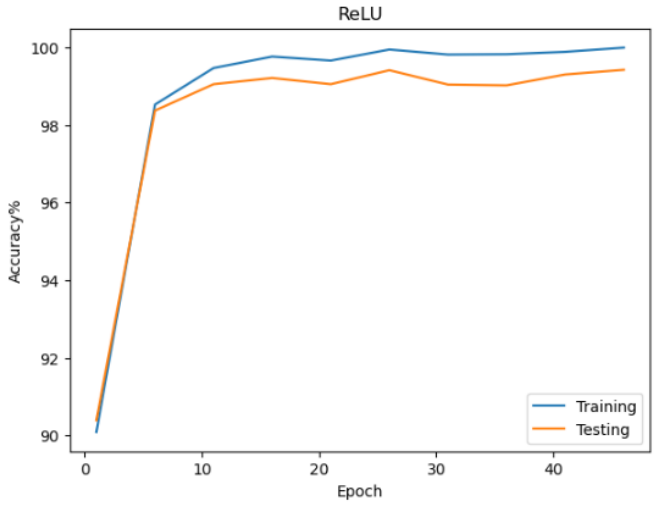
Assignment: 2

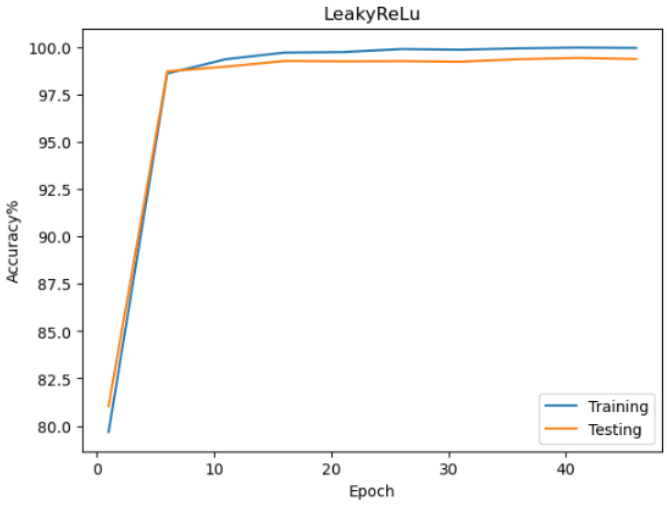
Q1

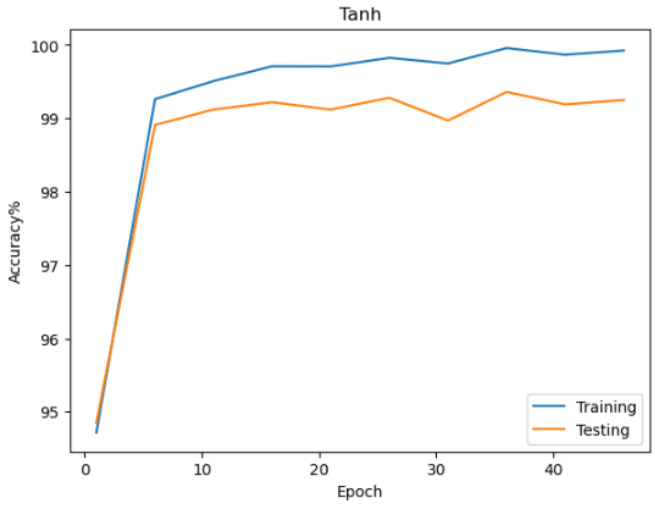
a) (Which Activation function is better?)

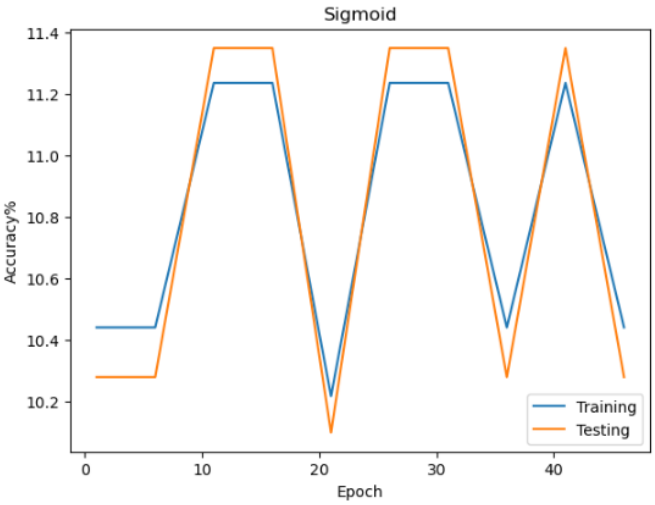
- only RELU activation
- only leaky RELU activation,
- only sigmoid activation,
- only tanh activation.

ReLU performed the best out of all the activation functions, LeakyReLU and Tanh performed similarly only sigmoid wasn't able to learn.







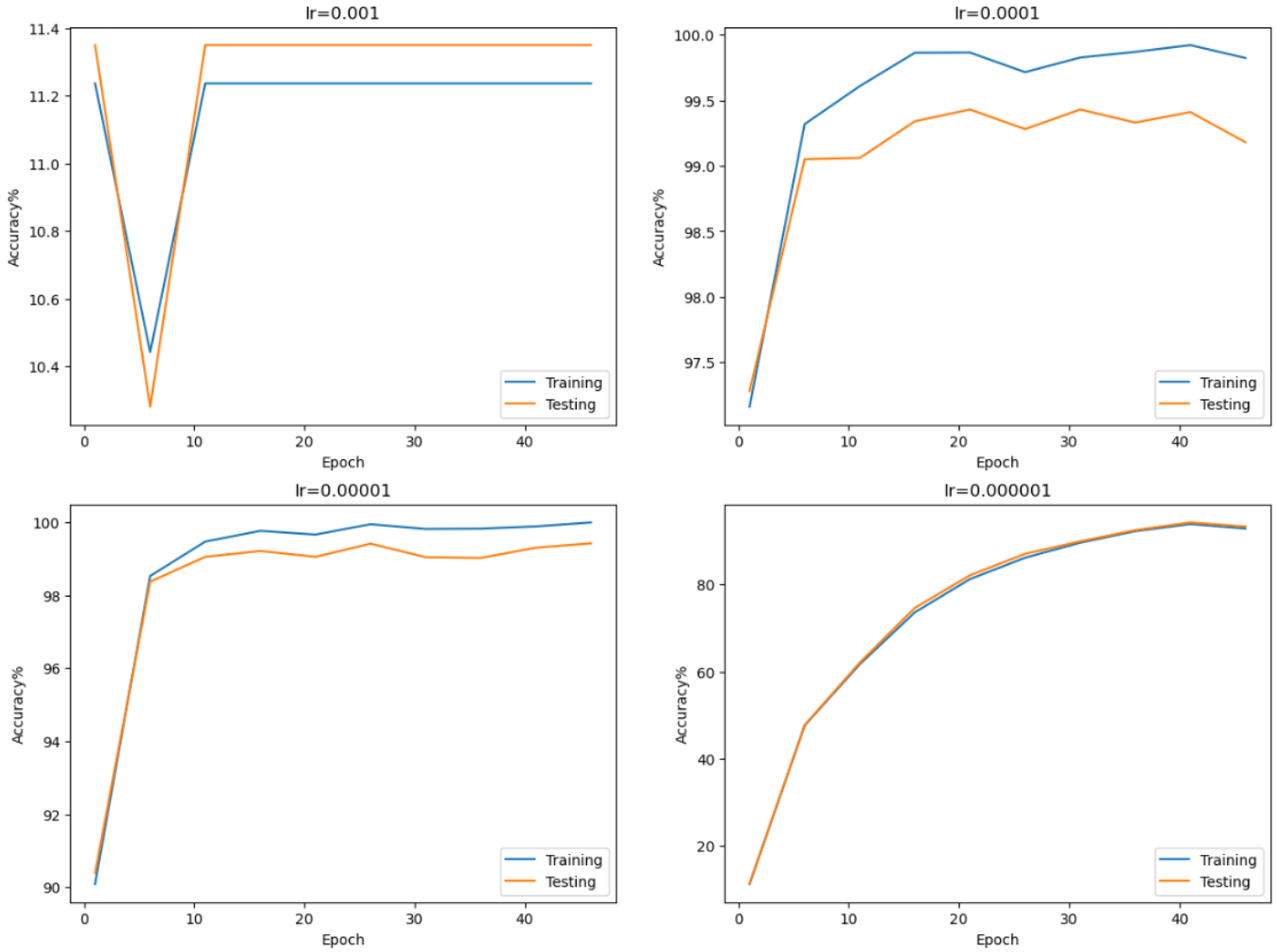


Optimizer: Adam learning rate 0.0001
Loss Function: Cross Entropy Loss

b) (What learning rate is better?)

