```
import keras
import numpy as np
from os import listdir
# from scipy.misc import imread, imresize
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from keras.datasets import mnist
import matplotlib.pyplot as plt
%matplotlib inline
import os
import numpy as np
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import ImageDataGenerator
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# print(os.listdir("/content/drive/MyDrive/Dogs/train/")
```

```
# Define the path to your data folder on Google Drive
data_dir = '/content/drive/MyDrive/Dogs/Train_Data'

# Define the image dimensions and batch size
img_width, img_height = 64, 64
batch_size = 1

# Create an ImageDataGenerator for data augmentation
datagen = ImageDataGenerator(
    rescale=1.0/255.0,  # Rescale pixel values to the range [0, 1]
    rotation_range=20,  # Randomly rotate images
    width_shift_range=0.2,  # Randomly shift the width
    height_shift_range=0.2,  # Randomly shift the height
    horizontal_flip=True,  # Randomly flip images horizontally
    validation_split=0.2  # Split the data into training and validation
)

# Load images and split them into training and validation sets
train_generator = datagen.flow_from_directory(
    data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical',  # Change this to suit your task
    subset='training'  # This specifies the training set
)

val_generator = datagen.flow_from_directory(
    data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical',  # Change this to suit your task
    subset='validation'  # This specifies the validation set
)
```

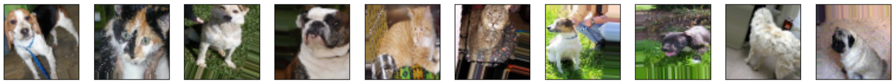
```
X_train = []
Y_train = []
for _ in range(len(train_generator)):
    batch_x, batch_y = train_generator.next()
    X_train.extend(batch_x)
    Y_train.extend(batch_y)

# Convert x_train and y_train to NumPy arrays
X_train = np.array(X_train)
Y_train = np.array(Y_train)

X_test = []
Y_test = []
for _ in range(len(val_generator)):
    batch_x, batch_y = val_generator.next()
    X_test.extend(batch_x)
    Y_test.extend(batch_y)
```

```
# Convert x_train and y_train to NumPy arrays
X_test = np.array(X_test)
Y_test = np.array(Y_test)

print(X_train.shape)
print(X_test.shape)
```

```
Found 1120 images belonging to 2 classes.
Found 279 images belonging to 2 classes.
(1120, 64, 64, 3)
(279, 64, 64, 3)
```

```
# import os
# import numpy as np
# from os import listdir
# from imageio import imread
# from skimage.transform import resize
# from sklearn.model_selection import train_test_split
# from keras.utils import to_categorical

# def get_img(file_path):
#     # Getting image array from path:
#     img_size = 64
#     img = imread(file_path)
#     img = resize(img, (img_size, img_size, 3))
#     return img

# def get_dataset(data_path='/content/drive/MyDrive/Train_Data'):
#     # Getting all data from data path:
#     data_files = listdir(data_path)
#     print(data_files)
#     X = []
#     Y = []
#     for i, file in enumerate(data_files):
#         img = get_img(os.path.join(data_path, file))
#         X.append(img)
#         # Assuming that the filename contains the class label, e.g., "class_1.jpg"
#         label = int(file.split("_")[1].split(".")[0])
#         Y.append(label)

#         # Create dataset:
#     X = np.array(X).astype('float32') / 255.
#     Y = np.array(Y).astype('float32')
#     Y = to_categorical(Y, num_classes=2)

#     if not os.path.exists('Data/npy_train_data/'):
#         os.makedirs('Data/npy_train_data/')
#     np.save('Data/npy_train_data/X.npy', X)
#     np.save('Data/npy_train_data/Y.npy', Y)

#     X, X_test, Y, Y_test = train_test_split(X, Y, test_size=0.1, random_state=42)
#     return X, X_test, Y, Y_test



# Getting Dataset:
# X_train, X_test, Y_train, Y_test = get_dataset()


img_size = X_train.shape[1] # 64
print('Training shape:', X_train.shape)
print(X_train.shape[0], 'sample,',X_train.shape[1] ,'x',X_train.shape[2] ,'size RGB image.\n')
print('Test shape:', X_test.shape)
print(X_test.shape[0], 'sample,',X_test.shape[1] ,'x',X_test.shape[2] ,'size RGB image.\n')

print('Examples:')
n = 10
plt.figure(figsize=(20, 4))
for i in range(1, n+1):
    # Display some data:
    ax = plt.subplot(1, n, i)
    plt.imshow(X_train[i])
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
```

```
Training shape: (1120, 64, 64, 3)
1120 sample, 64 x 64 size RGB image.

Test shape: (279, 64, 64, 3)
279 sample, 64 x 64 size RGB image.

Examples:
```



```
# Deep Learning Model:
from keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D, Dense
from keras.models import Model

input_img = Input(shape=(64, 64, 3))

x = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

x = Conv2D(64, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='same')(x)

autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='rmsprop', loss='mse')

autoencoder.summary()
```

