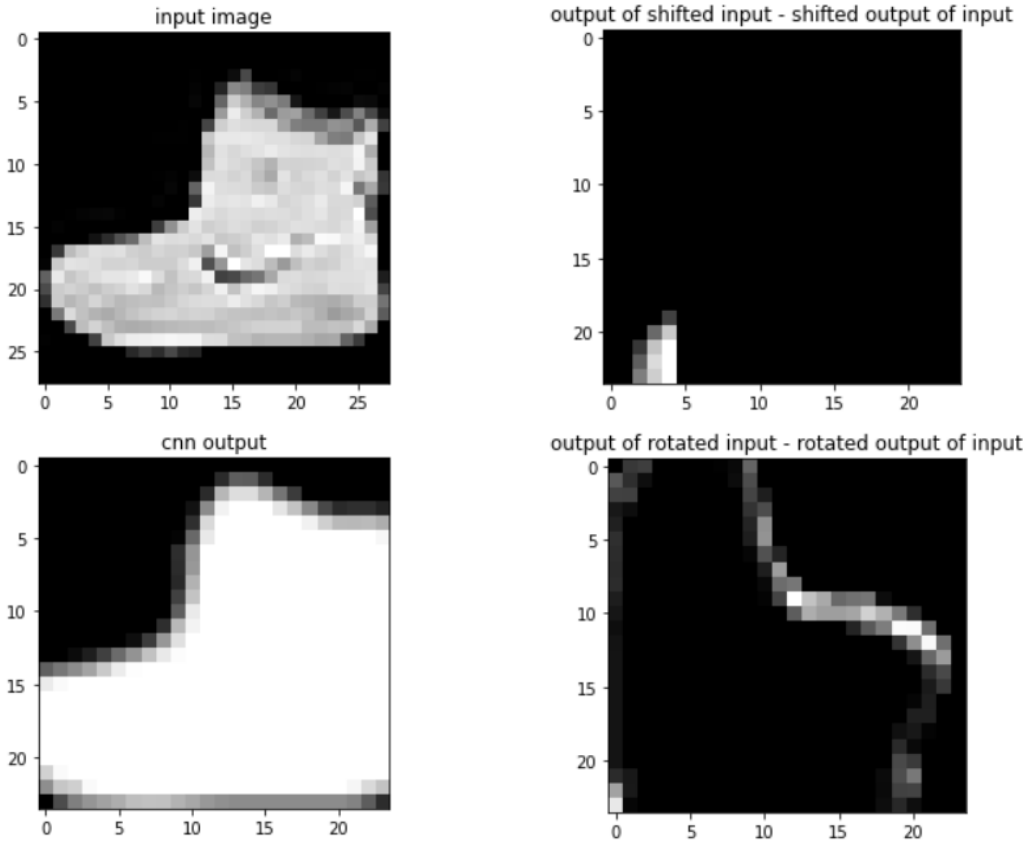
IFT 6390, Homework 3 - Practical Report

Abhay Puri(20209505), Saurabh Bodhe(20208545)

2) Experimenting on FashionMNIST Dataset

1)