Optimizer: Adam
Loss Function: Cross Entropy Loss
Activation Function: ReLU

Best learning rate was found to be 0.00001

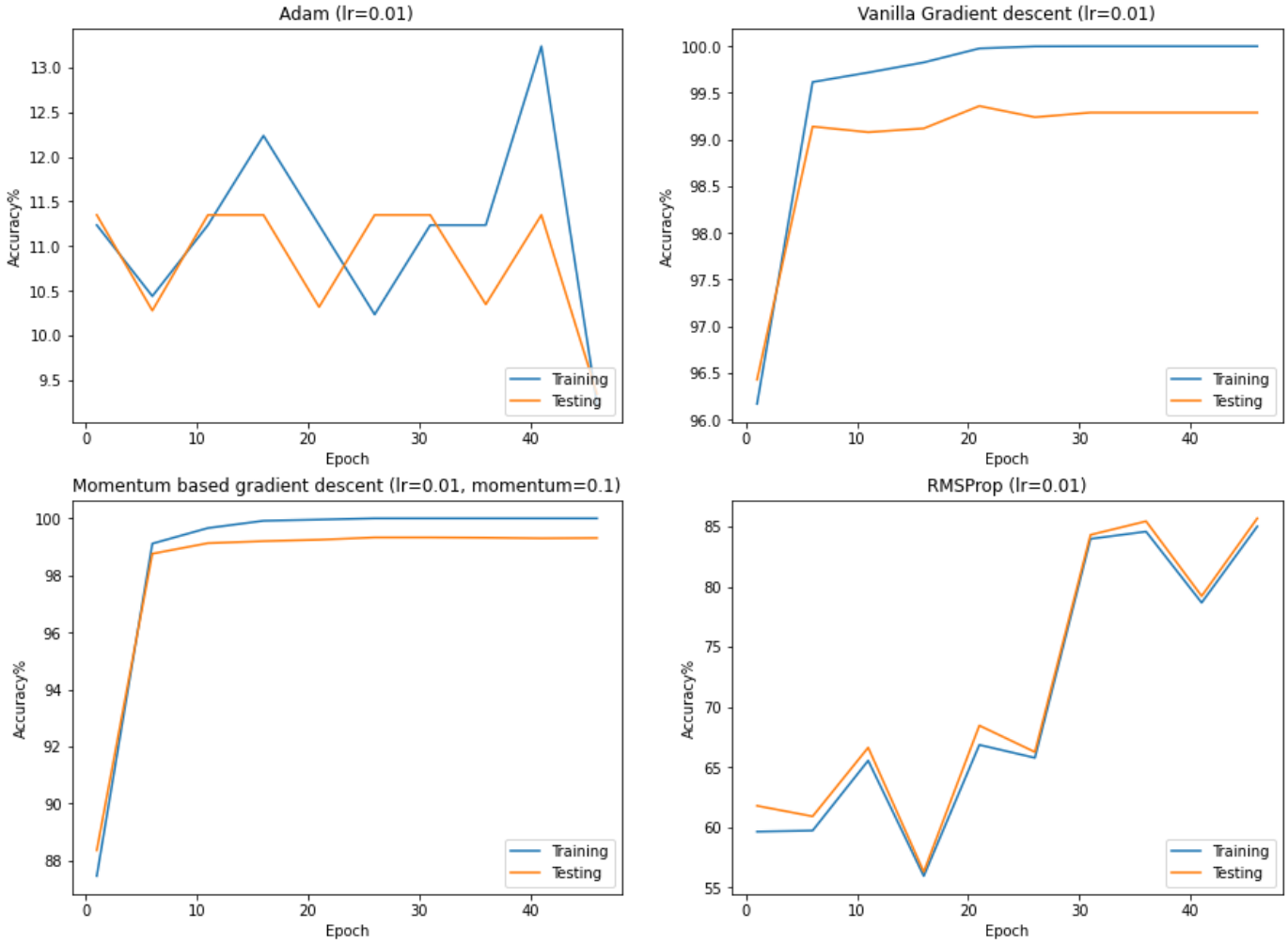


c) (Role of optimizer)

- Vanilla Gradient descent
- Momentum based gradient descent (momentum value of your choice)
- Adam.
- RMS prop.

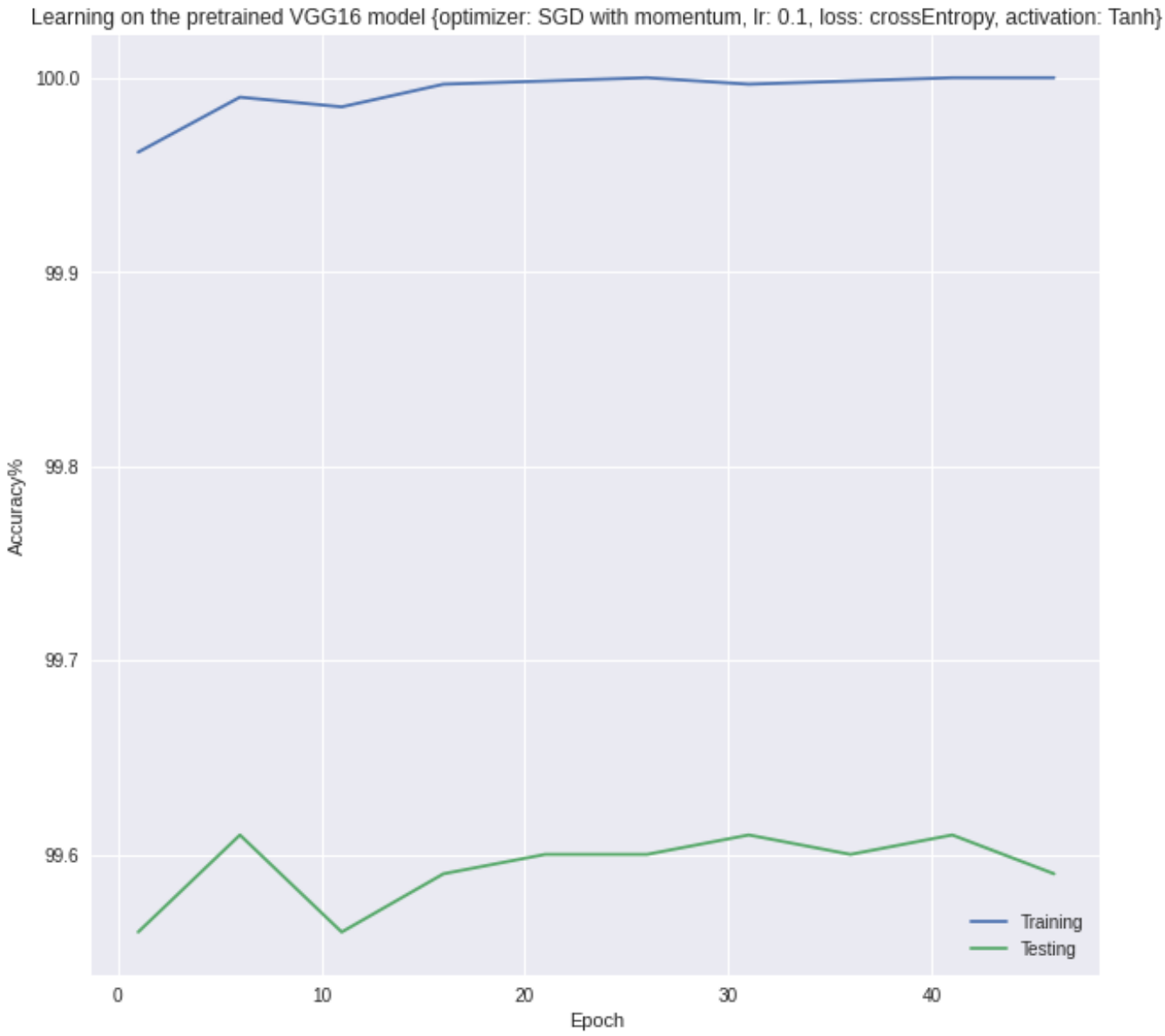
Loss Function: Cross Entropy Loss
Activation Function: ReLU

