DEEP LEARNING ASSIGNMENT-1 REPORT

Name:	**ARIJIT SAMAL**
Roll No.:	19051
University Name:	IISER Bhopal
Stream:	EECS

1 QUESTION-1

1.1 OBJECTIVE

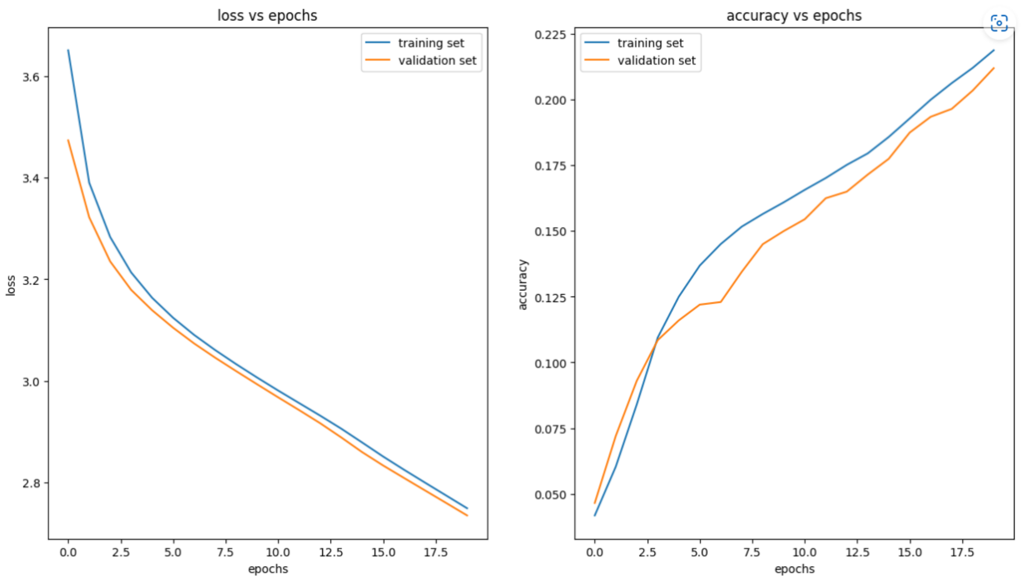
Implementing a two layer back propagation neural network from scratch on Letter Recognition dataset.

1.2 PREPARING THE DATA

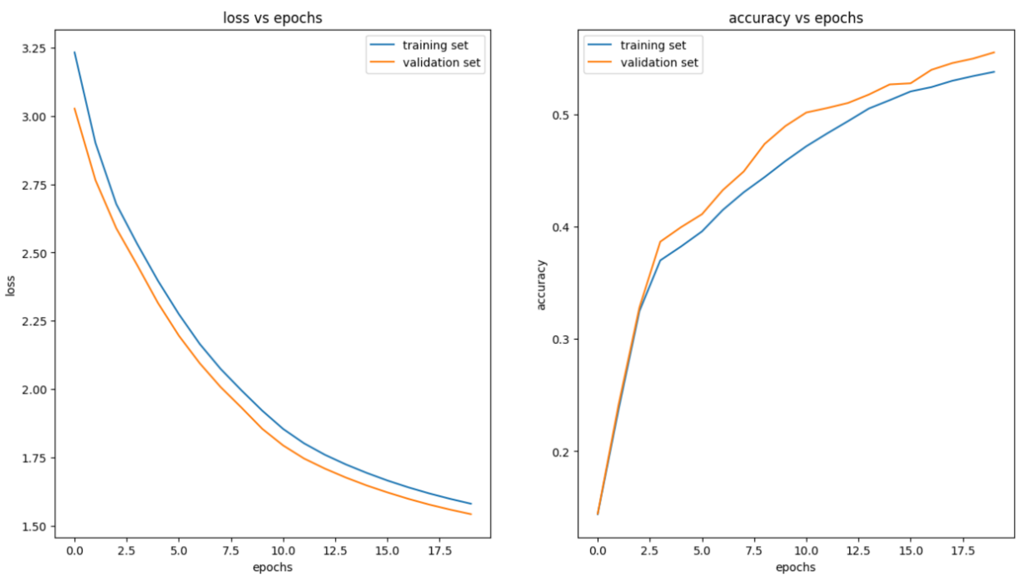
There are 17 columns and 20000 rows in the training dataset. we first label encode the target column and separate it out from the original data.Out of this we use train test split to make them into training testing and validation sets we have train set shape- (16000, 16), validation set shape- (2000,16) and test set size- (2000,16). we train on the train set. we use validation set for finding best parameters and we find accuracy on the test set.

1.3 RESULTS

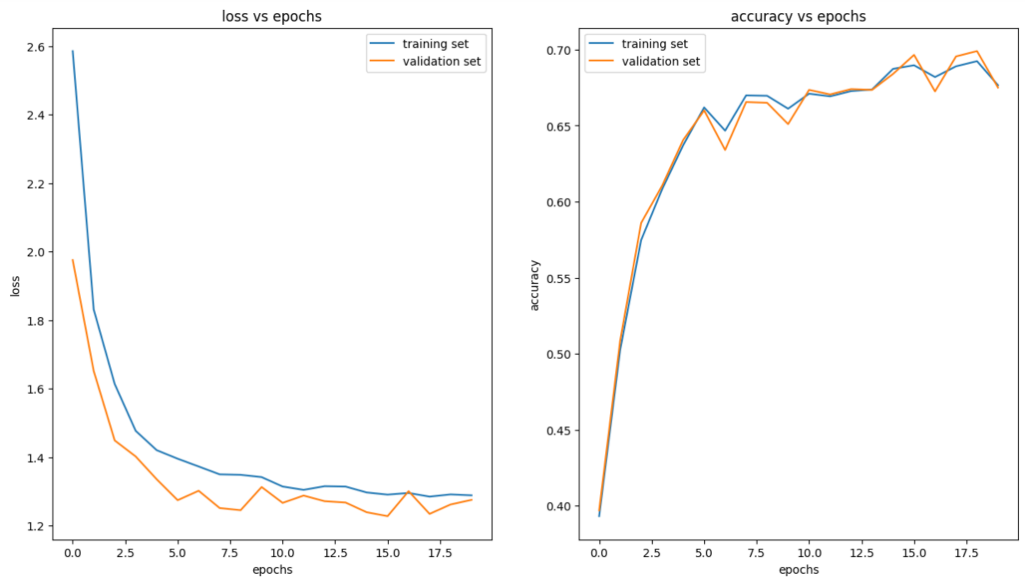
A. varying the learning rate
1. for tanh activation-



