```
import numpy as np
import h5py
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import cv2
from keras.layers import Input,Dense,Reshape,Conv2D,Dropout,multiply,Dot,Concatenate,subtr
from keras.layers import BatchNormalization,LeakyReLU,Flatten
from keras.layers import Conv2DTranspose as Deconv2d
from keras.models import Model
from tensorflow.keras.optimizers import Adam
from google.colab import files
from keras import backend as K
import smtplib
from sklearn.utils import shuffle
from google.colab import drive
```

3.Conv2DTranspose Layer The Conv2DTranspose or transpose convolutional layer is more complex than a simple upsampling layer.

A simple way to think about it is that it both performs the upsample operation and interprets the coarse input data to fill in the detail while it is upsampling. It is like a layer that combines the UpSampling2D and Conv2D layers into one layer. This is a crude understanding, but a practical starting point.

importing all the necessary required packages

Overfitting Overfitting occurs when our machine learning model tries to cover all the data points or more than the required data points present in the given dataset. Because of this, the model starts caching noise and inaccurate values present in the dataset, and all these factors reduce the efficiency and accuracy of the model. The overfitted model has low bias and high variance.

```
def plot(A,B,C,n):

    samples = [A,B,C]
    fig = plt.figure(figsize=(3,n))        #figsize(float, float): Width, height in inches
    gs = gridspec.GridSpec(3,n)             #no of rows and cols in grid
    g=0
    for i in range(3):
        for j in range(n):
            ax = plt.subplot(gs[g])
            g+=1
            plt.axis('off')
```

```
            ax.set_xticklabels([])          #setting labels in graph
            ax.set_yticklabels([])
            ax.set_aspect('equal')            #ratio of y axis and x-axis is set to equal
            if samples[i][j].shape == (32,32,1):
              plt.imshow(samples[i][j].reshape(32, 32))     # used to display an image in
            else:
              plt.imshow(samples[i][j].reshape(32,32,3))

    return fig
```