Gradient descent both Vanilla and momentum based performed comparatively better.



d) (Transfer Learning)

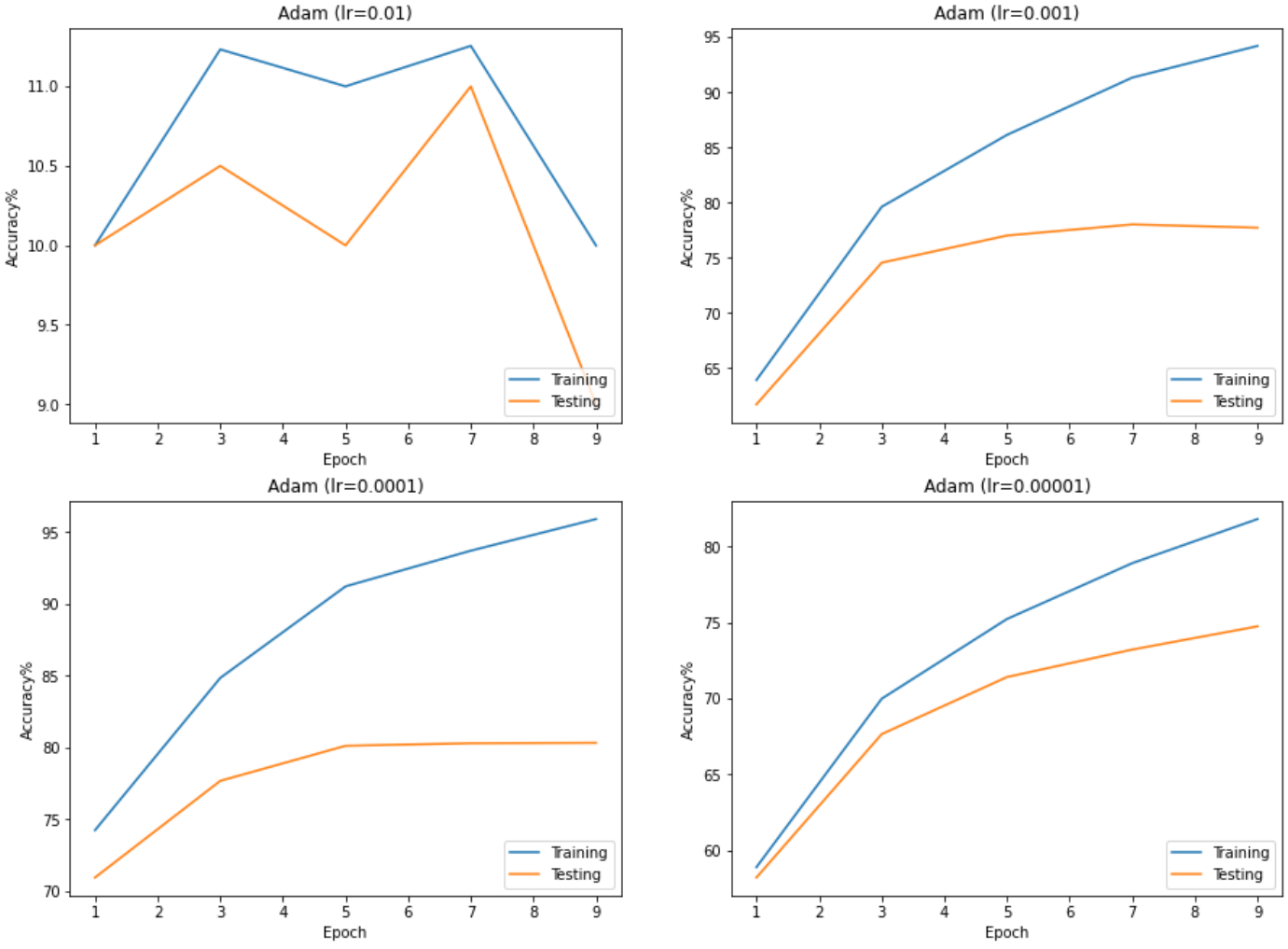
Optimal hyperparameters:
Optimizer: Momentum based gradient descent (lr: 0.01, momentum: 0.1)
Loss Function: Cross Entropy Loss
Activation Function: Tanh



Q2

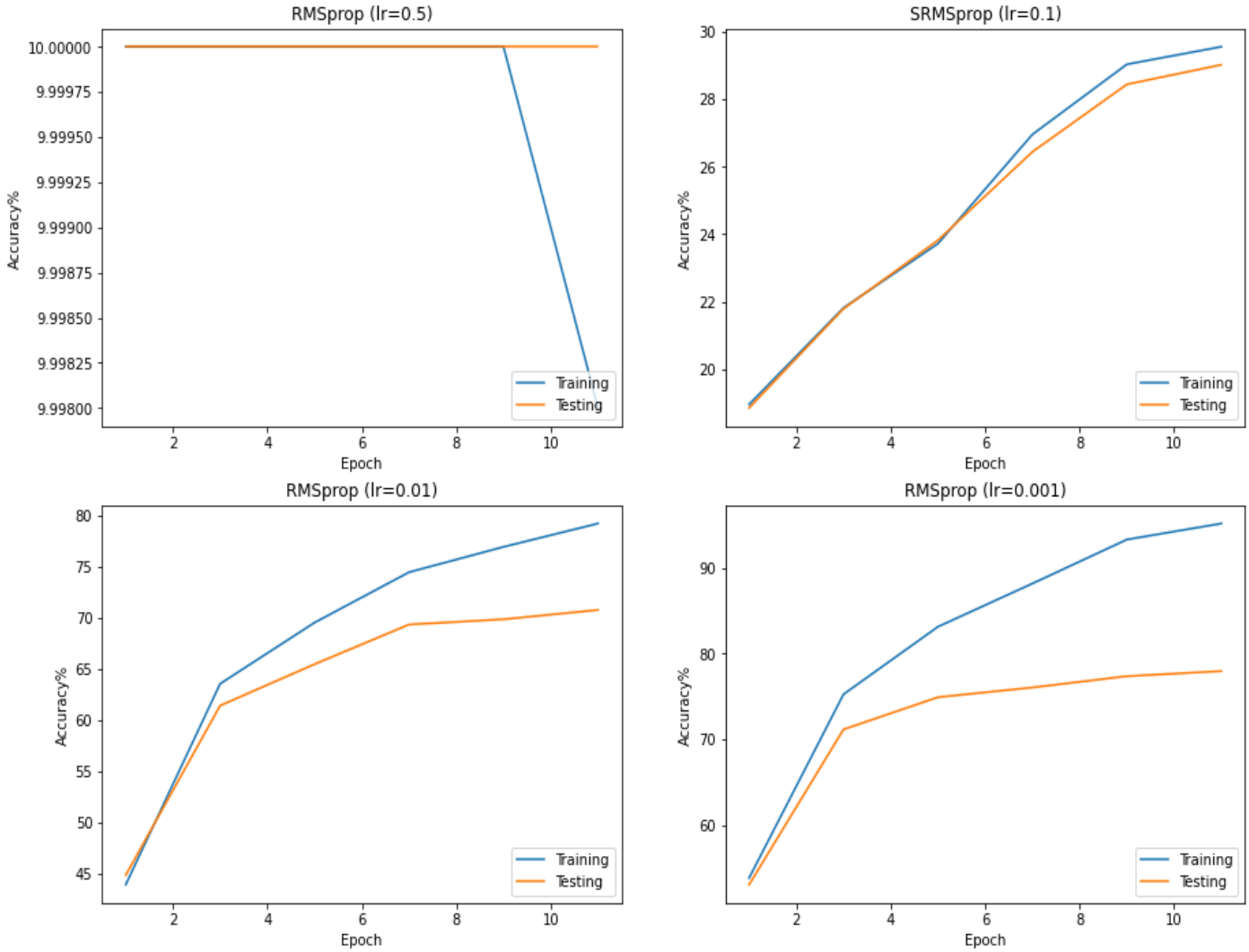
a) **Initialize the ResNet-18 network with pre-trained weights from ImageNet and then try to use these weights to improve the training for the CIFAR-10 dataset.**

