In [4]:
```
#importing the libraries

import numpy as np
import matplotlib.pyplot as plt
import torch
from torchvision import datasets, transforms
from torch.utils.data import DataLoader, random_split
import torch.nn as nn
from torch.autograd import Variable
from torch.optim import Adam
from PIL import Image
from tabulate import tabulate
import torch.nn.functional as F
import os

os.environ["KMP_DUPLICATE_LIB_OK"]="TRUE"
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

#visualization of the images in the data set

image = Image.open(r"D:\Uni-Siegen\4th Sem (SS 2022)\RAML\DeepFake\Dataset\train\train_bicubic\imagewoof\0.jpg")
print('The format of the image is: ',image.format)
print('The mode the image is: ',image.mode)
print('The size of the images in the data set is : ',image.size)
#plt.imshow(image)
#plt.show()




#defining the path of the training and testing data

data_dir = 'D:/Uni-Siegen/4th Sem (SS 2022)/RAML/DeepFake/Dataset/'
train_data_path = ['train/train_bicubic','train/train_bilinear','train/train_pixelshuffle','train/combo']
test_data_path = ['test/bicubic', 'test/bilinear', 'test/pixelshuffle','test/combo']

"""
#selecting of the desired data set for training ans testing the model

print('\nThe Training and Test Data is as follows')
table_1 = [[1,'Bicubic','Bicubic'],[2, 'Bilinear','Bilinear'], [3, 'Pixel shuffle','Pixel shuffle'],[4,'All the above']]
header = ["Sr No", "Training Data", "Testing Data"]
```

```
print(tabulate(table_1, header))



#assignment of the value of our selection


train_path = int(input(('Enter the Sr No. of the data set for training: ')))-1
test_path =  int(input(('Enter the Sr No. of the data set for testing: ')))-1


#selecting of the desired data set for training ans testing the model


print('Select the Training and Testing datasets: ')
display_data_1 = wdg.Dropdown(
    options=[('Bicubic', 0), ('Bilinear', 1), ('Pixel shuffle', 2), ('All the above', 3)],
    value = 0,
    description='Training:',
)
display_data_2 = wdg.Dropdown(
    options=[('Bicubic', 0), ('Bilinear', 1), ('Pixel shuffle', 2), ('All the above', 3)],
    value = 0,
    description='Testing:',
)
display(wdg.HBox([display_data_1, display_data_2]))
#display(display_data_1)
#display(display_data_2)



#assignment of the value of our selection

train_path = display_data_1.value
test_path = display_data_2.value
#print(train_path,test_path)
"""
#declaration of the lists to aggregate the losses and accuracies

combo_train_losses = []
combo_train_accu = []
combo_test_losses = []
combo_test_accu = []
combo_validation_losses = []
```

```
combo_validation_accu = []

#iterate over different training paths to train over Bicibic, Bilinear, Pixel Shuffle and combinesd data.
for idx_tr,trn in enumerate(train_data_path):

        #initilization of the hyperparamaters

        batch_size = 40
        num_epochs = 50
        best_accuracy=0.0
        min_validation_loss = np.inf
        the_last_loss = 100
        patience = 2
        trigger_times = 0
        final_epoch = 0
        image_size = [32,32]

        #declaration of lists to append the performance measurement parameters

        train_losses = []
        train_accu = []
        test_losses = []
        test_accu = []
        validation_losses = []
        validation_accu = []


        data_transforms = {
                'train': transforms.Compose([
                #transforms.Grayscale(1),
                transforms.Resize(image_size),
                transforms.RandomHorizontalFlip(),
                transforms.ToTensor()]),

                'test': transforms.Compose([
                #transforms.Grayscale(1),
                transforms.Resize(image_size) ,
                transforms.ToTensor()])}

        train_dataset = datasets.ImageFolder(os.path.join(data_dir,trn),data_transforms['train'])

        #Split the training data into traning and validation datasets in the 80:20 ratio
        percentage_split = 0.8
        val_percentage_split = round(1 - percentage_split, 1)
```

```
    val_data_len = int(val_percentage_split*train_dataset.__len__())
    train_data_len = train_dataset.__len__() - val_data_len
    #print(train_data_len,val_data_len)

    train_dataset, validation_dataset = random_split(train_dataset,[train_data_len,val_data_len])

    """
    train_sampler = SubsetRandomSampler(list(range(1600-320)))
    validation_sampler = SubsetRandomSampler(list(range(320)))

    train_loader = DataLoader(dataset=train_dataset, batch_size=batch_size, sampler = train_sampler)
    validation_loader = DataLoader(dataset=train_dataset, batch_size=batch_size, sampler = validation_sampler)
    """
    train_loader = DataLoader(dataset=train_dataset, batch_size=batch_size)
    validation_loader = DataLoader(dataset=validation_dataset, batch_size=batch_size)


    num_train_sam = train_loader.__len__()*train_loader.batch_size
    num_valid_sam = validation_loader.__len__()*validation_loader.batch_size

    print('\nThere are',num_train_sam, 'train samples')
    print('There are',num_valid_sam, 'validation samples')


    #displaying the labels for real and fake images

    print('\nThe Classification is as follows')
    table_2 = [[0,'Real Image'],[1, 'Fake Image']]
    header = ["Label", "Image Classification"]
    print(tabulate(table_2, header))

    """
    dataiter = iter(train_loader)
    images, labels = dataiter.next()
    #print(labels.shape)
    label= labels.numpy()
    #print(label)

    index = 0

    for index in range(len(label)):
        if (label[index] != 0):
            label[index] = 1
    """
```

```
#visualizing the images and their respective labels of the loaded images

dataiter = iter(train_loader)
images, labels = dataiter.next()
print('\nThe loaded batch of the images is a tensor with size: ',images.shape)
#print(labels.shape)
#
label= labels.numpy()
print('The loaded batch of the images is a tensor with labels: ', label[:4])
#print(label[:4])


classes = ['Real image', 'Fake image']

fig_1 = plt.figure()
for i in range(4):
  plt.subplot(2,2,i+1)
  plt.tight_layout()
  plt.imshow(images[i][0], cmap='gray', interpolation='none')
  plt.title("Reality: {}".format(classes[labels[i]]))
  plt.xticks([])
  plt.yticks([])
fig_1

"""
fig_2 = plt.figure()
for i, (images, labels) in enumerate(train_loader):
    plt.subplot(2,2,i+1)
    plt.tight_layout()
    plt.imshow(images[i][0], cmap='gray', interpolation='none')
    plt.title("Reality: {}".format(classes[labels[i]]))
    plt.xticks([])
    plt.yticks([])
fig_2

"""
print('\n')

#defining the CNN model

class ConvNet(nn.Module):
    def __init__(self):
        super(ConvNet,self).__init__()
```

```
        #Input image shape: (50,3,32,32)
        self.conv1=nn.Conv2d(in_channels=3,out_channels=12,kernel_size=3,stride=1,padding=1)
        #image shape: (50,12,32,32)
        self.bn1=nn.BatchNorm2d(num_features=12)
        #image shape: (50,12,32,32)
        self.relu1=nn.ReLU()
        #image shape: (50,12,32,32)
        self.pool=nn.MaxPool2d(kernel_size=2)
        #image shape: (50,12,16,16)
        self.conv2=nn.Conv2d(in_channels=12,out_channels=20,kernel_size=3,stride=1,padding=1)
        #image shape: (50,20,16,16)
        self.relu2=nn.ReLU()
        #image shape: (50,20,16,16)
        self.conv3=nn.Conv2d(in_channels=20,out_channels=32,kernel_size=3,stride=1,padding=1)
        #image shape: (50,32,16,16)
        self.bn3=nn.BatchNorm2d(num_features=32)
        #image shape: (50,32,16,16)
        self.relu3=nn.ReLU()
        #image shape: (50,32,16,16)

        """
        self.pool=nn.MaxPool2d(kernel_size=2)

        self.conv4=nn.Conv2d(in_channels=32,out_channels=45,kernel_size=3,stride=1,padding=1)

        self.bn4=nn.BatchNorm2d(num_features=45)

        self.relu4=nn.ReLU()

        """

        self.fc=nn.Linear(in_features=16 * 16 * 32,out_features=2)
        #self.fc1 = nn.Linear(in_features=16 * 16 * 32,out_features=5000)
        #self.fc2 = nn.Linear(in_features=5000,out_features=2)
        #self.fc3 = nn.Linear(in_features=2500,out_features=1250)
        #self.fc4 = nn.Linear(in_features=1250,out_features=2)


    def forward(self,input):
        output=self.conv1(input)
        output=self.bn1(output)
        output=self.relu1(output)

        output=self.pool(output)
```

```
        output=self.conv2(output)
        output=self.relu2(output)


        output=self.conv3(output)
        output=self.bn3(output)
        output=self.relu3(output)

        """
        output=self.pool(output)


        output=self.conv4(output)
        output=self.bn4(output)
        output=self.relu4(output)
        """

        output=output.view(-1, 32 * 16 * 16)


        output=self.fc(output)
        #output = F.relu(self.fc1(output))
        #output = F.relu(self.fc2(output))
        #output = F.relu(self.fc3(output))
        #output = F.relu(self.fc4(output))
        output = torch.sigmoid(output)

        return output


model = ConvNet().to(device)

#defining the loss function and the optimizer

loss_function=nn.CrossEntropyLoss().to(device)
#optimizer = torch.optim.SGD(model.parameters(), lr = 0.0001,weight_decay=0.001)
optimizer=Adam(model.parameters(),lr=0.0001,weight_decay=0.001)

print(device)

#training the model
for epoch in range(num_epochs):
```

```
            model.train()
            train_accuracy=0.0
            train_loss=0.0

            for i, (images,labels) in enumerate(train_loader):
                if torch.cuda.is_available():
                    images=Variable(images.cuda())
                    labels=Variable(labels.cuda())

                optimizer.zero_grad()

                outputs=model(images)
                loss = loss_function(outputs,labels)
                loss.backward()
                optimizer.step()


                train_loss+= loss.cpu().data*images.size(0)
                _,prediction=torch.max(outputs.data,1)

                train_accuracy+=int(torch.sum(prediction==labels.data))

            train_accuracy = train_accuracy/num_train_sam*100
            train_loss = train_loss/num_train_sam
            train_accu.append(train_accuracy)
            train_losses.append(train_loss)


            #validating the model

            model.eval()
            validation_accuracy=0.0
            validation_loss = 0.0

            for i, (images,labels) in enumerate(validation_loader):
                if torch.cuda.is_available():
                    images=Variable(images.cuda())
                    labels=Variable(labels.cuda())

                outputs=model(images)
                loss = loss_function(outputs,labels)

                validation_loss+= loss.cpu().data*images.size(0)
                _,prediction=torch.max(outputs.data,1)
```

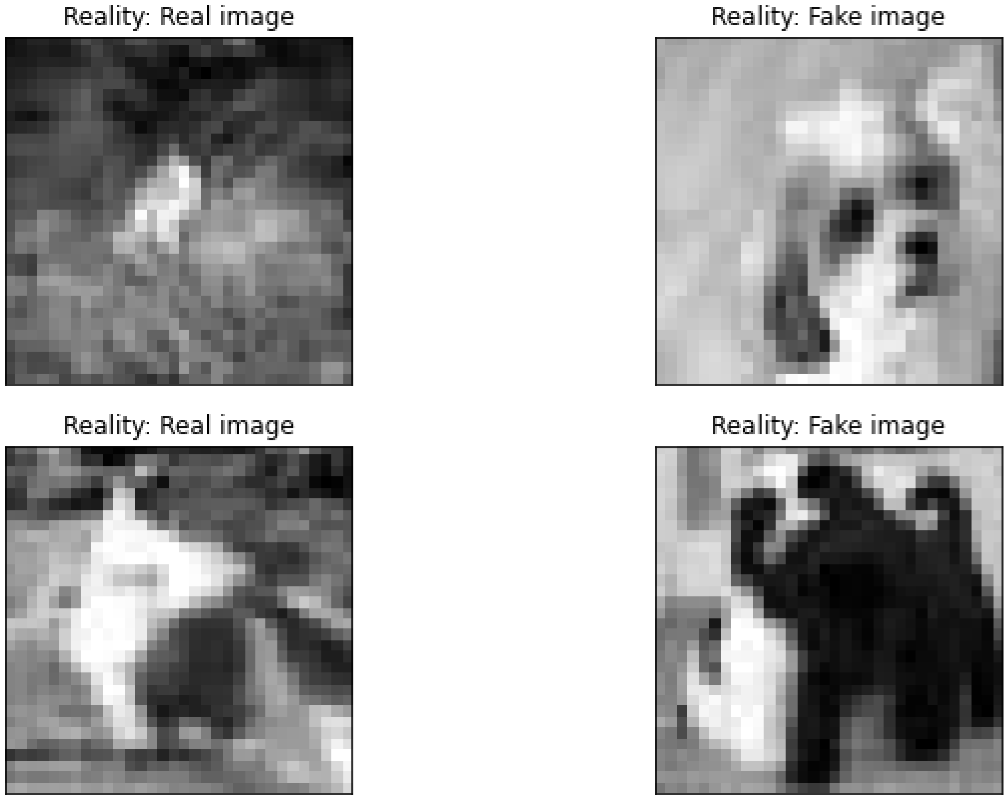
```
                validation_accuracy+=int(torch.sum(prediction==labels.data))

            validation_accuracy=validation_accuracy/num_valid_sam*100
            validation_loss = validation_loss/num_valid_sam
            validation_losses.append(validation_loss)
            validation_accu.append(validation_accuracy)



            #print('\n')
            if epoch < 9:
                print('Epoch:',epoch+1,'      Train Loss:',format(train_loss.item(),".4f"),'     Train Accuracy:',format(train_accu
                #print(tabulate([epoch,train_loss,train_accuracy,test_accuracy], headers = ['Epoch','Train Loss','Train Accuracy
            else:
                print('Epoch:',epoch+1,'     Train Loss:',format(train_loss.item(),".4f"),'     Train Accuracy:',format(train_accu


            #Implementation of Early stop to prevent overfitting

            if validation_loss > the_last_loss:
                trigger_times += 1
                print('Patirnce:', trigger_times)

                if trigger_times >= patience:
                    final_epoch = epoch
                    print('Early stopping\n\nStart of the testing process.\n')
                    break

            else:
                print('Patirnce: 0')
                trigger_times = 0

            the_last_loss = validation_loss


        combo_train_losses.append(train_loss.item())
        combo_validation_losses.append(validation_loss.item())
        combo_train_accu.append(train_accuracy)
        combo_validation_accu.append(validation_accuracy)


        #iterate over different testing paths to test over Bicibic, Bilinear, Pixel Shuffle and combinesd data.
```

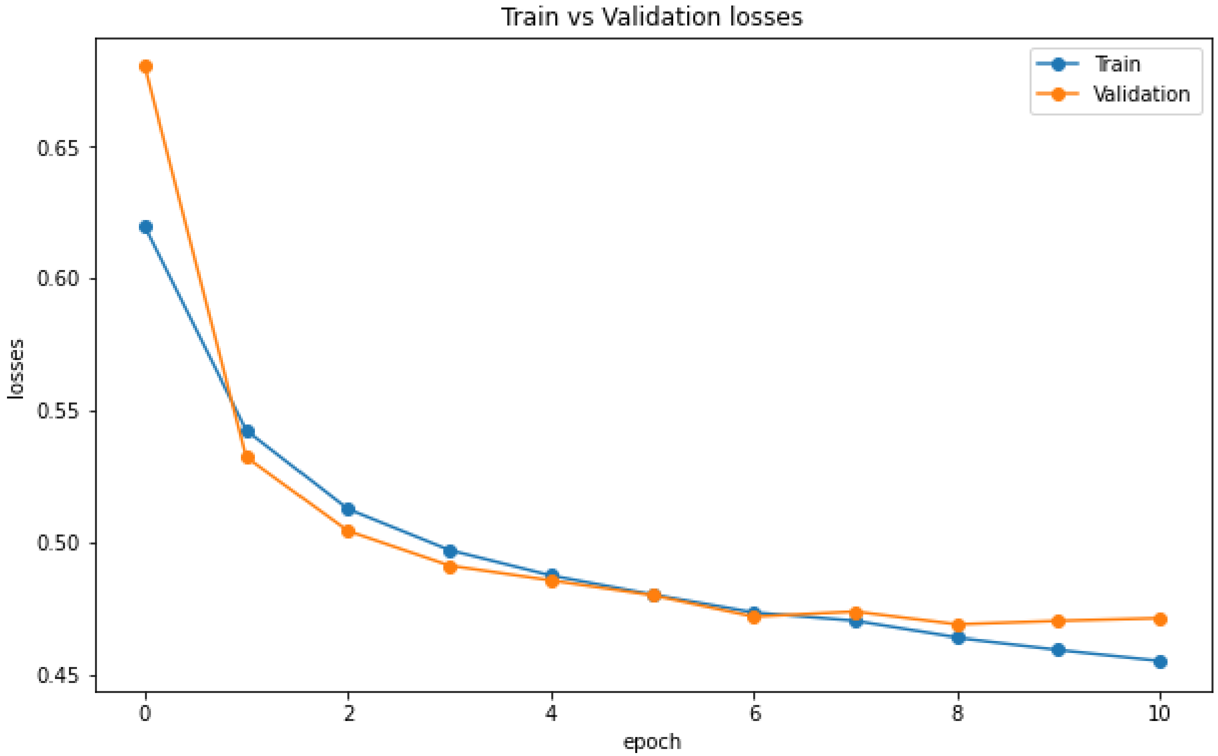
```
    for idx_tst,tst in enumerate(test_data_path):

        model.eval()
        test_accuracy=0.0
        test_loss = 0.0

        test_dataset = datasets.ImageFolder(os.path.join(data_dir,tst),data_transforms['test'])
        test_loader = DataLoader(dataset=test_dataset, batch_size=batch_size, shuffle=True)
        num_test_sam = test_loader.__len__()*test_loader.batch_size
        print('There are',num_test_sam, 'test samples\n')

        for i, (images,labels) in enumerate(test_loader):
            if torch.cuda.is_available():
                images=Variable(images.cuda())
                labels=Variable(labels.cuda())

            outputs=model(images)
            loss = loss_function(outputs,labels)

            test_loss+= loss.cpu().data*images.size(0)
            _,prediction=torch.max(outputs.data,1)


            test_accuracy+=int(torch.sum(prediction==labels.data))


        test_accuracy=test_accuracy/num_test_sam*100
        test_loss = test_loss/num_test_sam


        #test_losses.append(test_loss)
        #test_accu.append(test_accuracy)

        table_1 = ['Bicubic','Bilinear','Pixel shuffle','Combined data']

        print('After carring out the training with ' + str(table_1[idx_tr]) + ' dataset and testing the model with ' +str(tal
        if epoch < 9:
            print('Test Loss:',format(test_loss.item(),".4f"),  '     Test Accuracy:',format(test_accuracy,".4f"))
            #print(tabulate([epoch,train_loss,train_accuracy,test_accuracy], headers = ['Epoch','Train Loss','Train Accuracy
        else:
            print('Test Loss:',format(test_loss.item(),".4f"),  '     Test Accuracy:',format(test_accuracy,".4f"))

        #appending all the losses and accuracy to the respective lists
```

