Introduction to Machine Learning *Sai Venkatesh Kurella, VR62250*

HW3

- You will learn how to train a deep network using PyTorch tool. Please read the following tutorial. You may skip the data parallelism section.

https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

- CIFAR10 is a dataset of $60,000$ color images of size 32×32 from 10 categories. Please download the PyTorch tutorial code for CIFAR10 to start:

https://pytorch.org/tutorials/_downloads/cifar10_tutorial.py

- When you run the tutorial code, it will download CIFAR10 dataset for you. Please follow the instructions in the following link to install PyTorch: *https://pytorch.org/*

- To learn more, you can also find a tutorial for MNIST here:

https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html#sphx-glr-beginner-blitz-neural-networks-tutorial-py

and the sample model for MNIST here:

https://github.com/pytorch/examples/blob/master/mnist/main.py

and the sample code for Imagenet here:

https://github.com/pytorch/examples/blob/master/imagenet/main.py

- For all the following sections, train the model for 50 epochs and plot the curve for loss, training accuracy, and test accuracy evaluated every epoch.

1. Run the tutorial code out of the box and make sure you get reasonable results. You will report these results in Section 4, so no report needed here.

2. Change the code to have only a single fully connected layer. The model will have a single layer that connects the input to the output. What is the number of parameters? In PyTorch, "nn.Linear" can be used for fully connected layer.

   Ans: The number of parameters in a given layer is the count of learnable elements for a filter.
   The number of Parameters : 30730

   The following are the results:

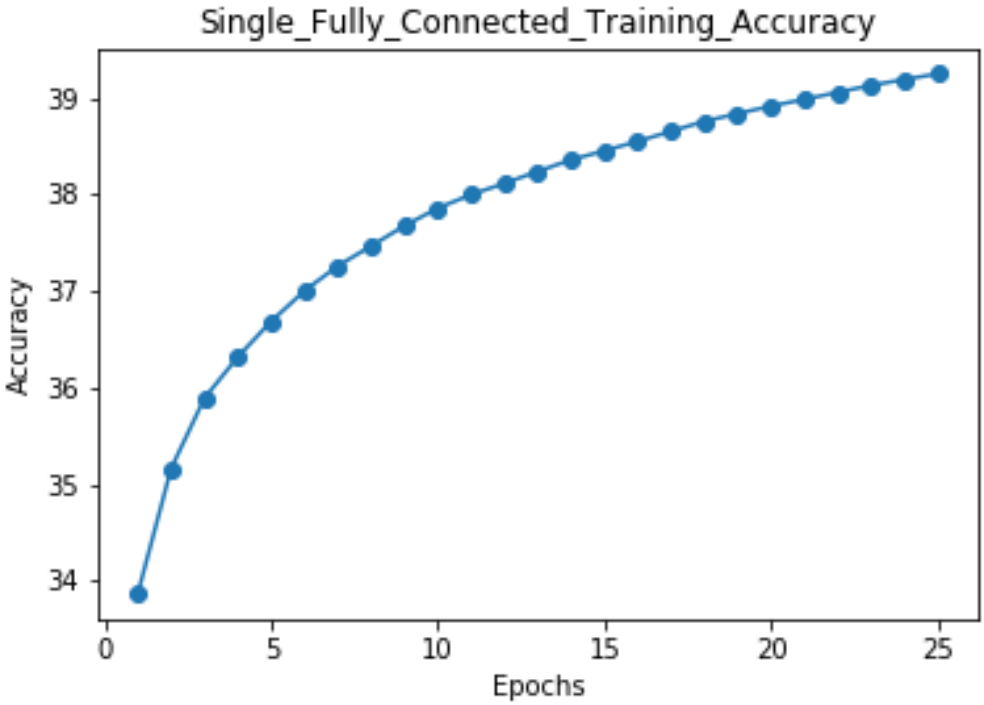


Figure 1: Single Fully Connected Layer - training accuracy

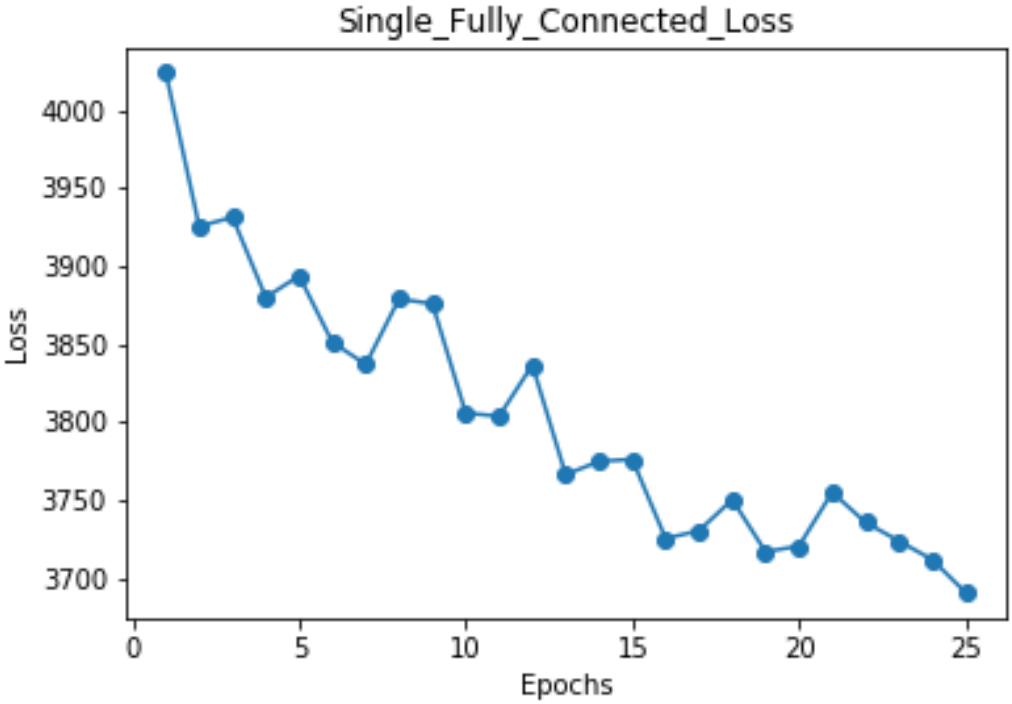


Figure 2: Single Fully Connected Layer - loss

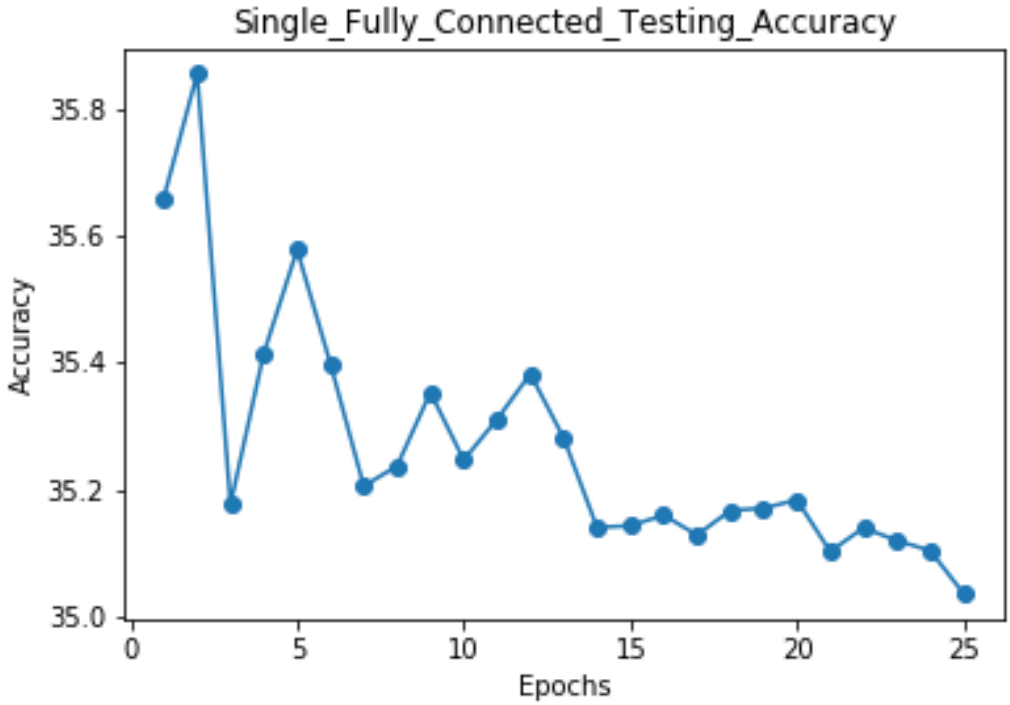


Figure 3: Single Fully Connected Layer - testing accuracy

3. Change the code to have multiple fully connected layers. Try having a layer from input to 120 neurons and then a layer to 84 neurons, and finally a layer to 10 neurons, one for each category. What happens if you do not use ReLU? Describe why.

   Ans:

   Parameters of the network: 379774

   The following are the results for Multiple Fully Connected Layer with ReLU:

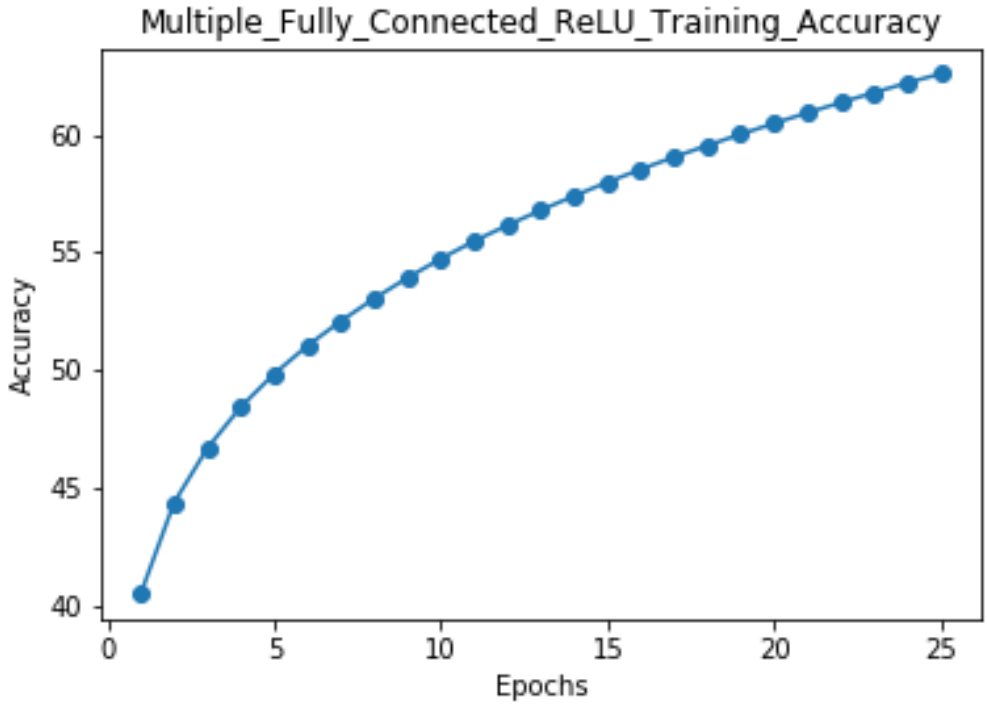


Figure 4: Multiple Fully Connected Layer with ReLU - training accuracy

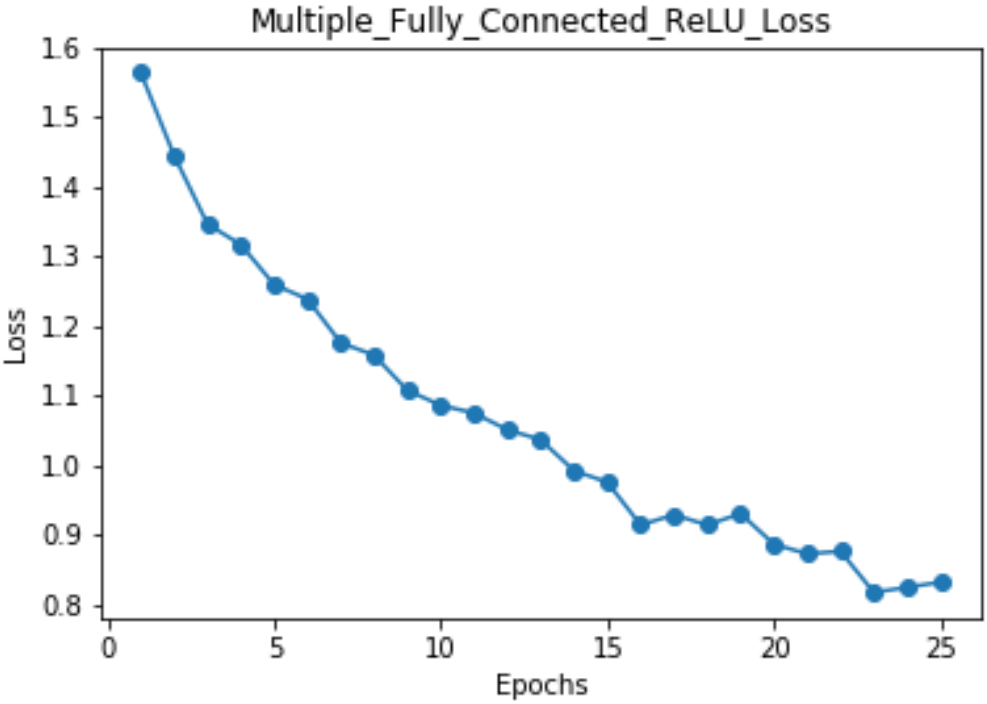


Figure 5: Multiple Fully Connected Layer with ReLU - loss

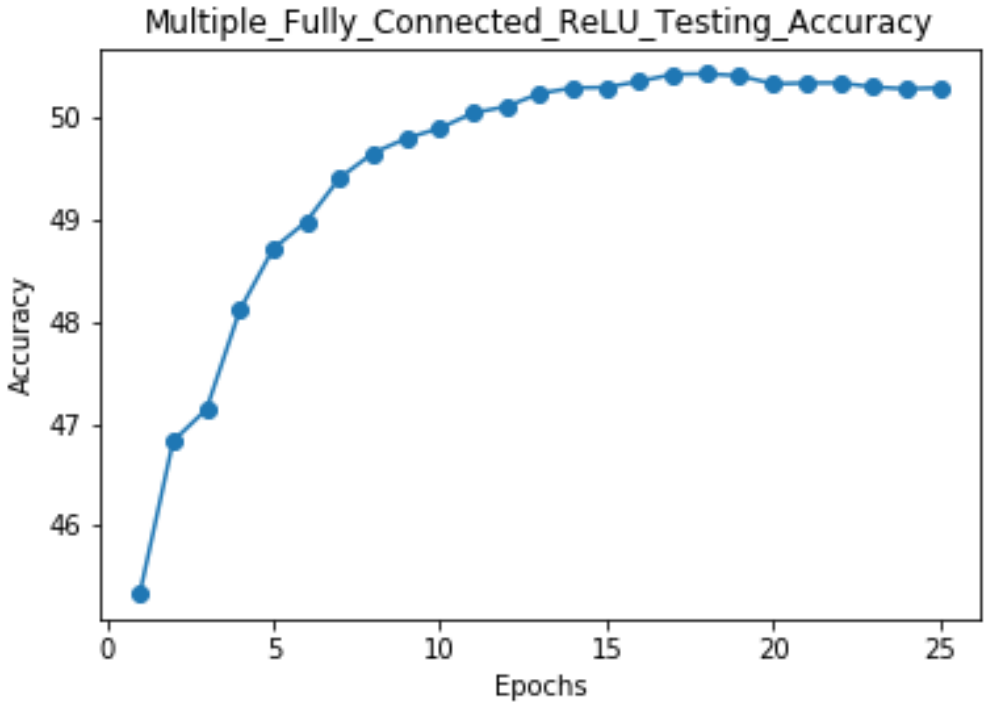


Figure 6: Multiple Fully Connected Layer with ReLU - testing accuracy

ReLU(Rectified Linear Unit) is a type of activation function, mathematically it is defined as y = max(0, x). ReLU layer applies ReLU function to the inputs and either turns the result to 0 or some number equal to x.

The rectified linear activation function is a piece-wise linear function that will output the input directly if is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

Without ReLU activation function, the Neural Network would only be a large linear mapping between inputs and outputs and would loose the classification capabilities because with linear mappings the output for different inputs probably would span a large range of values.

So, supposing a simple Neural Network with inputs x, an intermediate layer with variables u and output layer y, we have in the case there is a pure linear relation between the variables.

u=Ax+a

y=Bu+b=B(Ax+a)+b=BAx+(Ba+b)

which is the same as:

y=Cx+c

That is, there would be no point in using intermediate layers without a nonlinear activation function, because all could be transformed to a linear mapping from x to y.

The convolution in CNNs is also a linear operation. So, without activation functions in CNNs, this network would be similar to a chain of linear systems processing the input x to get the output y.

The following are the results for Multiple Fully Connected Layer without ReLU:

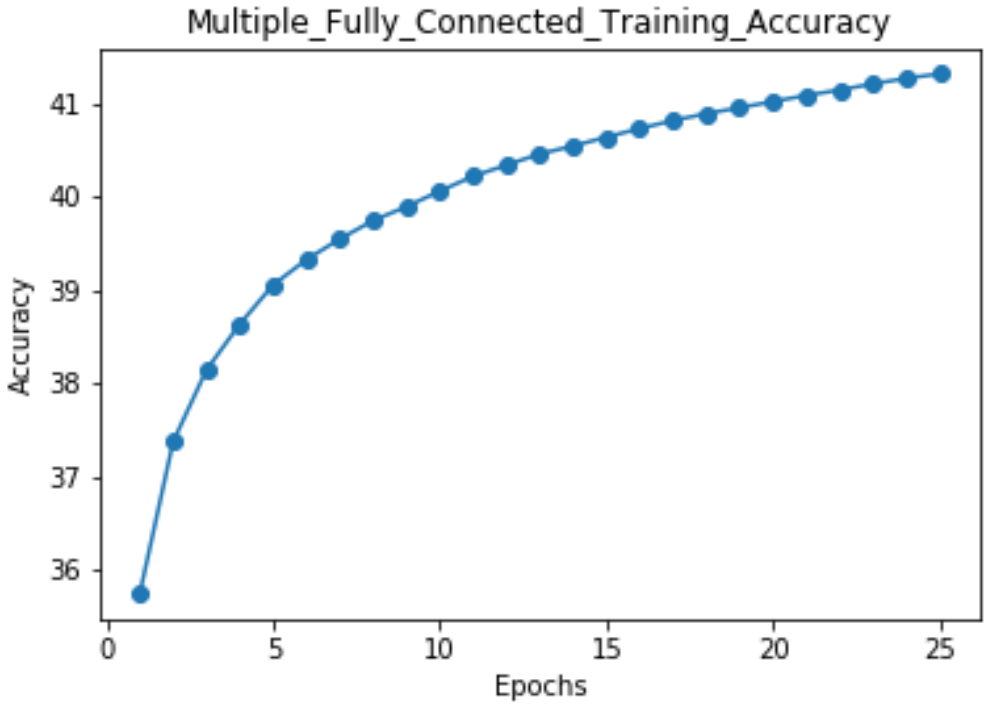


Figure 7: Multiple Fully Connected Layer - training accuracy

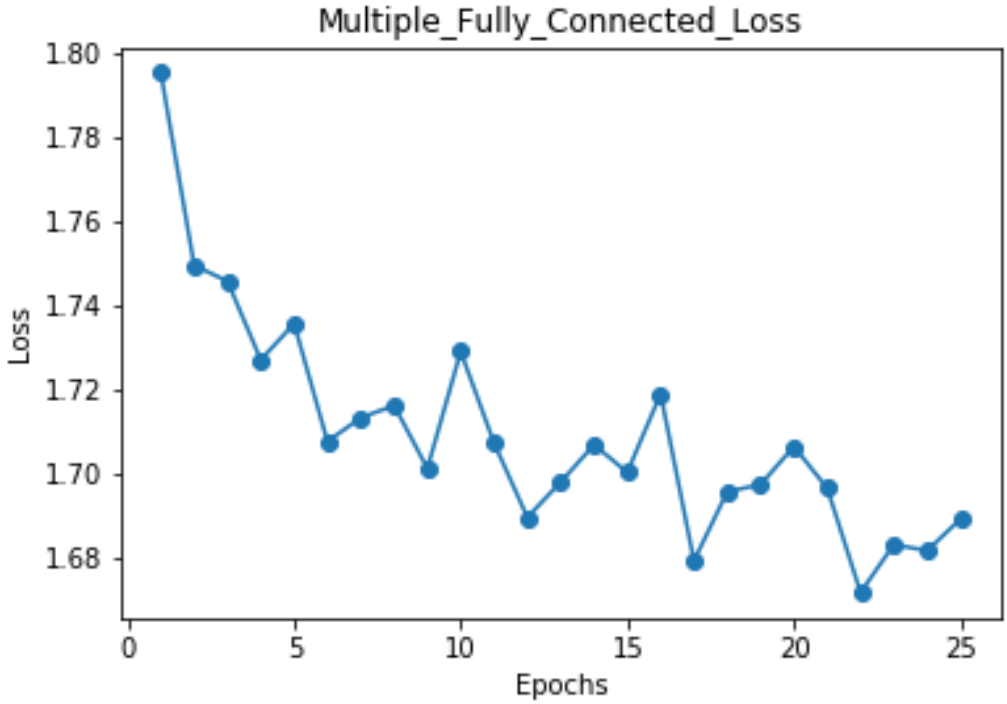


Figure 8: Multiple Fully Connected Layer- loss

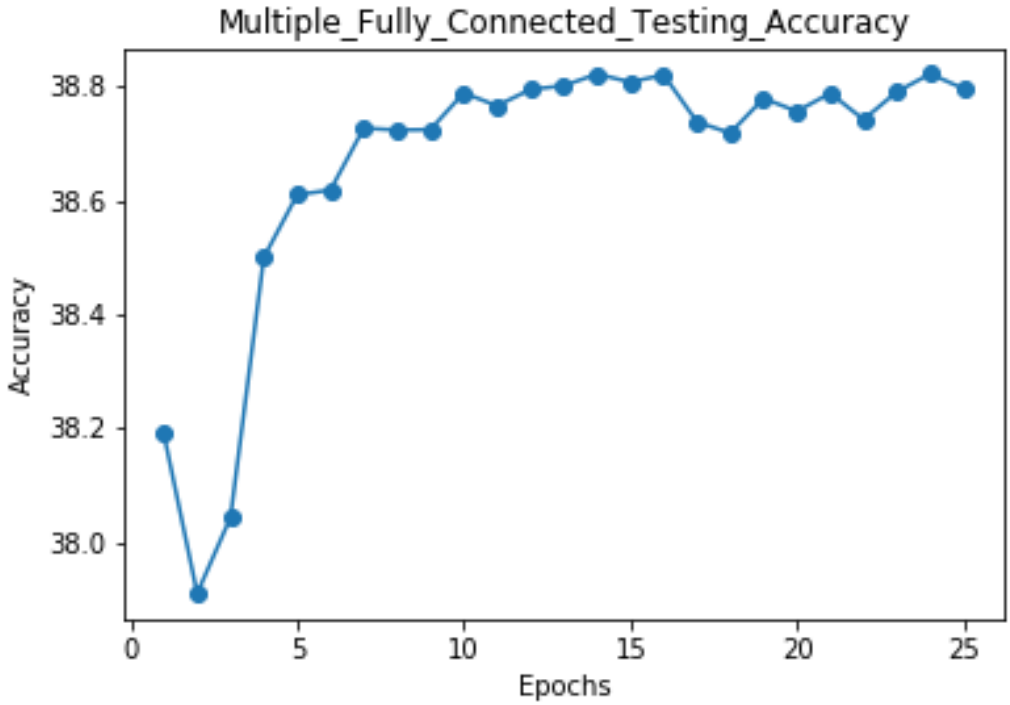


Figure 9: Multiple Fully Connected Layer- testing accuracy

4. Change the code by adding two convolutional layers along with maxpooling layers before the fully connected layers. This will be similar to the example in the tutorial. Use this model for the following sections.