hyperparameters:
Optimizer: Adam
Loss Function: Cross Entropy Loss
Activation Function: Tanh



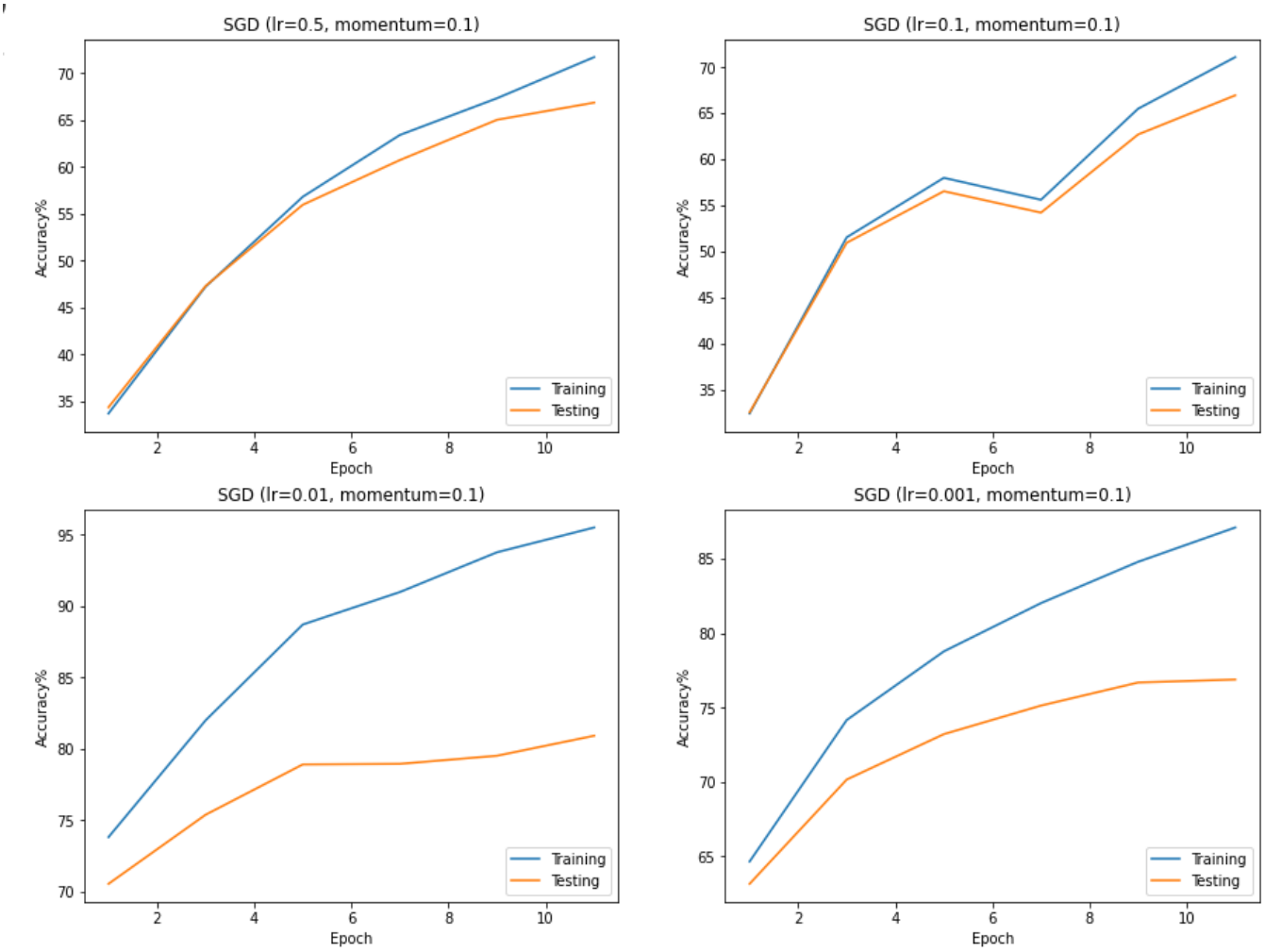
Best performing in Adam(lr=0.0001) Accuracy: 78.26%

hyperparameters:
Optimizer: RMSprop
Loss Function: Cross Entropy Loss
Activation Function: ReLU



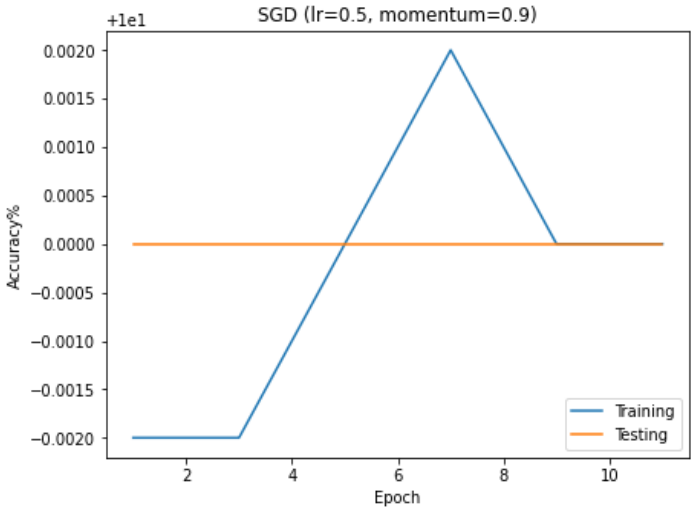
Best performing in RMSprop(lr=0.001) Accuracy: 77.92%