Double-click (or enter) to edit

```
#for plotting any two images in case

def ploty(A,B,n):

    samples = [A,B]
    fig = plt.figure(figsize=(3,n))
    gs = gridspec.GridSpec(3,n)
    g=0
    for i in range(2):
        for j in range(n):
            ax = plt.subplot(gs[g])
            g+=1
            plt.axis('off')
            ax.set_xticklabels([])
            ax.set_yticklabels([])
            ax.set_aspect('equal')
            if samples[i][j].shape == (32,32,1):
              plt.imshow(samples[i][j].reshape(32, 32, 1))
            else:
              plt.imshow(samples[i][j].reshape(32,32,3))

    return fig
```

```
drive.mount('/content/gdrive',force_remount=True)
```

```
Mounted at /content/gdrive
```

```
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

y=x_train
x=np.sum(y, axis=3)/(3*255)

y_test=x_test
x_test=np.sum(x_test, axis=3)/(3*255)   #for converting RGB into singe channel
x_test=x_test.reshape(10000, 32, 32, 1)


y=x_train/255
y=y*2-1
```

```
#x=x*2-1
#x=np.dot(y[...,:3], [0.299, 0.587, 0.114])/255
#x=x.reshape(50000,32, 32,1)

x=x.reshape(50000, 32, 32, 1)

print(x.shape)
print(y.shape)
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 2s 0us/step
170508288/170498071 [==============================] - 2s 0us/step
(50000, 32, 32, 1)
(50000, 32, 32, 3)
```

```
x_shape=(32,32,1)
y_shape=(32,32,3)
```

```
def Generator():
  X = Input(shape = x_shape)

  #C1 = ZeroPadding2D(padding=(1,1))(X)
  C1 = Conv2D(64,kernel_size = 1, strides = 1,input_shape = x_shape)(X)
  C1 = LeakyReLU(0.2)(C1)

  C2 = Conv2D(128,kernel_size = 2, strides = 2)(C1)
  C2 = LeakyReLU(0.2)(C2)

  C3 = Conv2D(256,kernel_size = 2, strides = 2)(C2)
  C3 = LeakyReLU(0.2)(C3)

  C4 = Conv2D(512,kernel_size = 2, strides = 2)(C3)
  C4 = LeakyReLU(0.2)(C4)

  C5 = Conv2D(512, kernel_size = 2, strides = 2)(C4)
  C5 = LeakyReLU(0.2)(C5)


  DC0 = Deconv2d(512, kernel_size = 2, strides = 2)(C5)
  DC0 = LeakyReLU(0.2)(DC0)
  DC0 = BatchNormalization()(DC0)
  DC0 = Dropout(0.5)(DC0)
  DC0 = Concatenate(axis=3)([DC0, C4])


  DC1 = Deconv2d(256,kernel_size=2, strides = 2)(DC0)
  DC1 = LeakyReLU(0.2)(DC1)
  DC1 = BatchNormalization()(DC1)
  DC1 = Dropout(0.5)(DC1)
  DC1 = Concatenate(axis=3)([DC1,C3])
```

```
  DC2 = Deconv2d(128,kernel_size=2, strides = 2)(DC1)
  DC2 = LeakyReLU(0.2)(DC2)
  DC2 = BatchNormalization()(DC2)
  DC2 = Concatenate(axis=3)([DC2,C2])

  DC3 = Deconv2d(64,kernel_size=2, strides = 2)(DC2)
  DC3 = LeakyReLU(0.2)(DC3)
  DC3 = BatchNormalization()(DC3)
  DC3 = Concatenate(axis=3)([DC3,C1])

  #DC4 = ZeroPadding2D(padding=(3,1))(DC3)
  CC4 = Conv2D(3,kernel_size=(1, 1), strides = (1, 1), activation="tanh")(DC3)

  m = Model(X,CC4)
  #m.summary()
  return m
```

```
def Discriminator():
  X = Input(shape = x_shape)
  Y = Input(shape = y_shape)

  In = Concatenate(axis=3)([X,Y])

  C1 = Conv2D(64,kernel_size = 2, strides = 2,input_shape = x_shape)(In)
  C1 = BatchNormalization()(C1)
  C1 = LeakyReLU(0.2)(C1)
  C2 = Conv2D(128,kernel_size = 2, strides = 2)(C1)
  C2 = BatchNormalization()(C2)
  C2 = LeakyReLU(0.2)(C2)

  C3 = Conv2D(256,kernel_size = 2, strides = 2)(C2)
  C3 = BatchNormalization()(C3)
  C3 = LeakyReLU(0.2)(C3)

  C4 = Conv2D(512,kernel_size = 1, strides = 1)(C3)
  C4 = BatchNormalization()(C4)
  C4 = LeakyReLU(0.2)(C4)

  D = Flatten()(C4)
  D = Dense(128)(D)
  D = Dense(1,activation='sigmoid')(D)

  m = Model([X,Y],D)
  #m.summary()
  return m
```

```
X = Input(shape = x_shape)
Y = Input(shape = y_shape)
```

```
gen = Generator()
dis = Discriminator()

out = gen(X)
comb = dis([X,out])

out = Flatten()(out)
org = Flatten()(Y)

cos_dis = Dot(axes = 1,normalize = True)([out,org])

combined = Model([X,Y],[comb,cos_dis])
```

```
genLoss=[]
disLoss=[]
```

```
epochs =50
batch_size = 50
n_example = 50000
batches = int(n_example/batch_size)
dis_updates = 2
gen_updates = 1
zero=np.zeros((batch_size,1))
one=np.ones((batch_size,1))*0.9
d_loss_factor = batches*2*dis_updates
g_loss_factor = batches*gen_updates
reuse = False
adams = Adam(lr = 0.0001)
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWarning: T
  super(Adam, self).__init__(name, **kwargs)
```

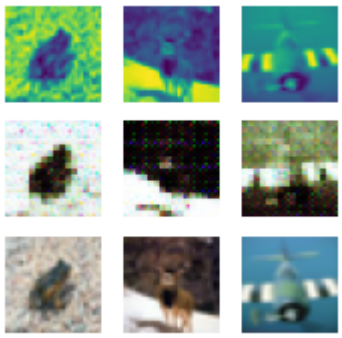
```
#location in drive where models are present.

if(reuse == True):
  gen.load_weights("gdrive/My Drive/newGAN/Generator.h5")
  dis.load_weights("gdrive/My Drive/newGAN/Discriminator.h5")
```

```
for epoch in range(epochs):
  print("###############")
  print("For Epoch:"+str(epoch))

  g_loss = 0
  d_loss = 0

  print("Training Discriminator")

  i = shuffle(range(n_example))
```