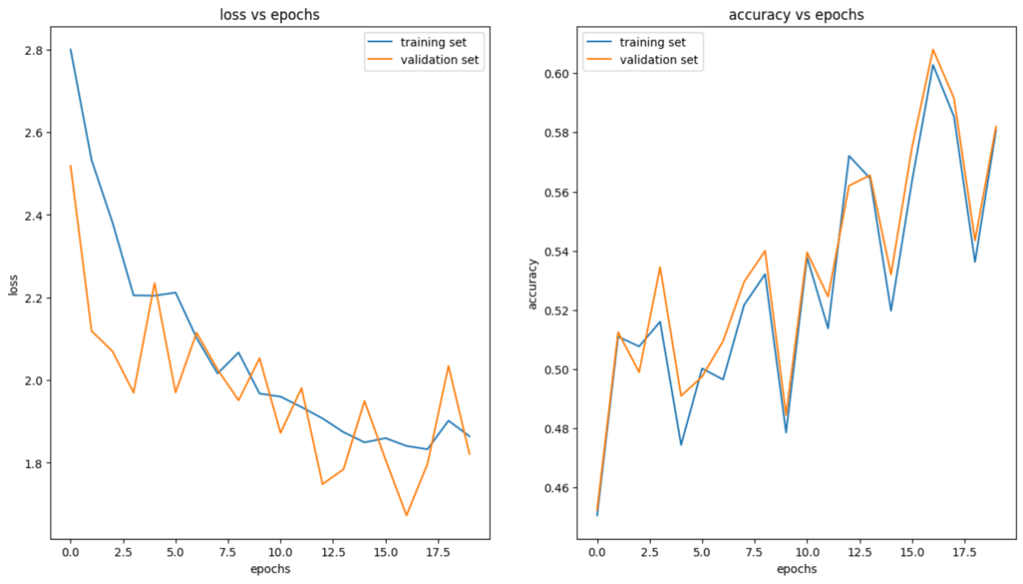
1.1. for learning rate 1e-5



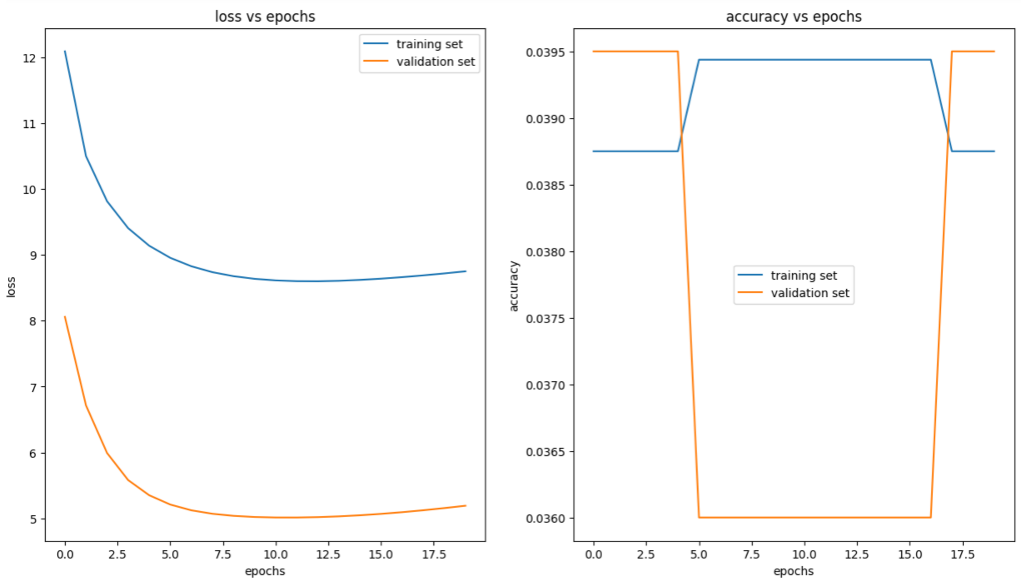
1.2. for learning rate 1e-4



1.3. for learning rate 1e-3

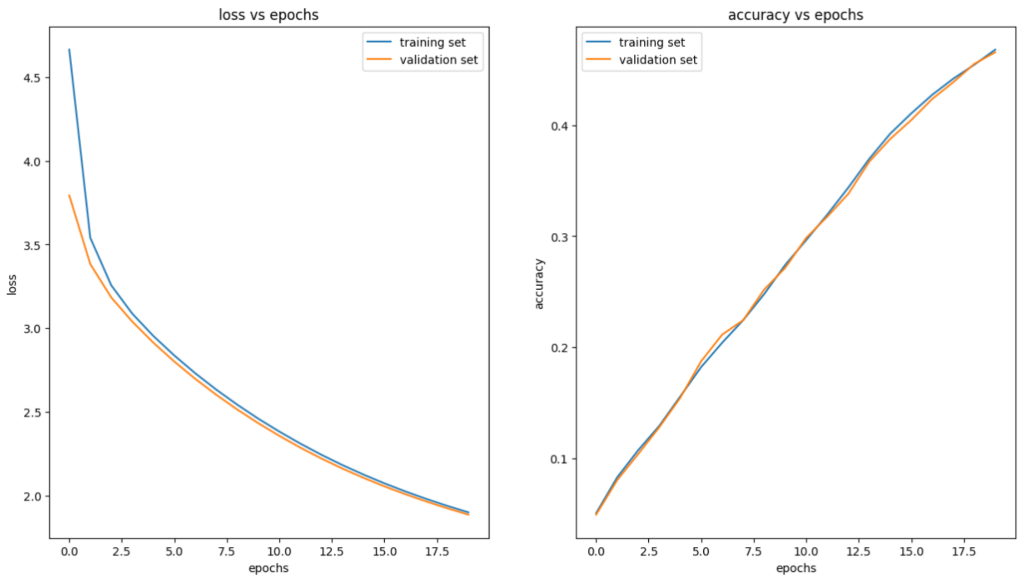


1.4. for learning rate 0.01



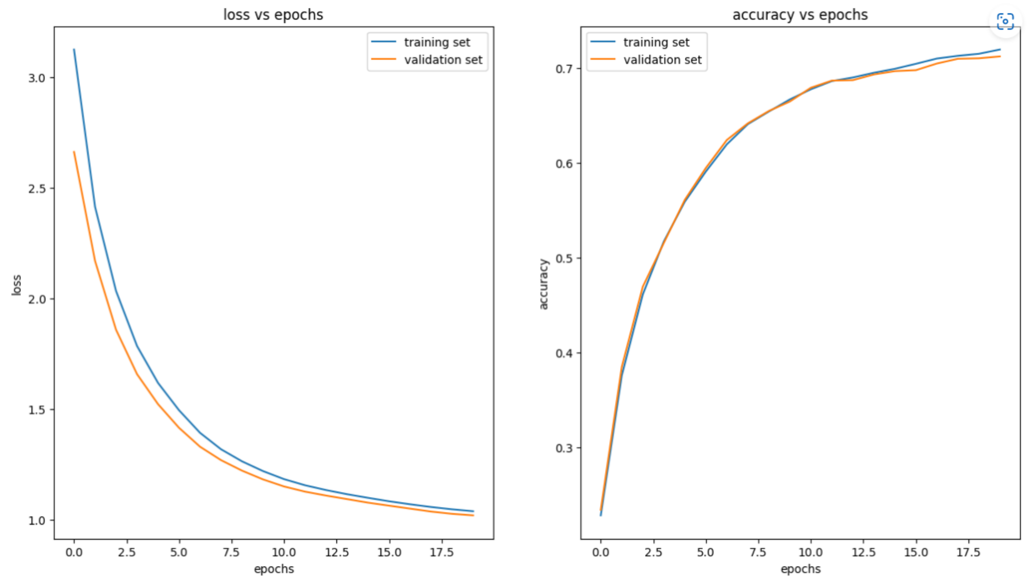
1.5. for learning rate 0.1

2. for ReLu activation-

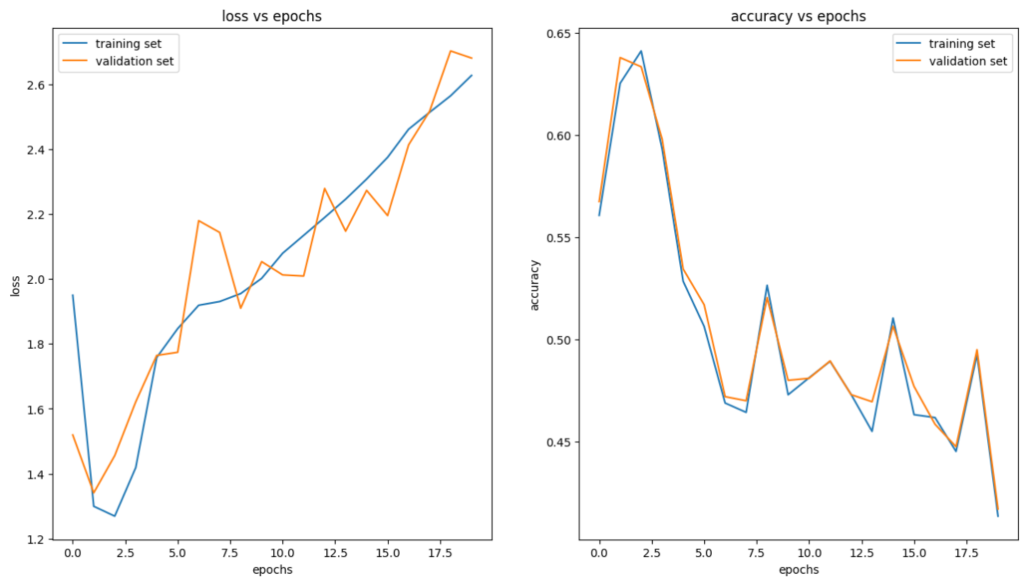


relu

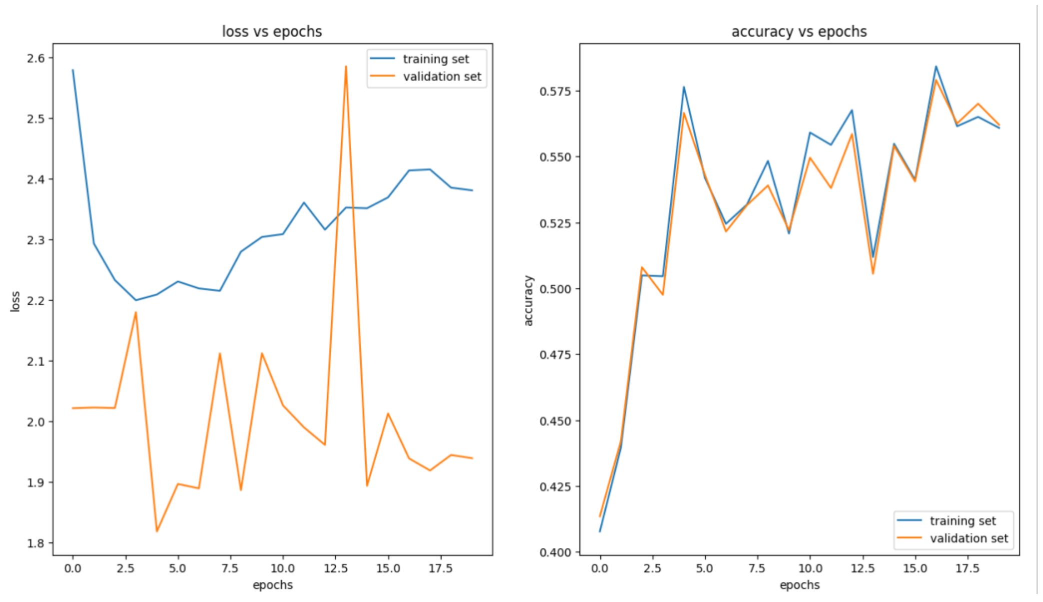