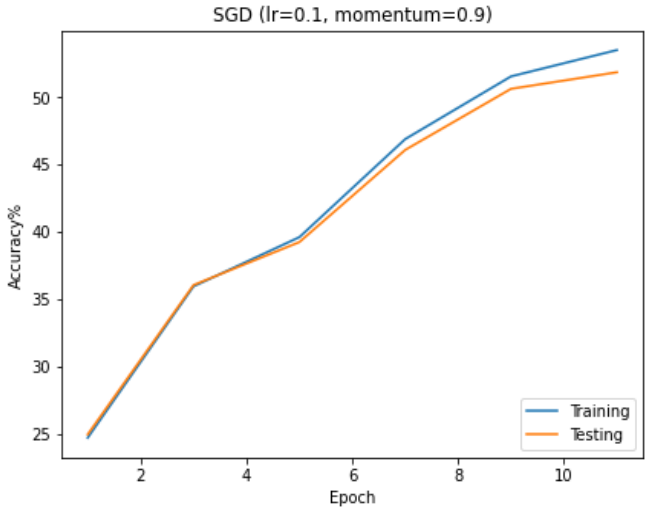
hyperparameters:
Optimizer: Momentum based gradient descent (momentum: 0.1)
Loss Function: Cross Entropy Loss
Activation Function: ReLU

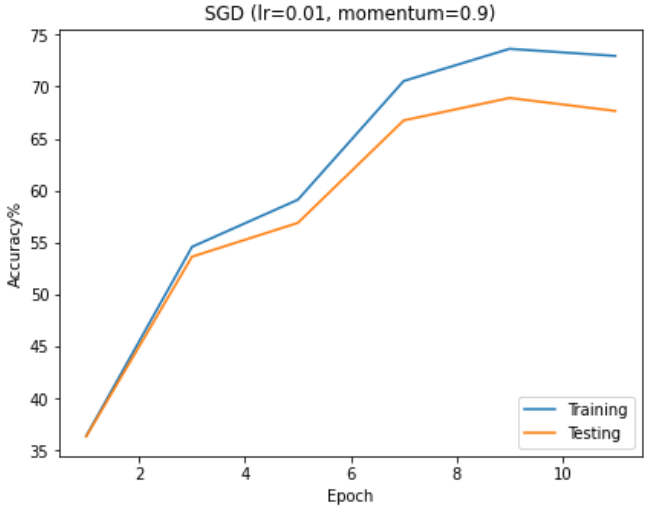


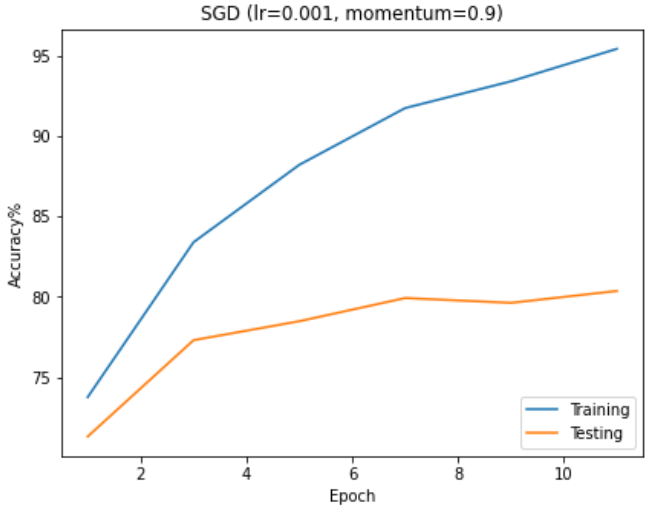
Best performing in SGD(lr=0.01, momentum=0.1) Accuracy: 80.15%

hyperparameters:
Optimizer: Momentum based gradient descent (momentum: 0.9)
Loss Function: Cross Entropy Loss
Activation Function: ReLU









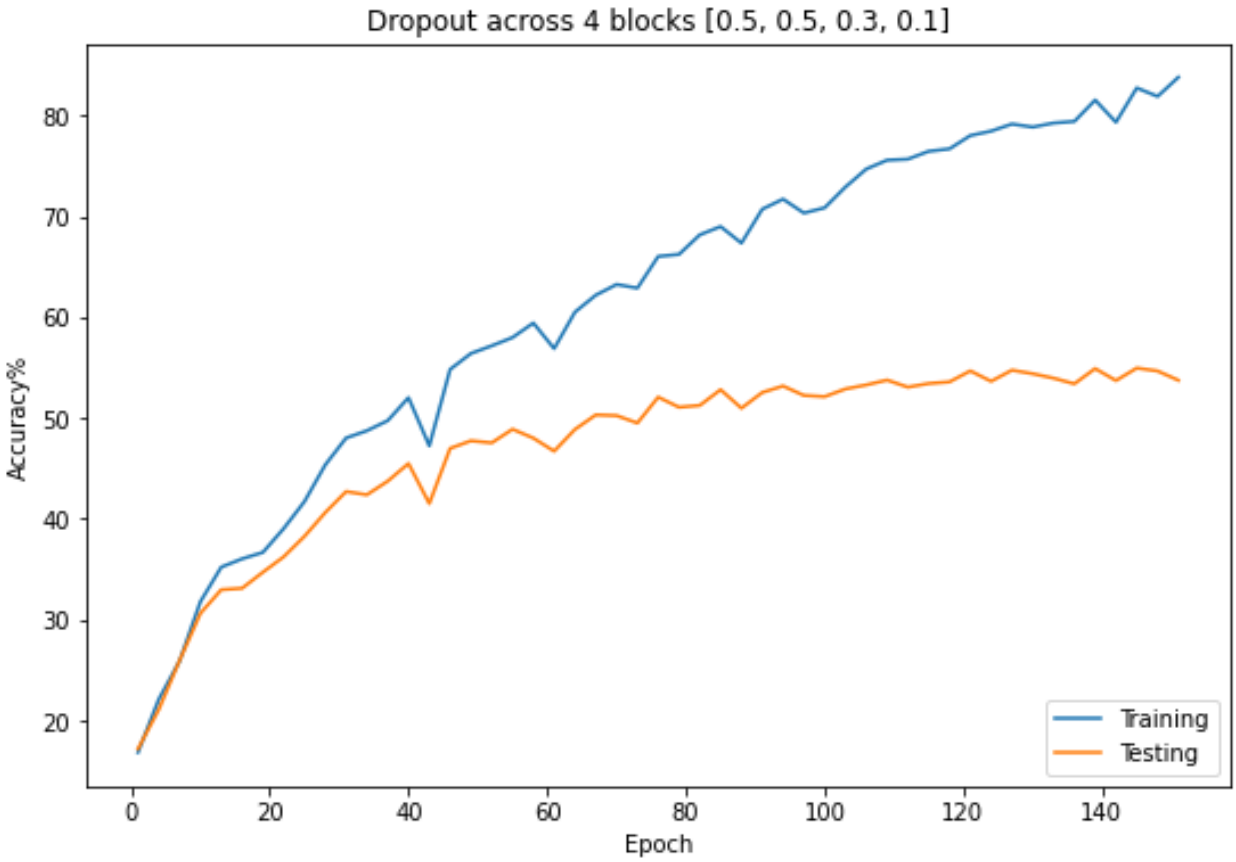
Best performing in SGD(lr=0.01, momentum=0.9) Accuracy: 79.88%