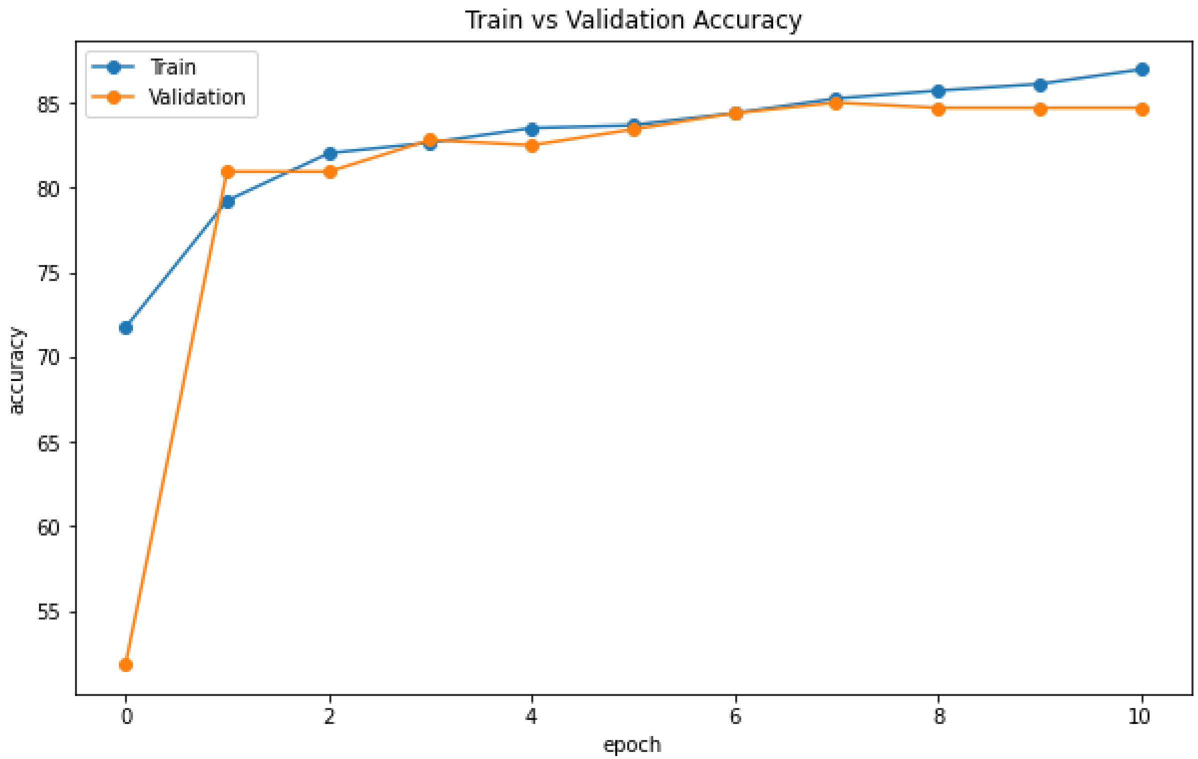
```
            combo_test_losses.append(test_loss.item())
            combo_test_accu.append(test_accuracy)
            combo_train_losses.append(0)
            combo_validation_losses.append(0)
            combo_train_accu.append(0)
            combo_validation_accu.append(0)


            plt.rcParams["figure.figsize"] = (10,6)

            #plotting the training vs validation lossses

            fig_3 = plt.figure()
            plt.plot(train_losses,'-o')
            #plt.plot(test_losses,'-o')
            plt.plot(validation_losses,'-o')
            plt.xlabel('epoch')
            plt.ylabel('losses')
            plt.legend(['Train','Validation'])
            plt.title('Train vs Validation losses')
            plt.show()

            #plotting the training vs validation accuracies

            fig_4 = plt.figure()
            plt.plot(train_accu,'-o')
            #plt.plot(test_accu,'-o')
            plt.plot(validation_accu,'-o')
            plt.xlabel('epoch')
            plt.ylabel('accuracy')
            plt.legend(['Train', 'Validation'])
            plt.title('Train vs Validation Accuracy')
            plt.show()
```

```
The format of the image is:  JPEG
The mode the image is:  RGB
The size of the images in the data set is :  (32, 32)

There are 1280 train samples
There are 320 validation samples

The Classification is as follows
  Label  Image Classification
-------  ----------------------
      0  Real Image
      1  Fake Image

The loaded batch of the images is a tensor with size:  torch.Size([40, 3, 32, 32])
The loaded batch of the images is a tensor with labels:  [0 1 0 1]


cpu
Epoch: 1      Train Loss: 0.6194     Train Accuracy: 71.7188      Validation Loss: 0.6799     Validation Accuracy: 51.8750
Patirnce: 0
Epoch: 2      Train Loss: 0.5424     Train Accuracy: 79.2188      Validation Loss: 0.5323     Validation Accuracy: 80.9375
Patirnce: 0
Epoch: 3      Train Loss: 0.5126     Train Accuracy: 82.0312      Validation Loss: 0.5043     Validation Accuracy: 80.9375
Patirnce: 0
Epoch: 4      Train Loss: 0.4970     Train Accuracy: 82.6562      Validation Loss: 0.4912     Validation Accuracy: 82.8125
Patirnce: 0
Epoch: 5      Train Loss: 0.4874     Train Accuracy: 83.5156      Validation Loss: 0.4856     Validation Accuracy: 82.5000
Patirnce: 0
Epoch: 6      Train Loss: 0.4802     Train Accuracy: 83.6719      Validation Loss: 0.4799     Validation Accuracy: 83.4375
Patirnce: 0
Epoch: 7      Train Loss: 0.4734     Train Accuracy: 84.3750      Validation Loss: 0.4718     Validation Accuracy: 84.3750
Patirnce: 0
Epoch: 8      Train Loss: 0.4703     Train Accuracy: 85.2344      Validation Loss: 0.4738     Validation Accuracy: 85.0000
Patirnce: 1
Epoch: 9      Train Loss: 0.4640     Train Accuracy: 85.7031      Validation Loss: 0.4689     Validation Accuracy: 84.6875
Patirnce: 0
Epoch: 10     Train Loss: 0.4594     Train Accuracy: 86.0938      Validation Loss: 0.4702     Validation Accuracy: 84.6875
Patirnce: 1
Epoch: 11     Train Loss: 0.4552     Train Accuracy: 86.9531      Validation Loss: 0.4713     Validation Accuracy: 84.6875
Patirnce: 2
Early stopping

Start of the testing process.
```

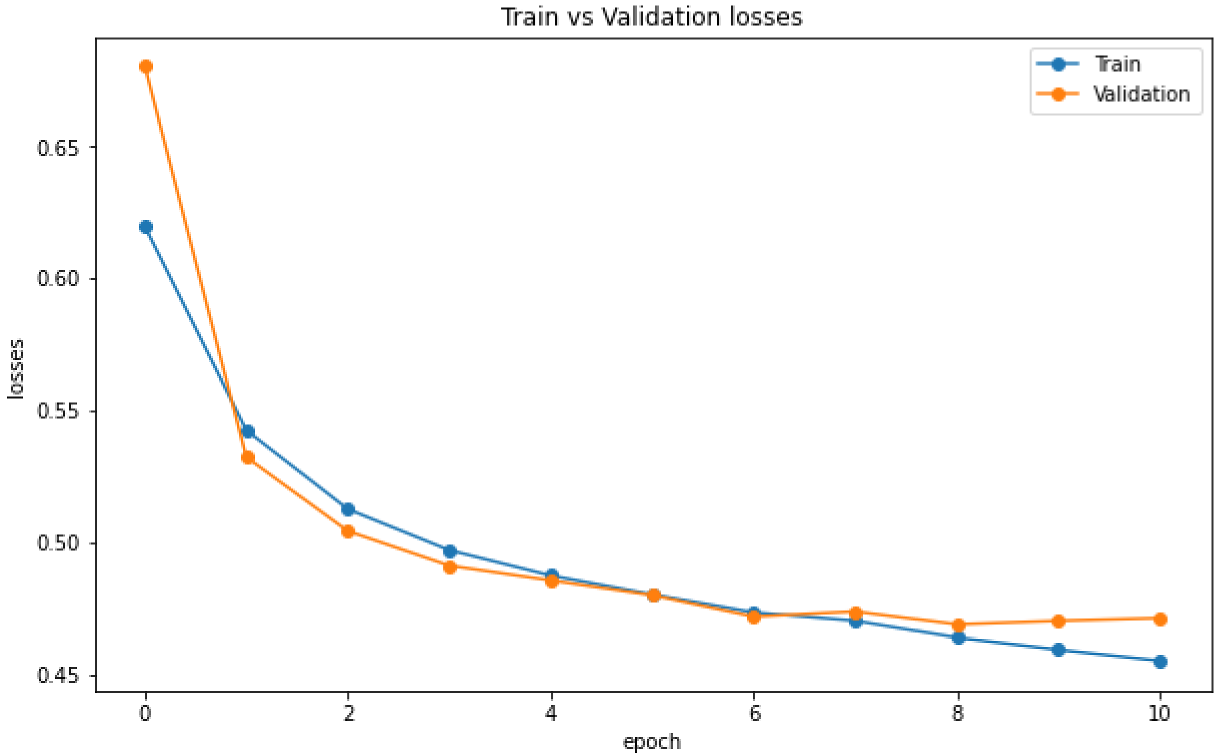
```
There are 400 test samples

After carring out the training with Bicubic dataset and testing the model with Bicubic we obtain:
Test Loss: 0.4953      Test Accuracy: 82.2500
```

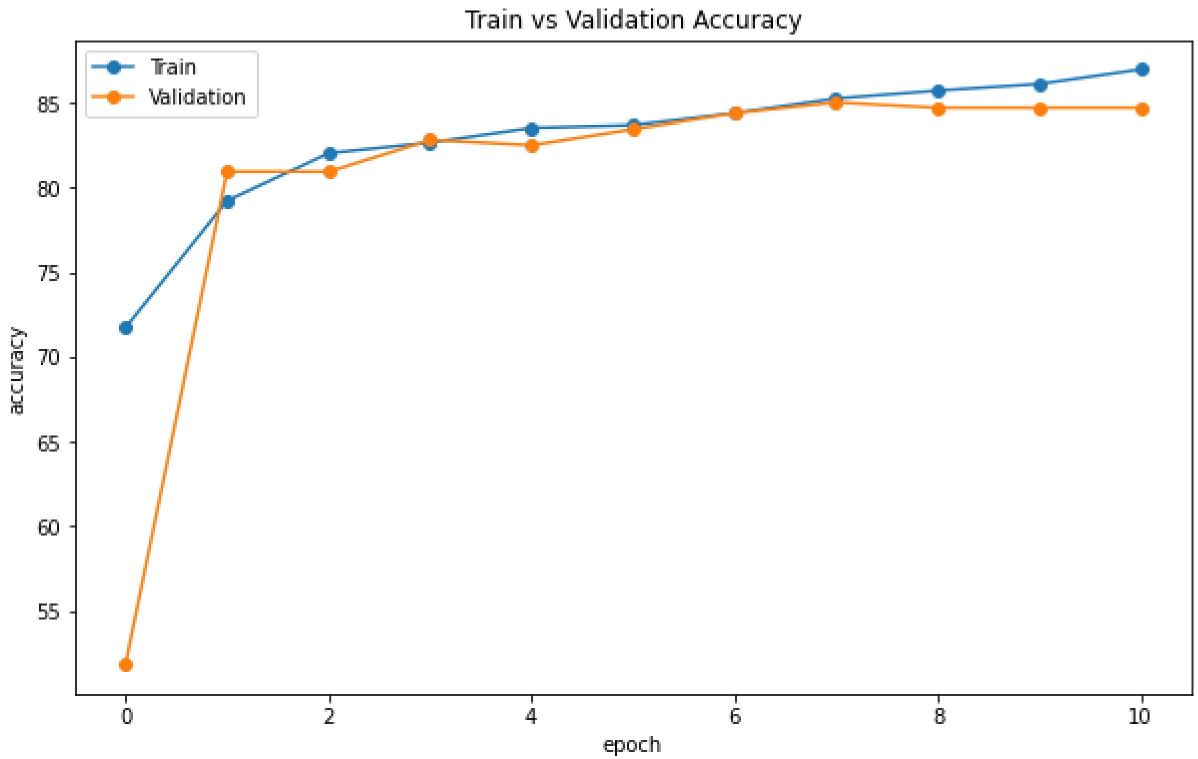



Train vs Validation losses
Train
Validation
0.65
0.60
0.55
0.50
0.45
losses
0
2
4
6
8
10
epoch

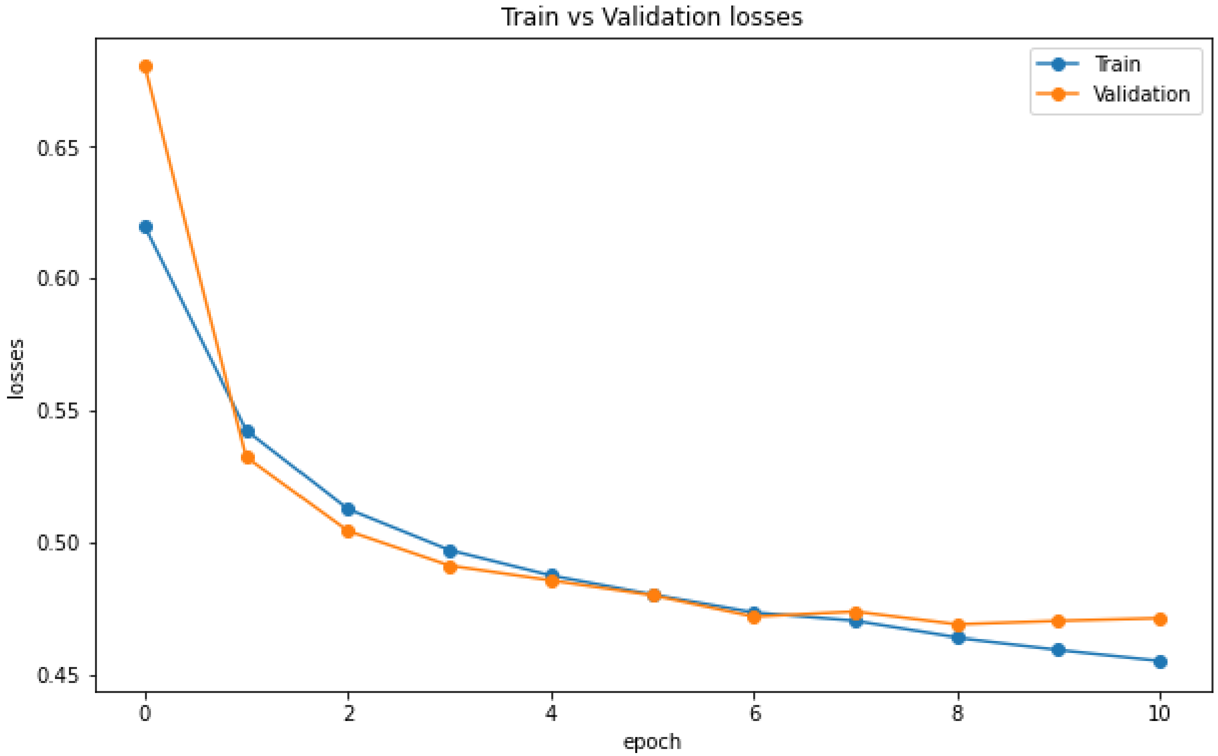


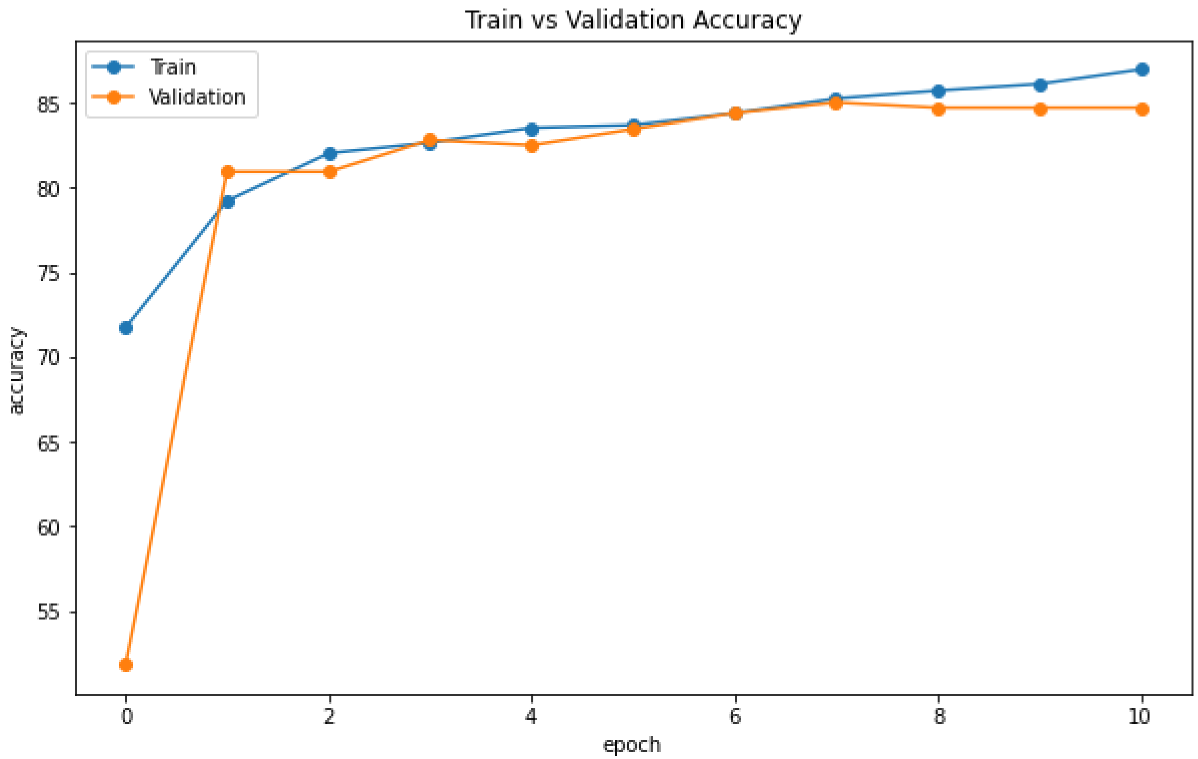

```
There are 400 test samples

After carring out the training with Bicubic dataset and testing the model with Bilinear we obtain:
Test Loss: 0.5707      Test Accuracy: 75.7500
```