2.1. for learning rate 1e-5



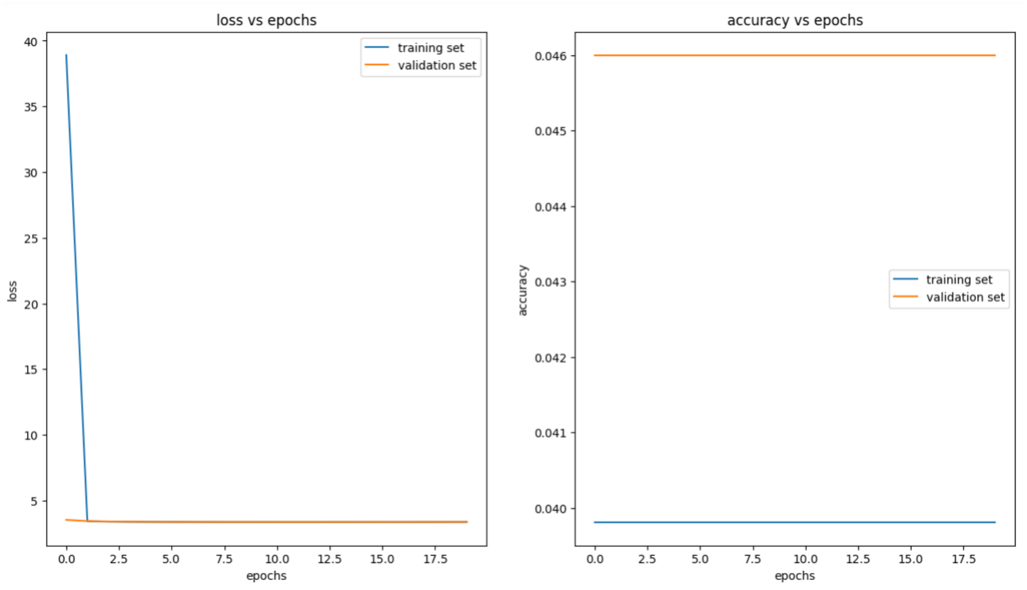
2.2. for learning rate 1e-4



2.3. for learning rate 1e-3



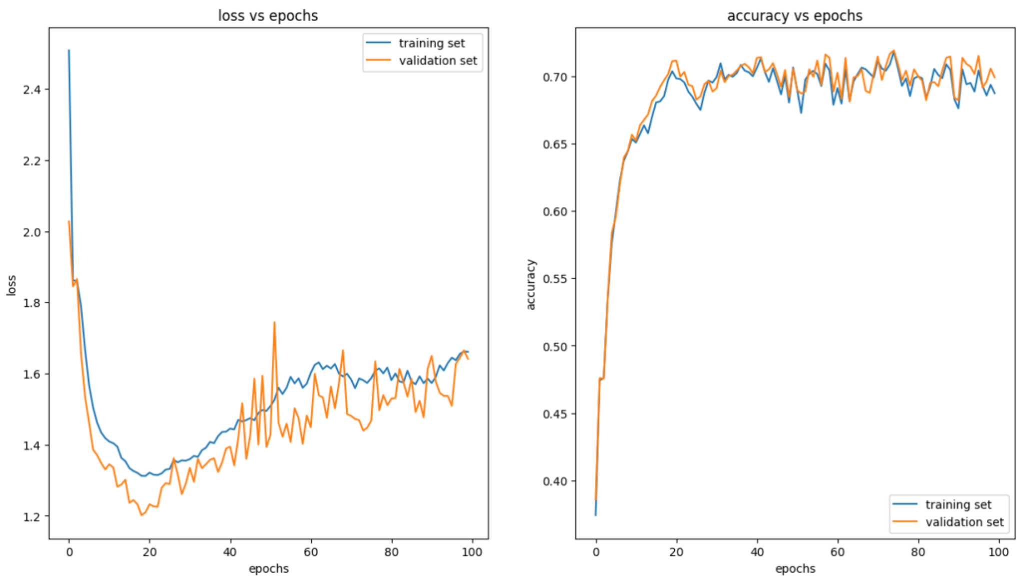
2.4. for learning rate 0.01



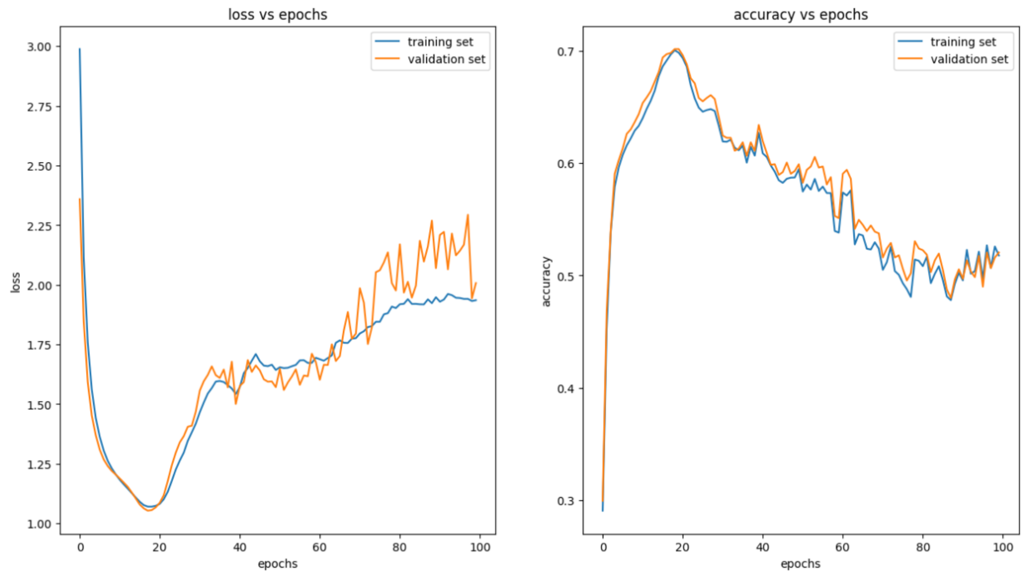
2.5. for learning rate 0.1

B. Varying the number of epochs

1. for tanh activation with 100 epochs-

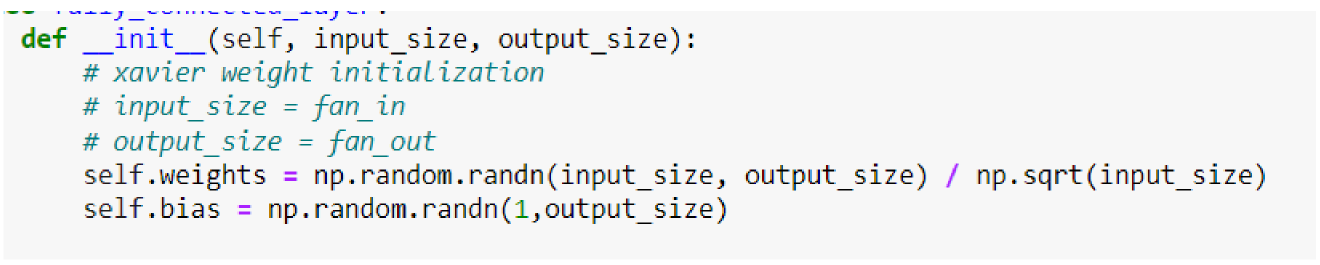


2. for ReLu activation with 100 epochs-