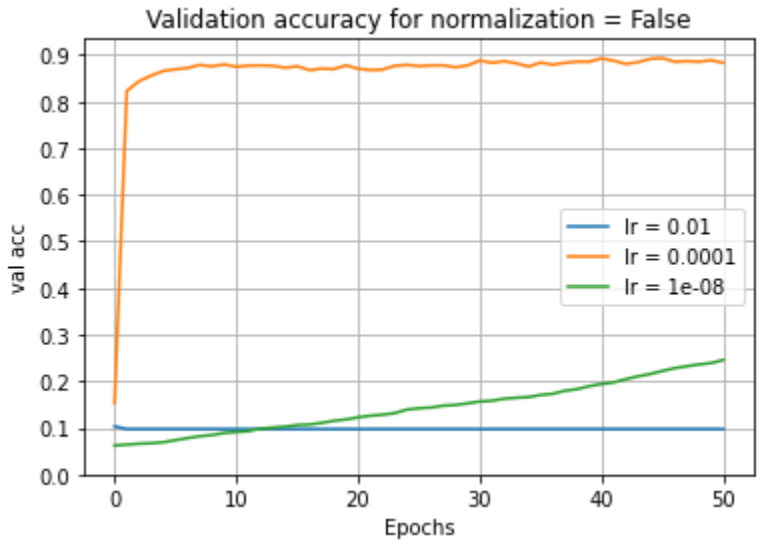


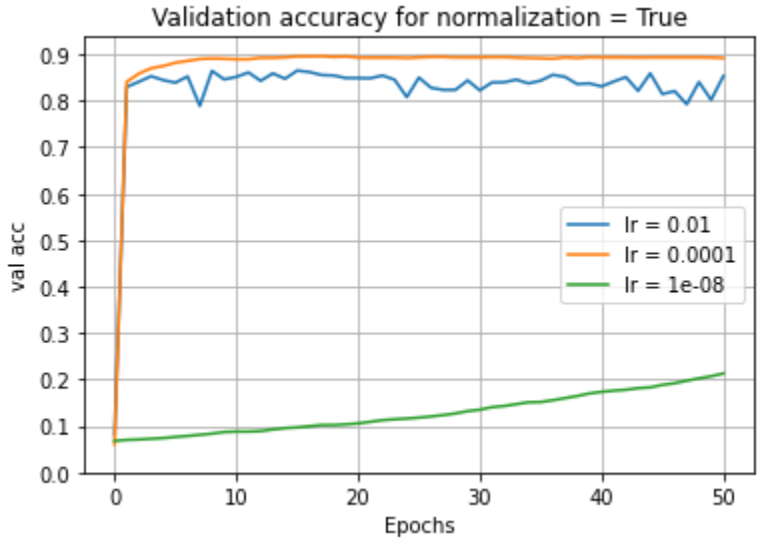
1.a) We can observe that convolutional layers are equivariant to translation but not to rotation. Ideally the difference between output of shifted input and shifted output should be all zeros but due the “cnn output” being clipped off on the bottom left, this leaves a small blob in the difference.

1.b) Some variations like translation are more important than others like rotation because the occurrence of translational variances in data is more frequent and significant than rotational variances which makes it necessary for the networks to work well in presence of these translational invariances.

It depends on both algorithms and data distribution. If the training dataset contains sufficient images containing these variances, it helps the algorithm work better on similar variations in unseen examples.

2)





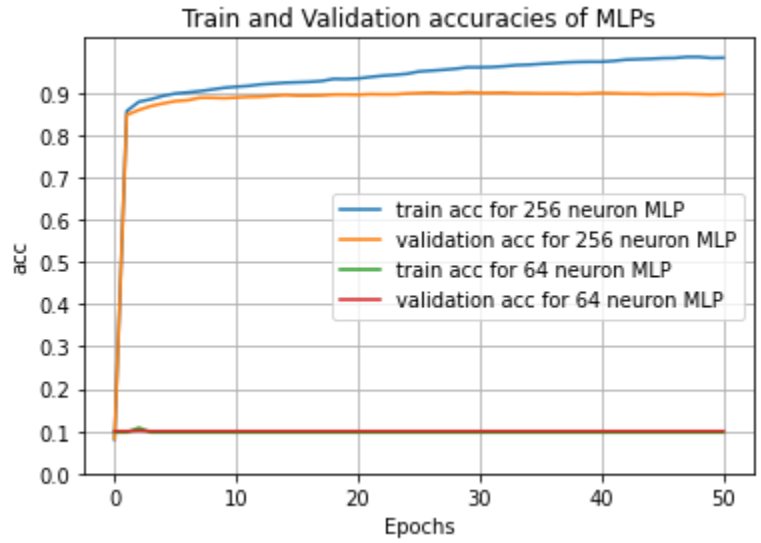
2.a) Normalization appears to improve generalization performance as seen in the above plots. It also helps high learning rate (0.01) train to some reasonable extent, while high learning rate without normalization hardly trains at all. It also appears to reduce noise and smoothen the learning as seen in moderate learning rate plot (0.0001)

2.b) Yes, it helps in convergence. Normalization scales the image pixel values to a certain narrow range. In the process of training, the network adjusts the weight by adding gradient errors multiplied by the learning rate. Since the learning rate is constant, If the image is not scaled properly, the network will overcompensate for large values or undercompensate for small values due to the widely varying feature ranges which won't let the network converge.