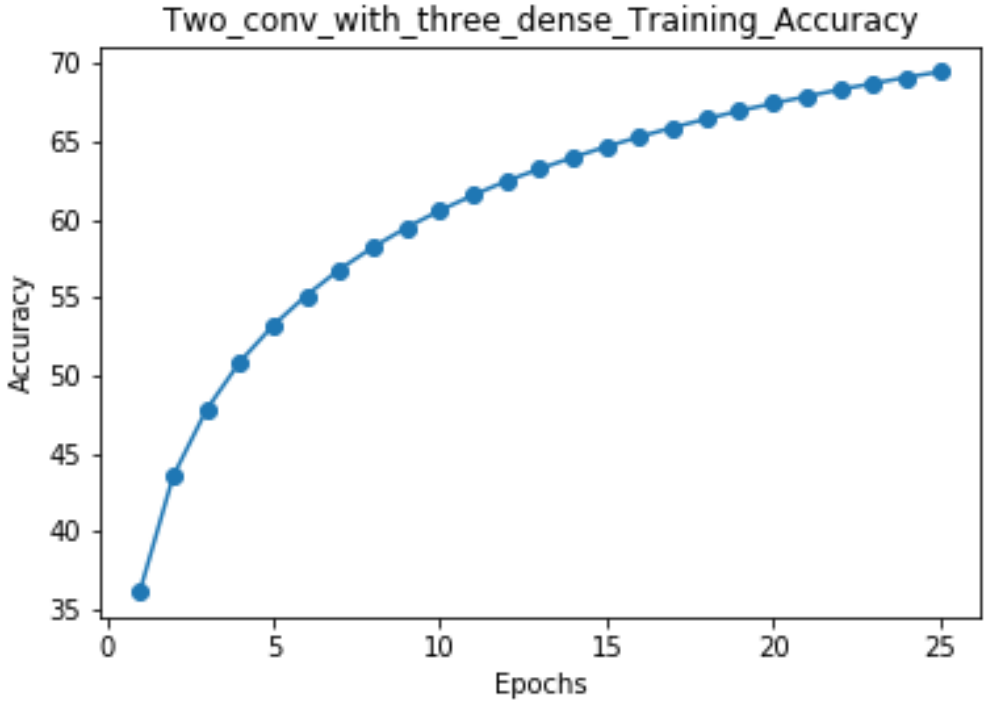


Figure 10: Two conv with three dense layers - training accuracy

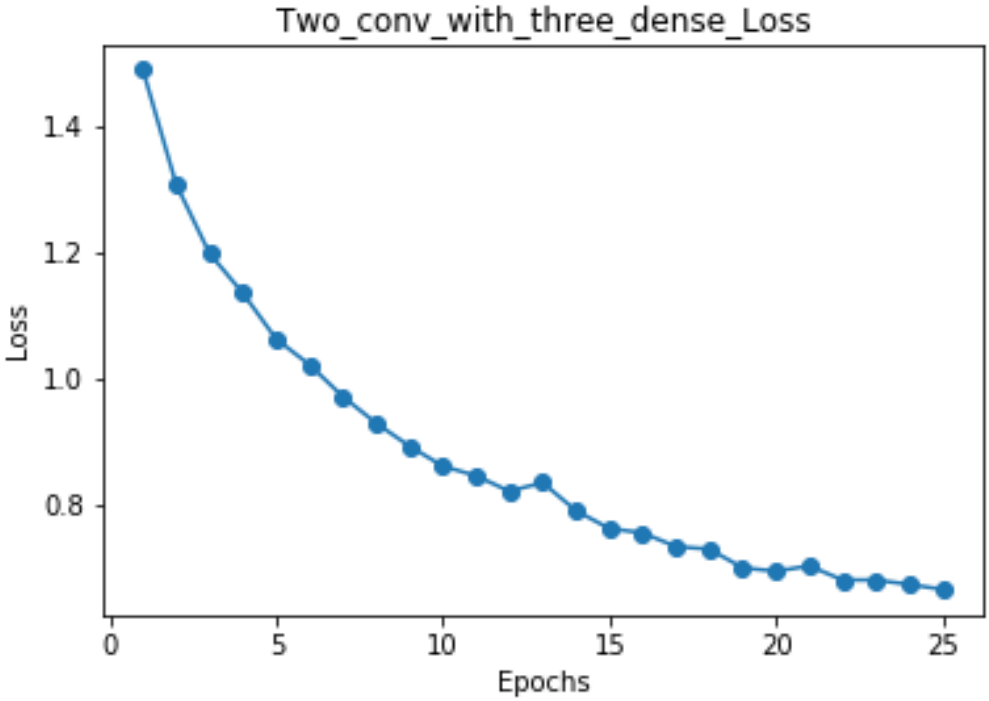


Figure 11: Two conv with three dense layers - loss

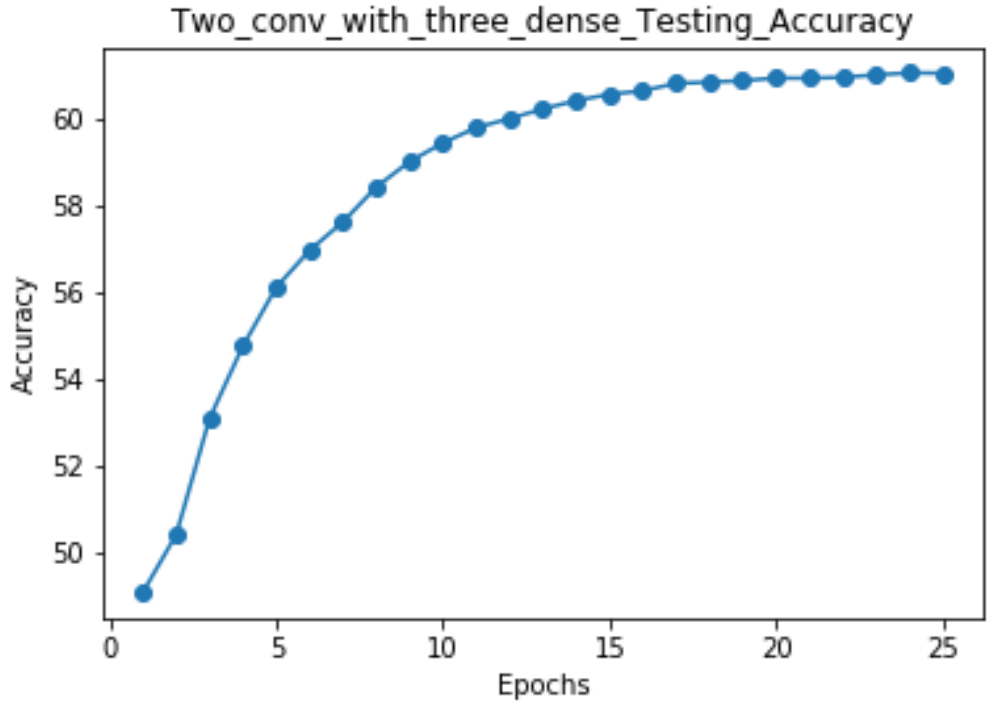


Figure 12: Two conv with three dense layers - testing accuracy

5. Try multiple batch sizes to see the effect and describe the findings. Please use batch size of 1, 4, and 1000. If 1000 does not fit into the memory of your machine, please feel free to reduce it to a largest possible number.

The following are the results obtained for batch sizes of 1, 4, and 1000:
Parameters of the network for batch size 1: 62006

The training accuracy in case of Batch size 1 increases evenly with increase in epochs ranging from 38 to 48 accuracy with 0-25 epoch range.
The loss function in case of Batch size 1 is unevenly distributed along the increase of epochs ranging from 0 to 25. The loss approximately began at 1.55 and varied across the epoch range until 1.60 to 1.70.
The test accuracy in case of Batch size 1 increases evenly with increase in epochs till 47.5 and then gradually decreases from that point ranging to 45 accuracy with 0-25 epoch range.

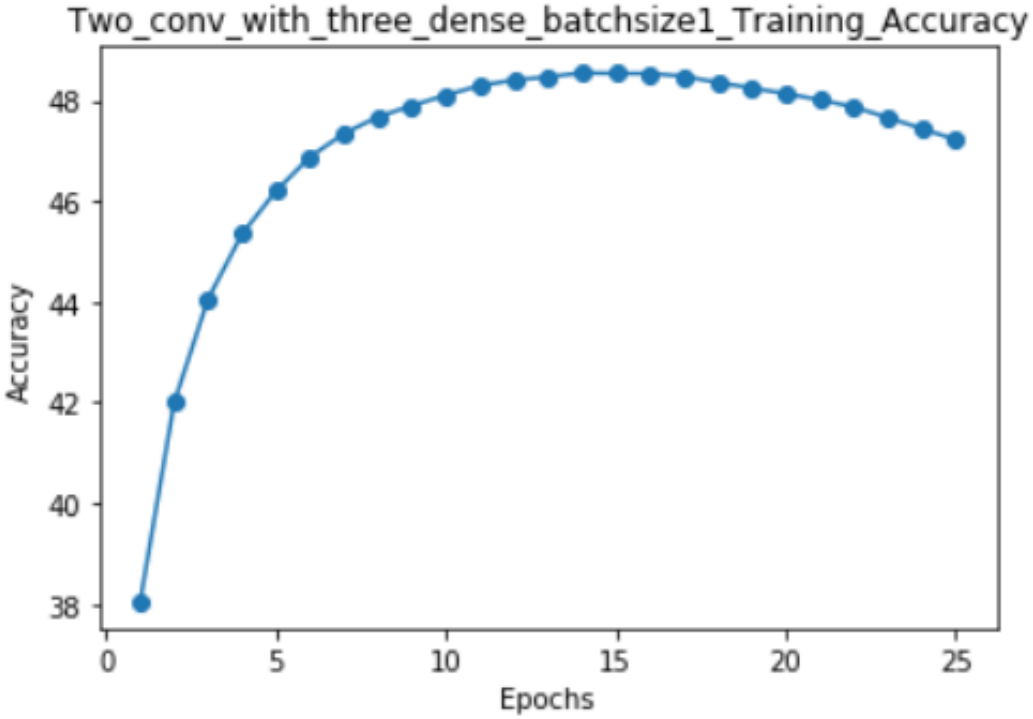


Figure 13: Two conv with three dense layers with 1 sample per batch - training accuracy

