Quiz3

Name: Mohamoud Hussein Mohamed

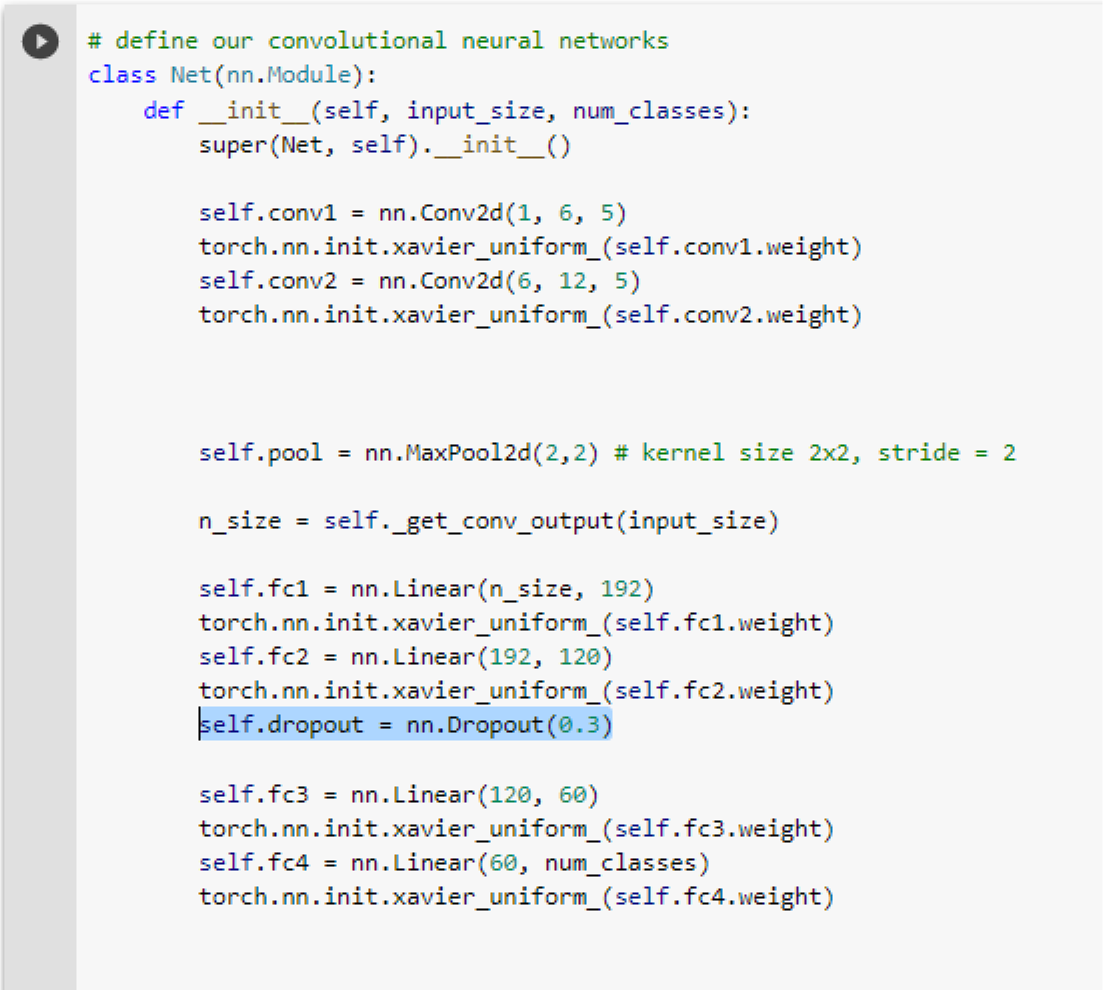
ID:G1914041

Question1:

After research the best suited loss function for this Connvolutional Neural Network is **Crossentropyloss**
The reason why this loss function compare to other loss functions that are available in pytorch is that:

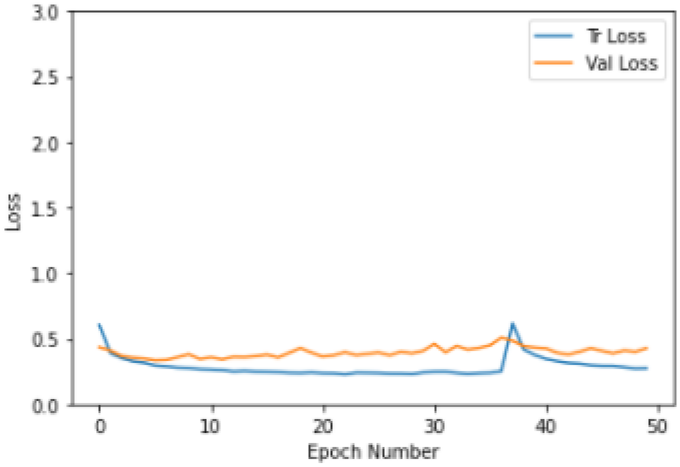
- Its one the most common loss function that is mostly used in training networks
- Using log-probabilities has the additional effect of keeping gradients from varying too widely
- Its more faster than others as its very uses log to get the data which is a simple algorithm
- Formula: $\text{loss}(x,class)=weight[class](-x[class]+\text{log}(j\sum\text{exp}(x[j])))$

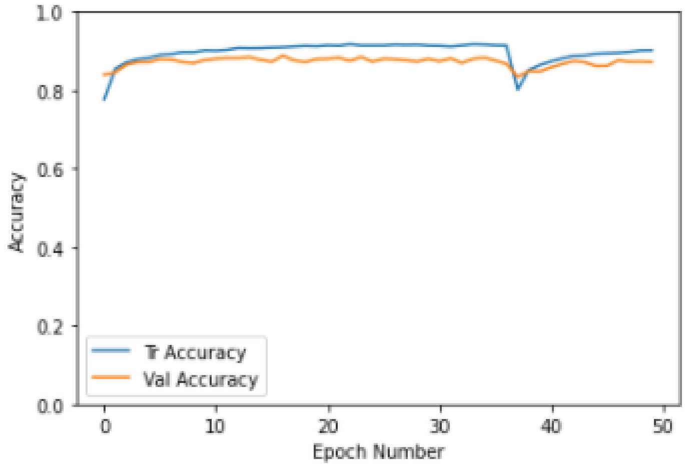
Question2:



```
# define our convolutional neural networks
class Net(nn.Module):
    def __init__(self, input_size, num_classes):
        super(Net, self).__init__()

        self.conv1 = nn.Conv2d(1, 6, 5)
        torch.nn.init.xavier_uniform_(self.conv1.weight)
        self.conv2 = nn.Conv2d(6, 12, 5)
        torch.nn.init.xavier_uniform_(self.conv2.weight)



        self.pool = nn.MaxPool2d(2,2) # kernel size 2x2, stride = 2

        n_size = self._get_conv_output(input_size)

        self.fc1 = nn.Linear(n_size, 192)
        torch.nn.init.xavier_uniform_(self.fc1.weight)
        self.fc2 = nn.Linear(192, 120)
        torch.nn.init.xavier_uniform_(self.fc2.weight)
        self.dropout = nn.Dropout(0.3)

        self.fc3 = nn.Linear(120, 60)
        torch.nn.init.xavier_uniform_(self.fc3.weight)
        self.fc4 = nn.Linear(60, num_classes)
        torch.nn.init.xavier_uniform_(self.fc4.weight)
```