C. Xavier weight initialization

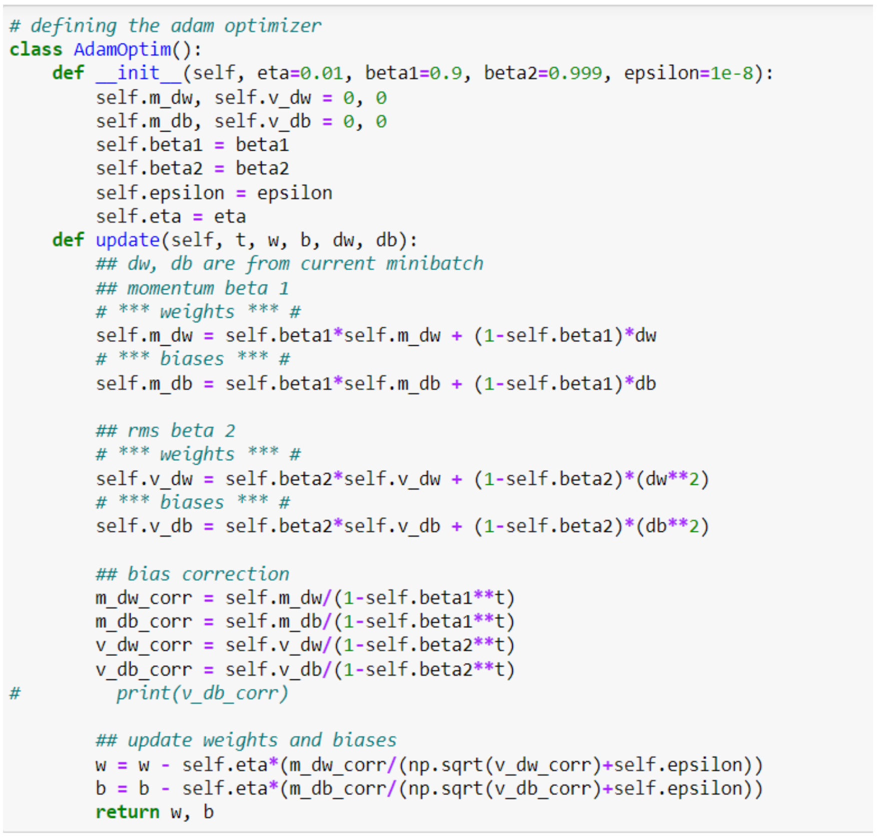
it is done using the formula for xavier weight initialization in cell 25



```python
    def __init__(self, input_size, output_size):
        # xavier weight initialization
        # input_size = fan_in
        # output_size = fan_out
        self.weights = np.random.randn(input_size, output_size) / np.sqrt(input_size)
        self.bias = np.random.randn(1,output_size)
```