Train vs Validation losses
Train
Validation
0.65
0.60
0.55
0.50
0.45
losses
0
2
4
6
8
10
epoch

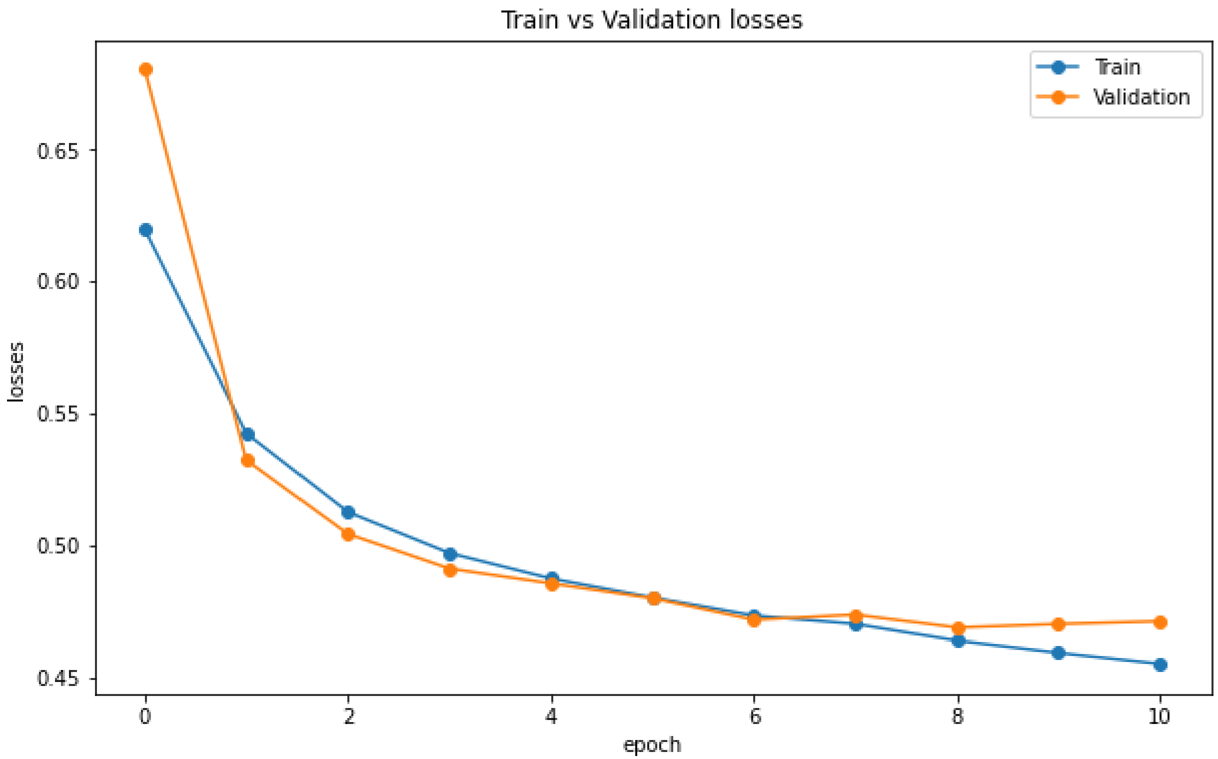


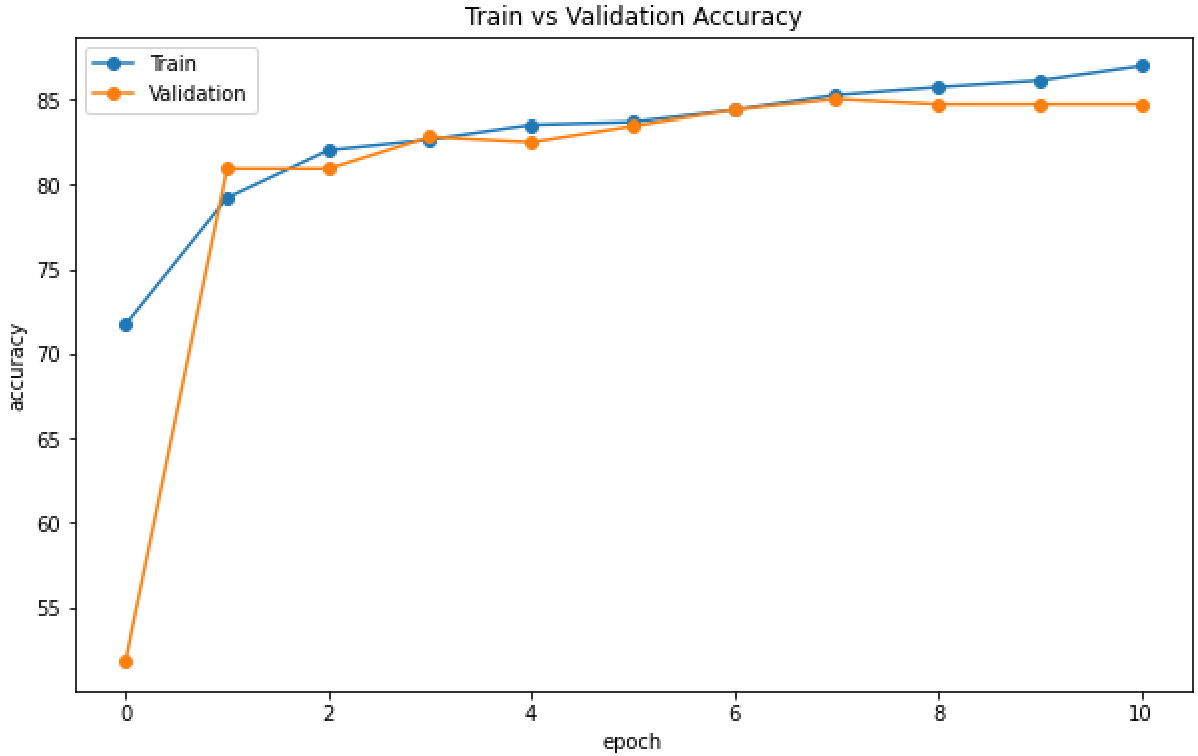

```
There are 400 test samples

After carring out the training with Bicubic dataset and testing the model with Pixel shuffle we obtain:
Test Loss: 0.5087      Test Accuracy: 80.2500
```


Train vs Validation losses
Train
Validation
0.65
0.60
0.55
0.50
0.45
losses
0
2
4
6
8
10
epoch




```
There are 800 test samples

After carring out the training with Bicubic dataset and testing the model with Combined data we obtain:
Test Loss: 1.0121      Test Accuracy: 19.8750
```


Train vs Validation losses
Train
Validation
losses
0.65
0.60
0.55
0.50
0.45
0
2
4
6
8
10
epoch

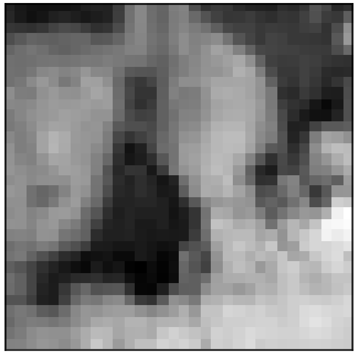

Train vs Validation Accuracy
Train
Validation
accuracy
epoch
85
80
75
70
65
60
55
0
2
4
6
8
10

```
There are 1280 train samples
There are 320 validation samples

The Classification is as follows
  Label  Image Classification
-------  ----------------------
      0  Real Image
      1  Fake Image

The loaded batch of the images is a tensor with size:  torch.Size([40, 3, 32, 32])
The loaded batch of the images is a tensor with labels:  [0 0 0 1]


cpu
Epoch: 1      Train Loss: 0.6440     Train Accuracy: 67.0312      Validation Loss: 0.6848     Validation Accuracy: 50.0000
Patirnce: 0
Epoch: 2      Train Loss: 0.5764     Train Accuracy: 77.4219      Validation Loss: 0.5643     Validation Accuracy: 76.2500
Patirnce: 0
Epoch: 3      Train Loss: 0.5455     Train Accuracy: 79.7656      Validation Loss: 0.5372     Validation Accuracy: 79.3750
Patirnce: 0
Epoch: 4      Train Loss: 0.5252     Train Accuracy: 81.3281      Validation Loss: 0.5282     Validation Accuracy: 78.7500
Patirnce: 0
Epoch: 5      Train Loss: 0.5105     Train Accuracy: 82.5000      Validation Loss: 0.5198     Validation Accuracy: 79.0625
Patirnce: 0
Epoch: 6      Train Loss: 0.5000     Train Accuracy: 83.6719      Validation Loss: 0.5132     Validation Accuracy: 80.3125
Patirnce: 0
Epoch: 7      Train Loss: 0.4881     Train Accuracy: 85.3906      Validation Loss: 0.5086     Validation Accuracy: 79.6875
Patirnce: 0
Epoch: 8      Train Loss: 0.4810     Train Accuracy: 85.5469      Validation Loss: 0.4986     Validation Accuracy: 81.5625
Patirnce: 0
Epoch: 9      Train Loss: 0.4729     Train Accuracy: 86.3281      Validation Loss: 0.4953     Validation Accuracy: 82.5000
Patirnce: 0
Epoch: 10     Train Loss: 0.4649     Train Accuracy: 87.5000      Validation Loss: 0.4934     Validation Accuracy: 81.5625
Patirnce: 0
Epoch: 11     Train Loss: 0.4590     Train Accuracy: 87.8906      Validation Loss: 0.4893     Validation Accuracy: 82.1875
Patirnce: 0
Epoch: 12     Train Loss: 0.4537     Train Accuracy: 88.5156      Validation Loss: 0.4840     Validation Accuracy: 84.3750
Patirnce: 0
Epoch: 13     Train Loss: 0.4461     Train Accuracy: 89.3750      Validation Loss: 0.4775     Validation Accuracy: 84.6875
Patirnce: 0
Epoch: 14     Train Loss: 0.4404     Train Accuracy: 90.0000      Validation Loss: 0.4726     Validation Accuracy: 85.3125
Patirnce: 0
Epoch: 15     Train Loss: 0.4327     Train Accuracy: 91.2500      Validation Loss: 0.4679     Validation Accuracy: 85.0000
Patirnce: 0
```

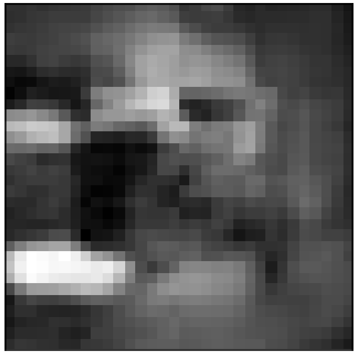
```
Epoch: 16     Train Loss: 0.4291     Train Accuracy: 91.5625      Validation Loss: 0.4604     Validation Accuracy: 87.5000
Patirnce: 0
Epoch: 17     Train Loss: 0.4232     Train Accuracy: 92.1875      Validation Loss: 0.4562     Validation Accuracy: 87.1875
Patirnce: 0
Epoch: 18     Train Loss: 0.4184     Train Accuracy: 92.8125      Validation Loss: 0.4520     Validation Accuracy: 88.1250
Patirnce: 0
Epoch: 19     Train Loss: 0.4128     Train Accuracy: 92.8906      Validation Loss: 0.4522     Validation Accuracy: 87.1875
Patirnce: 1
Epoch: 20     Train Loss: 0.4077     Train Accuracy: 93.3594      Validation Loss: 0.4475     Validation Accuracy: 88.7500
Patirnce: 0
Epoch: 21     Train Loss: 0.4051     Train Accuracy: 93.5156      Validation Loss: 0.4446     Validation Accuracy: 89.0625
Patirnce: 0
Epoch: 22     Train Loss: 0.4002     Train Accuracy: 94.0625      Validation Loss: 0.4450     Validation Accuracy: 88.4375
Patirnce: 1
Epoch: 23     Train Loss: 0.3992     Train Accuracy: 94.2969      Validation Loss: 0.4409     Validation Accuracy: 87.8125
Patirnce: 0
Epoch: 24     Train Loss: 0.3946     Train Accuracy: 94.6875      Validation Loss: 0.4381     Validation Accuracy: 89.3750
Patirnce: 0
Epoch: 25     Train Loss: 0.3920     Train Accuracy: 95.5469      Validation Loss: 0.4391     Validation Accuracy: 88.1250
Patirnce: 1
Epoch: 26     Train Loss: 0.3891     Train Accuracy: 95.2344      Validation Loss: 0.4347     Validation Accuracy: 89.0625
Patirnce: 0
Epoch: 27     Train Loss: 0.3842     Train Accuracy: 96.0156      Validation Loss: 0.4295     Validation Accuracy: 88.7500
Patirnce: 0
Epoch: 28     Train Loss: 0.3829     Train Accuracy: 95.9375      Validation Loss: 0.4283     Validation Accuracy: 89.3750
Patirnce: 0
Epoch: 29     Train Loss: 0.3814     Train Accuracy: 96.3281      Validation Loss: 0.4222     Validation Accuracy: 90.6250
Patirnce: 0
Epoch: 30     Train Loss: 0.3807     Train Accuracy: 95.9375      Validation Loss: 0.4297     Validation Accuracy: 90.0000
Patirnce: 1
Epoch: 31     Train Loss: 0.3772     Train Accuracy: 96.1719      Validation Loss: 0.4245     Validation Accuracy: 89.6875
Patirnce: 0
Epoch: 32     Train Loss: 0.3755     Train Accuracy: 96.7969      Validation Loss: 0.4261     Validation Accuracy: 90.0000
Patirnce: 1
Epoch: 33     Train Loss: 0.3722     Train Accuracy: 96.9531      Validation Loss: 0.4233     Validation Accuracy: 90.6250
Patirnce: 0
Epoch: 34     Train Loss: 0.3708     Train Accuracy: 96.9531      Validation Loss: 0.4201     Validation Accuracy: 91.2500
Patirnce: 0
Epoch: 35     Train Loss: 0.3702     Train Accuracy: 96.9531      Validation Loss: 0.4209     Validation Accuracy: 89.0625
Patirnce: 1
Epoch: 36     Train Loss: 0.3680     Train Accuracy: 97.0312      Validation Loss: 0.4171     Validation Accuracy: 91.2500
Patirnce: 0
Epoch: 37     Train Loss: 0.3670     Train Accuracy: 97.1094      Validation Loss: 0.4149     Validation Accuracy: 90.0000
Patirnce: 0
```

```
Epoch: 38     Train Loss: 0.3660     Train Accuracy: 97.1875      Validation Loss: 0.4202     Validation Accuracy: 89.0625
Patirnce: 1
Epoch: 39     Train Loss: 0.3632     Train Accuracy: 97.7344      Validation Loss: 0.4198     Validation Accuracy: 89.3750
Patirnce: 0
Epoch: 40     Train Loss: 0.3615     Train Accuracy: 97.7344      Validation Loss: 0.4214     Validation Accuracy: 89.3750
Patirnce: 1
Epoch: 41     Train Loss: 0.3596     Train Accuracy: 97.8125      Validation Loss: 0.4181     Validation Accuracy: 90.6250
Patirnce: 0
Epoch: 42     Train Loss: 0.3601     Train Accuracy: 97.4219      Validation Loss: 0.4157     Validation Accuracy: 90.0000
Patirnce: 0
Epoch: 43     Train Loss: 0.3559     Train Accuracy: 97.8906      Validation Loss: 0.4125     Validation Accuracy: 90.6250
Patirnce: 0
Epoch: 44     Train Loss: 0.3554     Train Accuracy: 98.1250      Validation Loss: 0.4132     Validation Accuracy: 90.0000
Patirnce: 1
Epoch: 45     Train Loss: 0.3547     Train Accuracy: 98.1250      Validation Loss: 0.4090     Validation Accuracy: 91.2500
Patirnce: 0
Epoch: 46     Train Loss: 0.3524     Train Accuracy: 98.2812      Validation Loss: 0.4100     Validation Accuracy: 90.3125
Patirnce: 1
Epoch: 47     Train Loss: 0.3518     Train Accuracy: 98.3594      Validation Loss: 0.4111     Validation Accuracy: 89.6875
Patirnce: 2
Early stopping

Start of the testing process.

There are 400 test samples

After carring out the training with Bilinear dataset and testing the model with Bicubic we obtain:
Test Loss: 0.6225      Test Accuracy: 65.2500
```