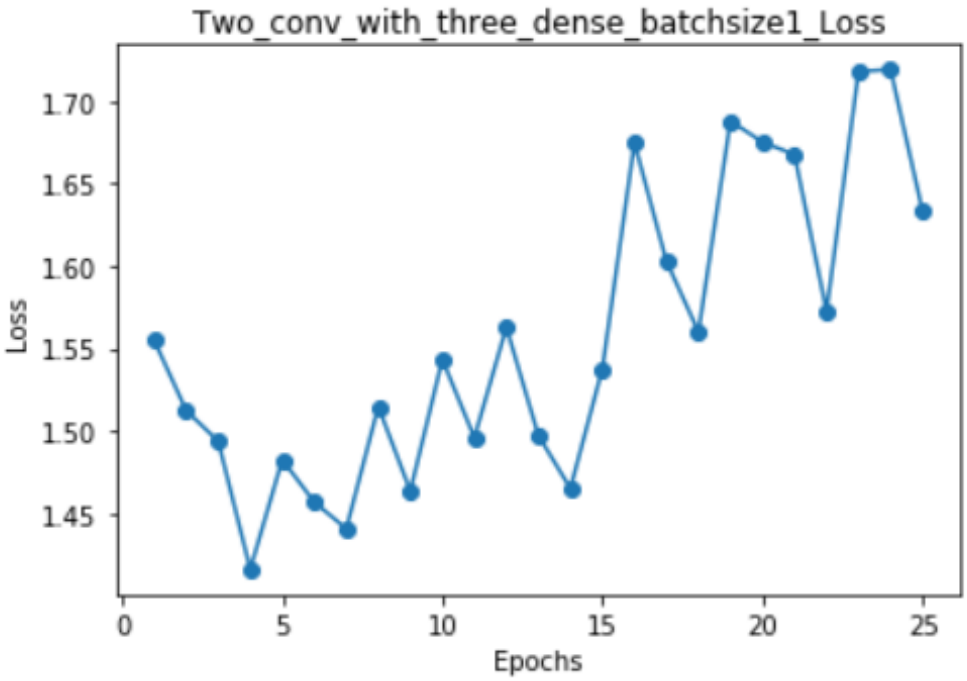


Figure 14: Two conv with three dense layers with 1 sample per batch - loss

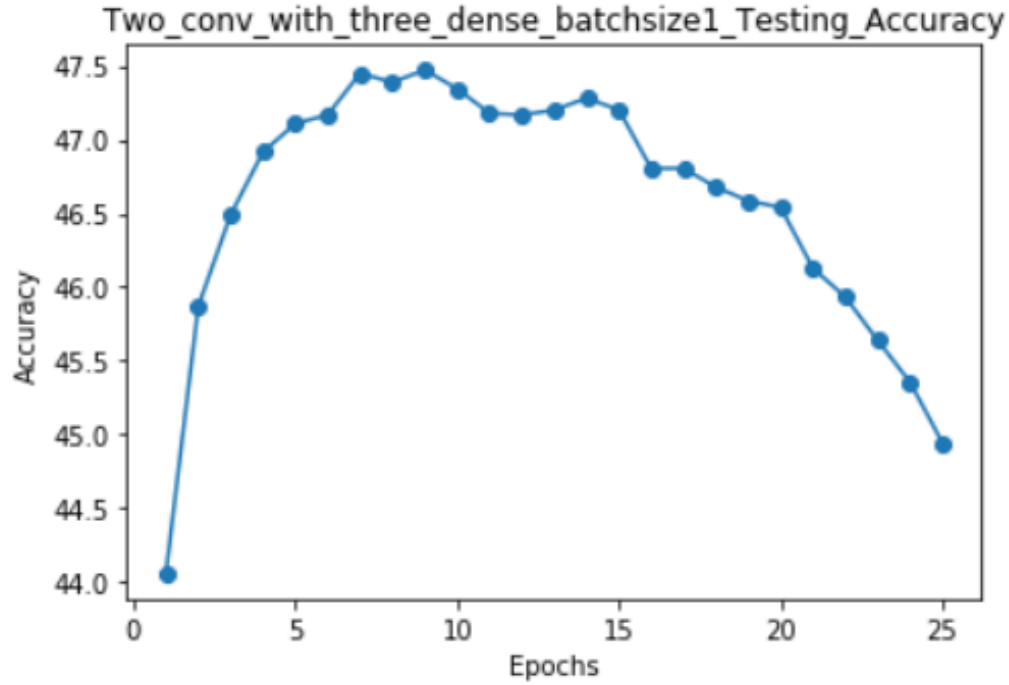


Figure 15: Two conv with three dense layers with 1 samples per batch - testing accuracy

Parameters of the network for batch size 4: 62006
The training accuracy in case of Batch size 4 increases evenly with increase in epochs ranging from 35 to 70 accuracy with 0-25 epoch range.
The loss function in case of Batch size 4 is evenly distributed along the increase of epochs ranging from 0 to 25. The loss approximately began at around 1.5 and steadily decreased across the epoch until almost 0.7 to 0.1.

The test accuracy in case of Batch size 4 increases evenly with increase in epochs from around accuracy 48 to accuracy 60 with in 0-25 epoch range.

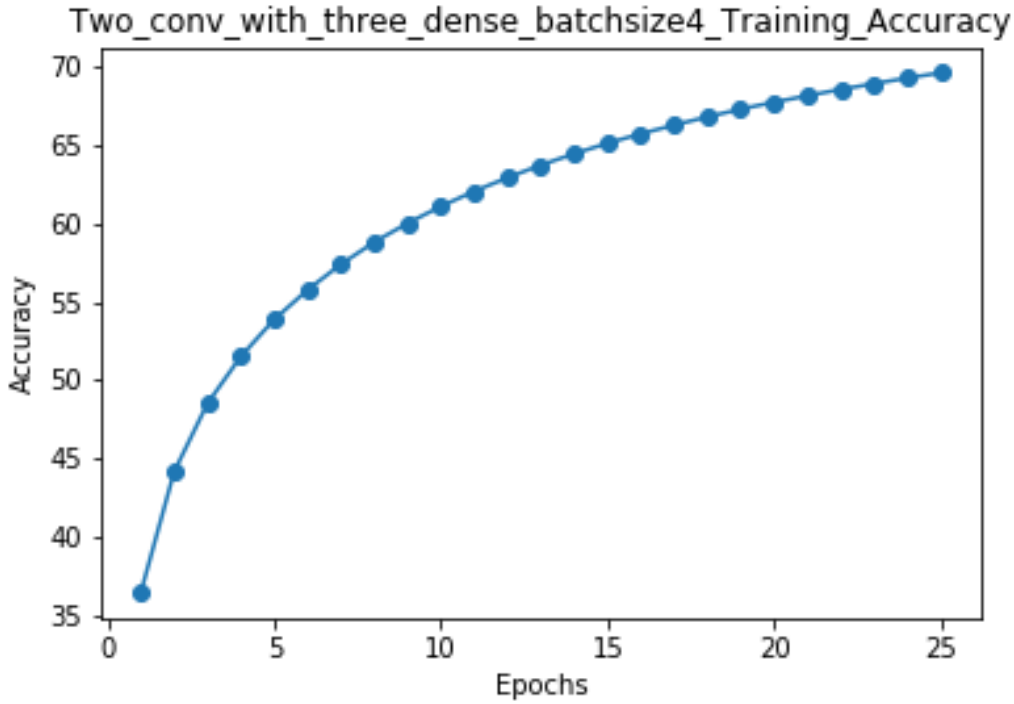


Figure 16: Two conv with three dense layers with 4 samples per batch - training accuracy

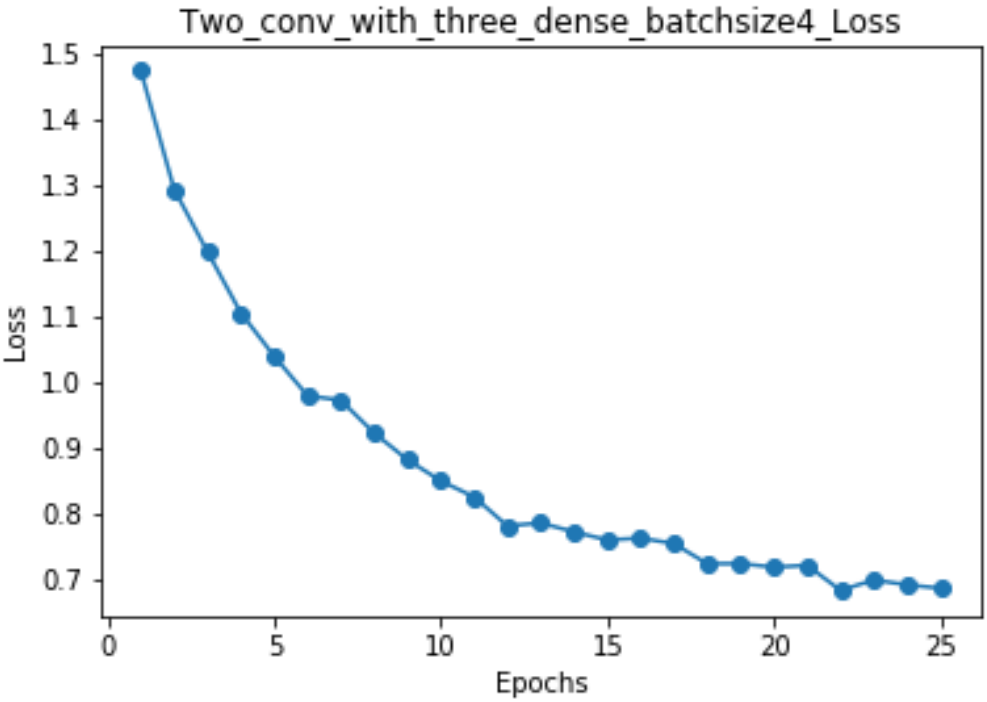


Figure 17: Two conv with three dense layers with 4 samples per batch - loss

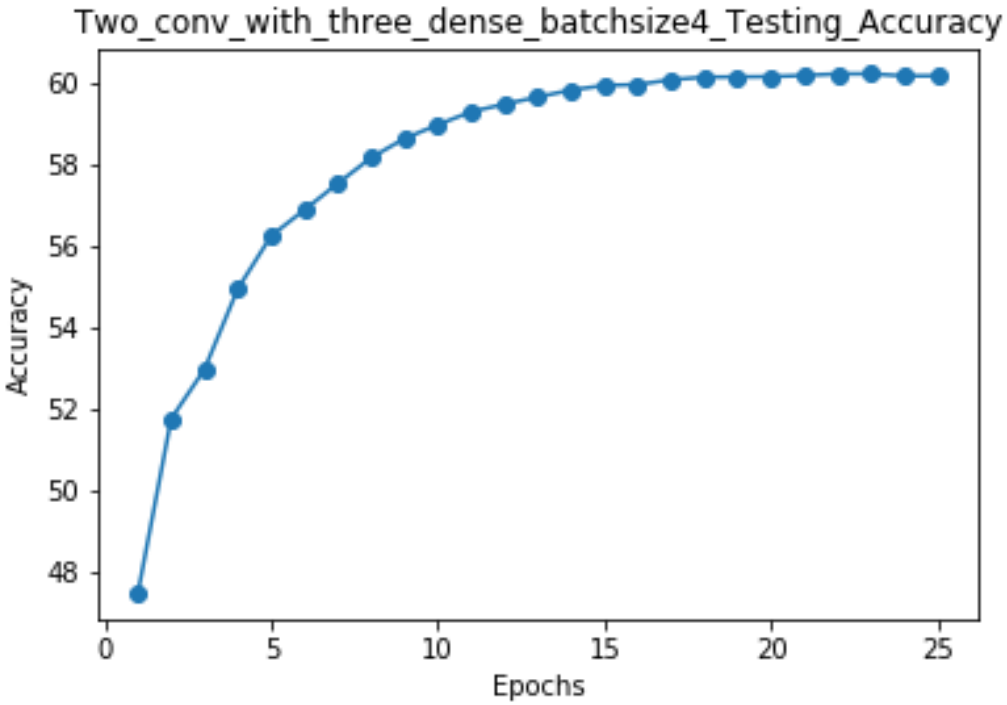


Figure 18: Two conv with three dense layers with 4 samples per batch - testing accuracy

Parameters of the network for batch size 1000: 62006
The training accuracy in case of Batch size 1000 increases evenly with increase in epochs ranging from 80 to 87 accuracy with 0-25 epoch range.
The test accuracy in case of Batch size 1000 increases evenly with increase in epochs from around accuracy 60 to accuracy 63 with in 0-25 epoch range.

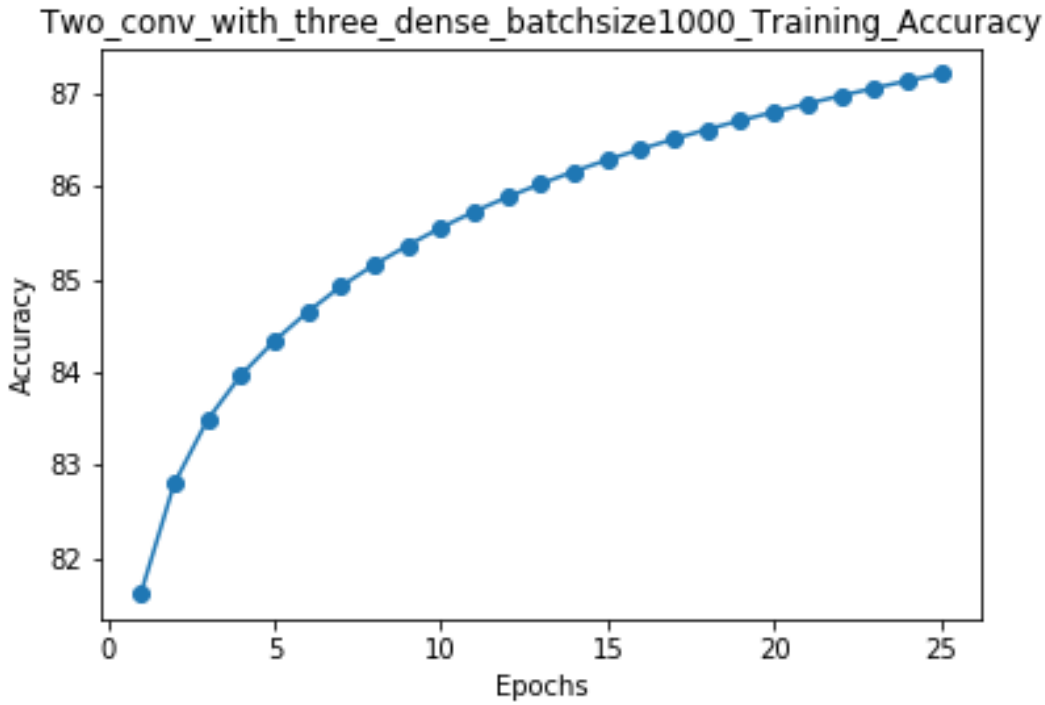


Figure 19: Two conv with three dense layers with 1000 samples per batch - training accuracy

Figure 20: Two conv with three dense layers with 1000 samples per batch - testing accuracy

6. Try multiple learning rates to see the effect and describe the findings. Please use learning rates of 10, 0.1, 0.01, and 0.0001.