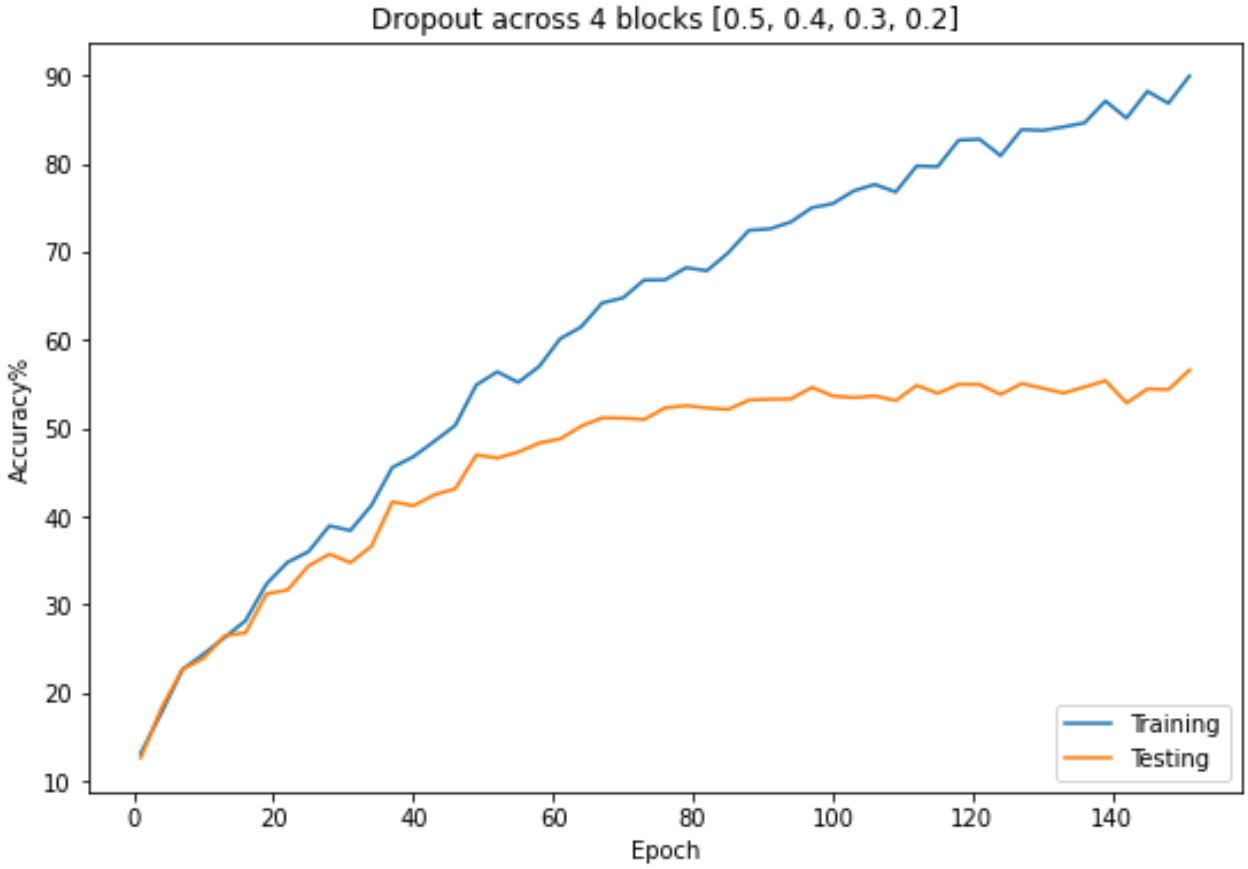
b) Train the network from scratch with the Tiny-CIFAR-10 dataset. Try different data augmentations and dropouts after different layers and with different dropout rates.

