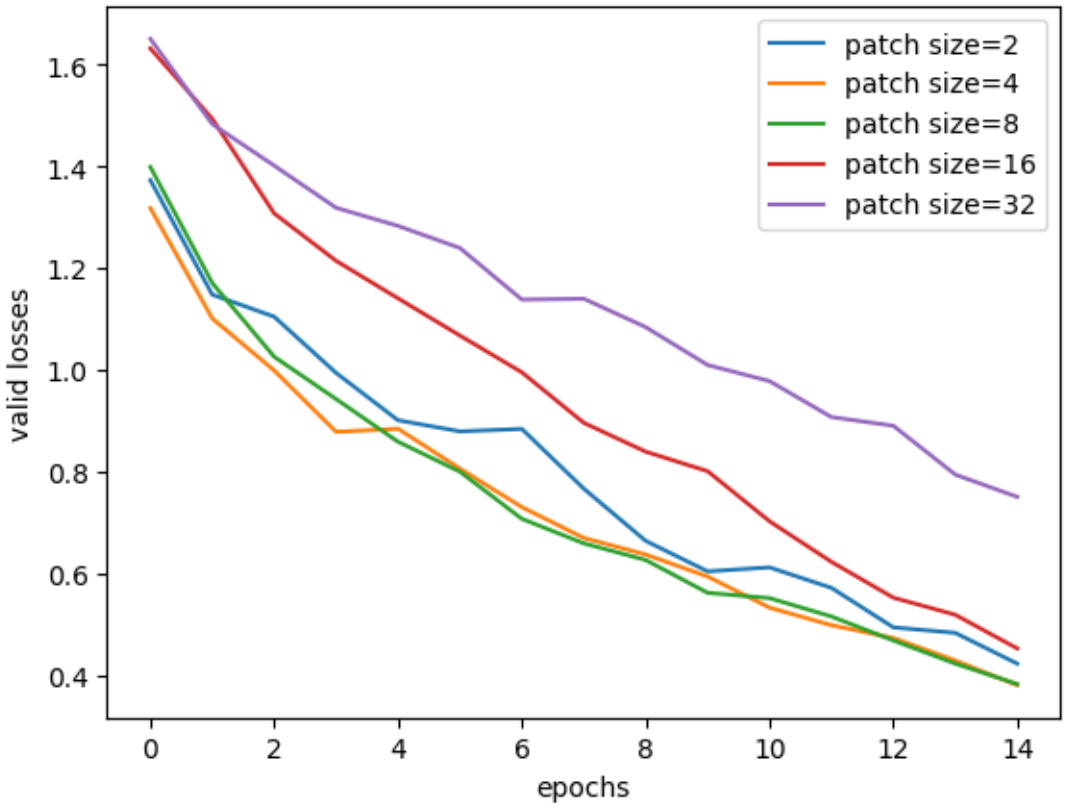
For patch sizes 16 and 32, the number of model parameters increases as the patch size increases. The average training time also increases as the patch size increases. As the patch size increases, the kernel size of convolution also increases, which results in an increase in the number of weight parameters used to project a patch into embeded dimensions. Each time the patch size increases by 2, The number of weights increases by a factor of 4. Training time increases because multiplications needed to project a patch into embeded dimension increases.

[3]:
```
from utils import generate_plots

list_of_dirs = ['exp/mlpmixer-2', 'exp/mlpmixer-4', 'exp/mlpmixer-8', 'exp/
 ↪mlpmixer-16', 'exp/mlpmixer-32']
names = ['patch size=2', 'patch size=4', 'patch size=8', 'patch size=16', 'patch␣
 ↪size=32']
save_path = 'figs/mlpmixer_exps'

generate_plots(list_of_dirs, names, save_path)
```




patch size=2
patch size=4
patch size=8
patch size=16
patch size=32
valid accs
epochs
0.4
0.5
0.6
0.7
0.8
0
2
4
6
8
10
12
14


patch size=2
patch size=4
patch size=8
patch size=16
patch size=32
train losses
epochs
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
2
4
6
8
10
12
14

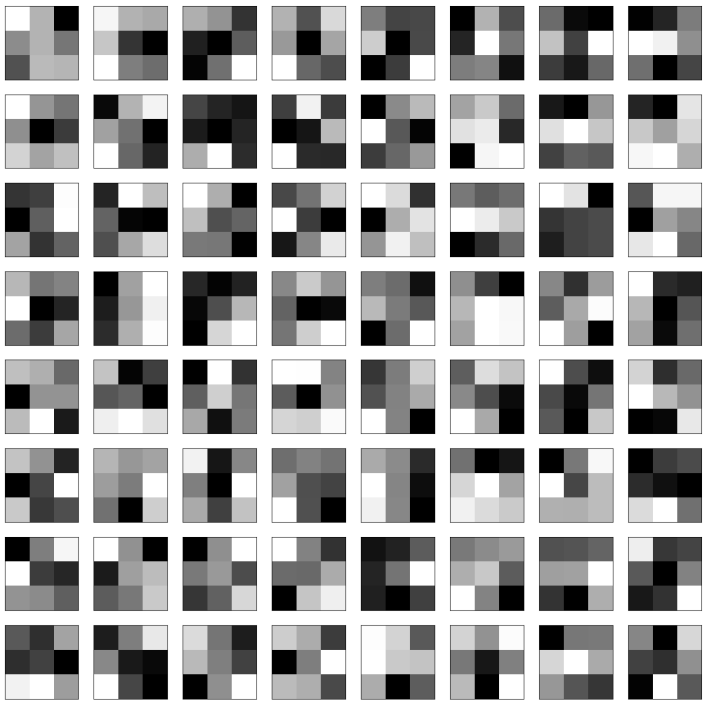


1.1.4 Q5

Best hyper-parameters:

- lr = 1e-3
- momentum = 0.9
- weight decay = 3e-4
- epochs = 100
- batch_size = 128

To visualize kernels of the first convolutional layer, I first take the kernels having a shape of (64, 3, 3, 3). Then I normalize the weights to the range (0,1) by subtracting the minimum and scaling by a factor of (min_weight - max_weight). Then I iterate over each of the 64 kernels and average the weights across its last 3 input channels. These steps give me a 3 x 3 x 3 images that I can plot as a gray-scale image. All 64 kernels of the first convolutional layer are shown underneath.



1.1.5 Q6

Best hyper-parameters for MLP-Mixer:

- lr = 1e-3
- momentum = 0.9
- weight decay = 4e-4
- epochs = 100
- batch_size = 128

Both CNN and MLP-mixer detect similar patterns of alternating opposing phases. CNN tend to focus on the local features such textures and frequncy changes, while token-mixing MLP aggregates spacial informations through dense matrix multiplications applied to every feature across all locations. It acts on the columns of patch x feature input tables and mixing features of every image patche. In other words, token-mixing MLPs allows global communications across all locations in the input image.

The figure below is a visualization of the first layer of the first block of token-MLP-mixer. As one can see from the visualization, opposing phases (alternating high and low stripes) in feature detection filters are more obvious in MLP-Mixer than in Resnet18. This shows that MLP-Mixer is better at detecting features that have structures, such as edges, than Resnet