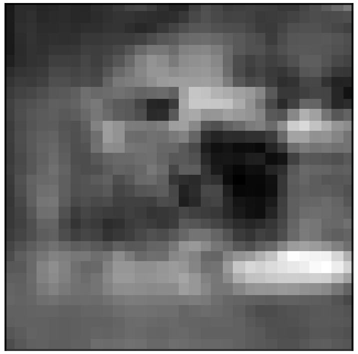
Reality: Real image



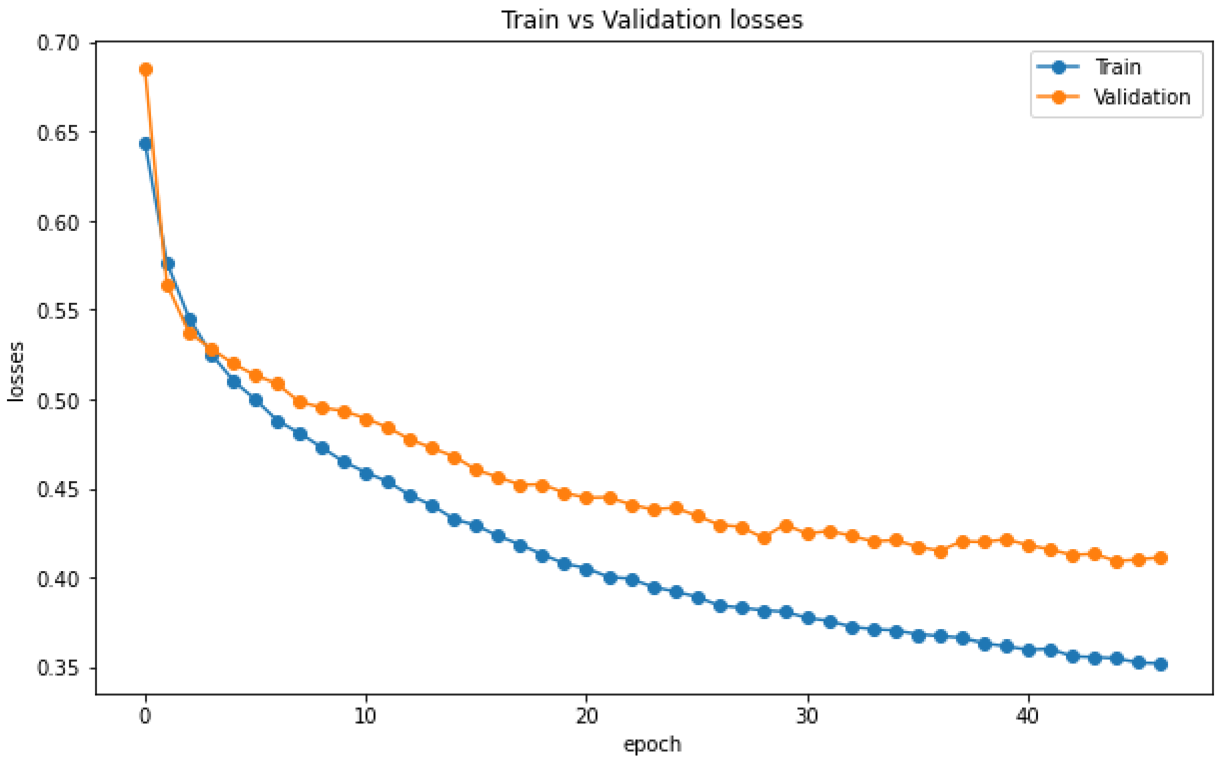
Reality: Real image

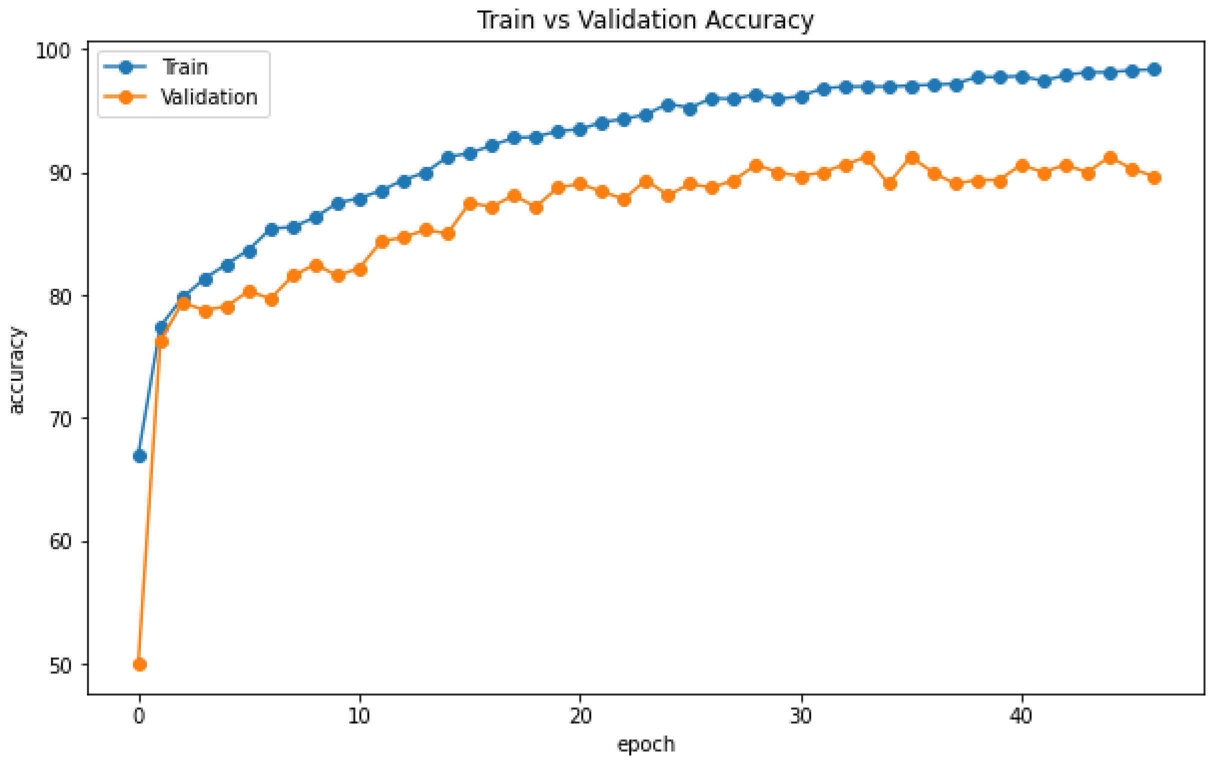


Reality: Real image

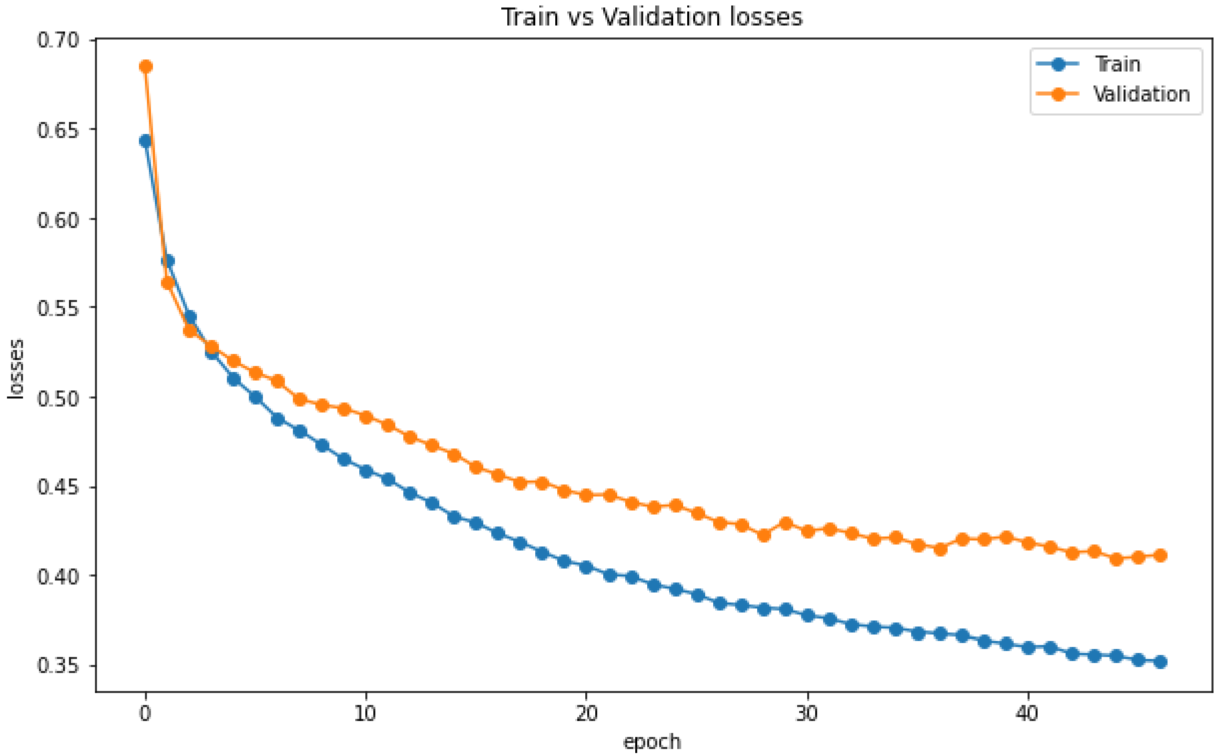


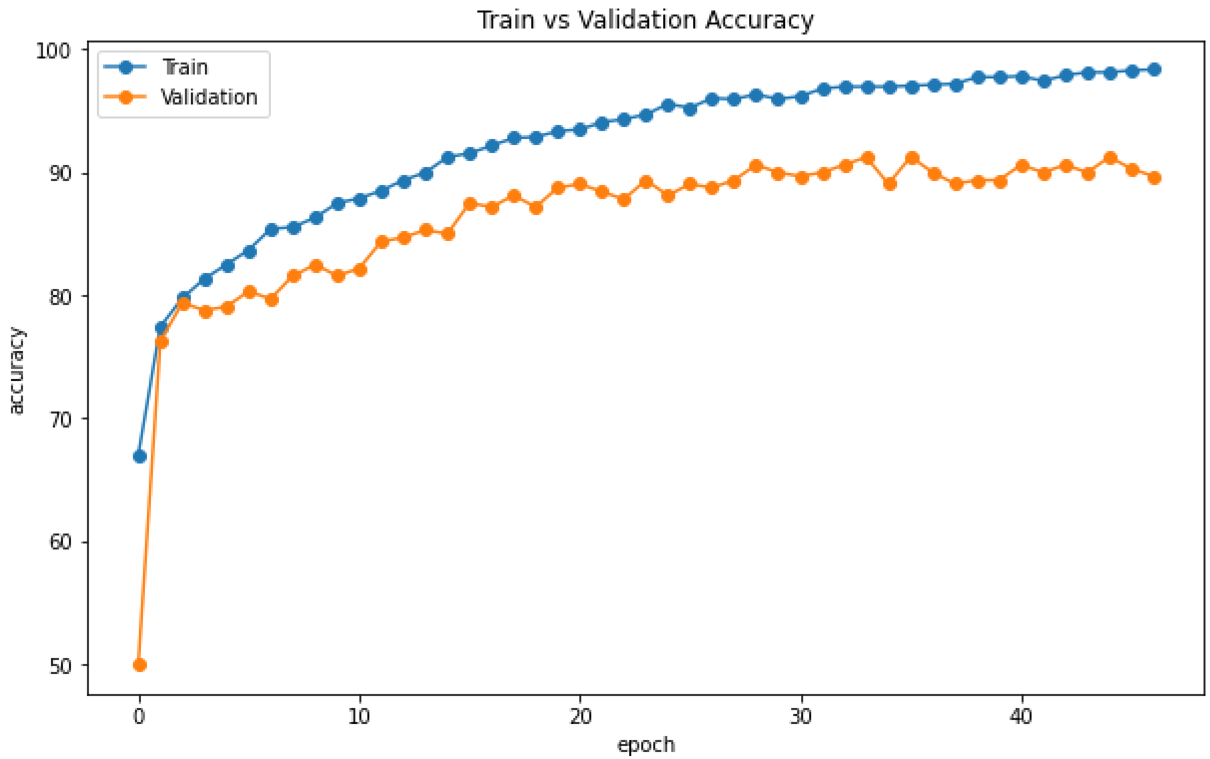
Reality: Fake image


Train vs Validation losses
Train
Validation
losses
epoch
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
10
20
30
40

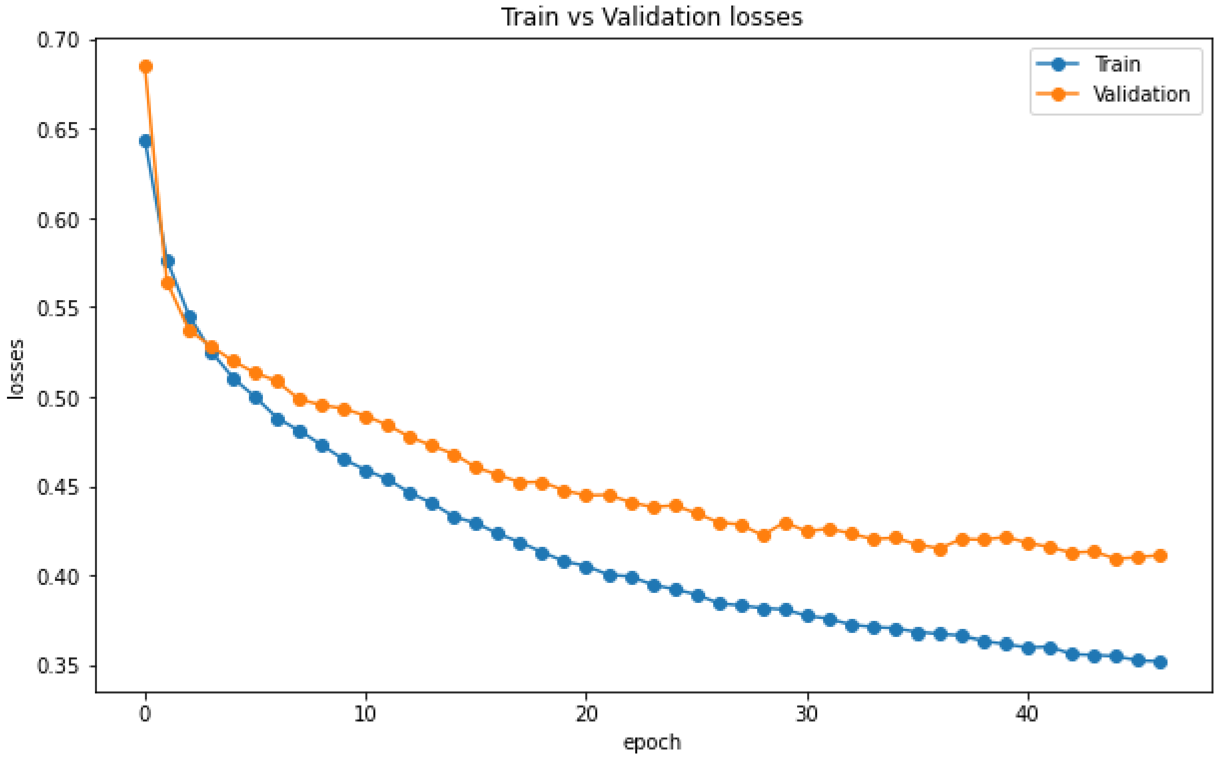


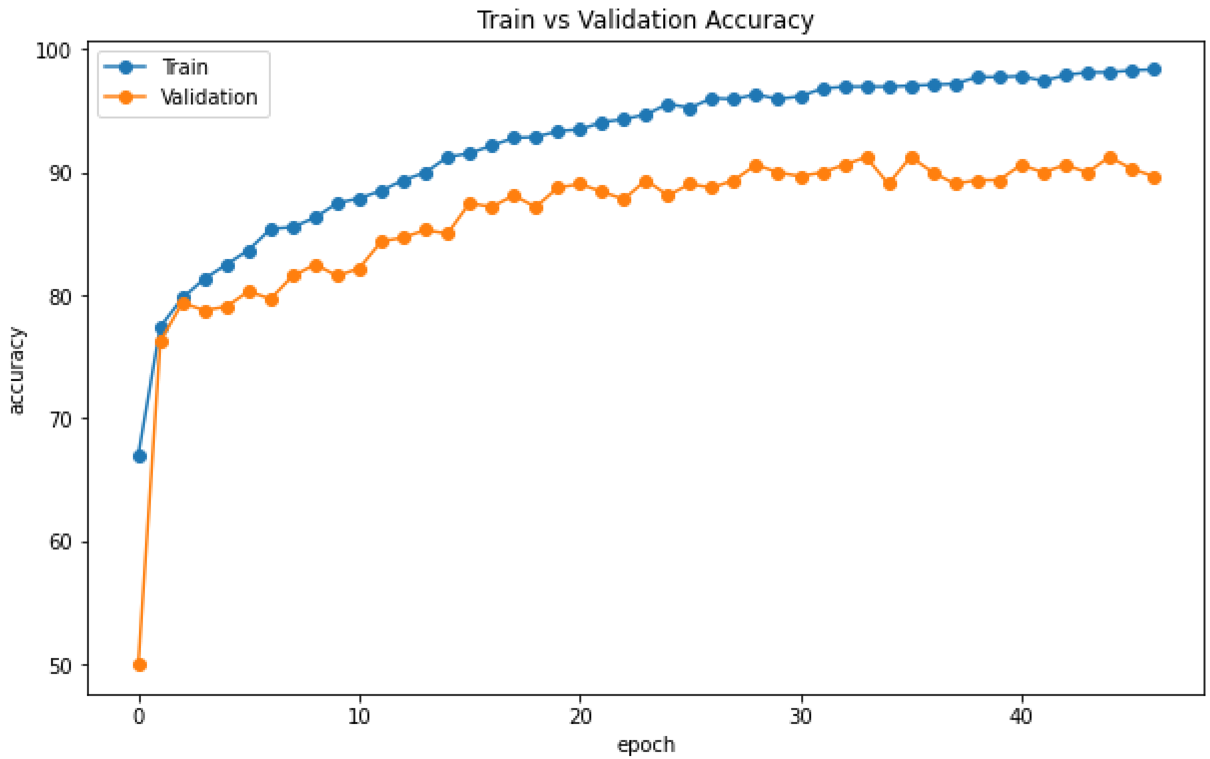

```
There are 400 test samples

After carring out the training with Bilinear dataset and testing the model with Bilinear we obtain:
Test Loss: 0.4166      Test Accuracy: 89.2500
```