D. using the adam optimizer

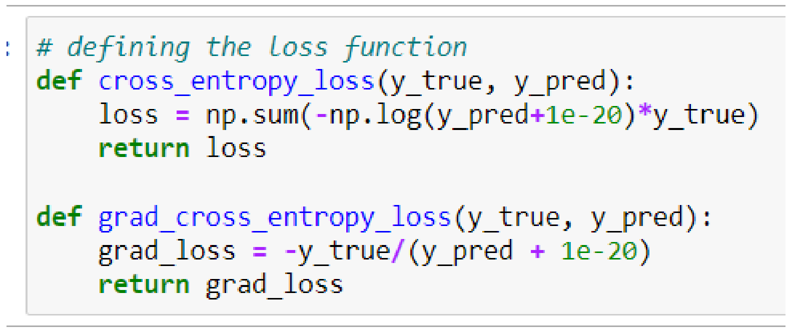
it is done in cell 30



```python
# defining the adam optimizer
class AdamOptim():
    def __init__(self, eta=0.01, beta1=0.9, beta2=0.999, epsilon=1e-8):
        self.m_dw, self.v_dw = 0, 0
        self.m_db, self.v_db = 0, 0
        self.beta1 = beta1
        self.beta2 = beta2
        self.epsilon = epsilon
        self.eta = eta
    def update(self, t, w, b, dw, db):
        ## dw, db are from current minibatch
        ## momentum beta 1
        # *** weights *** #
        self.m_dw = self.beta1*self.m_dw + (1-self.beta1)*dw
        # *** biases *** #
        self.m_db = self.beta1*self.m_db + (1-self.beta1)*db

        ## rms beta 2
        # *** weights *** #
        self.v_dw = self.beta2*self.v_dw + (1-self.beta2)*(dw**2)
        # *** biases *** #
        self.v_db = self.beta2*self.v_db + (1-self.beta2)*(db**2)

        ## bias correction
        m_dw_corr = self.m_dw/(1-self.beta1**t)
        m_db_corr = self.m_db/(1-self.beta1**t)
        v_dw_corr = self.v_dw/(1-self.beta2**t)
        v_db_corr = self.v_db/(1-self.beta2**t)
#          print(v_db_corr)

        ## update weights and biases
        w = w - self.eta*(m_dw_corr/(np.sqrt(v_dw_corr)+self.epsilon))
        b = b - self.eta*(m_db_corr/(np.sqrt(v_db_corr)+self.epsilon))
        return w, b
```

E. using the activation functions for tanh and relu

we have created the things and have already attached the results above for loss vs epochs and accuracy vs epochs of both the training and the validation sets the accuracy on the test set is mentioned in the ipynb notebook during the testing of the model.
We use the cross entropy loss here which can be found in cell 29



```python
# defining the loss function
def cross_entropy_loss(y_true, y_pred):
    loss = np.sum(-np.log(y_pred+1e-20)*y_true)
    return loss

def grad_cross_entropy_loss(y_true, y_pred):
    grad_loss = -y_true/(y_pred + 1e-20)
    return grad_loss
```

1.4 CONCLUSION

The conclusions from the above results are as follows-
from the above analysis we get that the best parameters for tanh are as follows-
1. learning rate- 0.001 or 1e-3
2. number of epochs- 100 epochs can be taken as test accuracy, loss and generalization gap does not change significantly.

we get accuracy on test set- 68.3%

from the above analysis we get that the best parameters for relu are as follows-
1. learning rate- 0.0001 or 1e-4
2. number of epochs- we must take 20 epochs because after that the loss increases and the accuracy decreases rapidly.

we get accuracy on test set- 63.5%

Finally, our neural network has tanh activation with learning rate 0.001 or 1e-3, number of epochs 100 and produce an accuracy of 68.3% on the test dataset

2 QUESTION-2

2.1 OBJECTIVE

we construct the following CNN architectures-
A. Conv-Pool-Conv-Pool-Conv-Pool-FC
B. Conv-Conv-Pool-Conv-Conv-Pool-FC
C. Conv-Pool-Conv-Pool-Conv-Pool-FC-FC
Then use the CIFAR-10 dataset and analyse the effect of max pooling, network size and presence of one or more fully connected layer on the results

2.2 PREPARING THE DATA

the data is already present in the pytorch library we just need to load the data by downloading it and storing the train and test sets in different tensors.

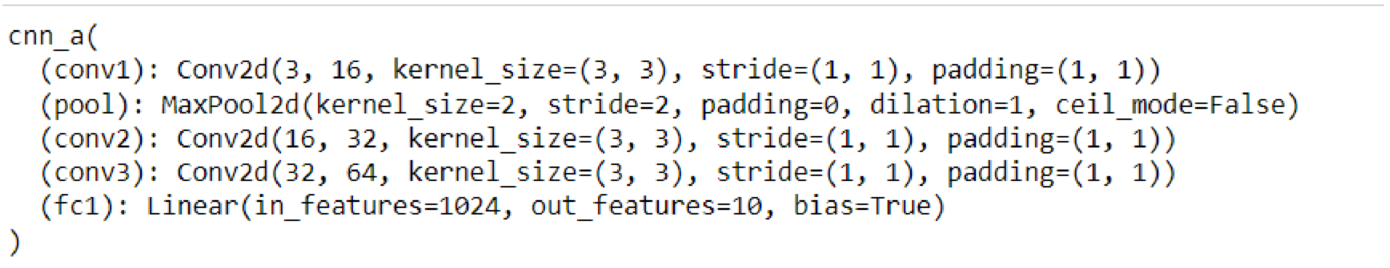
2.3 RESULTS

A. changing the network sizes

using the three networks mentioned on the objective we analyse the effect of network size on the accuracy of results on the test data

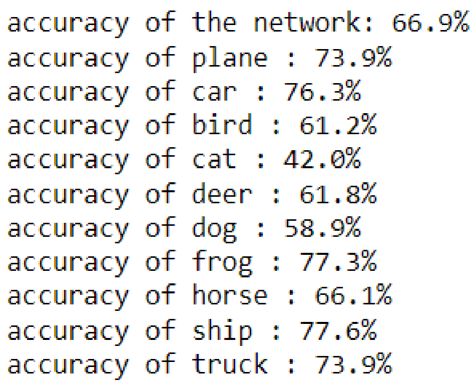
1. CNN model A

architecture of A-
Conv-Pool-Conv-Pool-Conv-Pool-FC



```
cnn_a(
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (fc1): Linear(in_features=1024, out_features=10, bias=True)
)
```