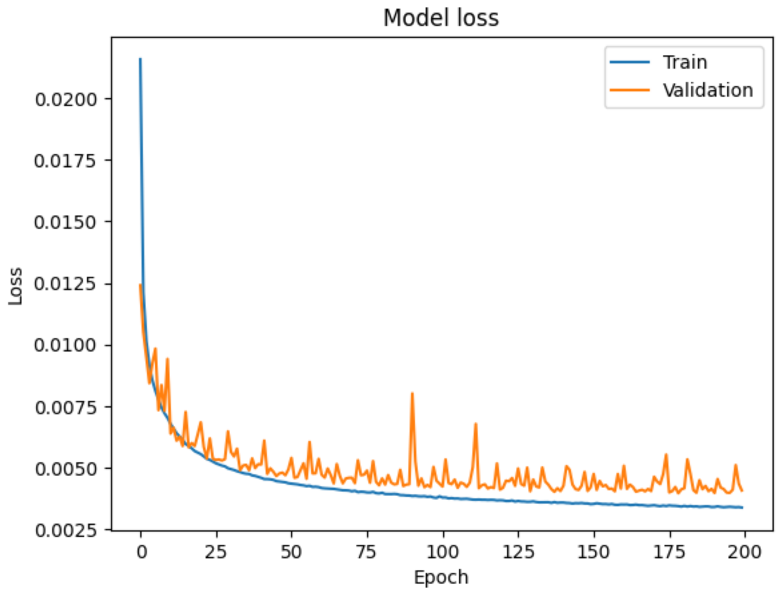
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_1 (InputLayer)        [(None, 64, 64, 3)]       0         
                                                                 
 conv2d (Conv2D)             (None, 64, 64, 32)        896       
                                                                 
 max_pooling2d (MaxPooling2  (None, 32, 32, 32)        0         
 D)                                                              
                                                                 
 conv2d_1 (Conv2D)           (None, 32, 32, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 16, 16, 64)        0         
 g2D)                                                            
                                                                 
 conv2d_2 (Conv2D)           (None, 16, 16, 64)        36928     
                                                                 
 max_pooling2d_2 (MaxPoolin  (None, 8, 8, 64)          0         
 g2D)                                                            
                                                                 
 conv2d_3 (Conv2D)           (None, 8, 8, 64)          36928     
                                                                 
 up_sampling2d (UpSampling2  (None, 16, 16, 64)        0         
 D)                                                              
                                                                 
 conv2d_4 (Conv2D)           (None, 16, 16, 64)        36928     
                                                                 
 up_sampling2d_1 (UpSamplin  (None, 32, 32, 64)        0         
 g2D)                                                            
                                                                 
 conv2d_5 (Conv2D)           (None, 32, 32, 32)        18464     
                                                                 
 up_sampling2d_2 (UpSamplin  (None, 64, 64, 32)        0         
 g2D)                                                            
                                                                 
 conv2d_6 (Conv2D)           (None, 64, 64, 3)         867       
                                                                 
=================================================================
Total params: 149507 (584.01 KB)
Trainable params: 149507 (584.01 KB)
```