Train vs Validation losses
Train
Validation
losses
epoch
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
10
20
30
40

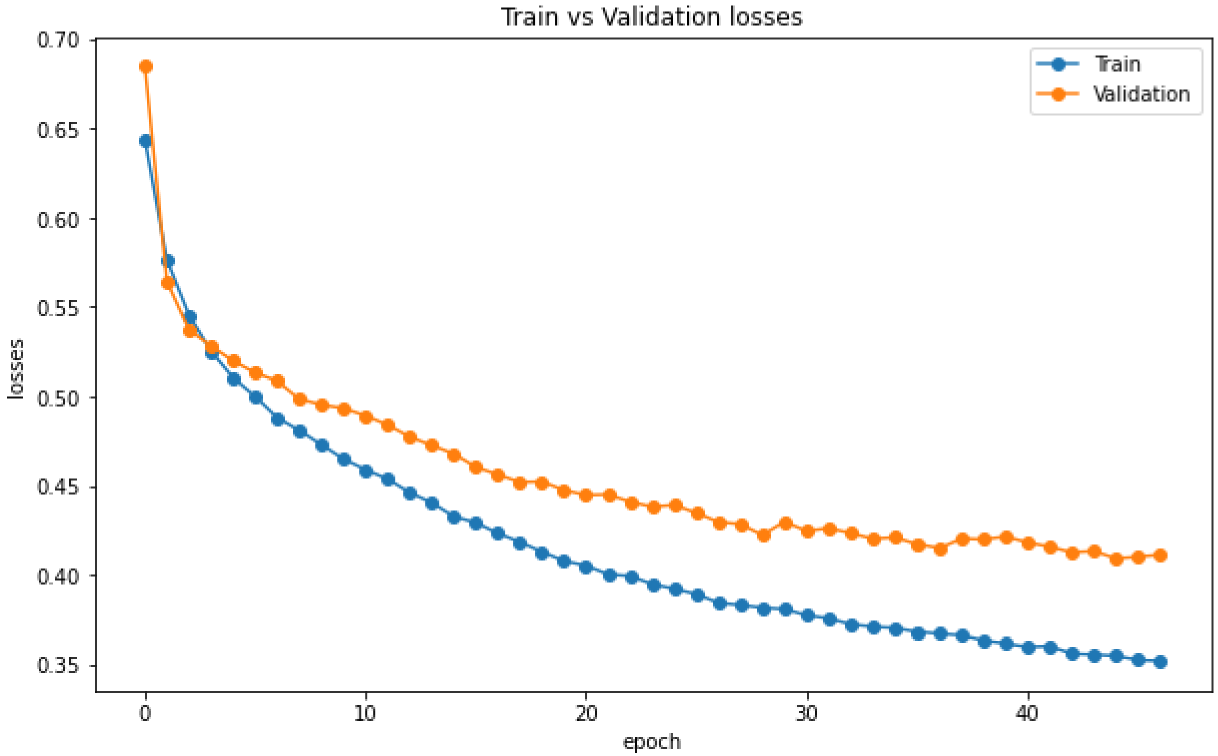


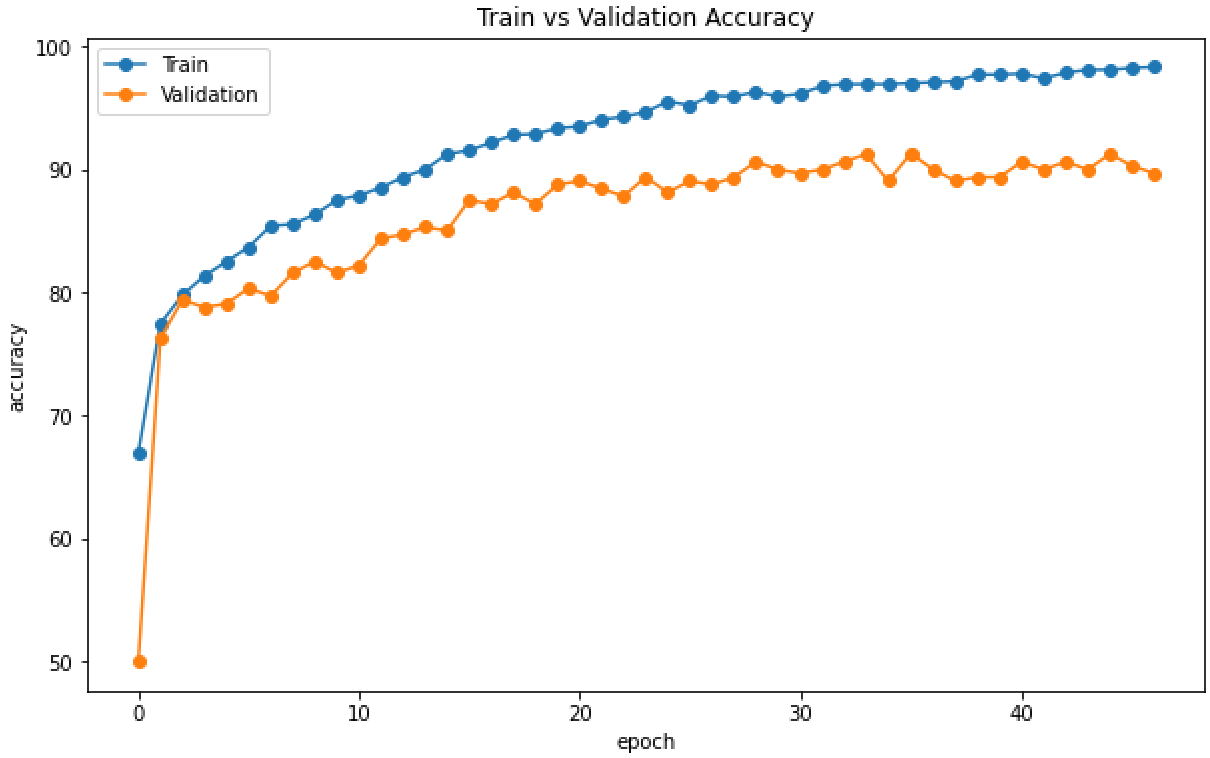

```
There are 400 test samples

After carring out the training with Bilinear dataset and testing the model with Pixel shuffle we obtain:
Test Loss: 0.6130      Test Accuracy: 68.0000
```


Train vs Validation losses
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
losses
0
10
20
30
40
epoch
Train
Validation




```
There are 800 test samples

After carring out the training with Bilinear dataset and testing the model with Combined data we obtain:
Test Loss: 0.9366      Test Accuracy: 32.2500
```

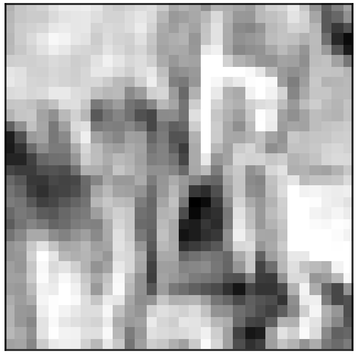

Train vs Validation losses
Train
Validation
losses
epoch
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
10
20
30
40


Train vs Validation Accuracy
Train
Validation
accuracy
epoch
100
90
80
70
60
50
0
10
20
30
40

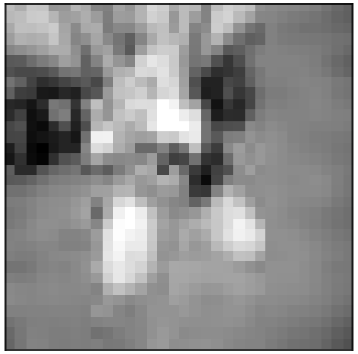
```
There are 1280 train samples
There are 320 validation samples

The Classification is as follows
  Label  Image Classification
-------  ----------------------
      0  Real Image
      1  Fake Image

The loaded batch of the images is a tensor with size:  torch.Size([40, 3, 32, 32])
The loaded batch of the images is a tensor with labels:  [0 1 1 0]


cpu
Epoch: 1      Train Loss: 0.6273     Train Accuracy: 68.5156      Validation Loss: 0.6833     Validation Accuracy: 50.0000
Patirnce: 0
Epoch: 2      Train Loss: 0.5362     Train Accuracy: 80.7812      Validation Loss: 0.5301     Validation Accuracy: 80.3125
Patirnce: 0
Epoch: 3      Train Loss: 0.5017     Train Accuracy: 82.9688      Validation Loss: 0.5069     Validation Accuracy: 80.9375
Patirnce: 0
Epoch: 4      Train Loss: 0.4851     Train Accuracy: 83.9062      Validation Loss: 0.4899     Validation Accuracy: 82.1875
Patirnce: 0
Epoch: 5      Train Loss: 0.4751     Train Accuracy: 84.5312      Validation Loss: 0.4799     Validation Accuracy: 83.7500
Patirnce: 0
Epoch: 6      Train Loss: 0.4666     Train Accuracy: 85.7031      Validation Loss: 0.4759     Validation Accuracy: 84.3750
Patirnce: 0
Epoch: 7      Train Loss: 0.4617     Train Accuracy: 86.0156      Validation Loss: 0.4719     Validation Accuracy: 85.0000
Patirnce: 0
Epoch: 8      Train Loss: 0.4564     Train Accuracy: 86.7188      Validation Loss: 0.4702     Validation Accuracy: 85.0000
Patirnce: 0
Epoch: 9      Train Loss: 0.4527     Train Accuracy: 87.1094      Validation Loss: 0.4699     Validation Accuracy: 85.3125
Patirnce: 0
Epoch: 10     Train Loss: 0.4492     Train Accuracy: 87.4219      Validation Loss: 0.4631     Validation Accuracy: 85.0000
Patirnce: 0
Epoch: 11     Train Loss: 0.4452     Train Accuracy: 87.5781      Validation Loss: 0.4620     Validation Accuracy: 85.0000
Patirnce: 0
Epoch: 12     Train Loss: 0.4417     Train Accuracy: 88.1250      Validation Loss: 0.4619     Validation Accuracy: 84.3750
Patirnce: 0
Epoch: 13     Train Loss: 0.4415     Train Accuracy: 87.9688      Validation Loss: 0.4618     Validation Accuracy: 85.3125
Patirnce: 0
Epoch: 14     Train Loss: 0.4369     Train Accuracy: 88.4375      Validation Loss: 0.4598     Validation Accuracy: 85.3125
Patirnce: 0
Epoch: 15     Train Loss: 0.4341     Train Accuracy: 89.1406      Validation Loss: 0.4557     Validation Accuracy: 85.9375
Patirnce: 0
```

```
Epoch: 16     Train Loss: 0.4329     Train Accuracy: 88.5938      Validation Loss: 0.4589     Validation Accuracy: 85.0000
Patirnce: 1
Epoch: 17     Train Loss: 0.4304     Train Accuracy: 89.3750      Validation Loss: 0.4571     Validation Accuracy: 86.5625
Patirnce: 0
Epoch: 18     Train Loss: 0.4272     Train Accuracy: 89.7656      Validation Loss: 0.4596     Validation Accuracy: 85.0000
Patirnce: 1
Epoch: 19     Train Loss: 0.4265     Train Accuracy: 90.0000      Validation Loss: 0.4580     Validation Accuracy: 85.0000
Patirnce: 0
Epoch: 20     Train Loss: 0.4236     Train Accuracy: 90.0000      Validation Loss: 0.4591     Validation Accuracy: 84.6875
Patirnce: 1
Epoch: 21     Train Loss: 0.4222     Train Accuracy: 90.2344      Validation Loss: 0.4556     Validation Accuracy: 85.6250
Patirnce: 0
Epoch: 22     Train Loss: 0.4197     Train Accuracy: 90.3906      Validation Loss: 0.4582     Validation Accuracy: 85.6250
Patirnce: 1
Epoch: 23     Train Loss: 0.4183     Train Accuracy: 90.2344      Validation Loss: 0.4574     Validation Accuracy: 85.9375
Patirnce: 0
Epoch: 24     Train Loss: 0.4173     Train Accuracy: 90.3125      Validation Loss: 0.4554     Validation Accuracy: 85.9375
Patirnce: 0
Epoch: 25     Train Loss: 0.4144     Train Accuracy: 90.5469      Validation Loss: 0.4593     Validation Accuracy: 85.3125
Patirnce: 1
Epoch: 26     Train Loss: 0.4117     Train Accuracy: 90.9375      Validation Loss: 0.4559     Validation Accuracy: 85.3125
Patirnce: 0
Epoch: 27     Train Loss: 0.4105     Train Accuracy: 91.5625      Validation Loss: 0.4612     Validation Accuracy: 85.0000
Patirnce: 1
Epoch: 28     Train Loss: 0.4088     Train Accuracy: 91.7969      Validation Loss: 0.4568     Validation Accuracy: 85.6250
Patirnce: 0
Epoch: 29     Train Loss: 0.4070     Train Accuracy: 91.8750      Validation Loss: 0.4519     Validation Accuracy: 87.1875
Patirnce: 0
Epoch: 30     Train Loss: 0.4053     Train Accuracy: 92.1094      Validation Loss: 0.4531     Validation Accuracy: 87.1875
Patirnce: 1
Epoch: 31     Train Loss: 0.4032     Train Accuracy: 92.7344      Validation Loss: 0.4598     Validation Accuracy: 84.6875
Patirnce: 2
Early stopping

Start of the testing process.

There are 400 test samples

After carring out the training with Pixel shuffle dataset and testing the model with Bicubic we obtain:
Test Loss: 0.5395      Test Accuracy: 76.5000
```