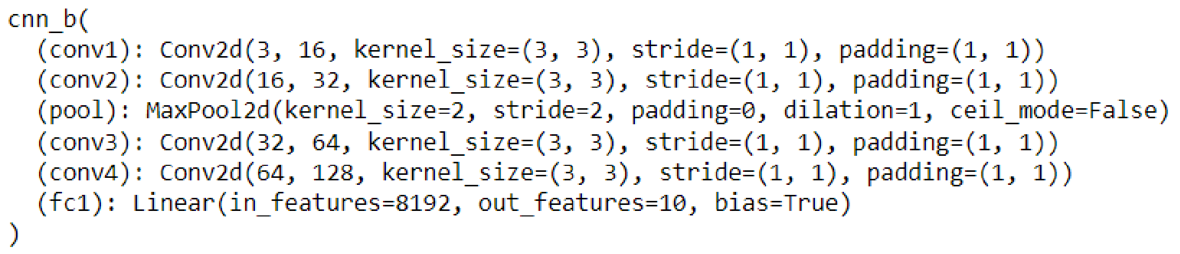
accuracy on test set-



```
accuracy of the network: 66.9%
accuracy of plane : 73.9%
accuracy of car : 76.3%
accuracy of bird : 61.2%
accuracy of cat : 42.0%
accuracy of deer : 61.8%
accuracy of dog : 58.9%
accuracy of frog : 77.3%
accuracy of horse : 66.1%
accuracy of ship : 77.6%
accuracy of truck : 73.9%
```

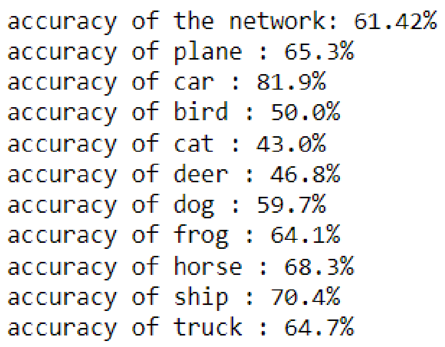
2. CNN model B

architecture of B-
Conv-Conv-Pool-Conv-Conv-Pool-FC



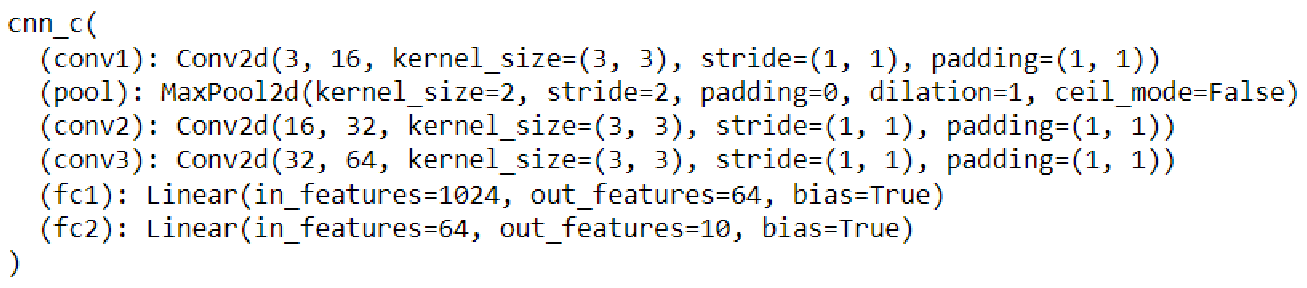
```
cnn_b(
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (fc1): Linear(in_features=8192, out_features=10, bias=True)
)
```

accuracy on test set-



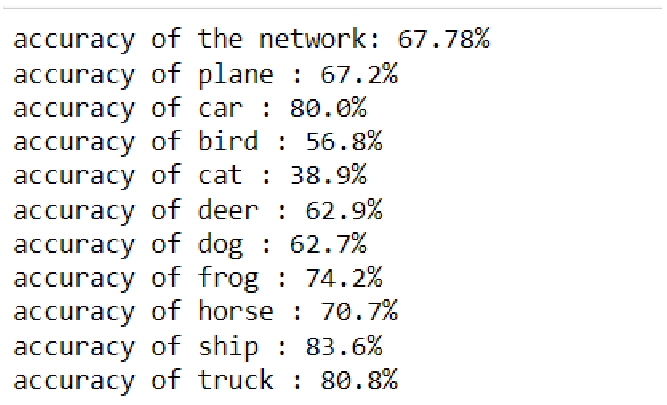
```
accuracy of the network: 61.42%
accuracy of plane : 65.3%
accuracy of car : 81.9%
accuracy of bird : 50.0%
accuracy of cat : 43.0%
accuracy of deer : 46.8%
accuracy of dog : 59.7%
accuracy of frog : 64.1%
accuracy of horse : 68.3%
accuracy of ship : 70.4%
accuracy of truck : 64.7%
```

3. CNN model C
architecture of C-
Conv-Pool-Conv-Pool-Conv-Pool-FC-FC



```
cnn_c(
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (fc1): Linear(in_features=1024, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=10, bias=True)
)
```

accuracy on test set-



```
accuracy of the network: 67.78%
accuracy of plane : 67.2%
accuracy of car : 80.0%
accuracy of bird : 56.8%
accuracy of cat : 38.9%
accuracy of deer : 62.9%
accuracy of dog : 62.7%
accuracy of frog : 74.2%
accuracy of horse : 70.7%
accuracy of ship : 83.6%
accuracy of truck : 80.8%
```

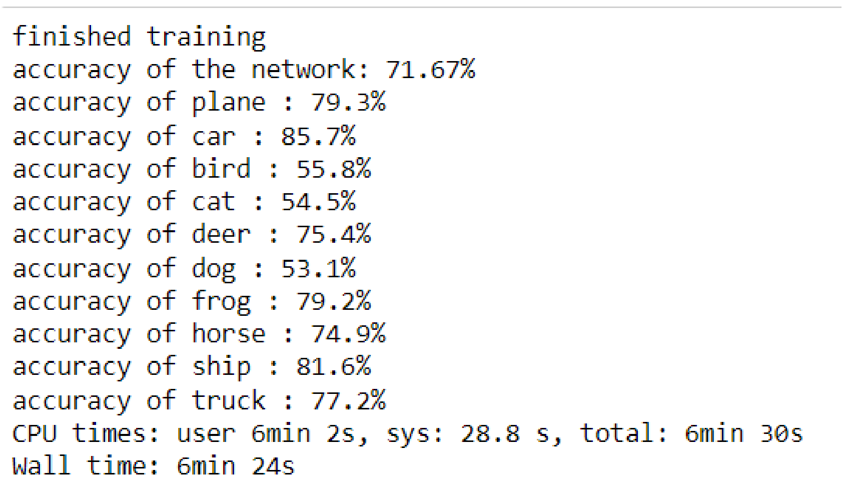
CNN model C has the highest accuracy on the test set followed by model A and B. Therefore, we conclude that network size and architecture affects the overall performance of the model.

B. using different sizes of pooling

we changed the kernel size in the maxpool layer for all the three models mentioned above. We observe that as the kernel size increases the size of image as input to the fully connected layer decreases. we obtain different results for accuracy on the network as shown below-

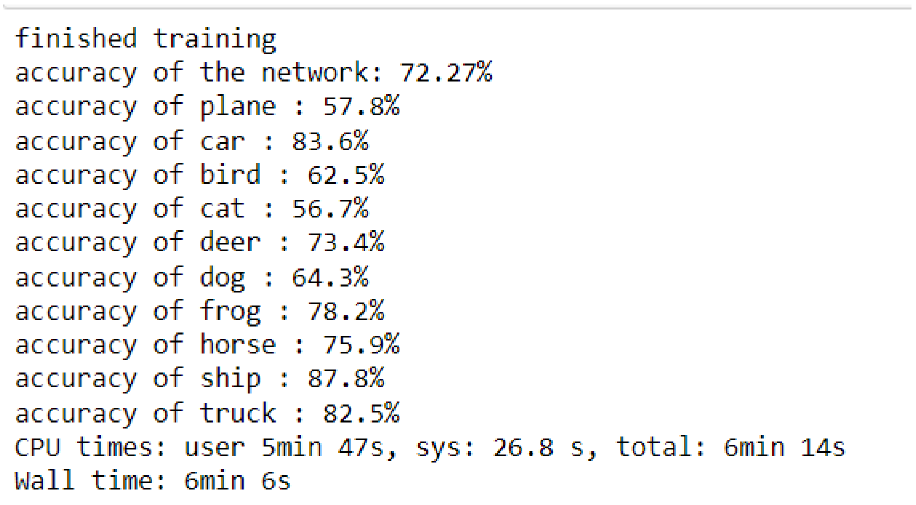
1.CNN MODEL A

1.1. for maxpool layer kernel size= 3



```
finished training
accuracy of the network: 71.67%
accuracy of plane : 79.3%
accuracy of car : 85.7%
accuracy of bird : 55.8%
accuracy of cat : 54.5%
accuracy of deer : 75.4%
accuracy of dog : 53.1%
accuracy of frog : 79.2%
accuracy of horse : 74.9%
accuracy of ship : 81.6%
accuracy of truck : 77.2%
CPU times: user 6min 2s, sys: 28.8 s, total: 6min 30s
Wall time: 6min 24s
```

1.2. for maxpool layer kernel size= 4

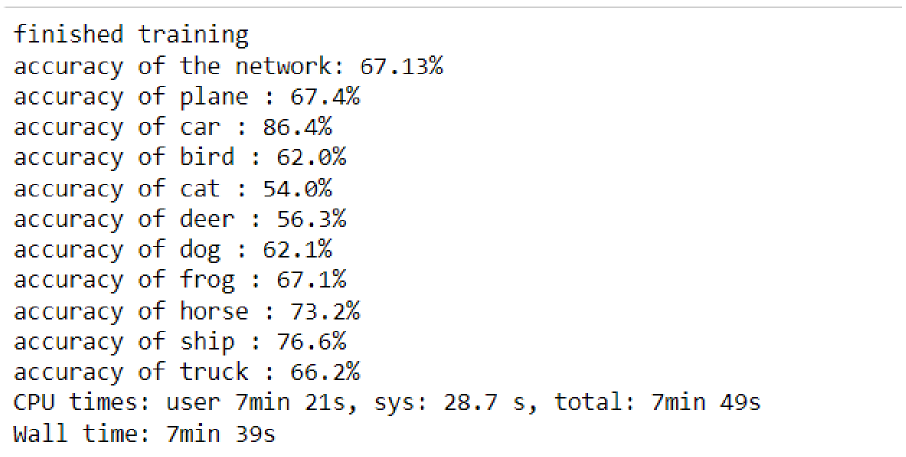


```
finished training
accuracy of the network: 72.27%
accuracy of plane : 57.8%
accuracy of car : 83.6%
accuracy of bird : 62.5%
accuracy of cat : 56.7%
accuracy of deer : 73.4%
accuracy of dog : 64.3%
accuracy of frog : 78.2%
accuracy of horse : 75.9%
accuracy of ship : 87.8%
accuracy of truck : 82.5%
CPU times: user 5min 47s, sys: 26.8 s, total: 6min 14s
Wall time: 6min 6s
```

we observe that the accuracy on test set increases with increase in kernel size of maxpool layer. continously increasing the size of maxpool layer finally reduces the image to a single pixel of 1x1 with multiple channels which can't be further used for training the cnn.

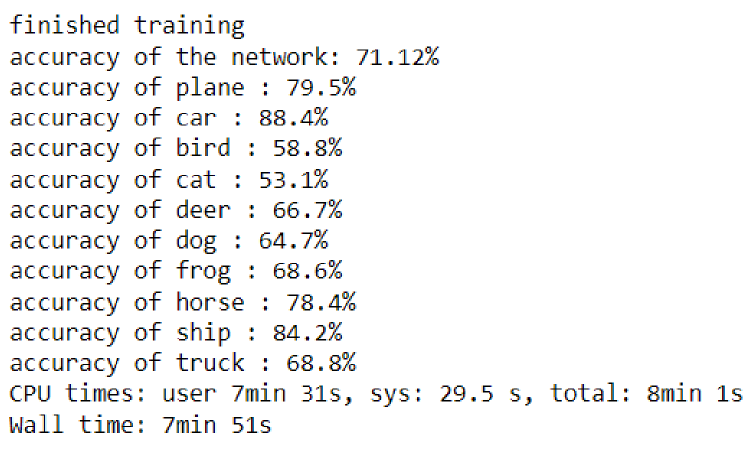
2.CNN MODEL B

2.1. for maxpool layer kernel size= 3



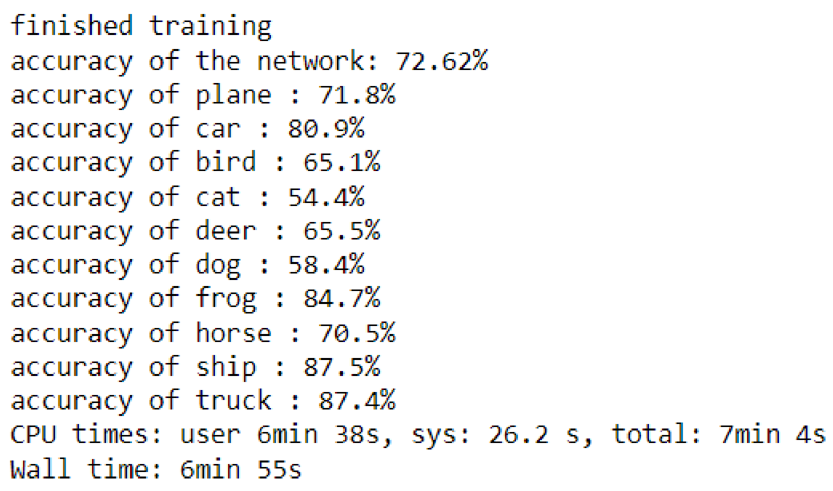
```
finished training
accuracy of the network: 67.13%
accuracy of plane : 67.4%
accuracy of car : 86.4%
accuracy of bird : 62.0%
accuracy of cat : 54.0%
accuracy of deer : 56.3%
accuracy of dog : 62.1%
accuracy of frog : 67.1%
accuracy of horse : 73.2%
accuracy of ship : 76.6%
accuracy of truck : 66.2%
CPU times: user 7min 21s, sys: 28.7 s, total: 7min 49s
Wall time: 7min 39s
```

2.2. for maxpool layer kernel size= 4



```
finished training
accuracy of the network: 71.12%
accuracy of plane : 79.5%
accuracy of car : 88.4%
accuracy of bird : 58.8%
accuracy of cat : 53.1%
accuracy of deer : 66.7%
accuracy of dog : 64.7%
accuracy of frog : 68.6%
accuracy of horse : 78.4%
accuracy of ship : 84.2%
accuracy of truck : 68.8%
CPU times: user 7min 31s, sys: 29.5 s, total: 8min 1s
Wall time: 7min 51s
```

2.3. for maxpool layer kernel size= 5

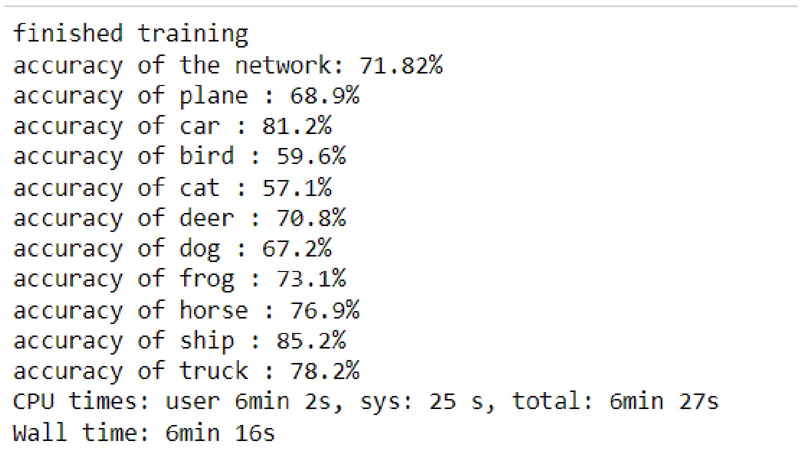


```
finished training
accuracy of the network: 72.62%
accuracy of plane : 71.8%
accuracy of car : 80.9%
accuracy of bird : 65.1%
accuracy of cat : 54.4%
accuracy of deer : 65.5%
accuracy of dog : 58.4%
accuracy of frog : 84.7%
accuracy of horse : 70.5%
accuracy of ship : 87.5%
accuracy of truck : 87.4%
CPU times: user 6min 38s, sys: 26.2 s, total: 7min 4s
Wall time: 6min 55s
```

Here also we observe the same trend as mentioned for cnn model A. The accuracy of the model increases with increase in the maxpool kernel size.

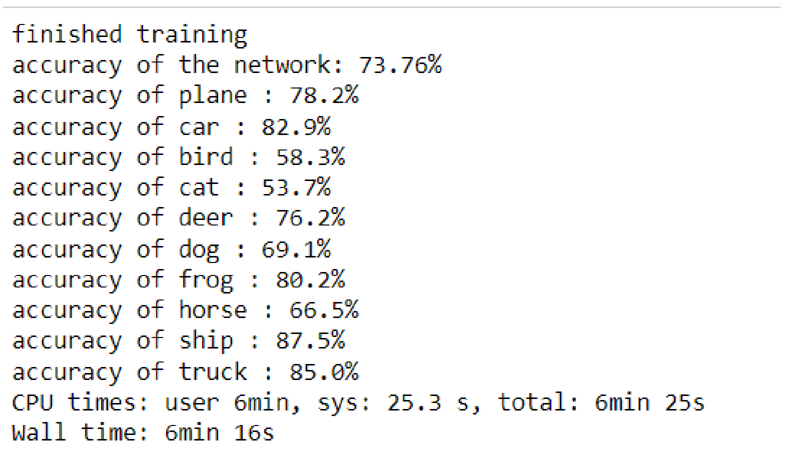
3.CNN MODEL C

3.1. for maxpool layer kernel size= 3



```
finished training
accuracy of the network: 71.82%
accuracy of plane : 68.9%
accuracy of car : 81.2%
accuracy of bird : 59.6%
accuracy of cat : 57.1%
accuracy of deer : 70.8%
accuracy of dog : 67.2%
accuracy of frog : 73.1%
accuracy of horse : 76.9%
accuracy of ship : 85.2%
accuracy of truck : 78.2%
CPU times: user 6min 2s, sys: 25 s, total: 6min 27s
Wall time: 6min 16s
```

3.2. for maxpool layer kernel size= 4



```
finished training
accuracy of the network: 73.76%
accuracy of plane : 78.2%
accuracy of car : 82.9%
accuracy of bird : 58.3%
accuracy of cat : 53.7%
accuracy of deer : 76.2%
accuracy of dog : 69.1%
accuracy of frog : 80.2%
accuracy of horse : 66.5%
accuracy of ship : 87.5%
accuracy of truck : 85.0%
CPU times: user 6min, sys: 25.3 s, total: 6min 25s
Wall time: 6min 16s
```

Here also we observe the same trend as mentioned for cnn model A and B. The accuracy of the model increases with increase in the maxpool kernel size.
Finally, we get the best accuracy of 73.76% from the test set using CNN model C and using a maxpool kernel of size (4x4).

C. Presence of one or more fully connected layer

presence of one or more fully connected layer does not affect the model much adding an extra fully connected layer boosts the accuracy slightly but there is no major advantage in using extra fully connected layer. The results can be interpreted from the models A and C which are similar with only model C having an extra fully connected layer

2.4 CONCLUSION

1. changing the network size and architecture changes the overall performance of the model even if we use same type of convolution and maxpool layers.
2. changing the max pool layer kernel size changes the performance of the model. The overall accuracy on the test set increases with increasing kernel size.
3. presence of one or more maxpooling layer does not affect the model much. It slightly boosts the accuracy of the model on the test set.

3 REFERENCES

1. FOR QUESTION-1

1.1.`https://www.analyticsvidhya.com/blog/2020/07/neural-networks-from-scratch-in-python-and-r/`
1.2.`https://towardsdatascience.com/neural-networks-from-scratch-easy-vs-hard-b26ddc2e89c7`
1.3.`https://github.com/TheIndependentCode/Neural-Network`

2. FOR QUESTION-2

2.1.`https://www.analyticsvidhya.com/blog/2021/09/convolutional-neural-network-pytorch-implement`
2.2.`https://blog.paperspace.com/pytorch-101-building-neural-networks/`
2.3.`https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`