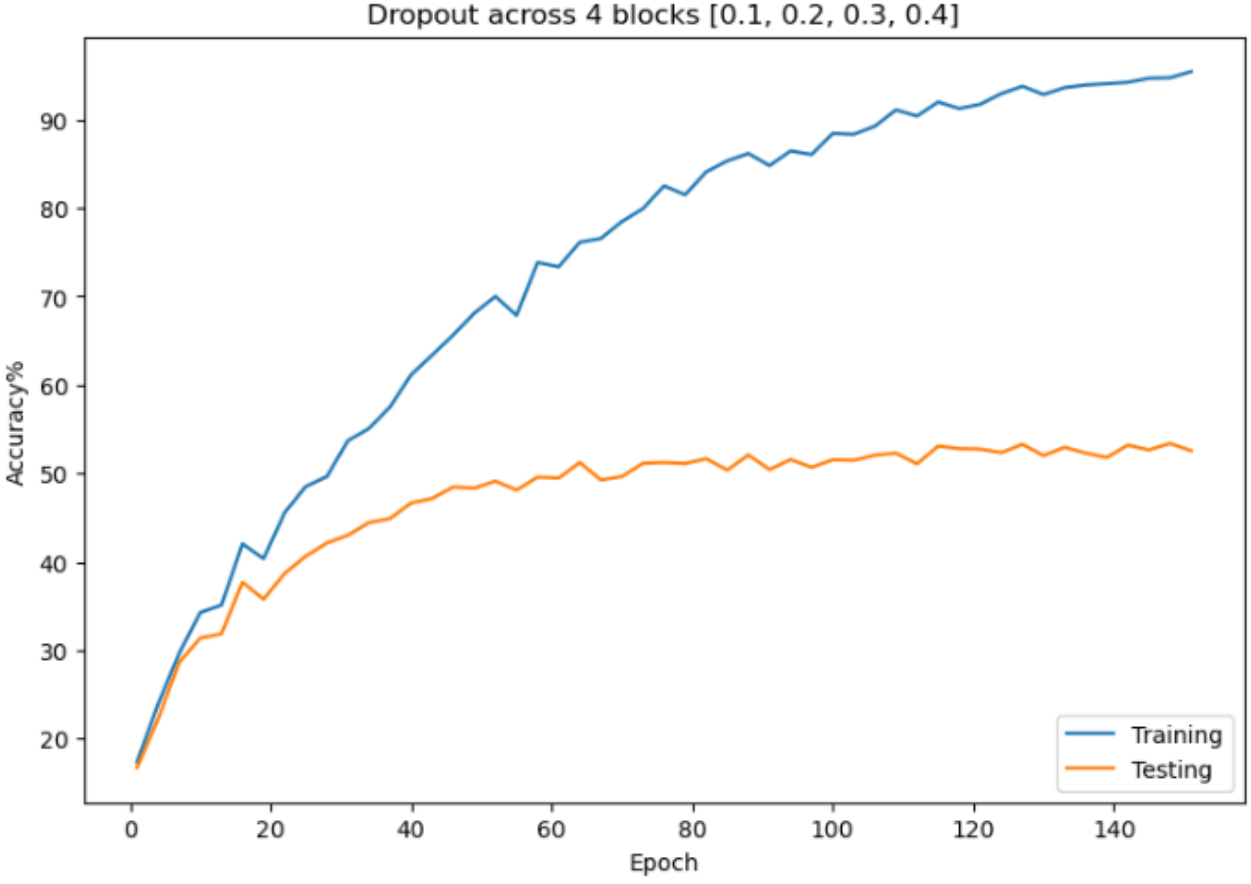
Following augmentations are used

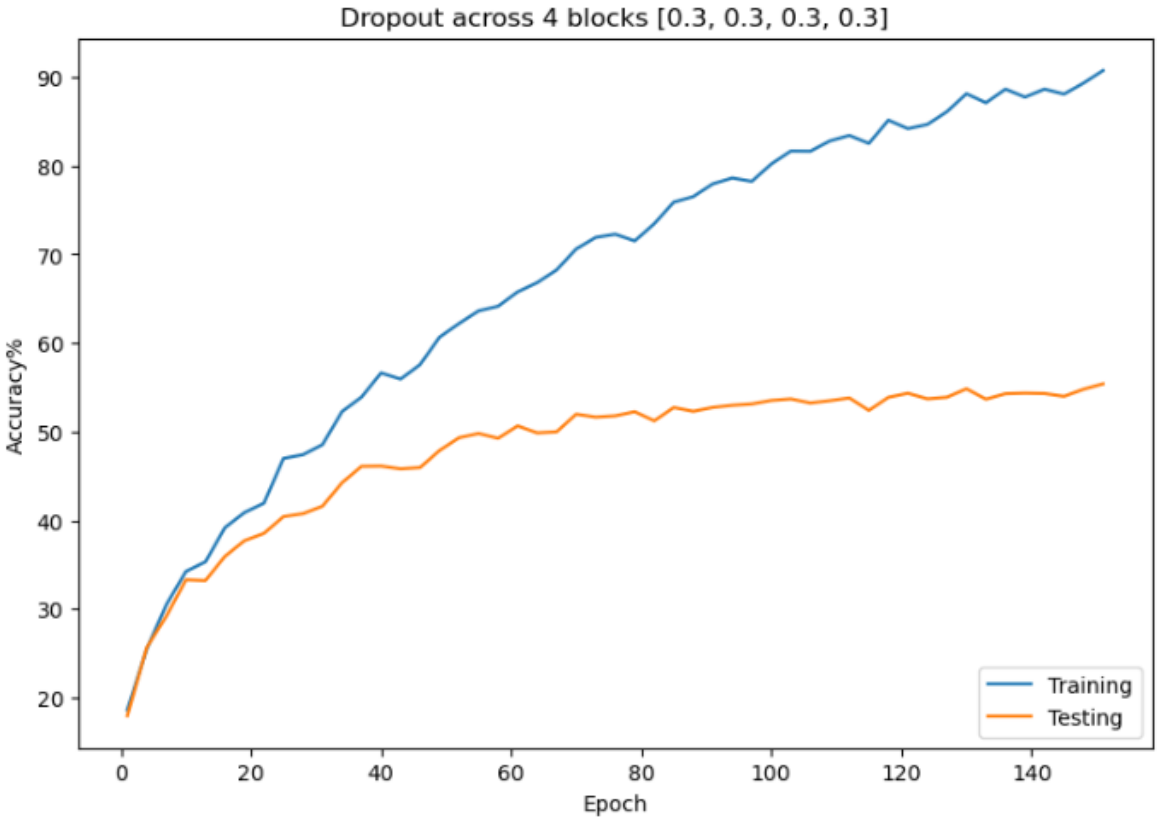
transforms.RandomAffine(degrees=(30, 70)),
transforms.RandomPerspective(distortion_scale=0.6, p=1.0),
transforms.ElasticTransform(alpha=150.0),
transforms.RandomSolarize(threshold=192.0),
transforms.ToTensor(),
transforms.Normalize(mean=(0.5, 0.5, 0.5), std=(0.5, 0.5, 0.5))

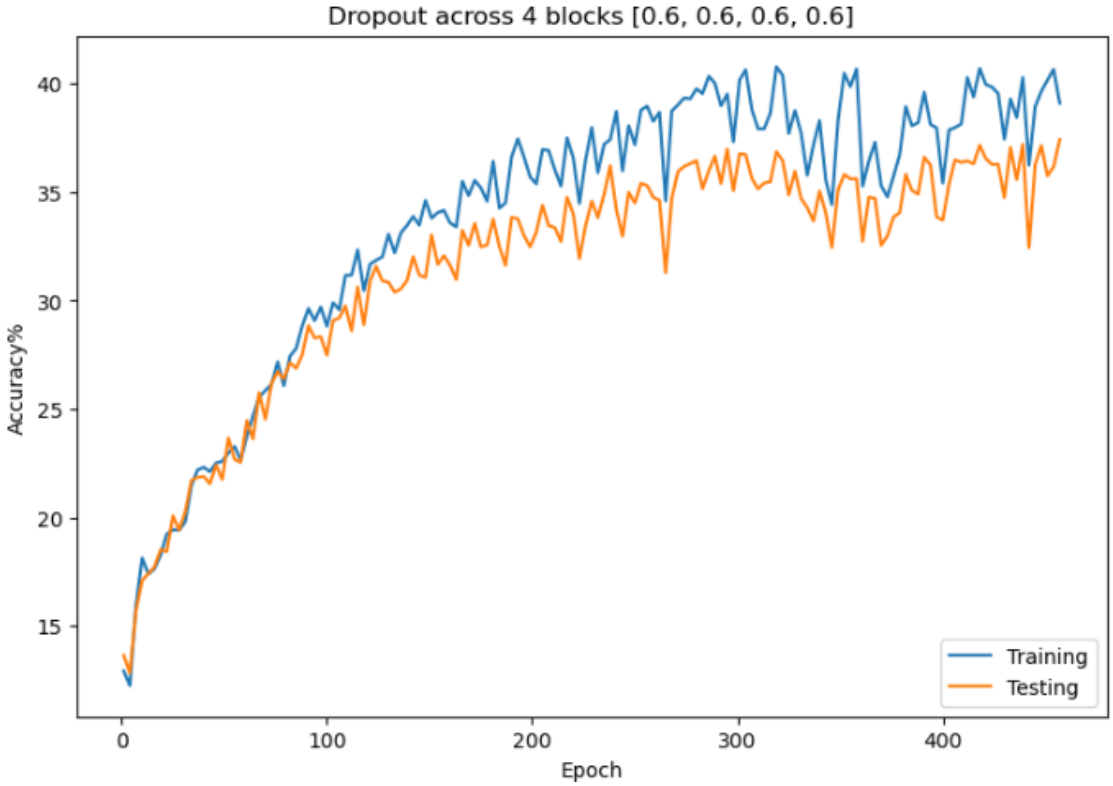
Result: however different the dropout rates network converges with test accuracy around 53-55%




Dropout across 4 blocks [0.5, 0.4, 0.3, 0.2]
Accuracy%
Epoch
Training
Testing
10
20
30
40
50
60
70
80
90
0
20
40
60
80
100
120
140

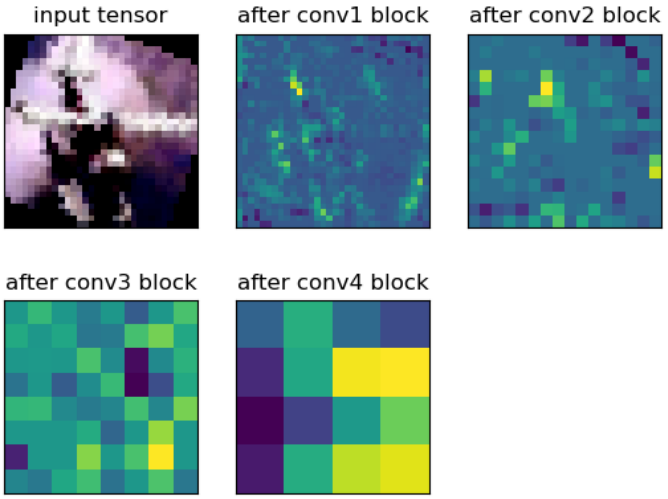

Dropout across 4 blocks [0.1, 0.2, 0.3, 0.4]
Accuracy%
Epoch
Training
Testing
20
30
40
50
60
70
80
90
0
20
40
60
80
100
120
140

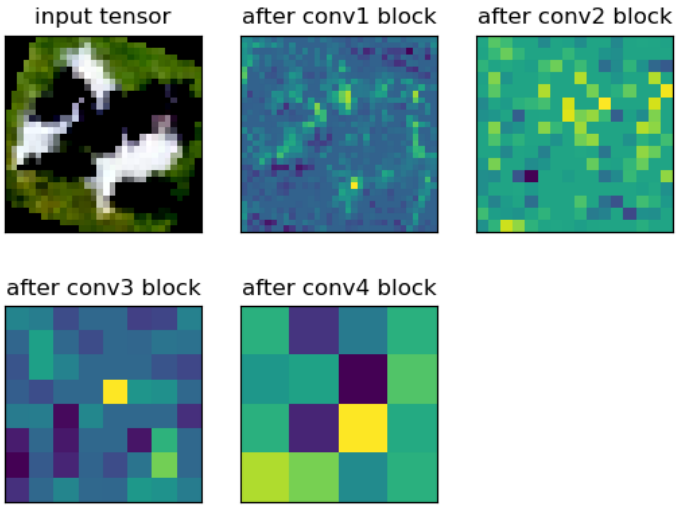

Dropout across 4 blocks [0.3, 0.3, 0.3, 0.3]
Accuracy%
Epoch
Training
Testing
90
80
70
60
50
40
30
20
0
20
40
60
80
100
120
140


Dropout across 4 blocks [0.6, 0.6, 0.6, 0.6]
Accuracy%
Epoch
Training
Testing
40
35
30
25
20
15
0
100
200
300
400

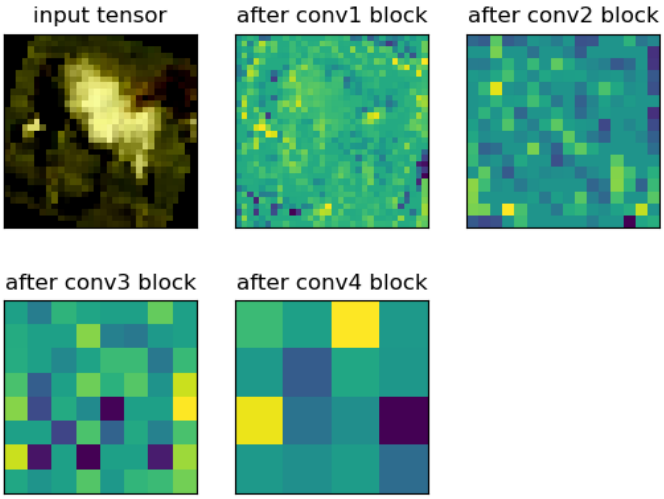
c) Visualize the activations of the CNN for a few test examples in each of the above cases.

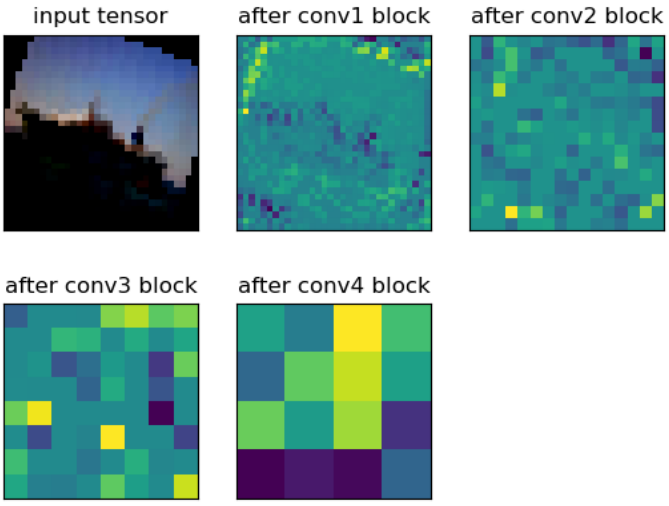
1) Dropout across 4 blocks [0.5, 0.5, 0.3, 0.1]



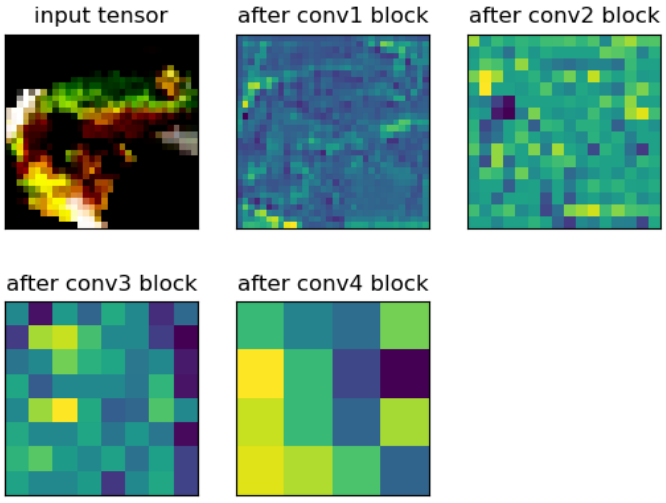


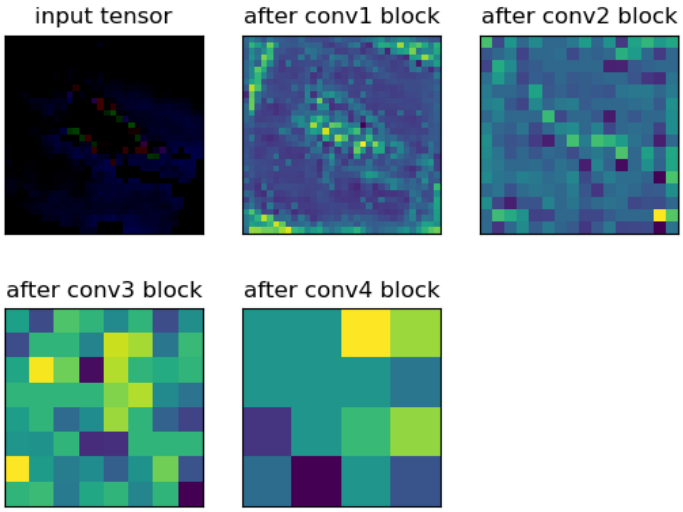
2) Dropout across 4 blocks [0.5, 0.4, 0.3, 0.2]



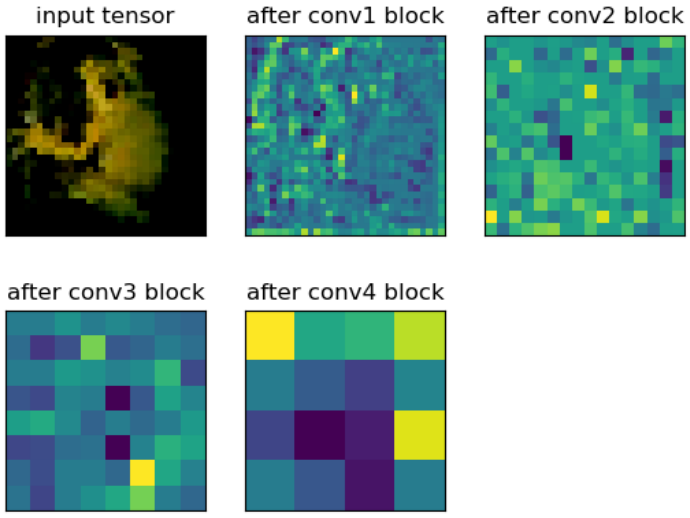


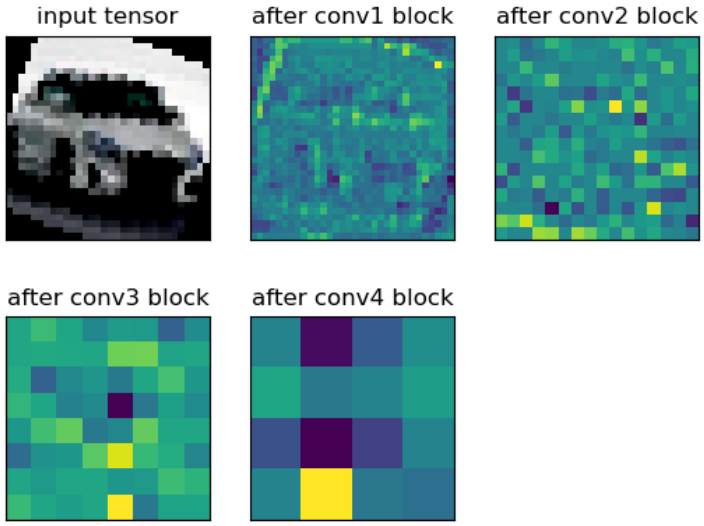
3) Dropout across 4 blocks [0.1, 0.2, 0.3, 0.4]





4) Dropout across 4 blocks [0.3, 0.3, 0.3, 0.3]





Results: It is visible that in the first few layers focus of the network is more towards borders whereas in the later layers it focuses on more complex features which are in the most part unintelligible for human perception.