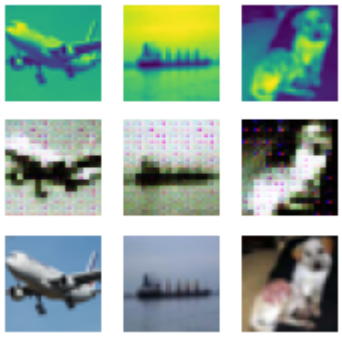
```
dis.trainable = True
dis.compile(loss = "binary_crossentropy",optimizer = adams)

for j in range(dis_updates):

  for b in range(batches):

    x_batch = x[i[b*batch_size:(b+1)*batch_size]]
    y_batch = y[i[b*batch_size:(b+1)*batch_size]]

    pre_batch = gen.predict(x_batch)

    d_loss += dis.train_on_batch([x_batch,y_batch],one)
    d_loss += dis.train_on_batch([x_batch,pre_batch],zero)

print("Training Generator")

dis.trainable = False
combined.compile(loss  = "binary_crossentropy", optimizer = adams)
dis.compile(loss = "binary_crossentropy",optimizer = adams)

for  j in range(gen_updates):

  for b in range(batches):

    x_batch = x[i[b*batch_size:(b+1)*batch_size]]
    y_batch = y[i[b*batch_size:(b+1)*batch_size]]


    #in case the mode collapse takes place....commenting next two lines might help.
    #if b%4==3:
      #gl,_,_ = combined.train_on_batch([x_batch,y_batch],[zero,one])

    gl,_,_ = combined.train_on_batch([x_batch,y_batch],[one,one])
    g_loss += gl

g_loss /= g_loss_factor
d_loss /= d_loss_factor

print("Discriminator Loss:"+str(d_loss))
print("Generator loss:"+str(g_loss))

genLoss.append(g_loss)
disLoss.append(d_loss)

gen.save_weights("gdrive/My Drive/newGAN/Generator.h5")
dis.save_weights("gdrive/My Drive/newGAN/Discriminator.h5")



plt_indices = np.random.randint(50000,size=3)
plt_a = x[plt_indices]
plt_b = gen.predict(plt_a)
plt_b = (plt_b+1)/2
plt_c = (y[plt_indices]+1)/2
```

```
  fig = plot(plt_a,plt_b,plt_c,3)
  plt.show()
  plt.close(fig)



plt.plot(genLoss, c='r', label="Generator Loss")
plt.plot(disLoss, c='b', label="Discriminator Loss")
plt.xlabel("Iterations")
plt.ylabel("Loss")
plt.legend()
plt.show()

files.download('gdrive/My Drive/GANModels/Generator.h5')
files.download('gdrive/My Drive/GANModels/Discriminator.h5')



#for recieving mail on completion of training.
  # server = smtplib.SMTP('smtp.gmail.com', 587)
  # server.starttls()
  # server.login("************@gmail.com", "*********")

  # msg = "COLAB WORK FINISH ALERT!"
  # server.sendmail("snaik4398@gmail.com", "********@vit.ac.in", msg)
  # server.quit()
```

```
###############
For Epoch:0
Training Discriminator
Training Generator
Discriminator Loss:0.2001913592489873
Generator loss:0.7948694915771485
```

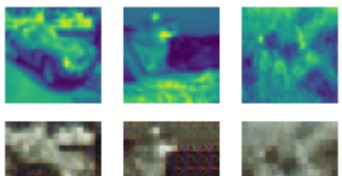


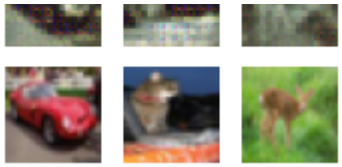
```
###############
For Epoch:1
Training Discriminator
Training Generator
Discriminator Loss:0.1715715838742235
Generator loss:0.6936229740381241
```



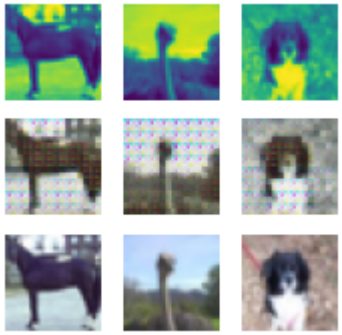
```
###############
For Epoch:2
Training Discriminator
Training Generator
Discriminator Loss:0.16729246332281672
Generator loss:0.6855872482061386
```

```
###############
For Epoch:3
Training Discriminator
Training Generator
Discriminator Loss:0.1667014088048286
Generator loss:0.6746737248897553
```



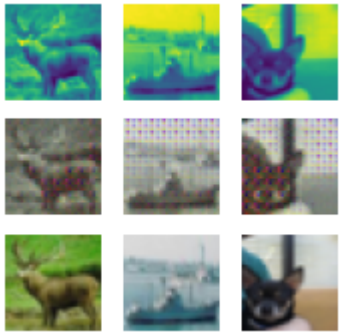


```
###############
For Epoch:4
Training Discriminator
Training Generator
Discriminator Loss:0.1651427747486623
Generator loss:0.6705417187213898
```

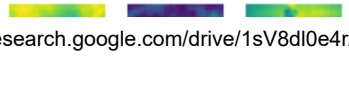


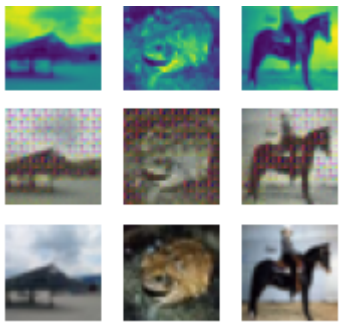
```
###############
For Epoch:5
Training Discriminator
Training Generator
Discriminator Loss:0.16545935005650061
Generator loss:0.6695241577029228
```

```
###############
For Epoch:6
Training Discriminator
Training Generator
Discriminator Loss:0.16476388082138693
Generator loss:0.6701616090536118
```

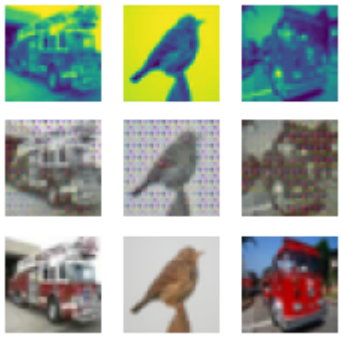


```
###############
For Epoch:7
Training Discriminator
Training Generator
Discriminator Loss:0.16493939571966756
Generator loss:0.6671749585866928
```

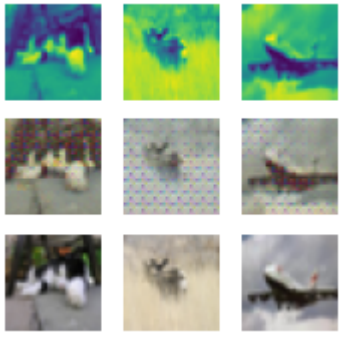




```
###############
For Epoch:8
Training Discriminator
Training Generator
Discriminator Loss:0.16520934328113934
Generator loss:0.668503820836544
```



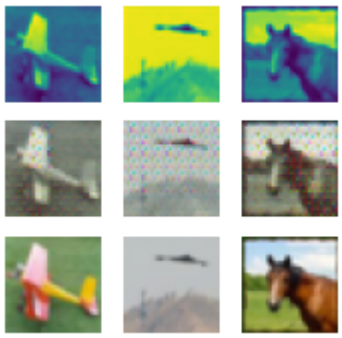
```
###############
For Epoch:9
Training Discriminator
Training Generator
Discriminator Loss:0.16582942774514958
Generator loss:0.6672675457000733
```



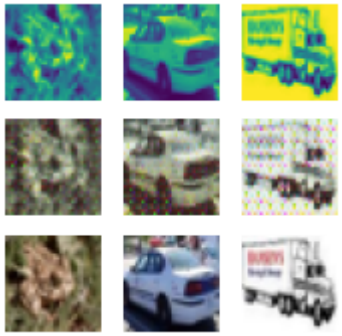
```
###############
For Epoch:10
Training Discriminator
Training Generator
Discriminator Loss:0.16502424667497917
Generator loss:0.6683814052939415
```

```
###############
For Epoch:11
```

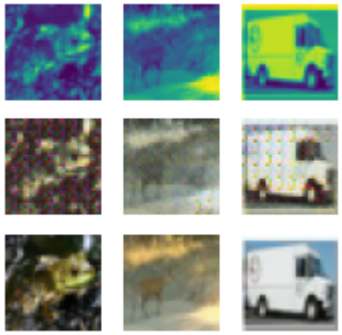
```
Training Discriminator
Training Generator
Discriminator Loss:0.16498170315802782
Generator loss:0.6706824194788933
```



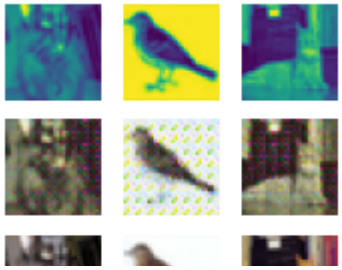
```
###############
For Epoch:12
Training Discriminator
Training Generator
Discriminator Loss:0.1646646920975586
Generator loss:0.6709591623544693
```



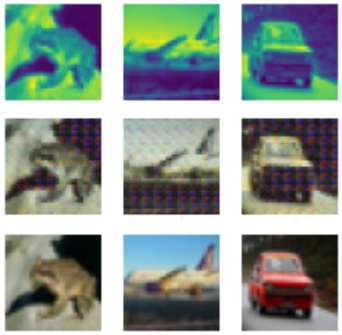
```
###############
For Epoch:13
Training Discriminator
Training Generator
Discriminator Loss:0.16435548746998394
Generator loss:0.6693950397968292
```



```
###############
For Epoch:14
Training Discriminator
Training Generator
Discriminator Loss:0.1639957057574893
Generator loss:0.6696532868742943
```



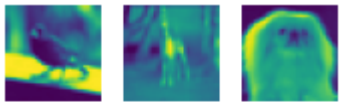
```
###############
For Epoch:15
Training Discriminator
Training Generator
Discriminator Loss:0.1641751960801239
Generator loss:0.6729533339142799
```

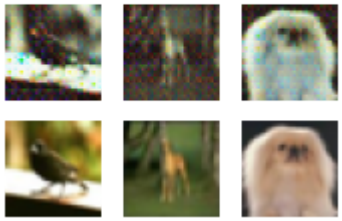


```
###############
For Epoch:16
Training Discriminator
Training Generator
Discriminator Loss:0.16336935695674812
Generator loss:0.6705266719460488
```

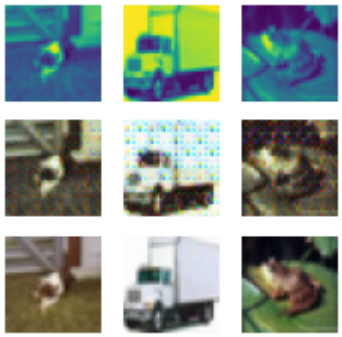
```
###############
For Epoch:17
Training Discriminator
Training Generator
Discriminator Loss:0.1633835805921035
Generator loss:0.6735583037137985
```

```
###############
For Epoch:18
Training Discriminator
Training Generator
Discriminator Loss:0.16307385235496918
Generator loss:0.6702078763246536
```



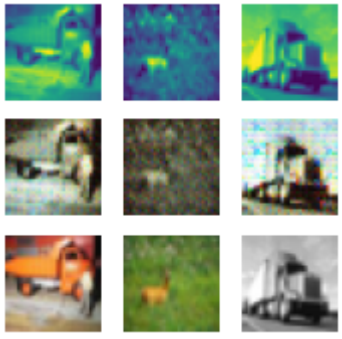


```
###############
For Epoch:19
Training Discriminator
Training Generator
Discriminator Loss:0.1632194431646458
Generator loss:0.6696241671442985
```



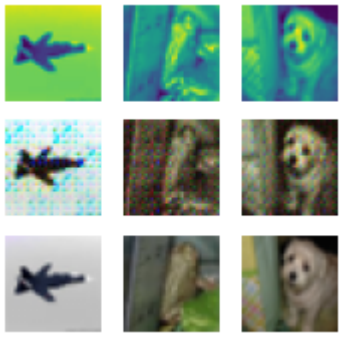
```
###############
For Epoch:20
Training Discriminator
Training Generator
Discriminator Loss:0.1630385489717346
Generator loss:0.670953309237957
```

```
###############
For Epoch:21
Training Discriminator
Training Generator
Discriminator Loss:0.16292553682323865
Generator loss:0.6697869750857354
```

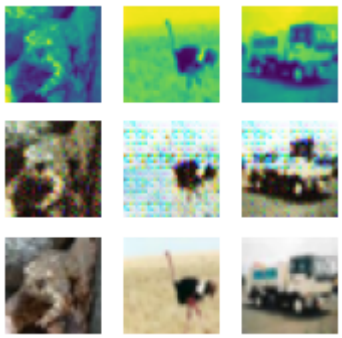


```
###############
For Epoch:22
Training Discriminator
Training Generator
Discriminator Loss:0.16285098506767348
```

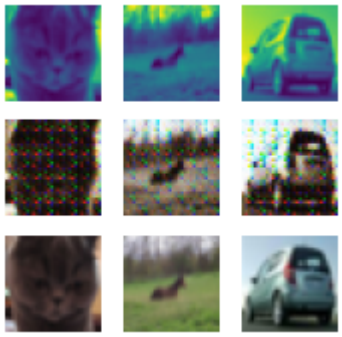
```
Generator loss:0.6704559684991837
```



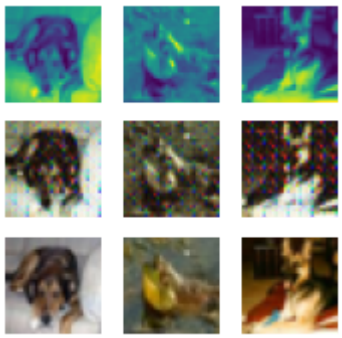
```
###############
For Epoch:23
Training Discriminator
Training Generator
Discriminator Loss:0.16281876400291856
Generator loss:0.6680046060681343
```



```
###############
For Epoch:24
Training Discriminator
Training Generator
Discriminator Loss:0.1628917342418381
Generator loss:0.6694869484901428
```

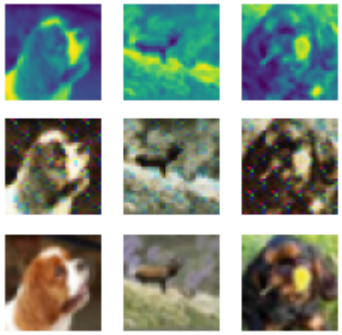


```
###############
For Epoch:25
Training Discriminator
Training Generator
Discriminator Loss:0.16295044648519671
Generator loss:0.6708984388709068
```

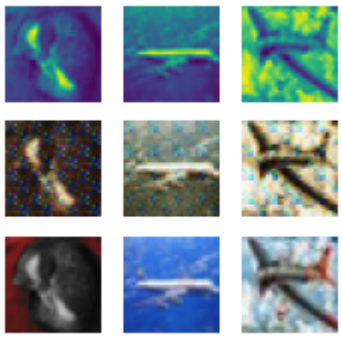


```
...................
```

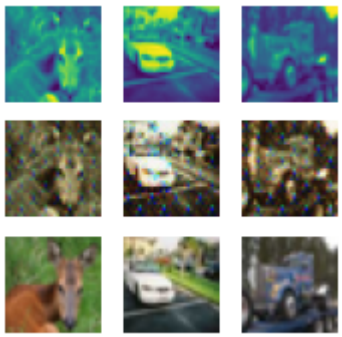
```
###############
For Epoch:26
Training Discriminator
Training Generator
Discriminator Loss:0.16297809908195415
Generator loss:0.6685514389872551
```



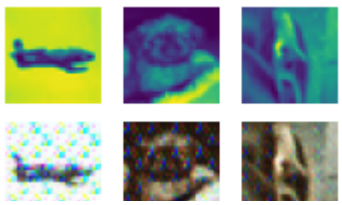
```
###############
For Epoch:27
Training Discriminator
Training Generator
Discriminator Loss:0.16279896932769453
Generator loss:0.668863990187645
```

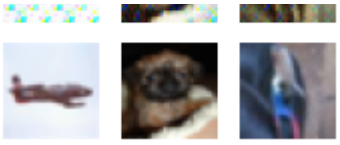


```
###############
For Epoch:28
Training Discriminator
Training Generator
Discriminator Loss:0.16268058442148306
Generator loss:0.6667280843257904
```

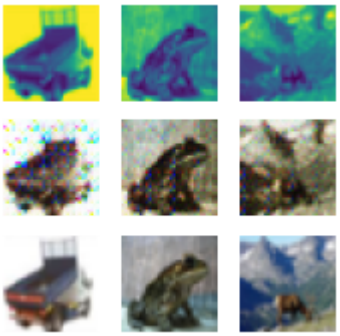


```
###############
For Epoch:29
Training Discriminator
Training Generator
Discriminator Loss:0.1627259402443884
Generator loss:0.665791107416153
```





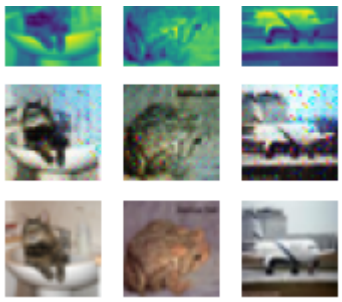
```
###############
For Epoch:30
Training Discriminator
Training Generator
Discriminator Loss:0.162629849890812
Generator loss:0.6645058681368827
```



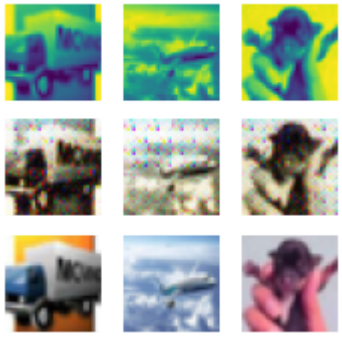
```
###############
For Epoch:31
Training Discriminator
Training Generator
Discriminator Loss:0.16263454858584844
Generator loss:0.6643786028027534
```

```
###############
For Epoch:32
Training Discriminator
Training Generator
Discriminator Loss:0.16282377454855781
Generator loss:0.6657028941512108
```

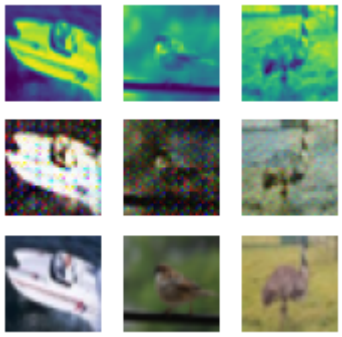
```
###############
For Epoch:33
Training Discriminator
Training Generator
Discriminator Loss:0.1626541676528816
Generator loss:0.6645093307495117
```



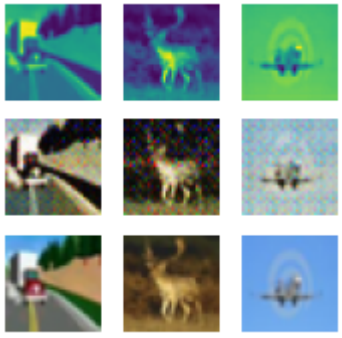
```
###############
For Epoch:34
Training Discriminator
Training Generator
Discriminator Loss:0.1626168895604219
Generator loss:0.6652793645262718
```



```
###############
For Epoch:35
Training Discriminator
Training Generator
Discriminator Loss:0.16262337673003105
Generator loss:0.6643437165021896
```

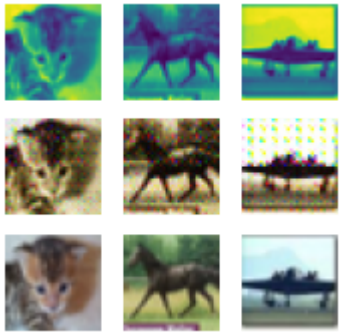


```
###############
For Epoch:36
Training Discriminator
Training Generator
Discriminator Loss:0.16267056236184704
Generator loss:0.6643946505188942
```



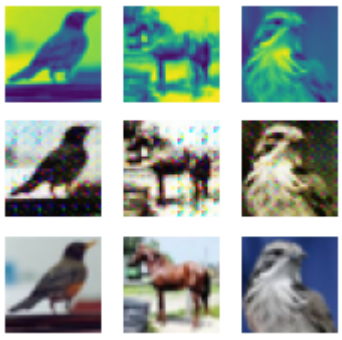
```
###############
For Epoch:37
Training Discriminator
```

```
Training Generator
Discriminator Loss:0.16266138081994966
Generator loss:0.663795507311821
```

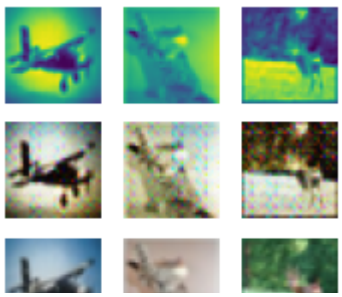


```
###############
For Epoch:38
Training Discriminator
Training Generator
Discriminator Loss:0.1626005861850747
Generator loss:0.6639219691753387
```

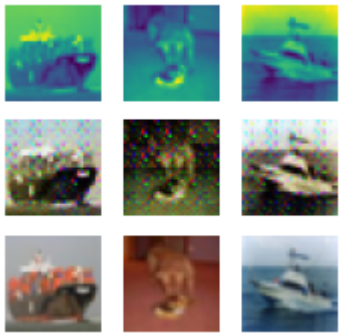
```
###############
For Epoch:39
Training Discriminator
Training Generator
Discriminator Loss:0.16262210607921707
Generator loss:0.6643993604183197
```



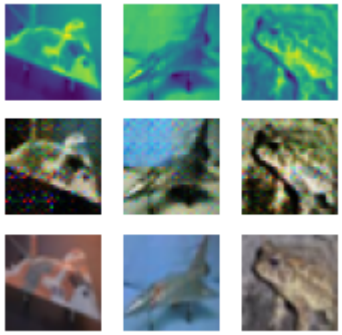
```
###############
For Epoch:40
Training Discriminator
Training Generator
Discriminator Loss:0.16260274850193768
Generator loss:0.6628811544179917
```



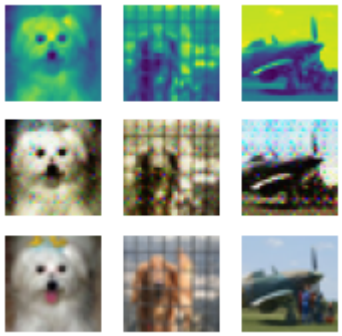
```
###############
For Epoch:41
Training Discriminator
Training Generator
Discriminator Loss:0.1625911410105966
Generator loss:0.6633278212547302
```



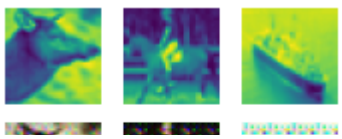
```
###############
For Epoch:42
Training Discriminator
Training Generator
Discriminator Loss:0.16258012370028194
Generator loss:0.665114381313324
```

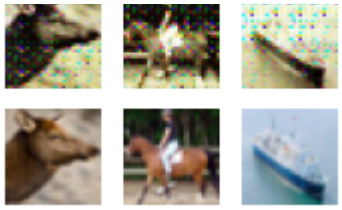


```
###############
For Epoch:43
Training Discriminator
Training Generator
Discriminator Loss:0.1626821110654053
Generator loss:0.6649027048349381
```

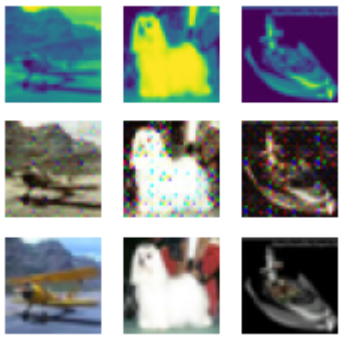


```
###############
For Epoch:44
Training Discriminator
Training Generator
Discriminator Loss:0.16276664168898008
Generator loss:0.6643109240531921
```



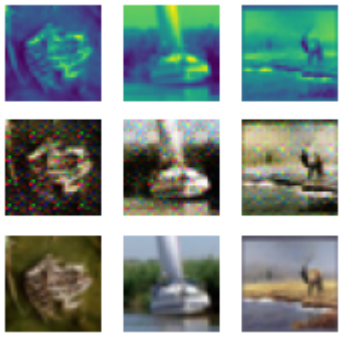


```
###############
For Epoch:45
Training Discriminator
Training Generator
Discriminator Loss:0.16261263331118853
Generator loss:0.6661321184635163
```



```
###############
For Epoch:46
Training Discriminator
Training Generator
Discriminator Loss:0.16267529108989112
Generator loss:0.6666202465295792
```

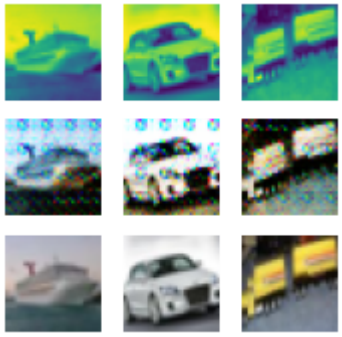
```
###############
For Epoch:47
Training Discriminator
Training Generator
Discriminator Loss:0.1625856142870918
Generator loss:0.6657525597810745
```



```
###############
For Epoch:48
Training Discriminator
Training Generator
Discriminator Loss:0.1625944320747853
Generator loss:0.6689751577973366
```

Generator loss:0.6689751577973366

```
###############
For Epoch:49
Training Discriminator
Training Generator
Discriminator Loss:0.16257633262809104
Generator loss:0.6763762025237083
```



35m 16s completed at 11:10 PM