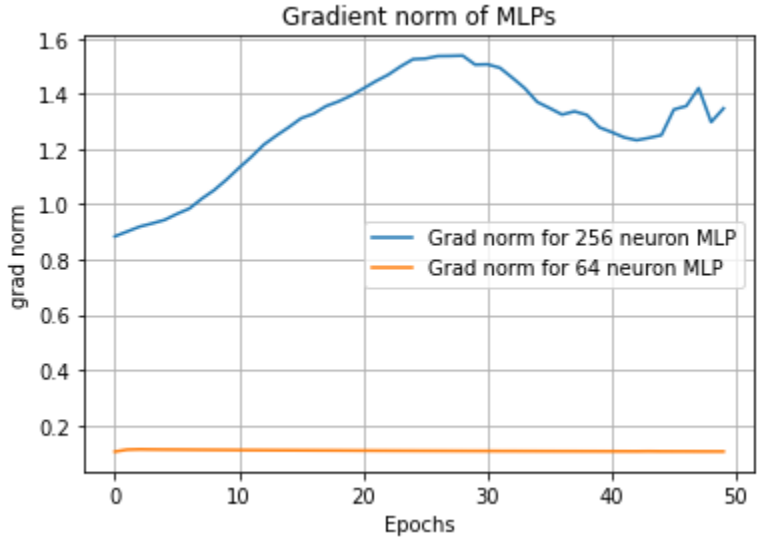
2.c) With normalization, too large learning rates make the network unable to converge as it keeps jumping over the optimum. If the learning rate is too small, the network will take small steps toward the optimum and take a very long time to converge.
Without normalization, too large learning rate seems to cause gradient overflow which creates NaN gradients, finally resulting in no increase in accuracy with epochs since it is not being trained.

3.a) The previous MLP with default params has 269322 parameters.

3.b.i) No. of hidden layers needed for 64 neuron MLP = 54



3.b.ii)

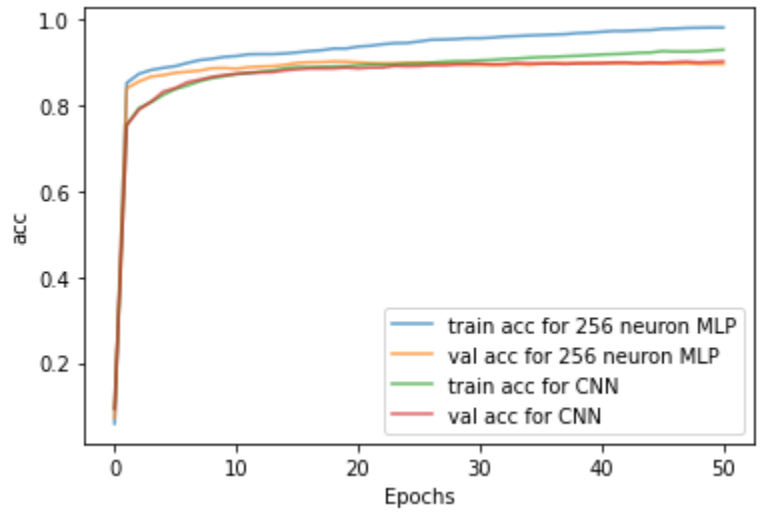


3.b.iii) Yes, we can observe that the gradient norm in the case of 64 neuron architecture is constant and unchanging, this shows that the weights are not being optimized and the network is not training, which finally results in extremely poor performance as seen in the plot. For 256 neuron MLP we see that the gradient is being updated every epoch and hence the accuracy keeps increasing.

4) kernel_sizes = (3, 3, 3) results in approximately the same number of params as the original MLP.

4.a)

train acc for mlp: 0.9820
train acc for cnn: 0.9300
val acc for mlp: 0.8970
val acc for cnn: 0.9030