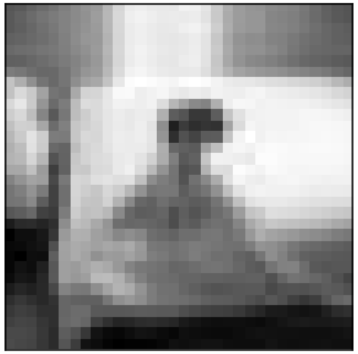
Reality: Real image



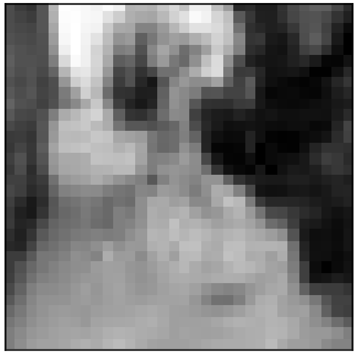
Reality: Fake image

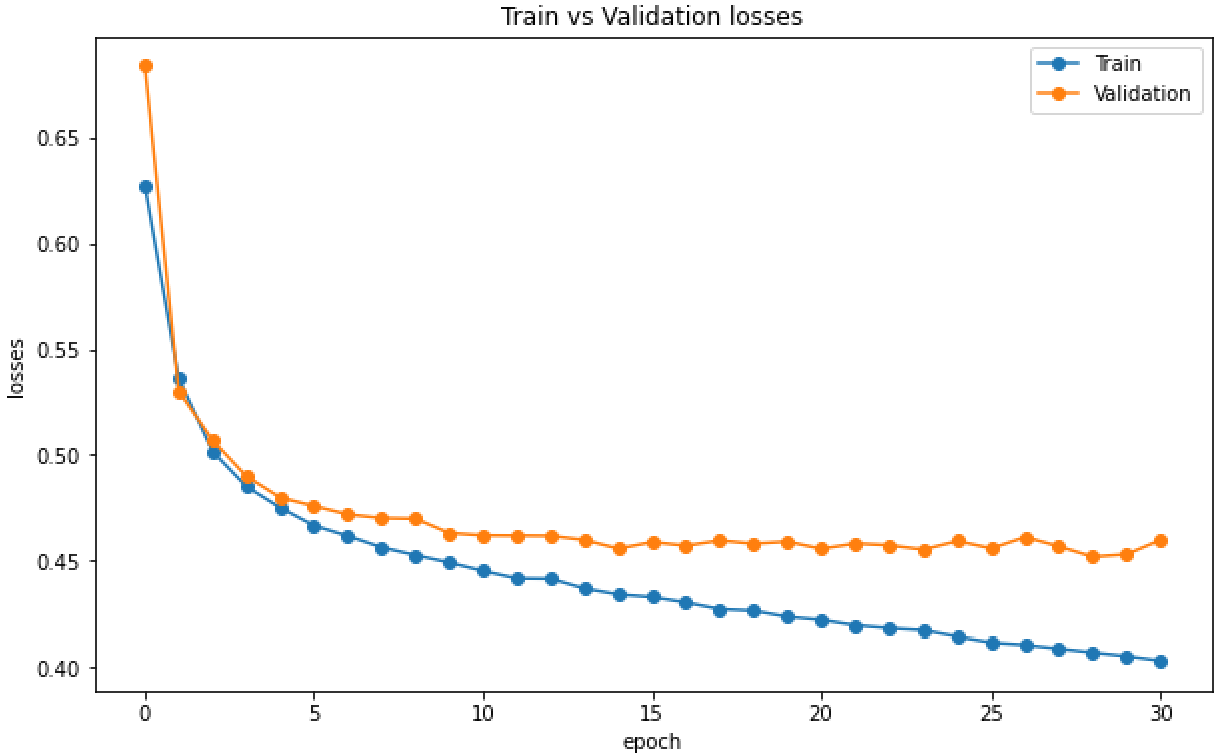


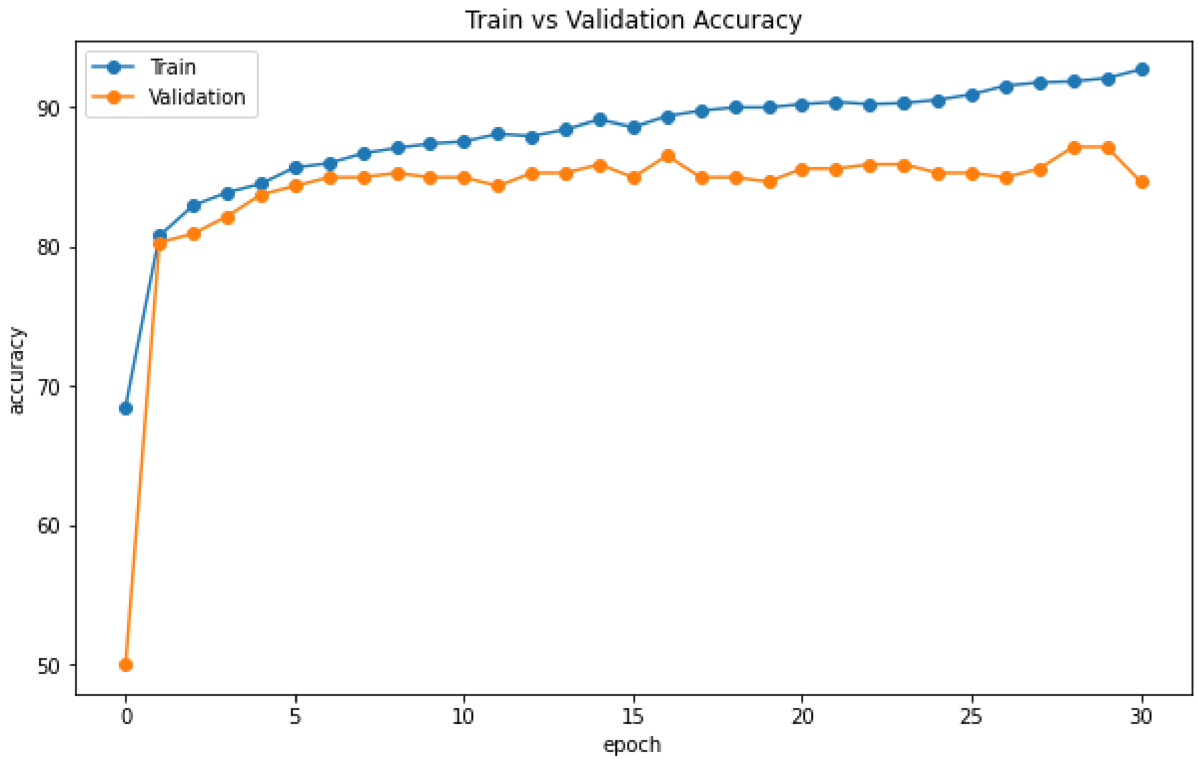
Reality: Fake image



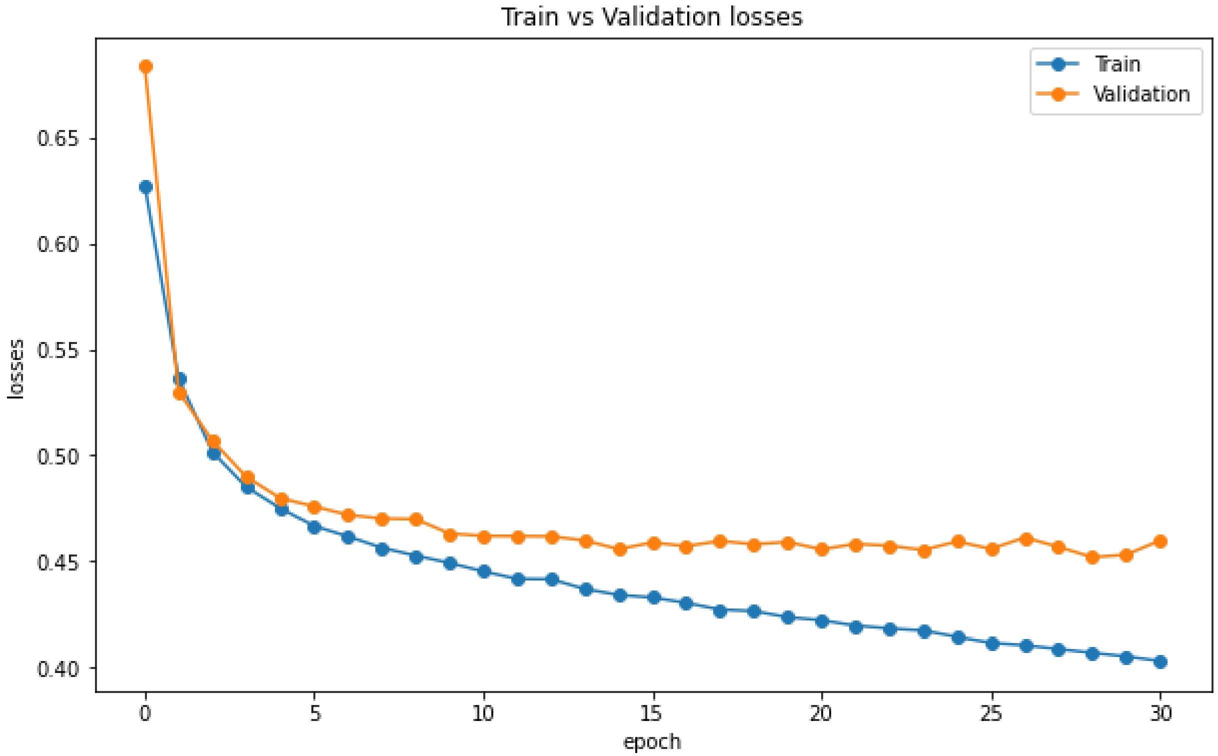
Reality: Real image

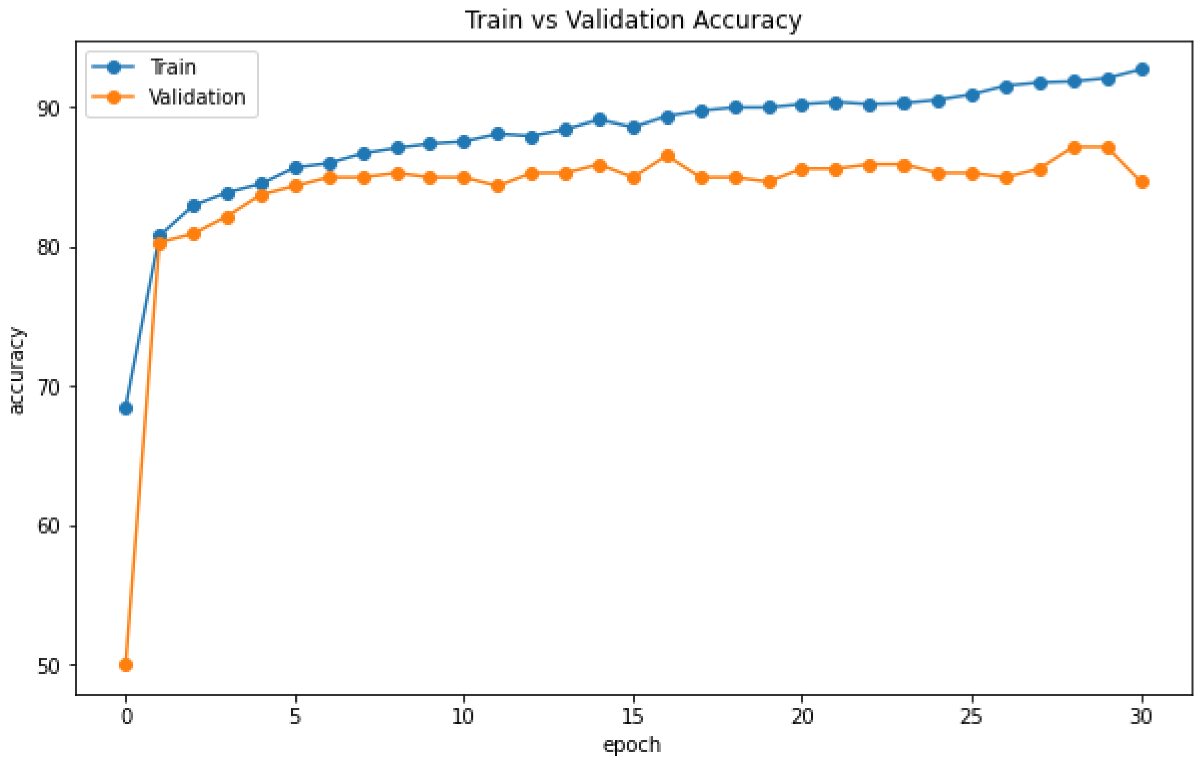



Train vs Validation losses
Train
Validation
losses
0.65
0.60
0.55
0.50
0.45
0.40
0
5
10
15
20
25
30
epoch

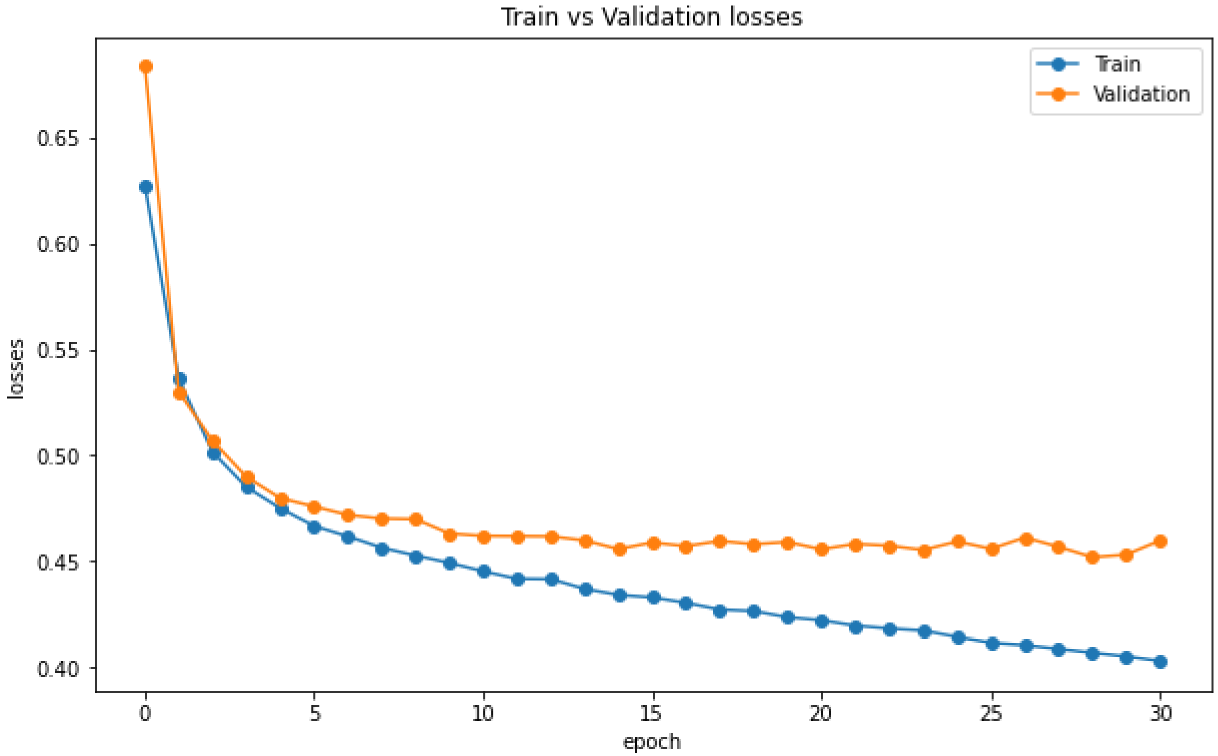


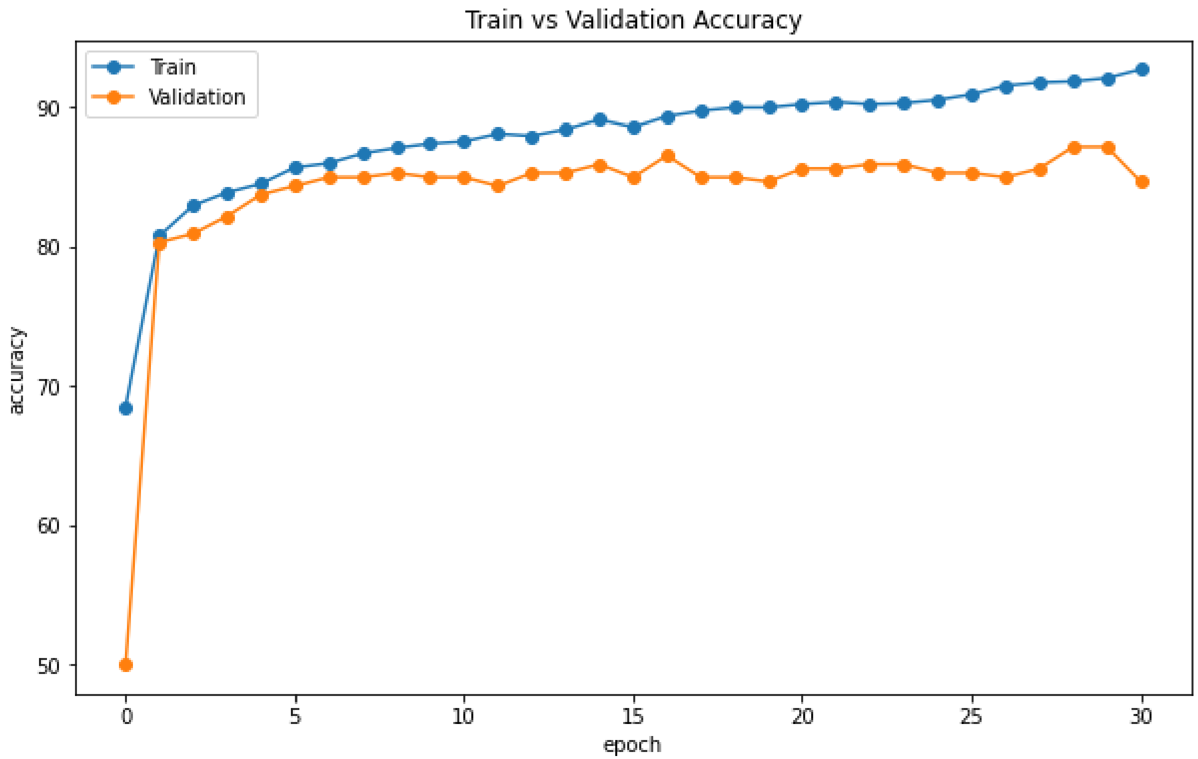

```
There are 400 test samples

After carring out the training with Pixel shuffle dataset and testing the model with Bilinear we obtain:
Test Loss: 0.6083      Test Accuracy: 66.7500
```


Train vs Validation losses
Train
Validation
0.65
0.60
0.55
0.50
0.45
0.40
losses
0
5
10
15
20
25
30
epoch