```
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# Checkpoints:
from keras.callbacks import ModelCheckpoint, TensorBoard
checkpoints = []
#checkpoints.append(TensorBoard(log_dir='/Checkpoints/logs'))


epochs = 200
batch_size = 1
history = autoencoder.fit(X_train, X_train, batch_size=batch_size, epochs=epochs, validation_data=(X_test, X_test), shuffle=True,
```

```
Epoch 172/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0044
Epoch 173/200
1120/1120 [==============================] - 33s 30ms/step - loss: 0.0034 - val_loss: 0.0043
Epoch 174/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0035 - val_loss: 0.0047
Epoch 175/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0055
Epoch 176/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0035 - val_loss: 0.0040
Epoch 177/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0035 - val_loss: 0.0040
Epoch 178/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0042
Epoch 179/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0040
Epoch 180/200
1120/1120 [==============================] - 31s 27ms/step - loss: 0.0034 - val_loss: 0.0041
Epoch 181/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0042
Epoch 182/200
1120/1120 [==============================] - 31s 28ms/step - loss: 0.0034 - val_loss: 0.0053
Epoch 183/200
1120/1120 [==============================] - 31s 28ms/step - loss: 0.0034 - val_loss: 0.0047
Epoch 184/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0041
Epoch 185/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0040
Epoch 186/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0045
Epoch 187/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0041
Epoch 188/200
1120/1120 [==============================] - 31s 28ms/step - loss: 0.0034 - val_loss: 0.0043
Epoch 189/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0040
Epoch 190/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0041
Epoch 191/200
1120/1120 [==============================] - 31s 27ms/step - loss: 0.0034 - val_loss: 0.0040
Epoch 192/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0045
Epoch 193/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0042
Epoch 194/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0041
Epoch 195/200
1120/1120 [==============================] - 31s 27ms/step - loss: 0.0034 - val_loss: 0.0040
Epoch 196/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0040
Epoch 197/200
1120/1120 [==============================] - 31s 28ms/step - loss: 0.0034 - val_loss: 0.0041
Epoch 198/200
1120/1120 [==============================] - 31s 27ms/step - loss: 0.0034 - val_loss: 0.0051
Epoch 199/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0044
Epoch 200/200
1120/1120 [==============================] - 30s 27ms/step - loss: 0.0034 - val_loss: 0.0041
```

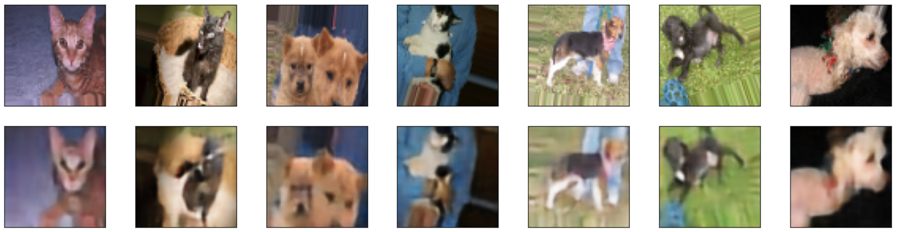
```
import matplotlib.pyplot as plt

history = autoencoder.history
# print(history)

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()
```



```
decoded_imgs = autoencoder.predict(X_test)
decoded_train_imgs = autoencoder.predict(X_train)
```

```
     9/9 [==============================] - 1s 125ms/step
     35/35 [==============================] - 5s 153ms/step
```

```
n = 7
plt.figure(figsize=(20, 5))
for i in range(1, n+1):
    # Display original:
    ax = plt.subplot(2, n, i)
    plt.imshow(X_test[i])
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Display reconstruction:
    ax = plt.subplot(2, n, i+n)
    plt.imshow(decoded_imgs[i])
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
```



```
n = 7
plt.figure(figsize=(20, 5))
for i in range(1, n+1):
    # Display original:
    ax = plt.subplot(2, n, i)
    plt.imshow(X_train[i])
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Display reconstruction:
    ax = plt.subplot(2, n, i+n)
    plt.imshow(decoded_train_imgs[i])
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
```