Figure 1: ConvNet Architecture





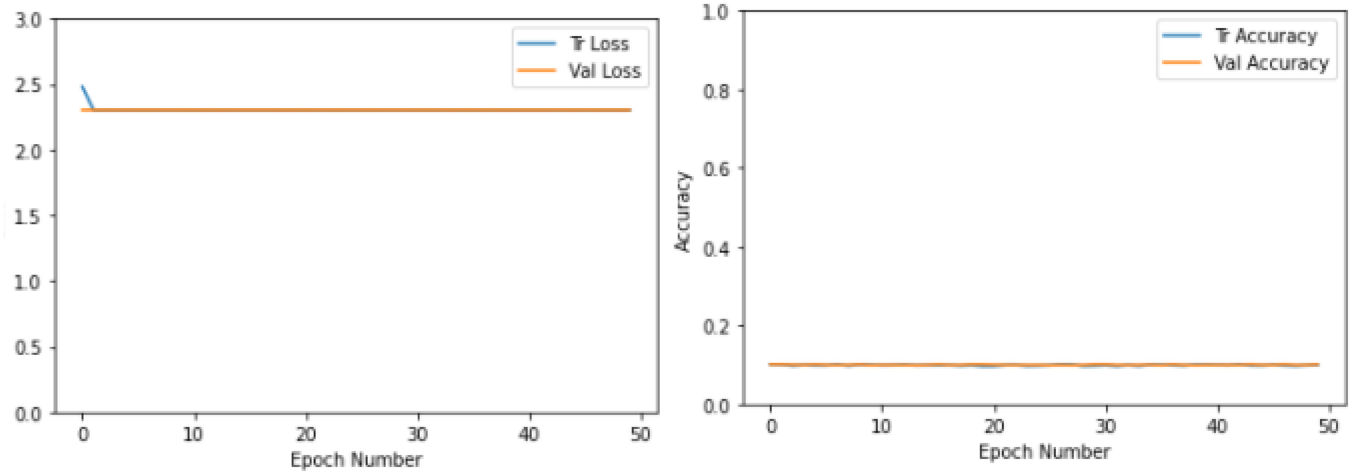


Figure 2: Loss plots of train and test data(Right),Accuracy plots of train and test data

- Optimizer=SGD optimizer
- Activation function=Relu
- Learning rate=0.1

Question3:

Activation Function	Accuracy	Loss
Tanh	Training Accuracy: 92.3019% Test Accuracy: 87.8560%	Training: Loss: 0.2022 Test Loss : 0.3762
Sigmoid	Train Accuracy: 9.9502% Test Accuracy: 9.9585%	Training: Loss: 2.3042 Test Loss : 2.3036
Elu	Train Accuracy: 10.0013% Test Accuracy: 9.8892%	Training: Loss: nan Test Loss : nan

B)Keepin ReLU Function and testing with different Learning Rate:

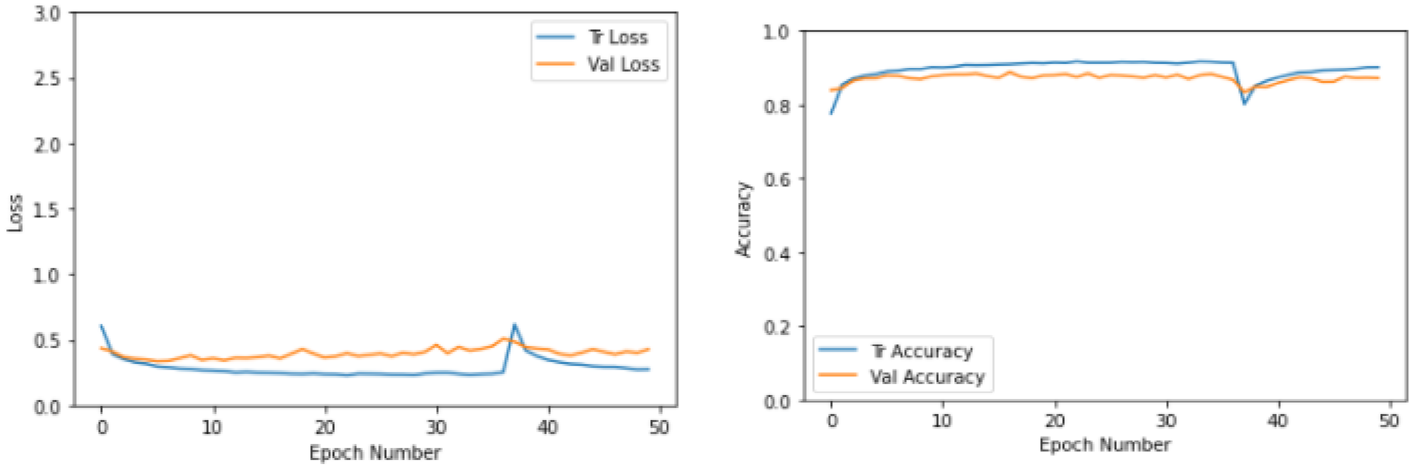


Figure 3: ReLU with Learning Rate=0.1

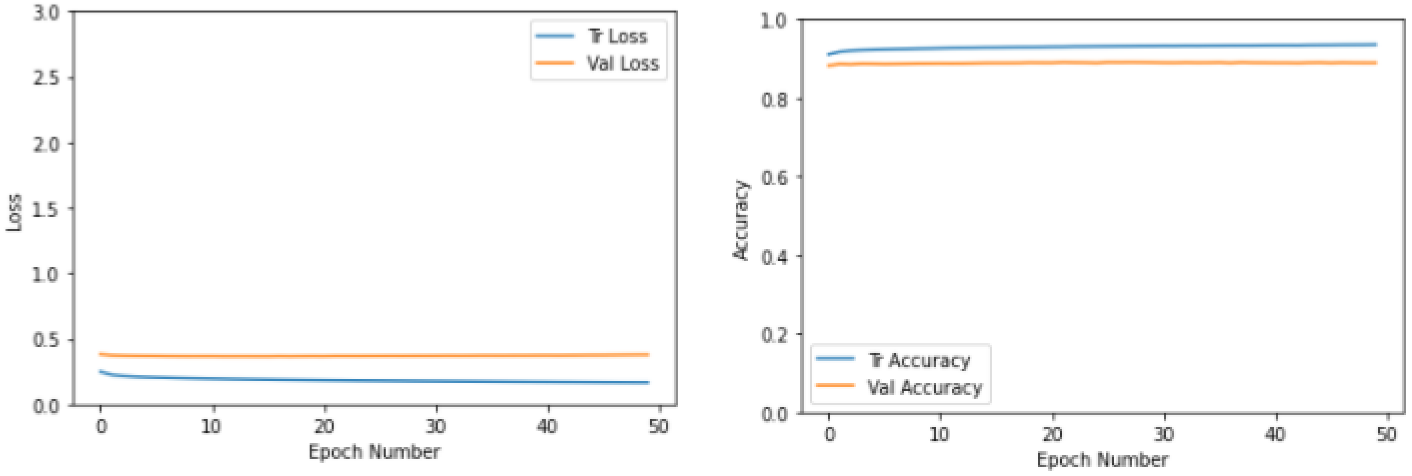


Figure 5: ReLU with Learning Rate=0.001

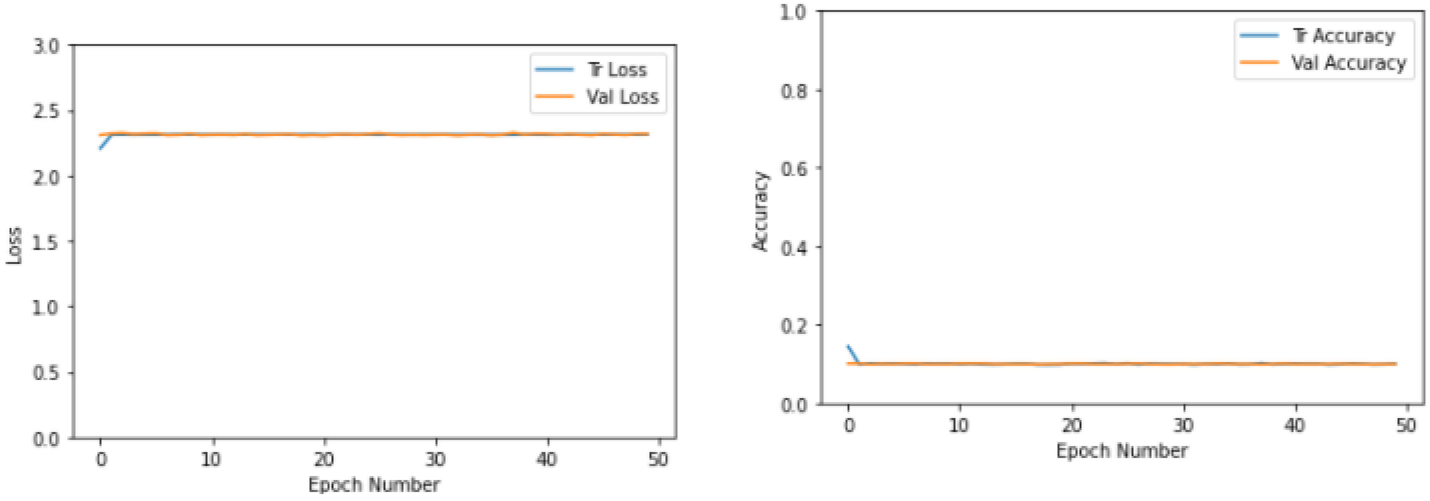