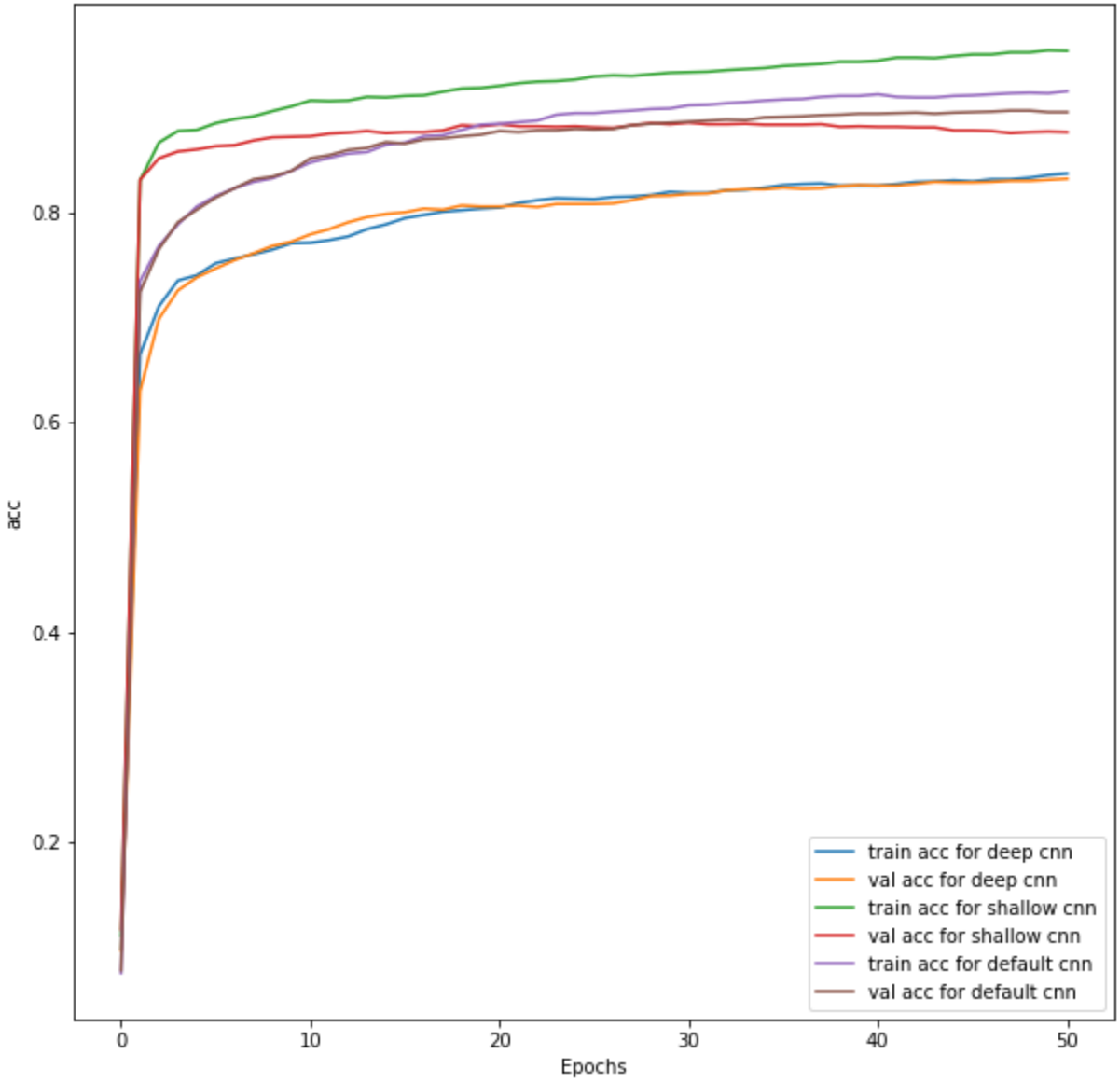


4.b) Looking at the final train accuracy it may seem like the MLP is performing better but the validation accuracy of CNN is actually slightly better than MLP. So effectively CNN performs better.
The very high train accuracy and relatively low validation accuracy of MLP is a sign of overfitting. So CNN generalizes better.

5.a) Since the kernel size is one, each neuron in a given feature map looks only at a single pixel of the image or previous layer. This is in extreme contrast to a fully connected layer where each neuron is connected to all the pixels.

5.b) Since the kernel size is the same as that of the image (28x28), each neuron in a given feature map is connected to the whole image i.e connected to all the pixels. This also means that each feature map only consists of one neuron. This is equivalent to what happens in a fully connected layer. Here the number of feature maps (n_channels) acts like the number of neurons in one FC layer.

5.c) train acc for deep cnn: 0.8375
train acc for shallow cnn: 0.9545
train acc for default cnn: 0.9160
val acc for deep cnn: 0.8325
val acc for shallow cnn: 0.8770
val acc for default cnn: 0.8960



Looking at the high validation accuracy of default CNN we can say that it has the best generalization performance.
Deep CNN has the lowest train and validation accuracies pointing towards underfitting.
Shallow CNN has high train accuracy but significantly lower validation accuracy which points to possible overfitting.

Hypothesis: It appears that for a fixed number of parameters, too many layers of CNN (deep CNN) causes underfitting possibly because every neuron in a given feature map is looking only

at one pixel of image or previous layer, this causes loss of contextual information given by the neighbouring pixels.

Too few layers of CNN (shallow CNN) causes overfitting because the single neuron in a given feature map is connected to all the pixels of images, causing it to fit too well to the training data.

A moderate number of layers (default CNN) causes the model to perform well due to the optimum number of neurons being present in each feature map and preserving local contextual information at the same time.