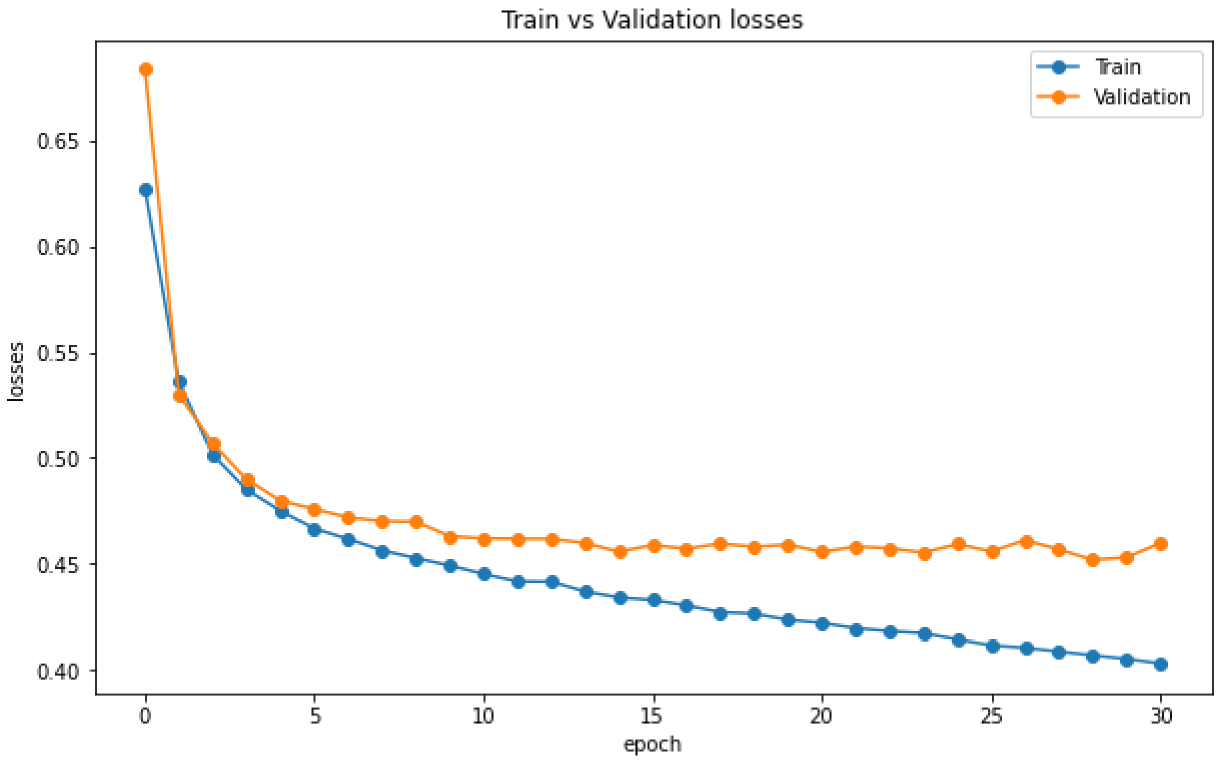


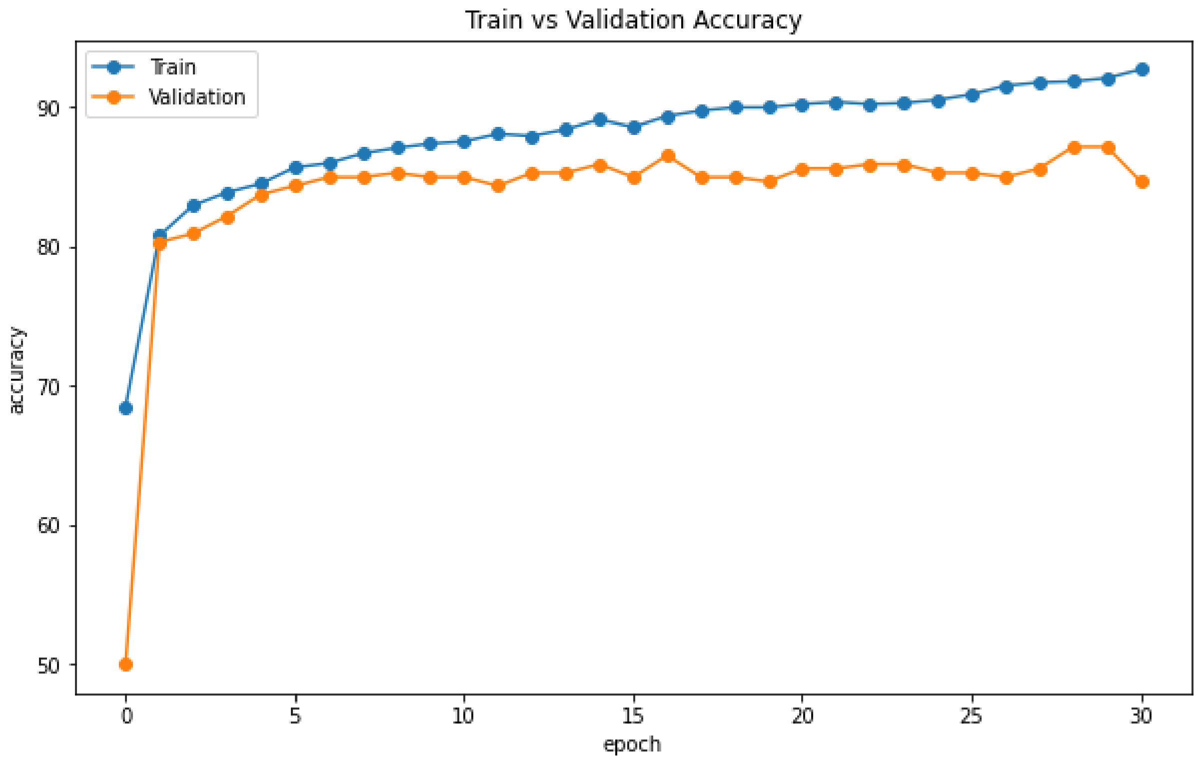

```
There are 400 test samples

After carring out the training with Pixel shuffle dataset and testing the model with Pixel shuffle we obtain:
Test Loss: 0.5025      Test Accuracy: 80.5000
```


Train vs Validation losses
Train
Validation
losses
0.65
0.60
0.55
0.50
0.45
0.40
0
5
10
15
20
25
30
epoch



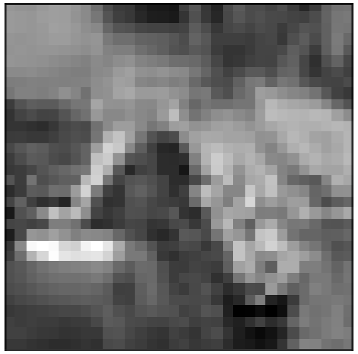

```
There are 800 test samples

After carring out the training with Pixel shuffle dataset and testing the model with Combined data we obtain:
Test Loss: 1.0161      Test Accuracy: 24.8750
```


Train vs Validation losses
Train
Validation
losses
0.40
0.45
0.50
0.55
0.60
0.65
0
5
10
15
20
25
30
epoch

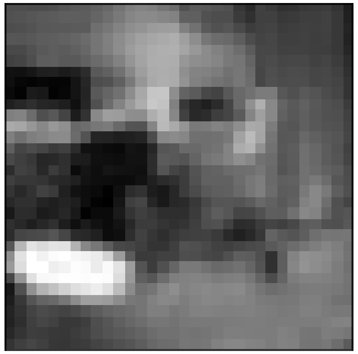

Train vs Validation Accuracy
Train
Validation
accuracy
epoch
90
80
70
60
50
0
5
10
15
20
25
30

```
There are 2560 train samples
There are 640 validation samples

The Classification is as follows
  Label  Image Classification
-------  ----------------------
      0  Real Image
      1  Fake Image

The loaded batch of the images is a tensor with size:  torch.Size([40, 3, 32, 32])
The loaded batch of the images is a tensor with labels:  [1 1 0 1]


cpu
Epoch: 1      Train Loss: 0.5569     Train Accuracy: 75.1953      Validation Loss: 0.5241     Validation Accuracy: 79.0625
Patirnce: 0
Epoch: 2      Train Loss: 0.5110     Train Accuracy: 79.4141      Validation Loss: 0.4917     Validation Accuracy: 81.2500
Patirnce: 0
Epoch: 3      Train Loss: 0.4891     Train Accuracy: 82.5391      Validation Loss: 0.4843     Validation Accuracy: 81.7188
Patirnce: 0
Epoch: 4      Train Loss: 0.4752     Train Accuracy: 84.3359      Validation Loss: 0.4768     Validation Accuracy: 83.5938
Patirnce: 0
Epoch: 5      Train Loss: 0.4672     Train Accuracy: 84.8438      Validation Loss: 0.4735     Validation Accuracy: 83.7500
Patirnce: 0
Epoch: 6      Train Loss: 0.4599     Train Accuracy: 85.6641      Validation Loss: 0.4690     Validation Accuracy: 83.4375
Patirnce: 0
Epoch: 7      Train Loss: 0.4553     Train Accuracy: 85.8984      Validation Loss: 0.4666     Validation Accuracy: 83.7500
Patirnce: 0
Epoch: 8      Train Loss: 0.4495     Train Accuracy: 86.6797      Validation Loss: 0.4647     Validation Accuracy: 84.2188
Patirnce: 0
Epoch: 9      Train Loss: 0.4455     Train Accuracy: 87.1094      Validation Loss: 0.4627     Validation Accuracy: 83.9062
Patirnce: 0
Epoch: 10     Train Loss: 0.4395     Train Accuracy: 88.3594      Validation Loss: 0.4631     Validation Accuracy: 83.7500
Patirnce: 1
Epoch: 11     Train Loss: 0.4368     Train Accuracy: 88.2422      Validation Loss: 0.4632     Validation Accuracy: 84.0625
Patirnce: 2
Early stopping

Start of the testing process.

There are 400 test samples

After carring out the training with Combined data dataset and testing the model with Bicubic we obtain:
Test Loss: 1.0110      Test Accuracy: 25.0000
```

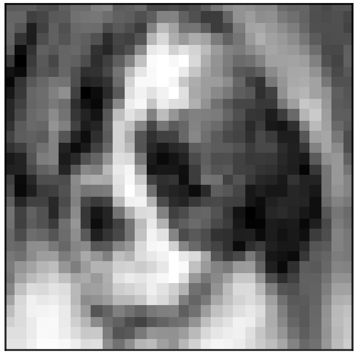
Reality: Fake image



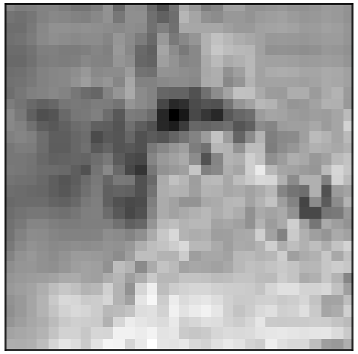
Reality: Fake image

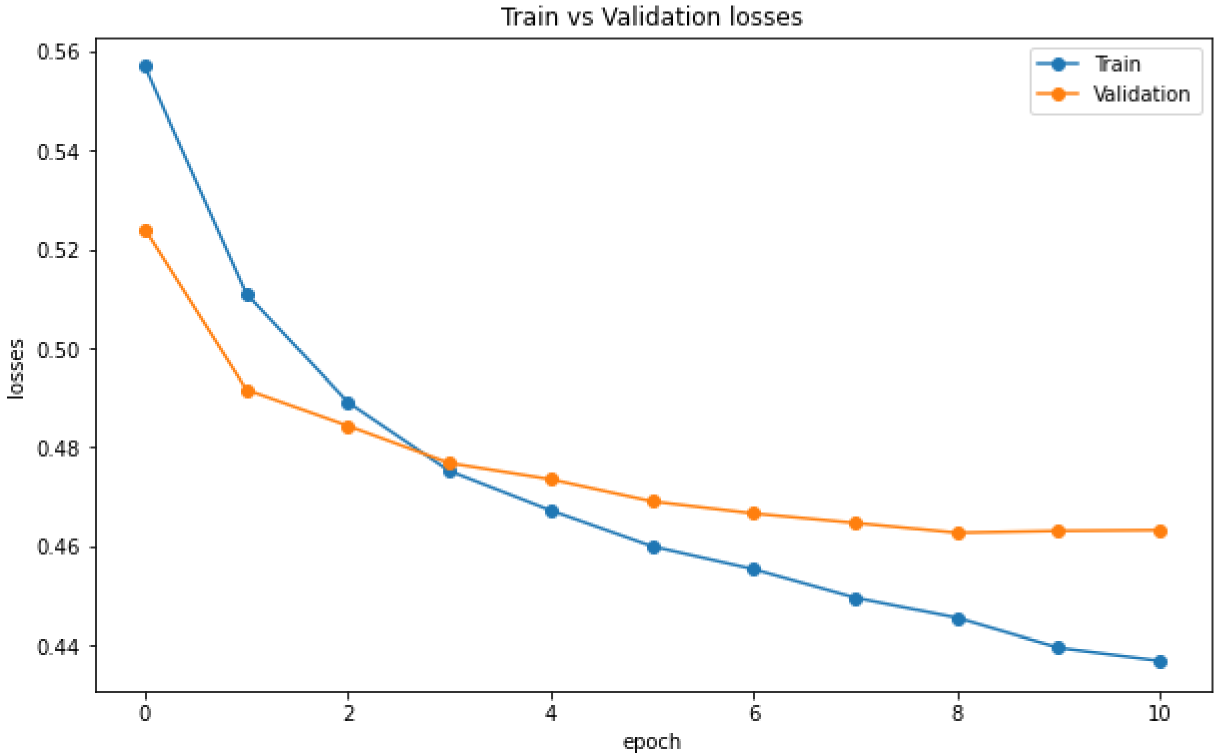


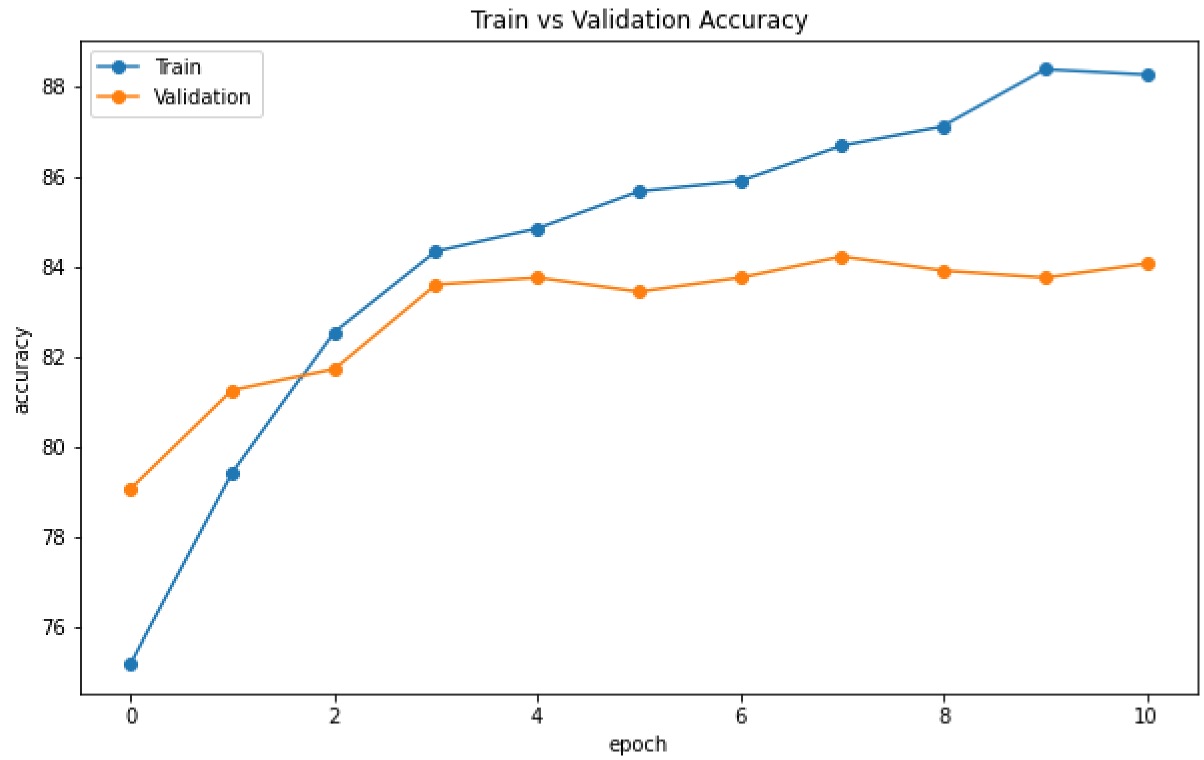
Reality: Real image



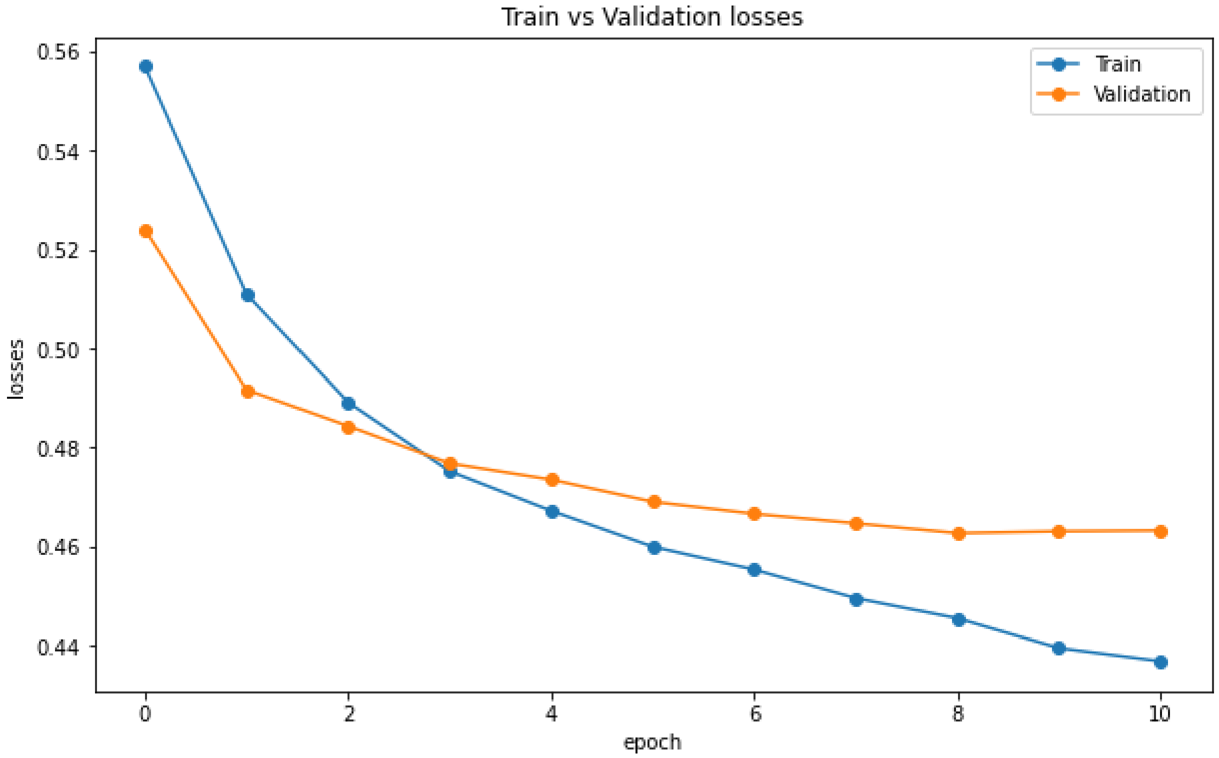
Reality: Fake image

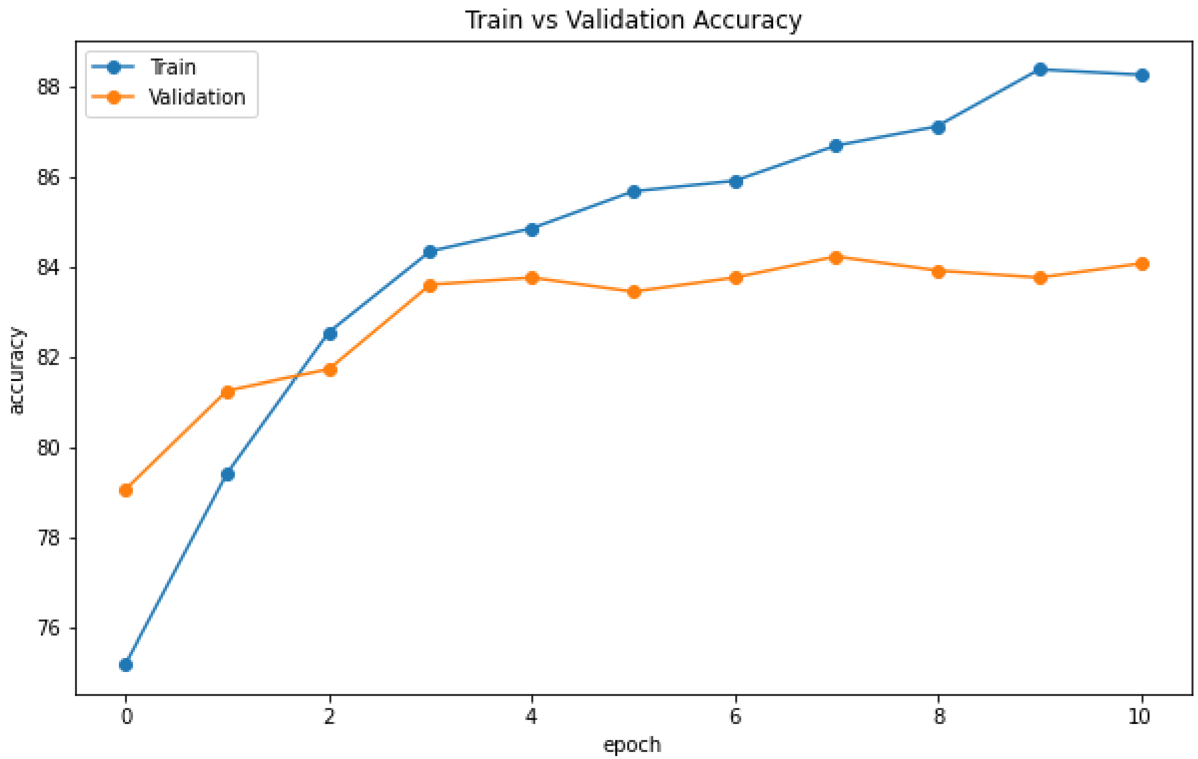



Train vs Validation losses
Train
Validation
0.56
0.54
0.52
0.50
0.48
0.46
0.44
losses
0
2
4
6
8
10
epoch