Figure 4:ReLU with learning Rate=0.5

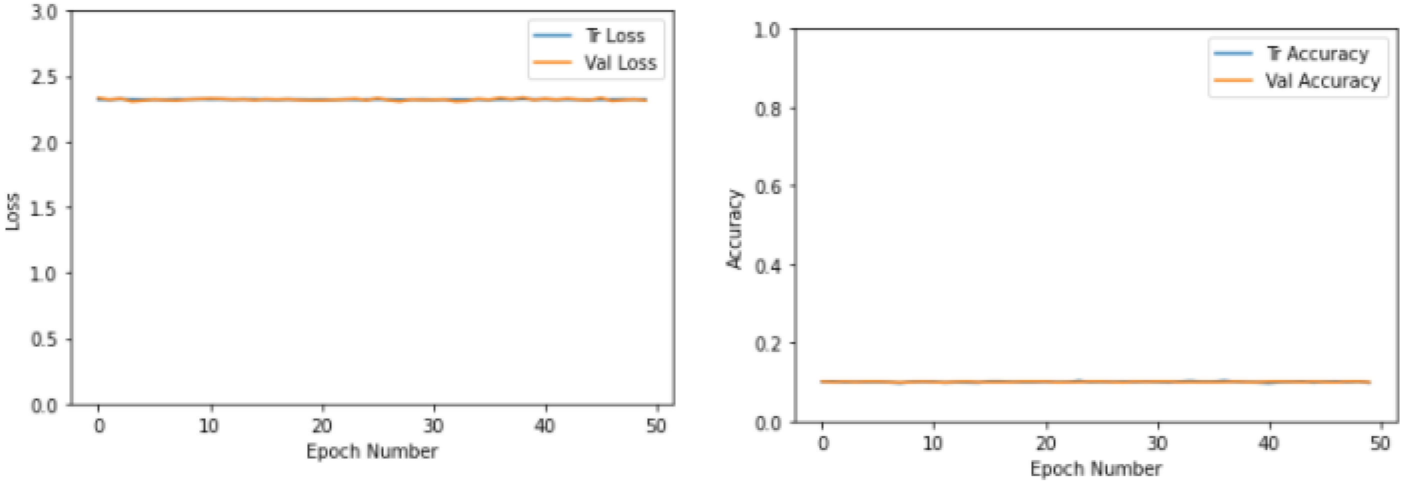


Figure 6:Relu with Learning rate=1

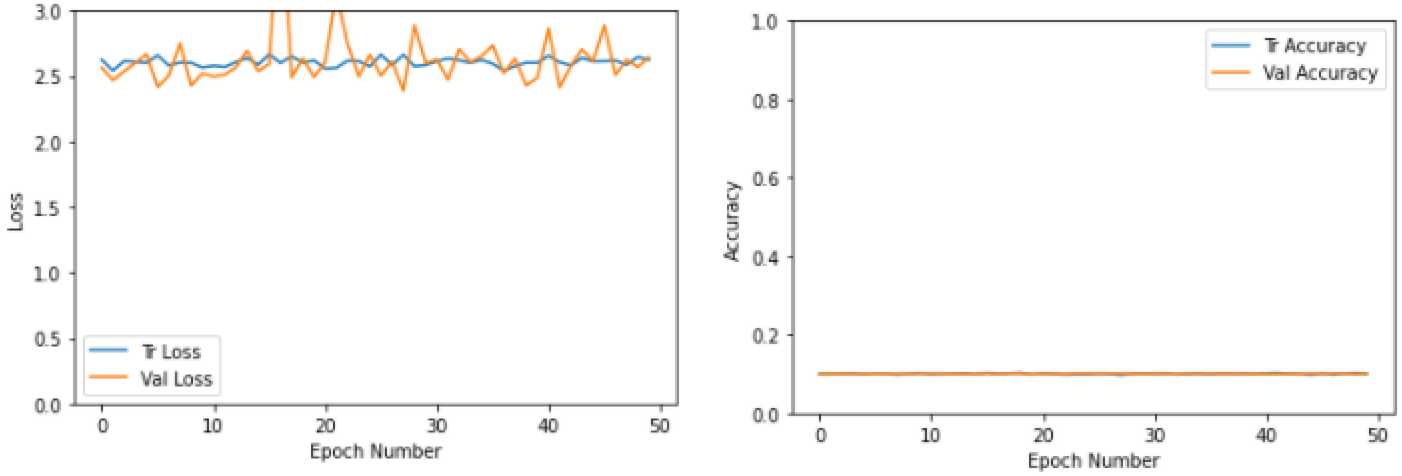


Figure 7:ReLU with Learning Rate=10

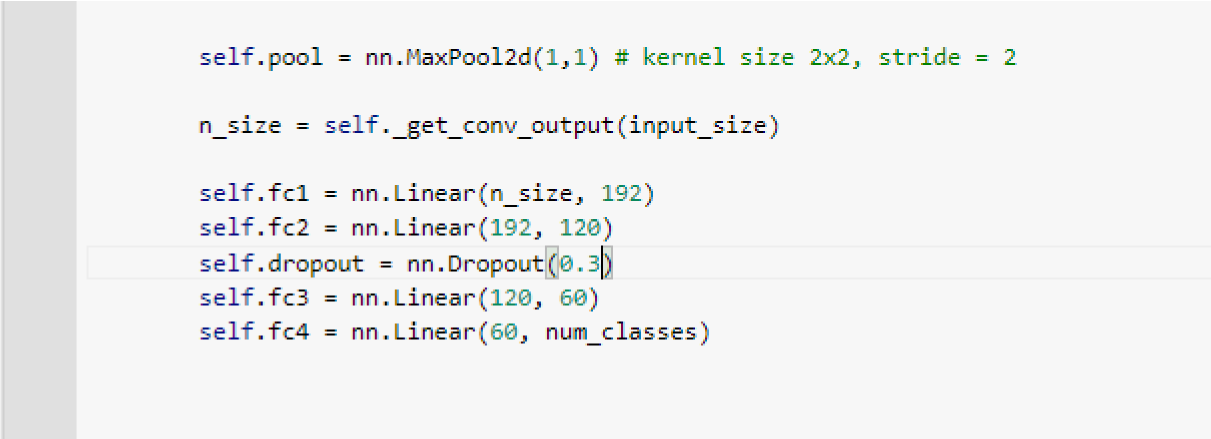
Conclusion:

As shown above diagram it looks the learning rate 0.001 is the best as the accuracy is very high and the loss is very loss , from 0.5 to higher learning rate the accuracy is very low.

But we can also observe that as the learning rate reaches 10 the loss graph start to randomly going up and down as the higher learning rate causes high change in the weights during back progression.

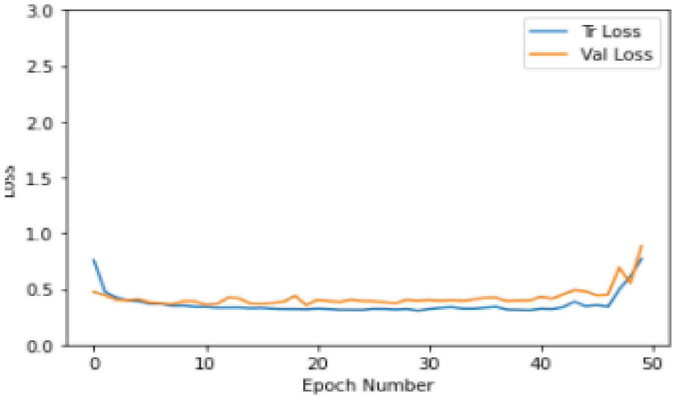
Question4:

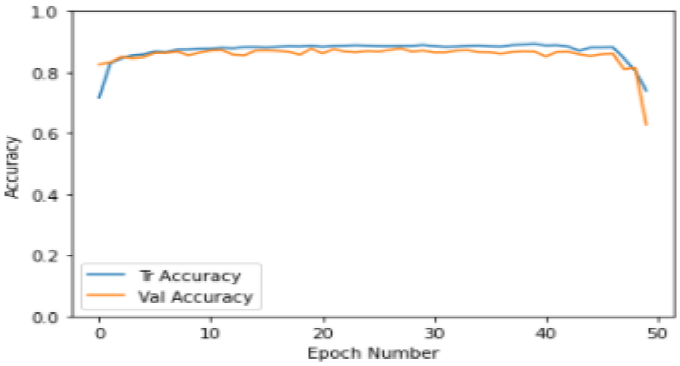
```
We add a dropout of 0.3 in the second fully connected layer
```



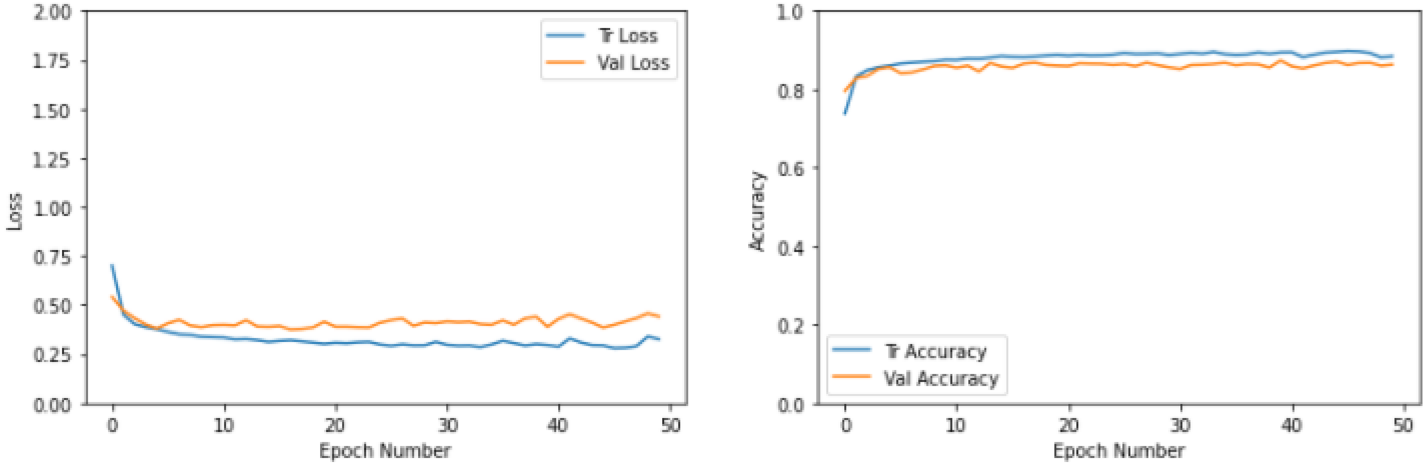
```
        self.pool = nn.MaxPool2d(1,1) # kernel size 2x2, stride = 2

        n_size = self._get_conv_output(input_size)

        self.fc1 = nn.Linear(n_size, 192)
        self.fc2 = nn.Linear(192, 120)
        self.dropout = nn.Dropout(0.3)
        self.fc3 = nn.Linear(120, 60)
        self.fc4 = nn.Linear(60, num_classes)
```

- Dropout=0.3

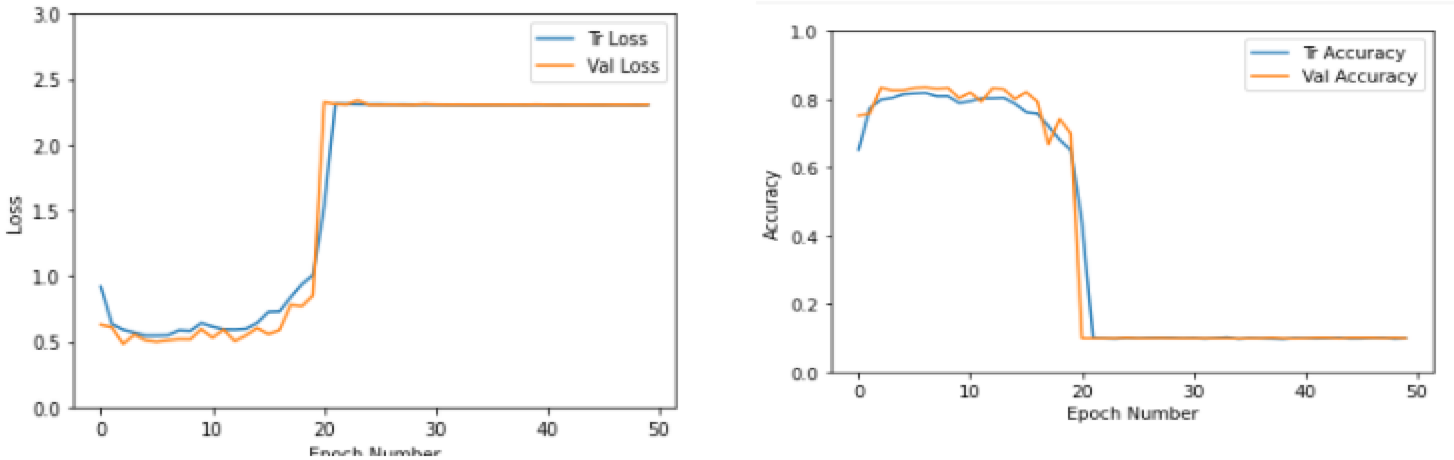




- DropOut=0.1



- Dropout=0.8



Conclusion:

As shown in the above results, if the dropout is low the gap between the training data and the test data is bigger compare when the dropout =0.3.

And if the dropout is too high it effect the accuracy and the loss as it make the accuracy low and the rate of loss degrading is very low.