Parameters of the network with learning rate 10: 62006
The training accuracy in case of learning rate 10 decreases from accuracy 10.15 to almost 9.5 within the epoch range of 1 to 10 and then increases up to a range of accuracy 10 with increase in epochs ranging from 10-25 epoch range.
The loss function in case of learning rate 10 varies unevenly from 19.5 to 18.5 range with increase in epoch from 0 to 25.
The test accuracy in case of learning rate 10 is almost steady at the range of 10 accuracy with in 0-25 epoch range. Here, the model doesn't seem to learn.

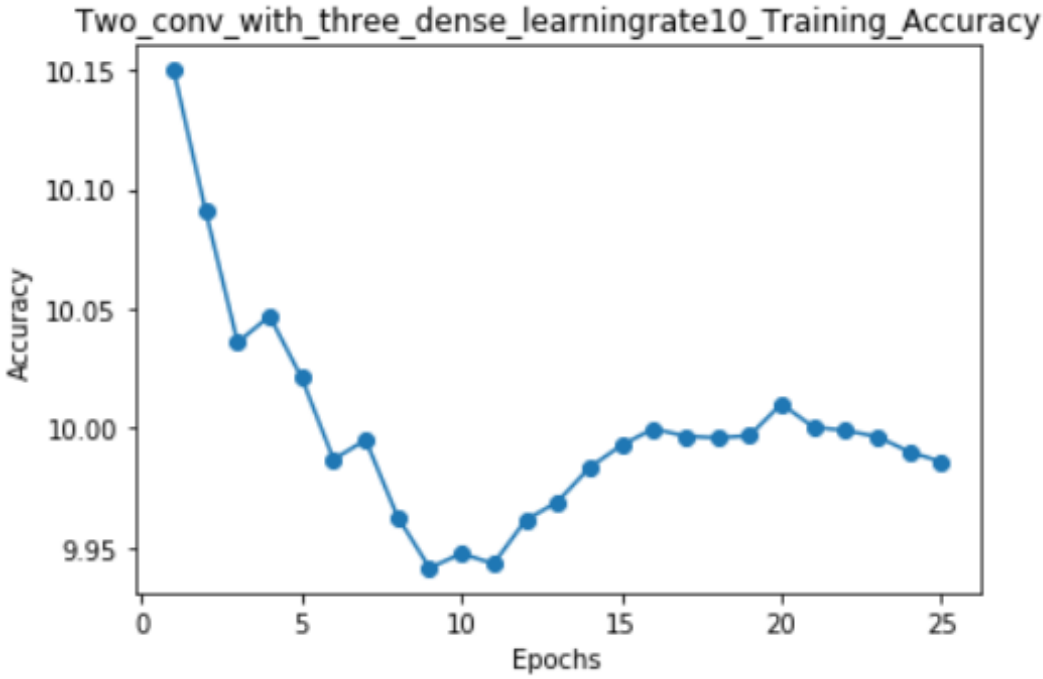


Figure 21: Two conv with three dense layers with a learning rate of 10 - training accuracy

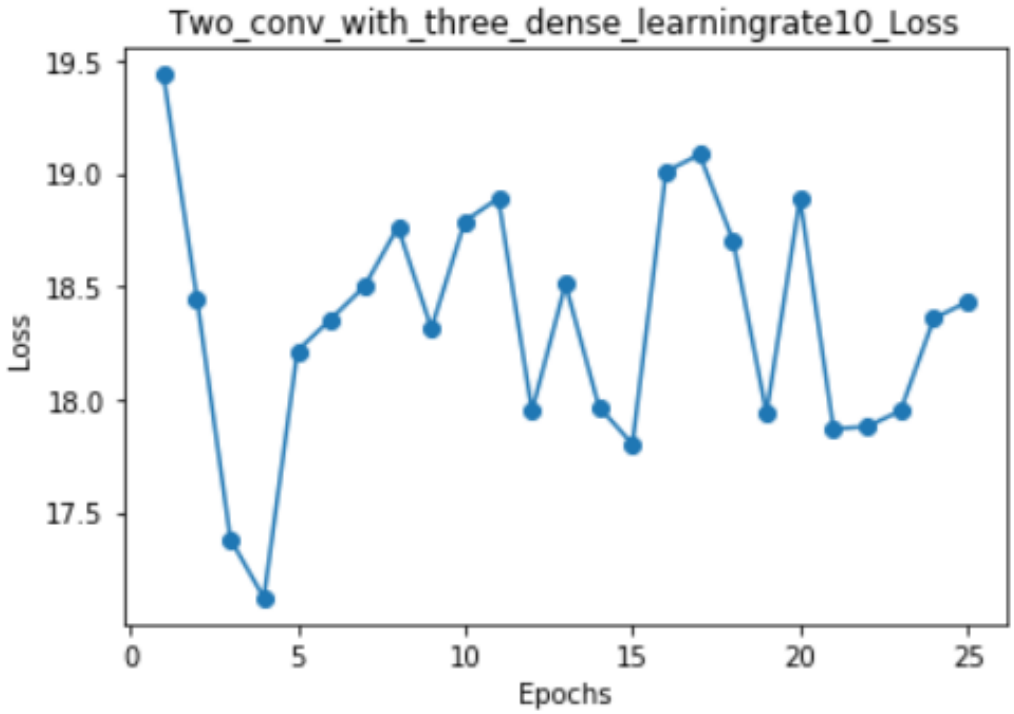


Figure 22: Two conv with three dense layers with a learning rate of 10 - loss

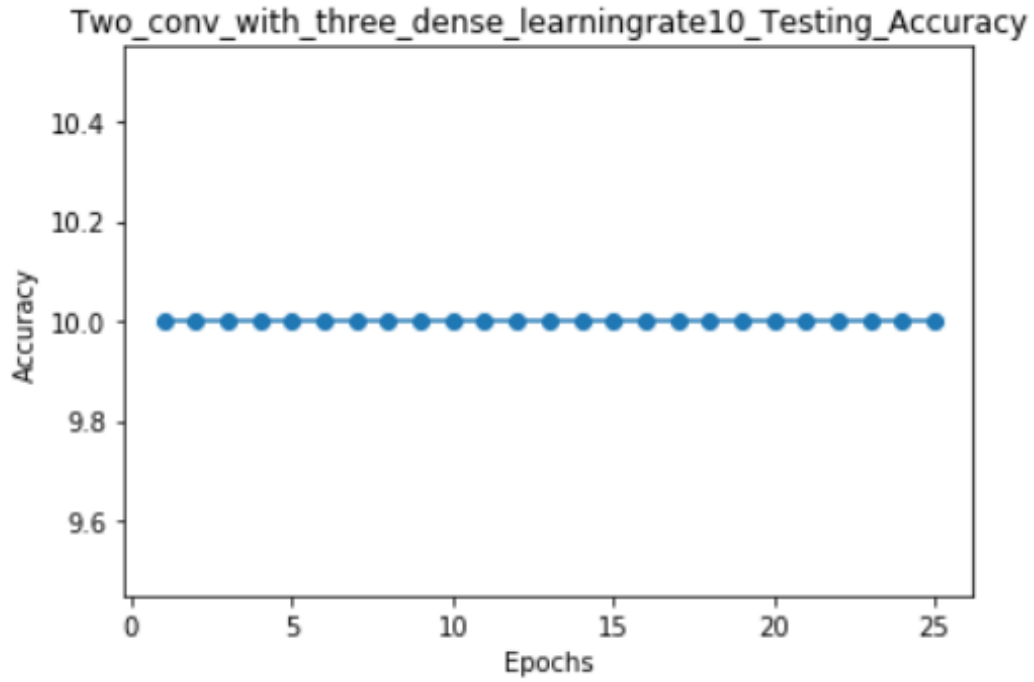


Figure 23: Two conv with three dense layers with a learning rate of 10 - testing accuracy

Parameters of the network with learning rate 0.1: 62006
The training accuracy in case of learning rate 0.1 increases in a irregular fashion from accuracy 9.88 to almost 10.0 within the epoch range of 0 to 25.
The loss function in case of learning rate 0.1 varies unevenly from 2.358 to 2.360 range with increase in epoch from 0 to 25. Here, the model doesn't seem to learn.

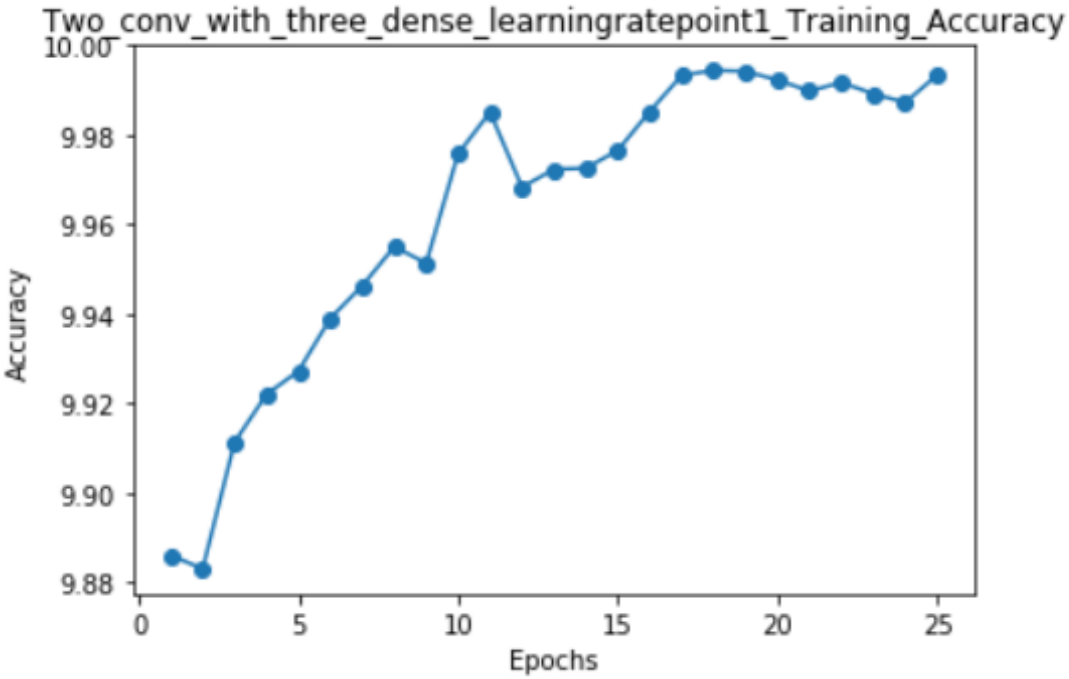


Figure 24: Two conv with three dense layers with a learning rate of 0.1 - training accuracy

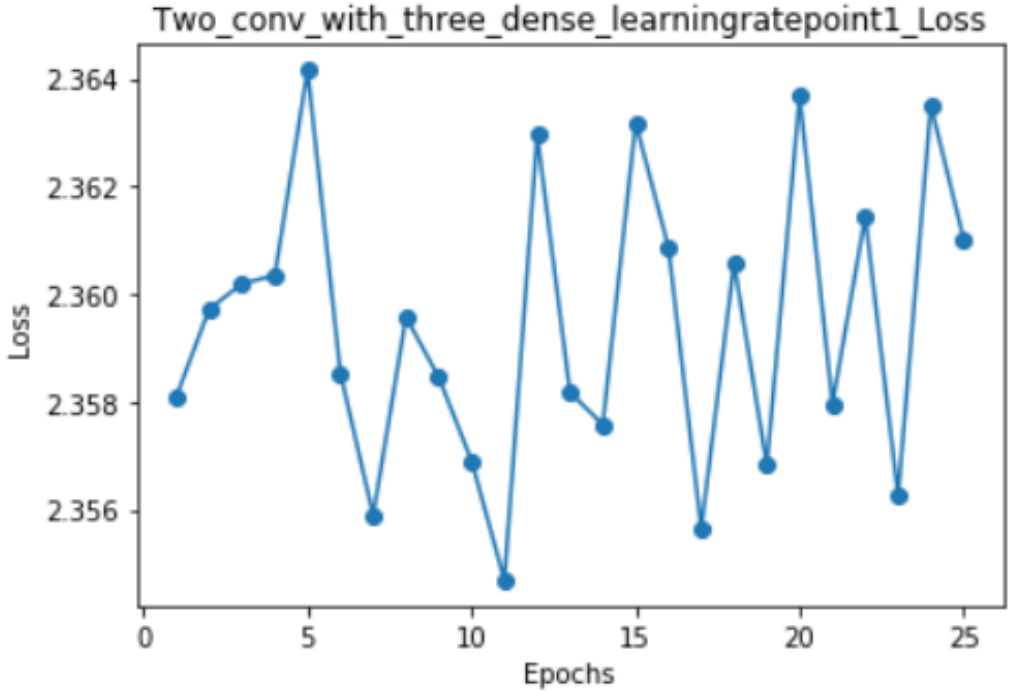


Figure 25: Two conv with three dense layers with a learning rate of 0.1 - loss

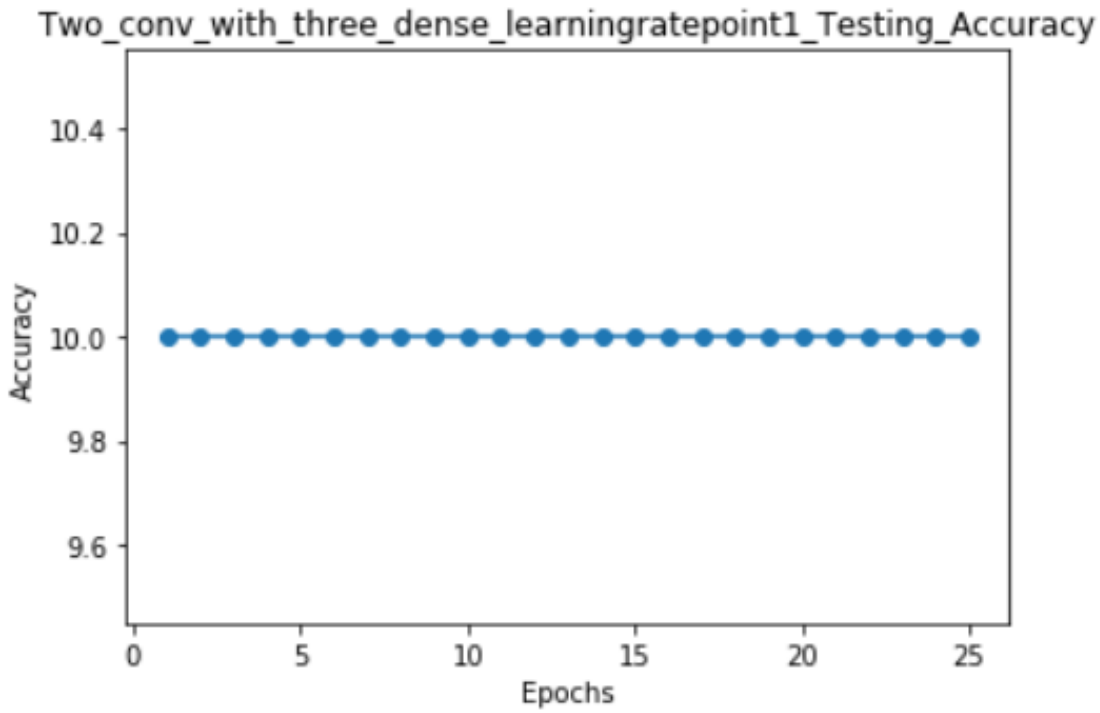


Figure 26: Two conv with three dense layers with a learning rate of 0.1 - testing accuracy

Parameters of the network with learning rate 0.01: 62006

The training accuracy in case of learning rate 0.01 increases in a irregular fashion from accuracy 9.92 to almost 9.96 within the epoch range of 0 to 25.
The loss function in case of learning rate 0.01 varies unevenly from 2.3080 to 2.3085 range with increase in epoch from 0 to 25. Here, the model doesn't seem to learn.

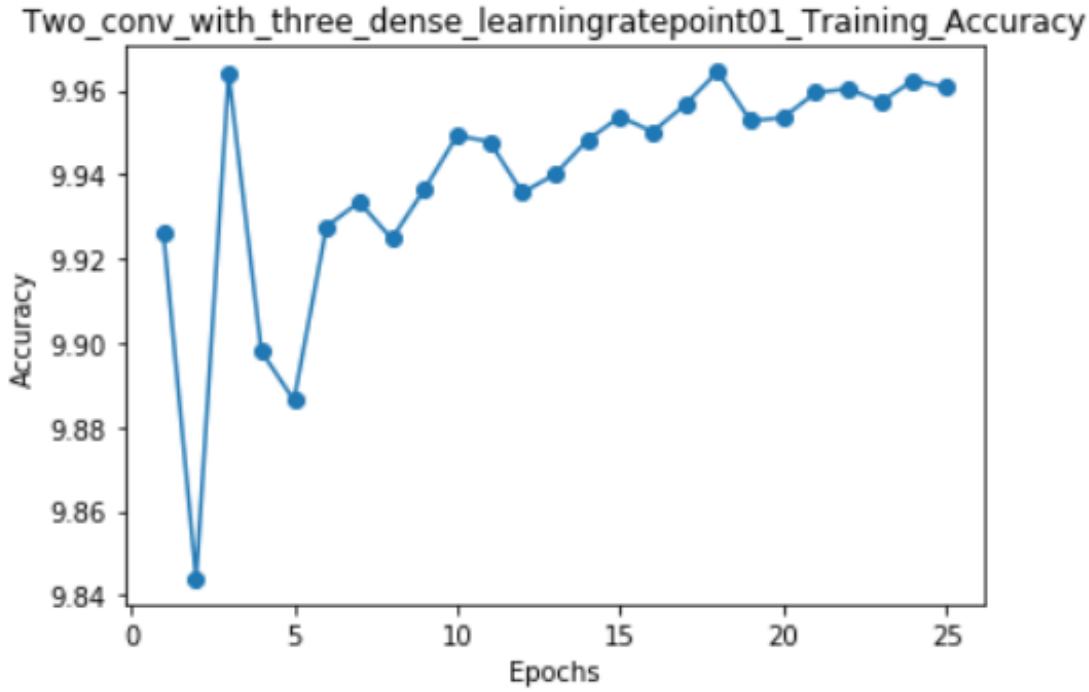


Figure 27: Two conv with three dense layers with a learning rate of 0.01 - training accuracy

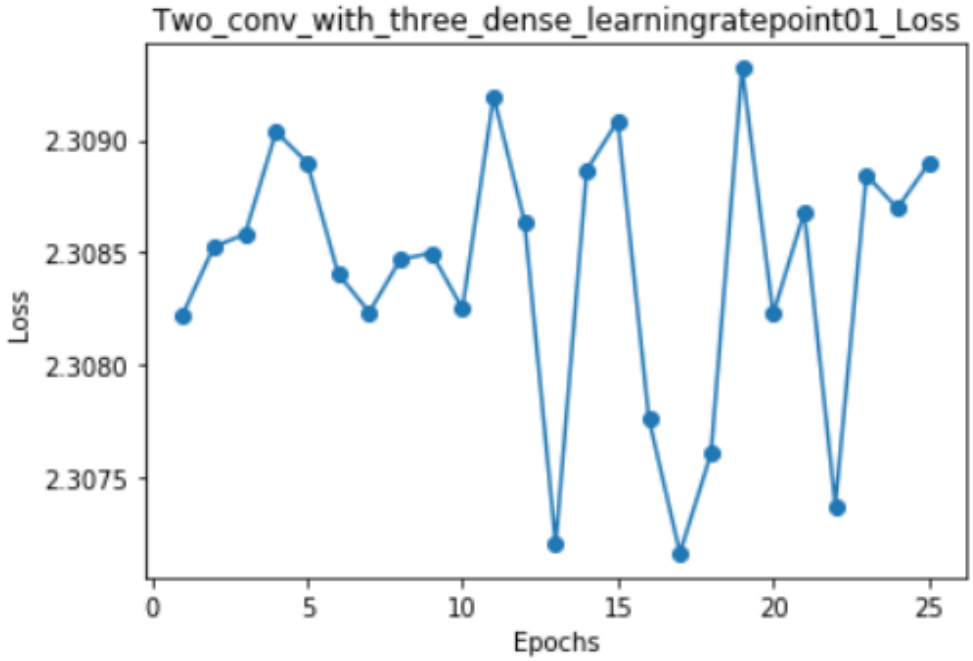


Figure 28: Two conv with three dense layers with a learning rate of 0.01 - loss

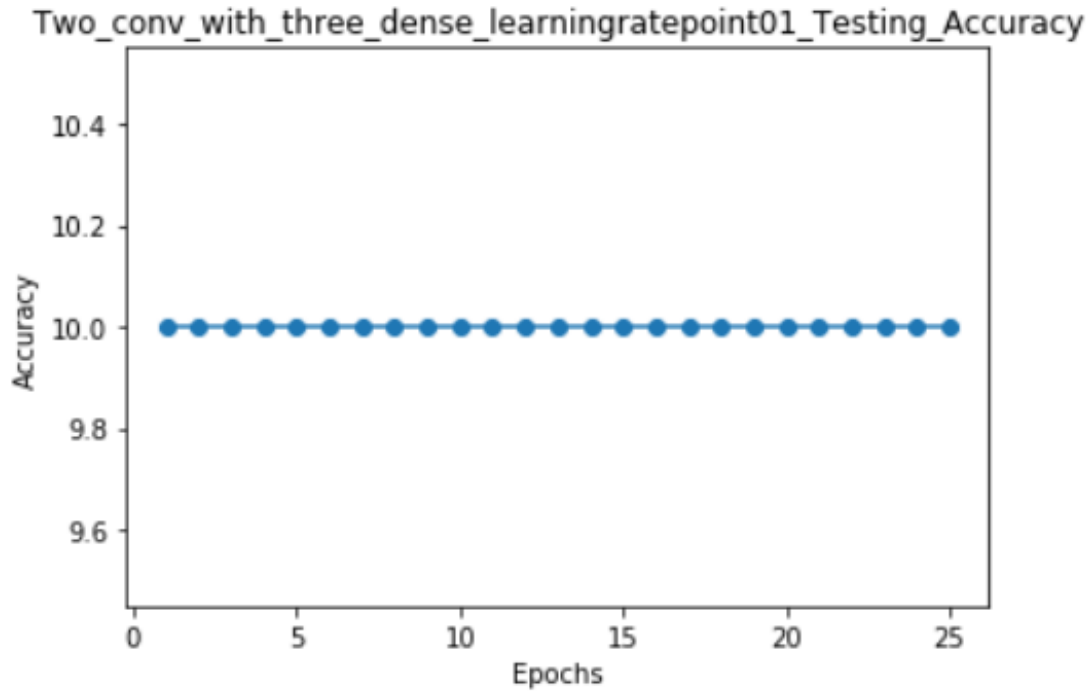


Figure 29: Two conv with three dense layers with a learning rate of 0.01 - testing accuracy

Parameters of the network with learning rate 0.0001: 62006
The training accuracy in case of learning rate 0.0001 increases from 10.0 to almost 60 above within the epoch range of 0 to 50.
The loss function in case of learning rate 0.0001 decreases from 2.25 to 0.50 range with increase in epoch from 0 to 50.

The training accuracy in case of learning rate 0.0001 increases from a range of 10 to 60 with increase in epoch from 0 to 50.

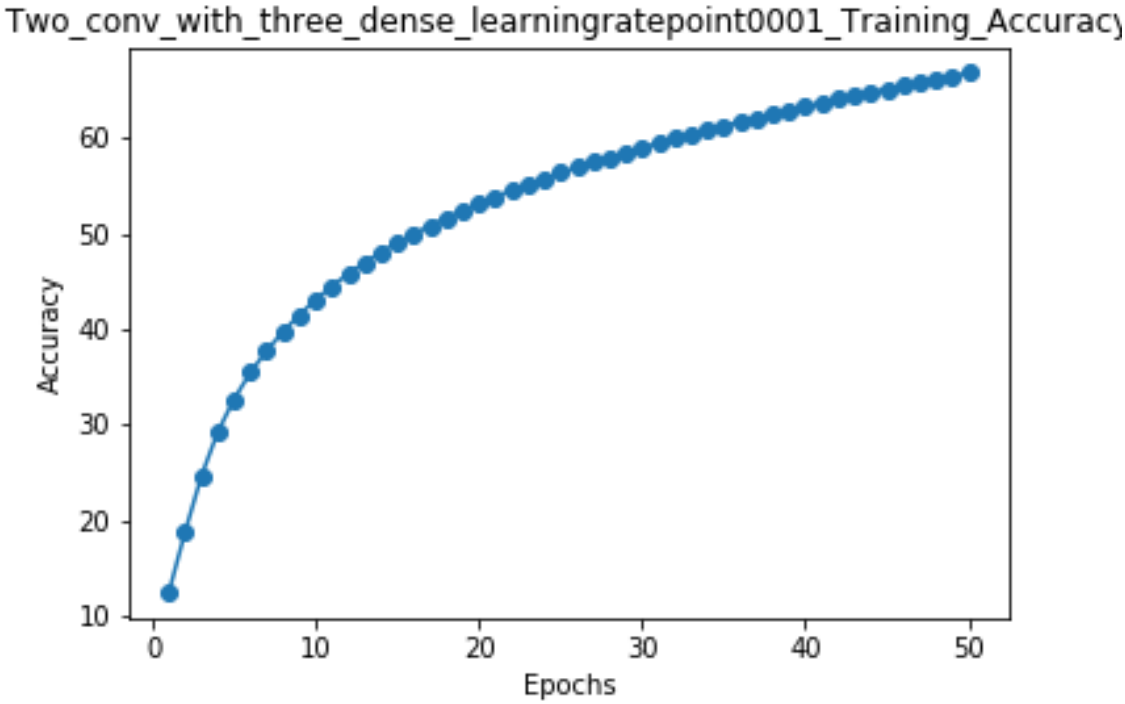


Figure 30: Two conv with three dense layers with a learning rate of 0.0001 - training accuracy

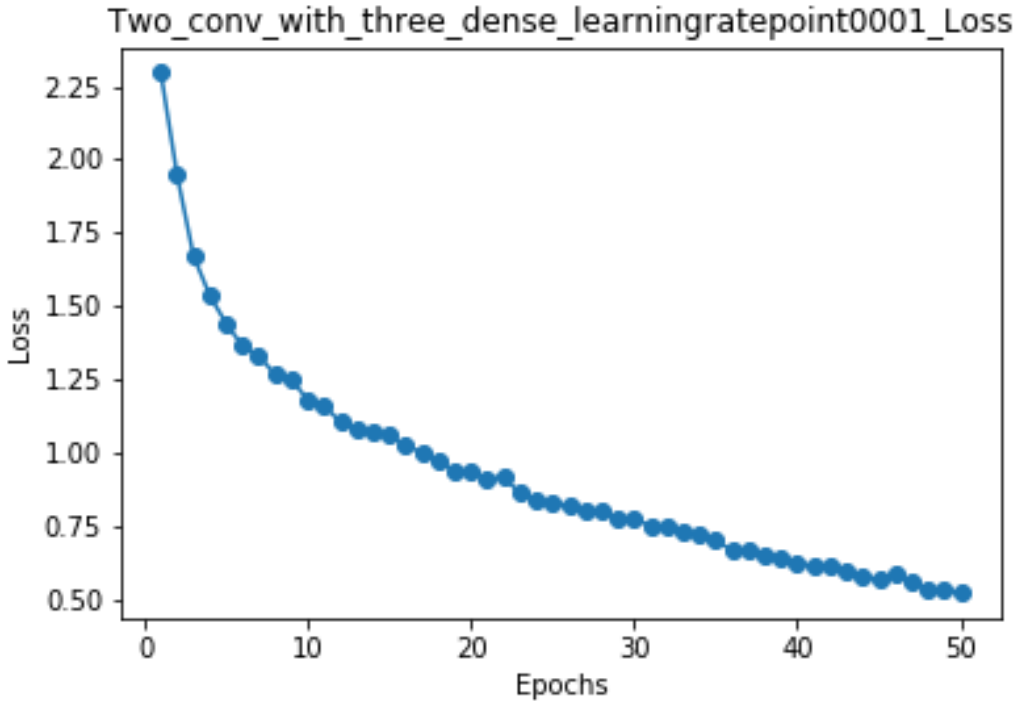


Figure 31: Two conv with three dense layers with a learning rate of 0.0001 - loss

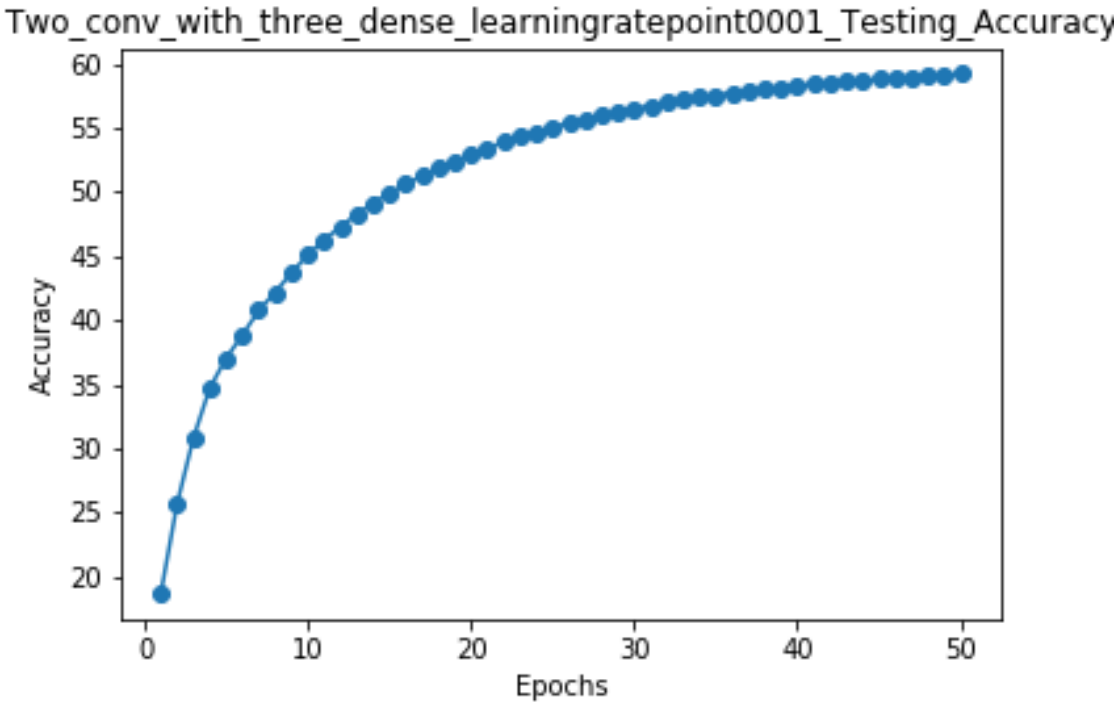


Figure 32: Two conv with three dense layers with a learning rate of 0.0001 - Test accuracy

7. Please add some data augmentation to avoid overfitting. Note that you need to do this only for the training and not the testing. You may use line 191 from Imagenet sample code:

   https://github.com/pytorch/examples/blob/master/imagenet/main.py

   "RandomResizedCrop" samples a random patch from the image to train the model on. "RandomHorizontalFlip" flips randomly chosen images horizontally.

   Parameters of the network: 62006

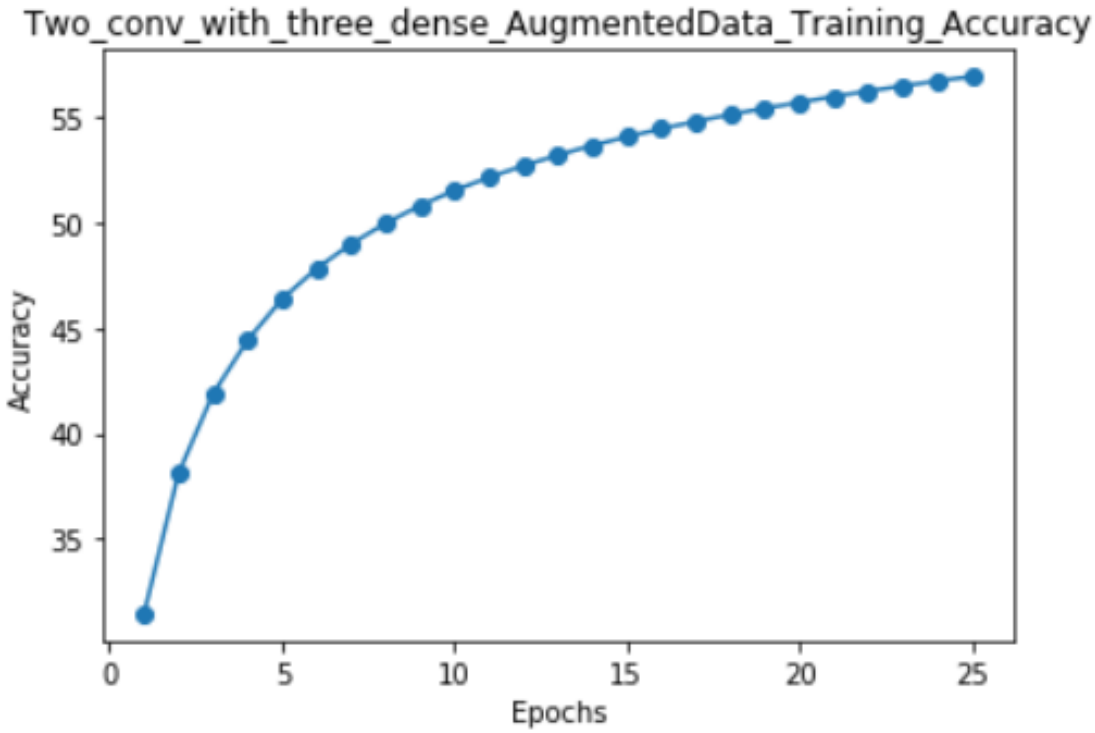


Figure 33: Two conv with three dense layers with Augmented Data - training accuracy

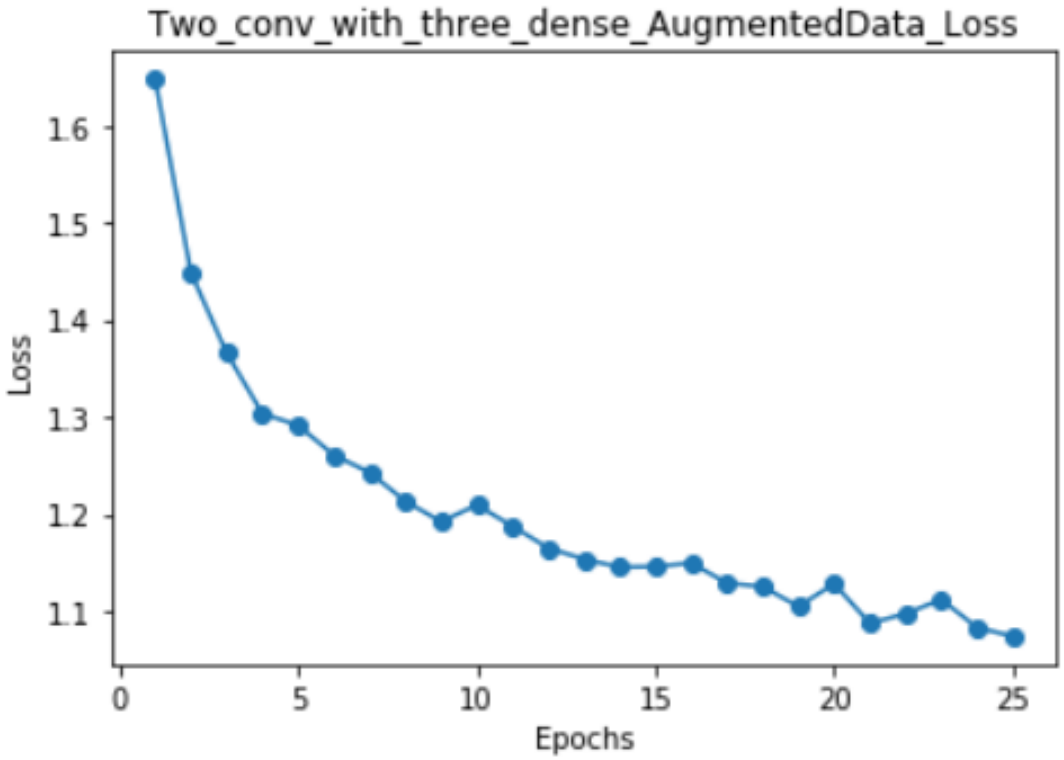


Figure 34: Two conv with three dense layers with Augmented Data - loss

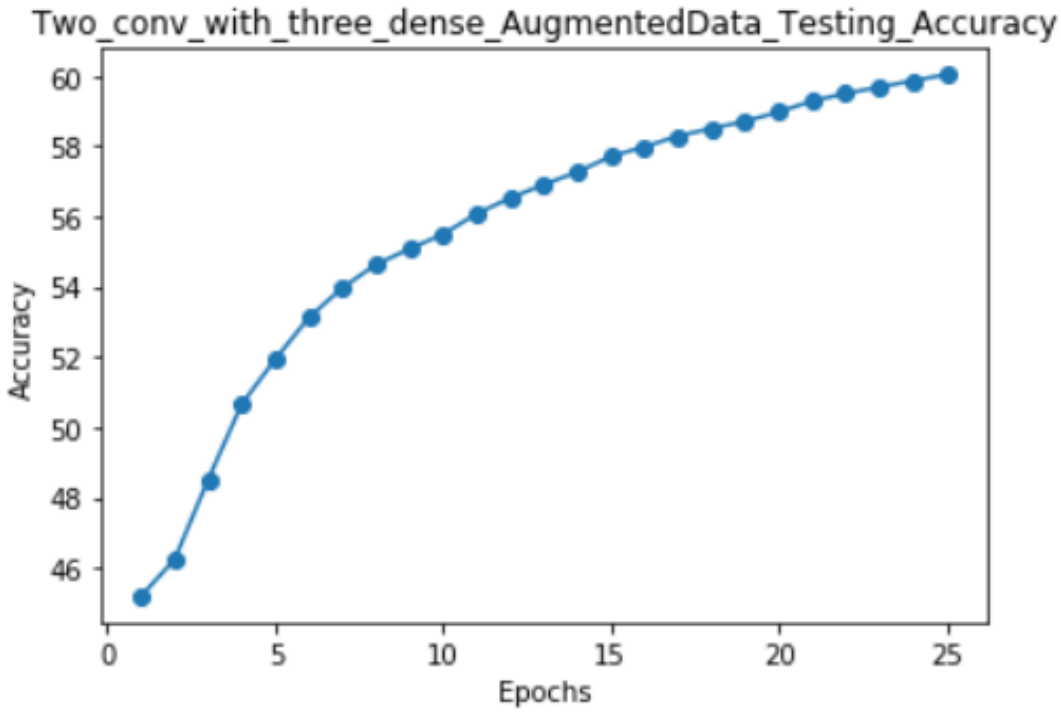
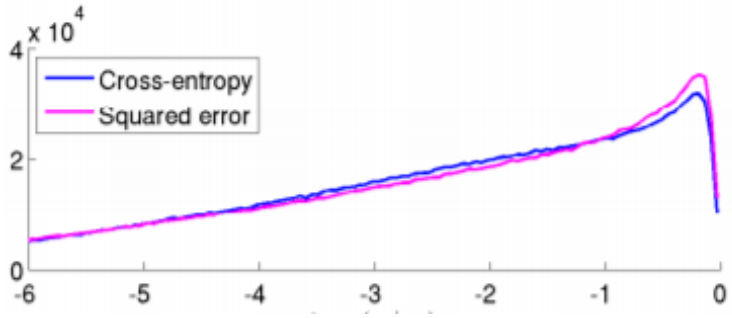


Figure 35: Two conv with three dense layers with Augmented Data - testing accuracy

8. Change the loss function from Cross Entropy to Mean Squared Error and report the effect.

   Cross-entropy is preferred for classification, while mean squared error is one of the best choices for regression. This comes directly from the statement of the problems itself - in classification we work with very particular set of possible output values thus, MSE is badly defined as it does not have this kind of knowledge thus penalizes errors in incompatible way.



   When we derive the cost function from the aspect of probability and distribution, it can observed that MSE is compatible when we assume the error follows Normal Distribution and cross entropy when we assume binomial distribution. It means that implicitly when we use MSE, we are doing regression (estimation) and when you use CE, we are doing classification. The following are the observed results by Changing the loss function from Cross Entropy to Mean Squared Error:
   As we observe the difference between the outputs of the graphs using Mean Squared error and Cross Entropy, the training and the test accuracy results are almost similar in both Mean Squared error and Cross Entropy whereas, the data loss in case of MSE is less when compared to Cross Entropy.
   Parameters of the network: 62006

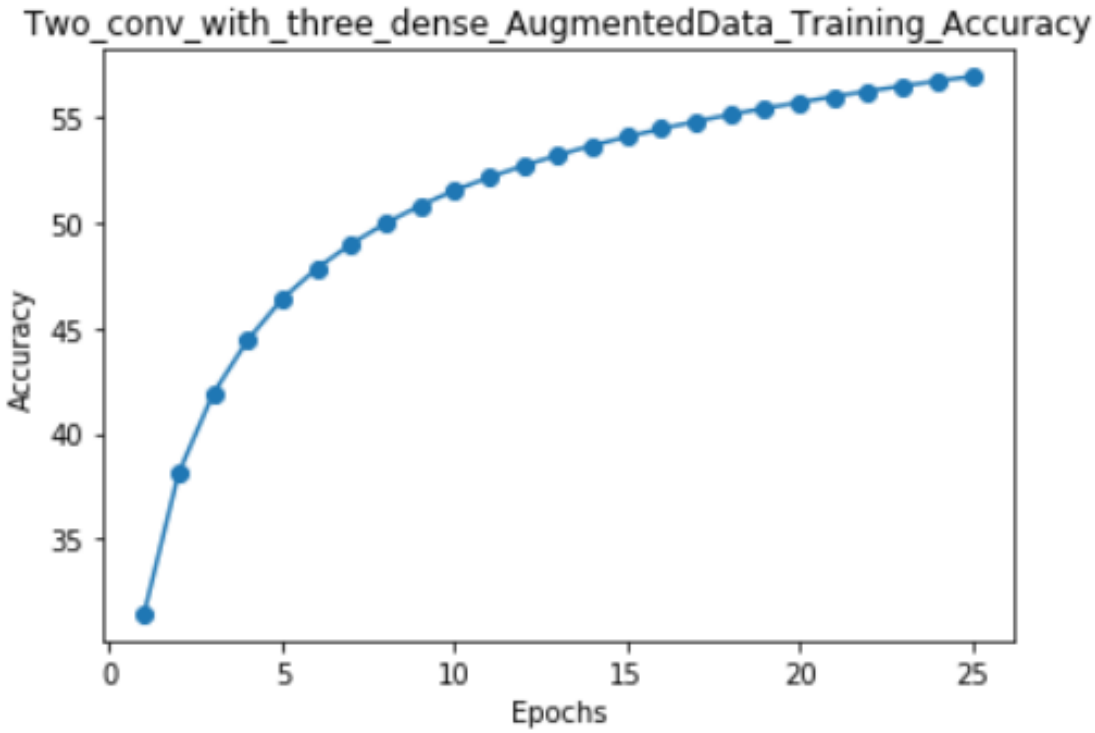


Figure 36: Two conv with three dense layers with MSELoss - training accuracy

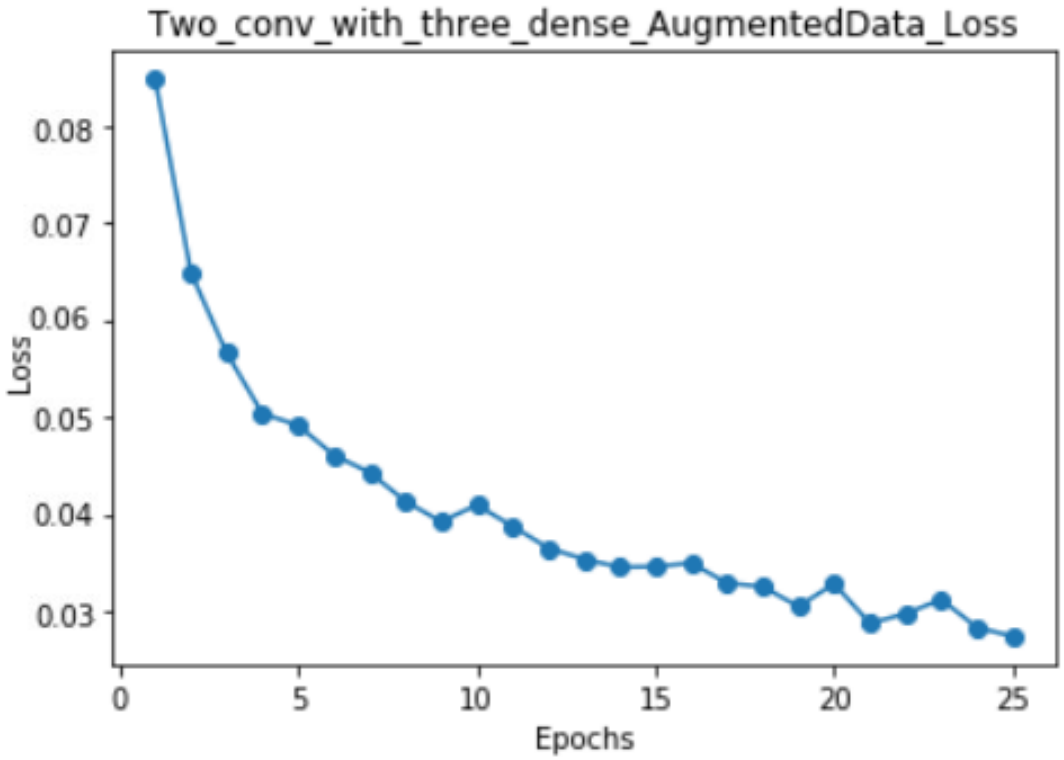


Figure 37: Two conv with three dense layers with MSELoss - loss

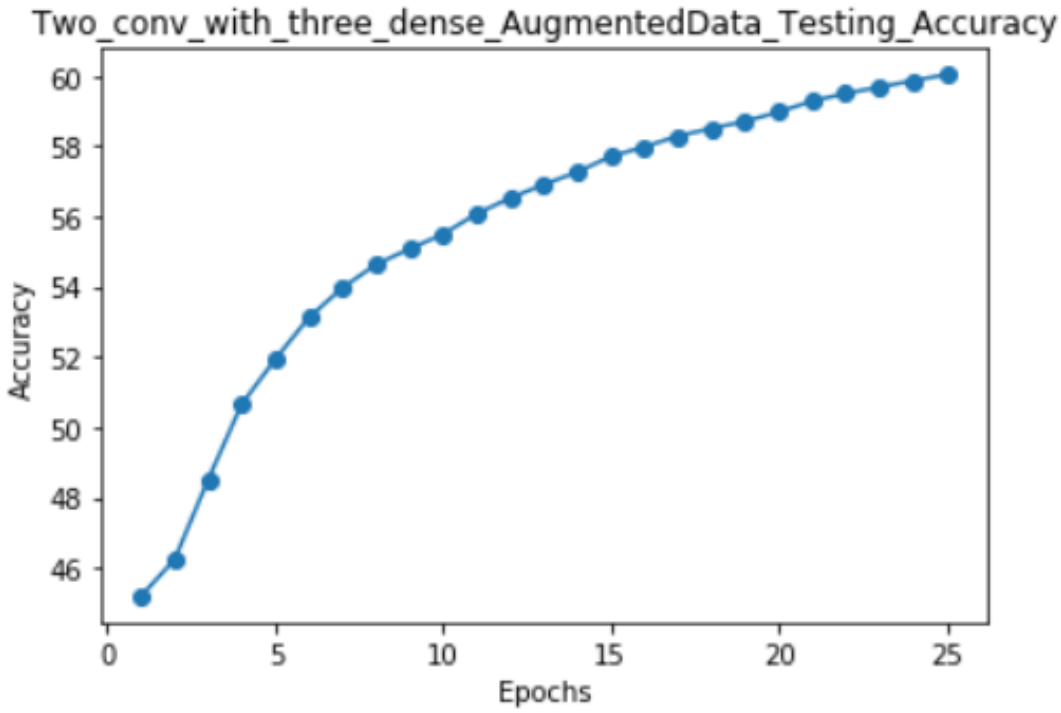


Figure 38: Two conv with three dense layers with MSELoss - testing accuracy