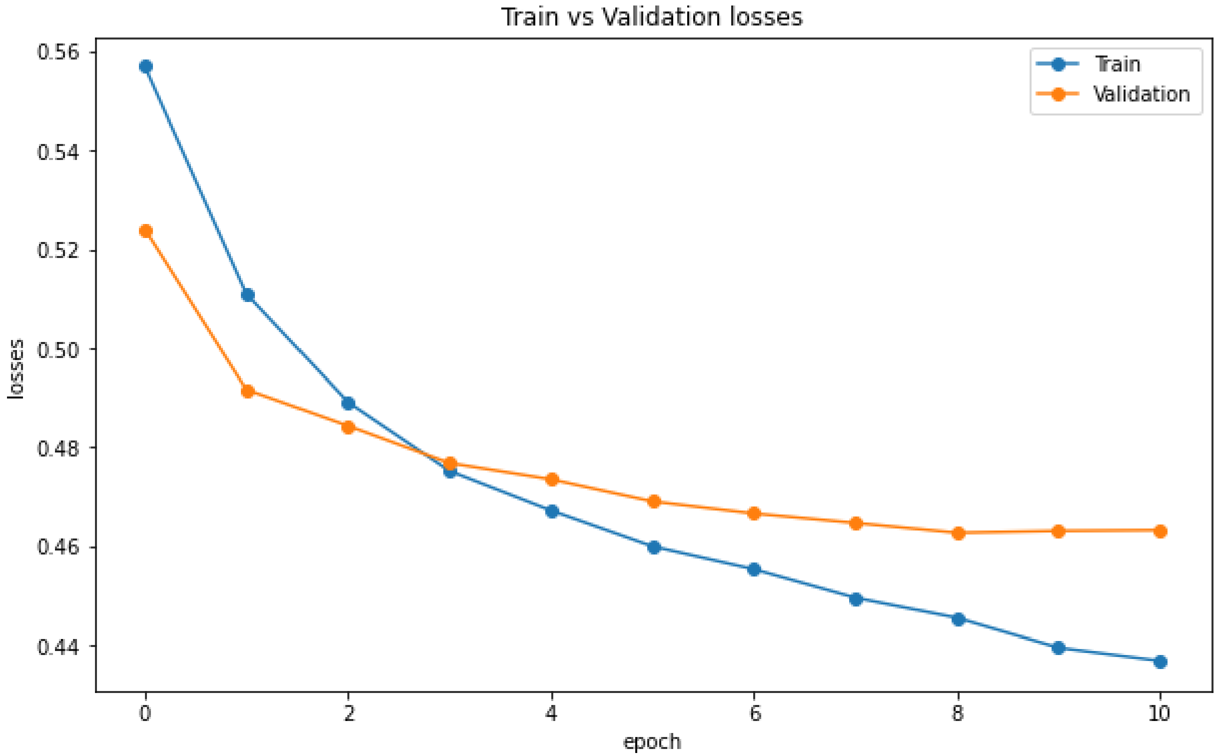


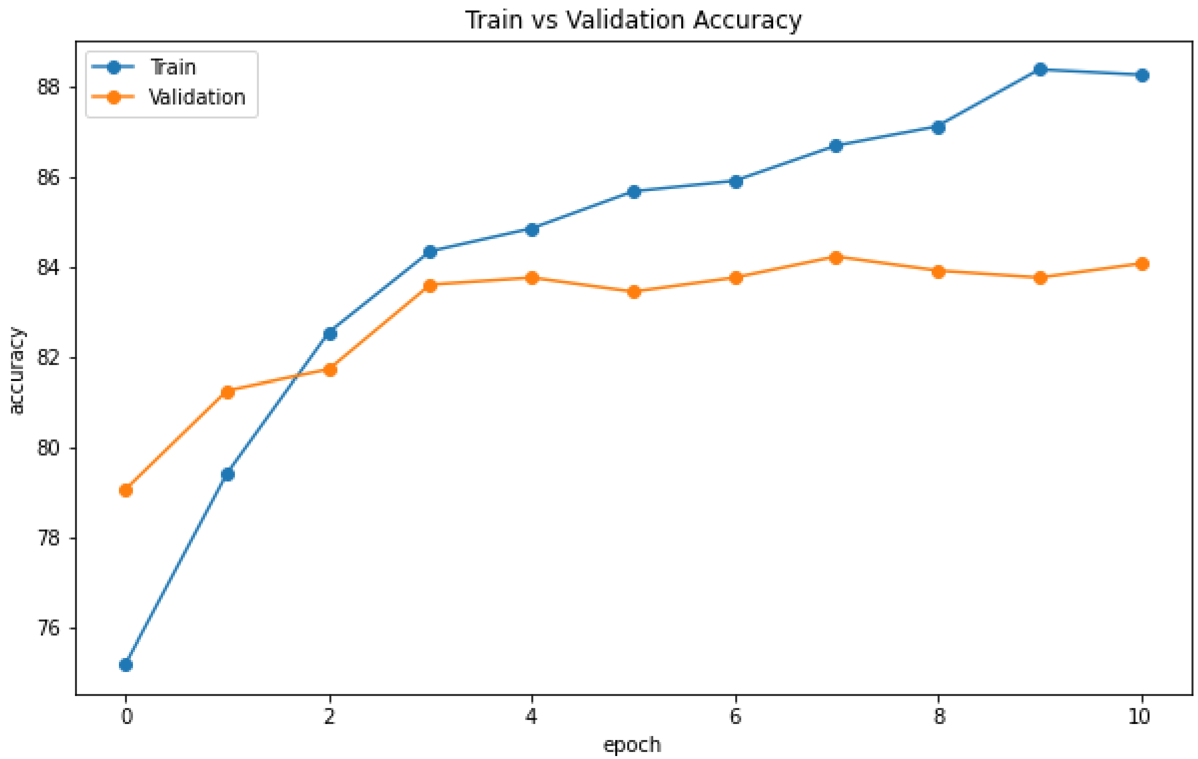

```
There are 400 test samples

After carring out the training with Combined data dataset and testing the model with Bilinear we obtain:
Test Loss: 1.0186      Test Accuracy: 24.2500
```


Train vs Validation losses
Train
Validation
losses
epoch
0.56
0.54
0.52
0.50
0.48
0.46
0.44
0
2
4
6
8
10

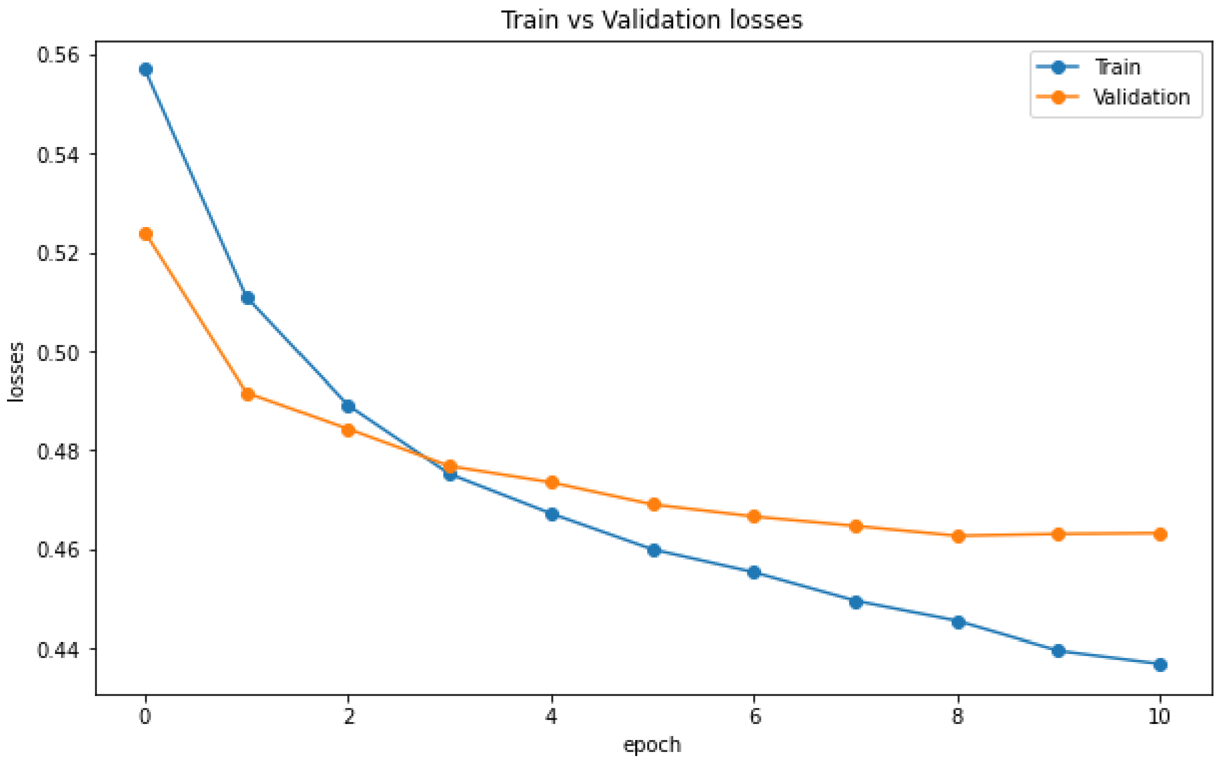


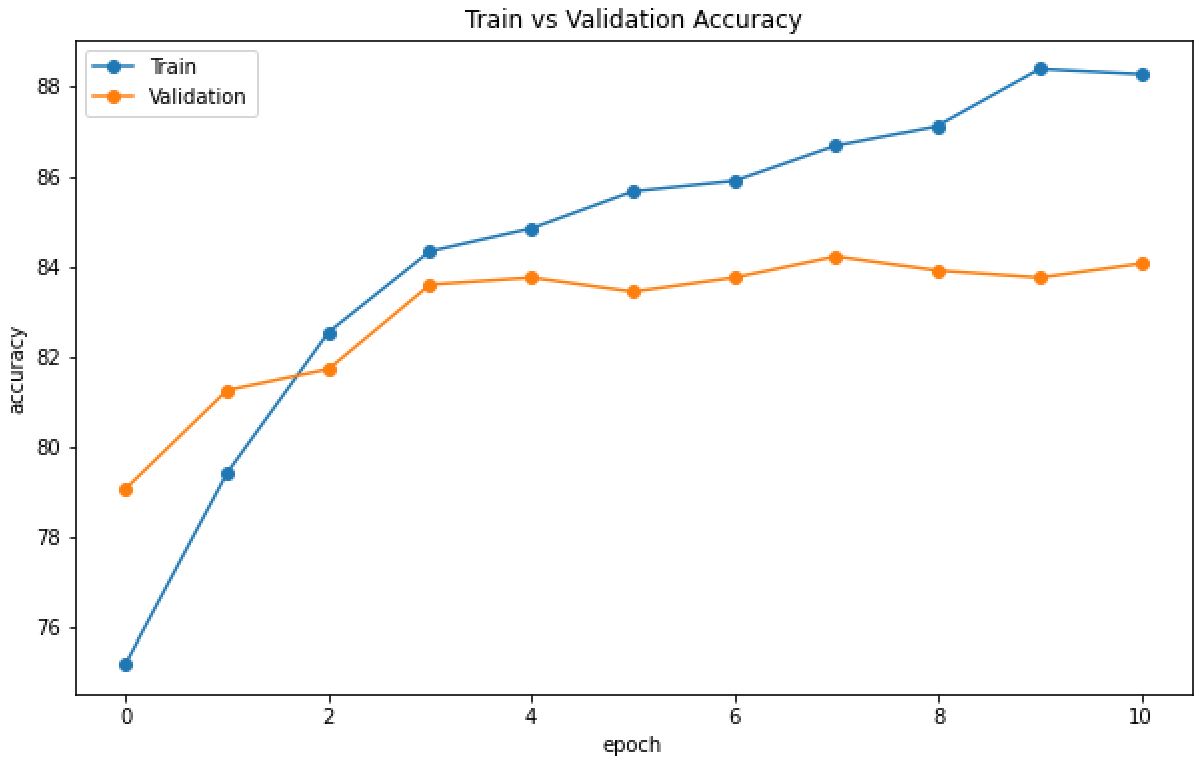

```
There are 400 test samples

After carring out the training with Combined data dataset and testing the model with Pixel shuffle we obtain:
Test Loss: 1.0135      Test Accuracy: 24.0000
```


Train vs Validation losses
0.56
0.54
0.52
0.50
0.48
0.46
0.44
losses
0
2
4
6
8
10
epoch
Train
Validation




```
There are 800 test samples

After carring out the training with Combined data dataset and testing the model with Combined data we obtain:
Test Loss: 0.4651      Test Accuracy: 84.1250
```


Train vs Validation losses
Train
Validation
0.56
0.54
0.52
0.50
0.48
0.46
0.44
losses
0
2
4
6
8
10
epoch



In [5]:

```
#representation of the losses and accuracies in a table

table_3 = ['Bicubic','Bicubic','Bicubic','Bicubic','Bilinear','Bilinear','Bilinear','Bilinear','Pixel shuffle','Pixel shuffle','I
table_4 = ['Bicubic','Bilinear','Pixel shuffle','Combined data','Bicubic','Bilinear','Pixel shuffle','Combined data','Bicubic','I

print('\nFinal Conclusion\n\n')

final_table = []

for i in range(16):
    final_table.append([table_3[i],table_4[i],combo_train_losses[i],combo_train_accu[i],combo_test_losses[i],combo_test_accu[i]]
#print(final_table)

header = ["Training Data","Test Data", "Training Loss", "Training Accuracy", "Testing Loss", "Testing Accuracy"]
print(tabulate(final_table, header))

table_3 = ['Bicubic','Bicubic','Bicubic','Bicubic','Bilinear','Bilinear','Bilinear','Bilinear','Pixel shuffle','Pixel shuffle','I
table_4 = ['Bicubic','Bilinear','Pixel shuffle','Combined data','Bicubic','Bilinear','Pixel shuffle','Combined data','Bicubic','I
```

```
print('\n\n')

final_table_2 = []

for i in range(16):
    final_table_2.append([table_3[i],table_4[i],combo_validation_losses[i],combo_validation_accu[i]])
#print(final_table)

header_2 = ["Training Data","Test Data","Validation Loss", "Validation Accuracy"]
print(tabulate(final_table_2, header_2))
```

Final Conclusion

Training Data	Test Data	Training Loss	Training Accuracy	Testing Loss	Testing Accuracy
Bicubic	Bicubic	0.455224	86.9531	0.495277	82.25
Bicubic	Bilinear	0	0	0.570726	75.75
Bicubic	Pixel shuffle	0	0	0.508732	80.25
Bicubic	Combined data	0	0	1.01208	19.875
Bilinear	Bicubic	0	0	0.622479	65.25
Bilinear	Bilinear	0.35182	98.3594	0.416583	89.25
Bilinear	Pixel shuffle	0	0	0.612952	68
Bilinear	Combined data	0	0	0.936553	32.25
Pixel shuffle	Bicubic	0	0	0.539475	76.5
Pixel shuffle	Bilinear	0	0	0.608277	66.75
Pixel shuffle	Pixel shuffle	0.403154	92.7344	0.502521	80.5
Pixel shuffle	Combined data	0	0	1.01606	24.875
Combined data	Bicubic	0	0	1.01103	25
Combined data	Bilinear	0	0	1.01856	24.25
Combined data	Pixel shuffle	0	0	1.01347	24
Combined data	Combined data	0.436809	88.2422	0.465142	84.125

Training Data	Test Data	Validation Loss	Validation Accuracy
Bicubic	Bicubic	0.471251	84.6875
Bicubic	Bilinear	0	0
Bicubic	Pixel shuffle	0	0
Bicubic	Combined data	0	0
Bilinear	Bicubic	0	0
Bilinear	Bilinear	0.411141	89.6875
Bilinear	Pixel shuffle	0	0
Bilinear	Combined data	0	0
Pixel shuffle	Bicubic	0	0
Pixel shuffle	Bilinear	0	0
Pixel shuffle	Pixel shuffle	0.459778	84.6875
Pixel shuffle	Combined data	0	0
Combined data	Bicubic	0	0
Combined data	Bilinear	0	0
Combined data	Pixel shuffle	0	0
Combined data	Combined data	0.463211	84.0625

In [ ]:

In [ ]:

In [ ]: