In [1]:

```python
"""
Image Super-Resolution using Convolutional
Autoencoders
Author: Amruth Karun M V
Date: 06-Nov-2021
"""

import cv2
import numpy as np
import random
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras import Model, Input, regularizers
from tensorflow.keras.layers import (
    Dense, Conv2D, MaxPool2D,
    UpSampling2D, Add)
from tensorflow.keras.callbacks import EarlyStopping
from keras.preprocessing import image
import glob
from tqdm import tqdm
import matplotlib.pyplot as plt
import warnings;
warnings.filterwarnings('ignore')


INPUT_PATH = '../input/cifar10/cifar10_sample/'

def show_sample_image():
    """
    Displays original 32 x 32 images.
    Arguments: None
    Returns: Displays the image
    """

    cifar_sample = glob.glob(INPUT_PATH + '*.png')
    random_index = random.randint(0, 19)
    print("Image: ", random_index)
    img_path = cifar_sample[random_index]
    img = cv2.imread(img_path)
    plt.imshow(img)

def load_images():
    """
```

```
    Loads sample images from cifar10 dataset.
    2 images are taken from each class. Total 20 images.
    Arguments: None
    Returns: Train and val images
    """

    cifar_sample = glob.glob(INPUT_PATH + '*.png')
    print("Total images = ", len(cifar_sample))
    all_images = []
    for i in tqdm(cifar_sample):
        img = image.load_img(i, target_size=(32,32,3))
        img = image.img_to_array(img)
        img = img/255.
        all_images.append(img)
    all_images = np.array(all_images)
    train_x, val_x = train_test_split(all_images, random_state=32, test_size=0.2
)
    return train_x, val_x

def pixalate_image(image, scale_percent = 50):
    """
    Lower the resolution of input image without
    reducing the size
    Arguments:
        image         -- input image
        scale_percent -- amount to be reduced
    Returns: Pixalated image
    """

    width = int(image.shape[1] * scale_percent / 100)
    height = int(image.shape[0] * scale_percent / 100)
    dim = (width, height)
    small_image = cv2.resize(image, dim, interpolation = cv2.INTER_AREA)
    width = int(small_image.shape[1] * 100 / scale_percent)
    height = int(small_image.shape[0] * 100 / scale_percent)
    dim = (width, height)
    low_res_image = cv2.resize(small_image, dim, interpolation = cv2.INTER_AREA)
    return low_res_image


def get_low_res_image(train_x, val_x):
    """
    Get low resolution images for train
    and validation set
    Arguments:
```

```
        train_x  -- train set
        val_x    -- validation set
    Returns: Low resolution data
    """

    # get low resolution images for the train set
    train_x_px = []

    for i in range(train_x.shape[0]):
        temp = pixalate_image(train_x[i,:,:,:])
        train_x_px.append(temp)
    train_x_px = np.array(train_x_px)

    # get low resolution images for the validation set
    val_x_px = []
    for i in range(val_x.shape[0]):
        temp = pixalate_image(val_x[i,:,:,:])
        val_x_px.append(temp)
    val_x_px = np.array(val_x_px)

    return train_x_px, val_x_px

def train_model(train_x_px, val_x_px):
    """
    Trains the  Autoencoder network
    Arguments:
        train_x_px    -- low resolution train set
        val_x_px      -- low resolution validation set
    Returns: Autoencoder Network
    """

    Input_img = Input(shape=(32, 32, 3))
    #encoding architecture
    x1 = Conv2D(64, (3, 3), activation='relu', padding='same', kernel_regularize
r=regularizers.l1(10e-10))(Input_img)
    x2 = Conv2D(64, (3, 3), activation='relu', padding='same', kernel_regularize
r=regularizers.l1(10e-10))(x1)
    x3 = MaxPool2D(padding='same')(x2)
    x4 = Conv2D(128, (3, 3), activation='relu', padding='same', kernel_regulariz
er=regularizers.l1(10e-10))(x3)
    x5 = Conv2D(128, (3, 3), activation='relu', padding='same', kernel_regulariz
er=regularizers.l1(10e-10))(x4)
    x6 = MaxPool2D(padding='same')(x5)
    encoded = Conv2D(256, (3, 3), activation='relu', padding='same', kernel_regu
larizer=regularizers.l1(10e-10))(x6)
```

```
    # decoding architecture
    x7 = UpSampling2D()(encoded)
    x8 = Conv2D(128, (3, 3), activation='relu', padding='same', kernel_regulariz
er=regularizers.l1(10e-10))(x7)
    x9 = Conv2D(128, (3, 3), activation='relu', padding='same', kernel_regulariz
er=regularizers.l1(10e-10))(x8)
    x10 = Add()([x5, x9])
    x11 = UpSampling2D()(x10)
    x12 = Conv2D(64, (3, 3), activation='relu', padding='same', kernel_regulariz
er=regularizers.l1(10e-10))(x11)
    x13 = Conv2D(64, (3, 3), activation='relu', padding='same', kernel_regulariz
er=regularizers.l1(10e-10))(x12)
    x14 = Add()([x2, x13])
    decoded = Conv2D(3, (3, 3), padding='same',activation='relu', kernel_regular
izer=regularizers.l1(10e-10))(x14)
    autoencoder = Model(Input_img, decoded)
    autoencoder.compile(optimizer='adam', loss='mse', metrics=['accuracy'])

    autoencoder.summary()

    early_stopper = EarlyStopping(monitor='val_loss', min_delta=0.0001, patience
=8, verbose=1, mode='auto')
    history = autoencoder.fit(train_x_px, train_x,
                              epochs=256,
                              batch_size=64,
                              shuffle=True,
                              validation_data=(val_x_px, val_x),
                              callbacks=[early_stopper])

    autoencoder.save_weights("autoencoder.h5")
    return autoencoder

def get_results(autoencoder, val_x, val_x_px):
    """
    Evaluate the autoencoder model using
    validation data and predict results
    Arguments:
        autoencoder    -- trained autoencoder model
        val_x          -- original validation data
        val_x_px       -- low resolution validation data
    """

    results = autoencoder.evaluate(val_x_px, val_x)
    print('val_loss = {}, val_accuracy = {}'.format(results[0], results[1]))
```

```
predictions = autoencoder.predict(val_x_px)

n = 4
plt.figure(figsize= (20,10))
for i in range(n):
    ax1 = plt.subplot(3, n, i+1)
    plt.imshow(val_x_px[i])
    ax1.get_xaxis().set_visible(False)
    ax1.get_yaxis().set_visible(False)
    ax1.title.set_text('Pixelated Image')

    ax2 = plt.subplot(3, n, i+1+n)
    plt.imshow(predictions[i])
    ax2.get_xaxis().set_visible(False)
    ax2.get_yaxis().set_visible(False)
    ax2.title.set_text('Predicted Image')

plt.show()
```

In [2]:

```
# Show 1 random image from train set
# show_sample_image()
train_x, val_x = load_images()
train_x_px, val_x_px = get_low_res_image(train_x, val_x)

# Train the model
autoencoder = train_model(train_x_px, val_x_px)
get_results(autoencoder, val_x, val_x_px)
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Co
nnected to
==================================================================================================
input_1 (InputLayer)            [(None, 32, 32, 3)]  0
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 32, 32, 64)   1792        in
put_1[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 32, 32, 64)   36928       co
nv2d[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 16, 16, 64)   0           co
nv2d_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 16, 16, 128)  73856       ma
x_pooling2d[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 16, 16, 128)  147584      co
nv2d_2[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 8, 8, 128)    0           co
nv2d_3[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 8, 8, 256)    295168      ma
x_pooling2d_1[0][0]
__________________________________________________________________________________________________
up_sampling2d (UpSampling2D)    (None, 16, 16, 256)  0           co
nv2d_4[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 16, 16, 128)  295040      up
_sampling2d[0][0]
```

```
__________________________________________________________________
______________________________
conv2d_6 (Conv2D)               (None, 16, 16, 128)  147584      co
nv2d_5[0][0]
__________________________________________________________________
______________________________
add (Add)                       (None, 16, 16, 128)  0           co
nv2d_3[0][0]
                                                                 co
nv2d_6[0][0]
__________________________________________________________________
______________________________
up_sampling2d_1 (UpSampling2D)  (None, 32, 32, 128)  0           ad
d[0][0]
__________________________________________________________________
______________________________
conv2d_7 (Conv2D)               (None, 32, 32, 64)   73792       up
_sampling2d_1[0][0]
__________________________________________________________________
______________________________
conv2d_8 (Conv2D)               (None, 32, 32, 64)   36928       co
nv2d_7[0][0]
__________________________________________________________________
______________________________
add_1 (Add)                     (None, 32, 32, 64)   0           co
nv2d_1[0][0]
                                                                 co
nv2d_8[0][0]
__________________________________________________________________
______________________________
conv2d_9 (Conv2D)               (None, 32, 32, 3)    1731        ad
d_1[0][0]
==================================================================
==============================
Total params: 1,110,403
Trainable params: 1,110,403
Non-trainable params: 0
__________________________________________________________________
______________________________
Epoch 1/256
```

```
2021-11-06 14:56:09.485213: I tensorflow/compiler/mlir/mlir_graph_o
ptimization_pass.cc:185] None of the MLIR Optimization Passes are e
nabled (registered 2)
```

```
1/1 [==============================] - 2s 2s/step - loss: 0.2602 -
accuracy: 0.1879 - val_loss: 0.2291 - val_accuracy: 0.3660
Epoch 2/256
1/1 [==============================] - 0s 335ms/step - loss: 0.2234
- accuracy: 0.1685 - val_loss: 0.1495 - val_accuracy: 0.3660
Epoch 3/256
1/1 [==============================] - 0s 334ms/step - loss: 0.1438
- accuracy: 0.1685 - val_loss: 0.0832 - val_accuracy: 0.3667
Epoch 4/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0877
- accuracy: 0.1703 - val_loss: 0.0470 - val_accuracy: 0.3765
Epoch 5/256
1/1 [==============================] - 0s 348ms/step - loss: 0.0555
- accuracy: 0.2001 - val_loss: 0.0368 - val_accuracy: 0.5352
Epoch 6/256
1/1 [==============================] - 0s 337ms/step - loss: 0.0418
- accuracy: 0.5106 - val_loss: 0.0262 - val_accuracy: 0.4985
Epoch 7/256
1/1 [==============================] - 0s 336ms/step - loss: 0.0290
- accuracy: 0.5726 - val_loss: 0.0232 - val_accuracy: 0.4189
Epoch 8/256
1/1 [==============================] - 0s 349ms/step - loss: 0.0240
- accuracy: 0.4974 - val_loss: 0.0246 - val_accuracy: 0.2380
Epoch 9/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0232
- accuracy: 0.3387 - val_loss: 0.0268 - val_accuracy: 0.1758
Epoch 10/256
1/1 [==============================] - 0s 331ms/step - loss: 0.0236
- accuracy: 0.2985 - val_loss: 0.0283 - val_accuracy: 0.1641
Epoch 11/256
1/1 [==============================] - 0s 336ms/step - loss: 0.0240
- accuracy: 0.2819 - val_loss: 0.0283 - val_accuracy: 0.1633
Epoch 12/256
1/1 [==============================] - 0s 336ms/step - loss: 0.0236
- accuracy: 0.2809 - val_loss: 0.0271 - val_accuracy: 0.1641
Epoch 13/256
1/1 [==============================] - 0s 330ms/step - loss: 0.0227
- accuracy: 0.2849 - val_loss: 0.0252 - val_accuracy: 0.1689
Epoch 14/256
1/1 [==============================] - 0s 390ms/step - loss: 0.0215
- accuracy: 0.3007 - val_loss: 0.0230 - val_accuracy: 0.1807
Epoch 15/256
1/1 [==============================] - 0s 337ms/step - loss: 0.0202
- accuracy: 0.3250 - val_loss: 0.0209 - val_accuracy: 0.2034
```

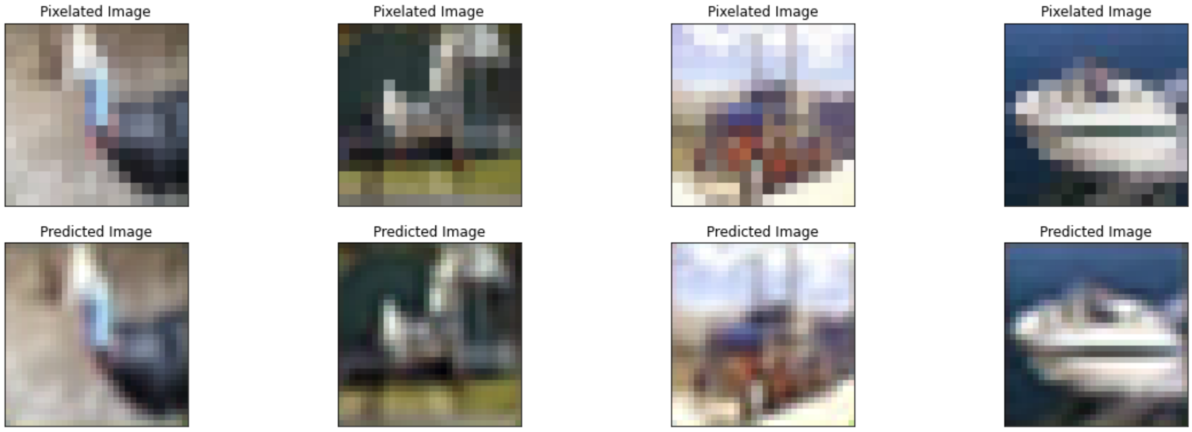
```
Epoch 16/256
1/1 [==============================] - 0s 334ms/step - loss: 0.0191
- accuracy: 0.3649 - val_loss: 0.0191 - val_accuracy: 0.2349
Epoch 17/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0181
- accuracy: 0.4069 - val_loss: 0.0177 - val_accuracy: 0.2773
Epoch 18/256
1/1 [==============================] - 0s 334ms/step - loss: 0.0174
- accuracy: 0.4540 - val_loss: 0.0165 - val_accuracy: 0.3789
Epoch 19/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0166
- accuracy: 0.4990 - val_loss: 0.0156 - val_accuracy: 0.4431
Epoch 20/256
1/1 [==============================] - 0s 334ms/step - loss: 0.0159
- accuracy: 0.5351 - val_loss: 0.0145 - val_accuracy: 0.4714
Epoch 21/256
1/1 [==============================] - 0s 330ms/step - loss: 0.0152
- accuracy: 0.5490 - val_loss: 0.0135 - val_accuracy: 0.4795
Epoch 22/256
1/1 [==============================] - 0s 334ms/step - loss: 0.0144
- accuracy: 0.5515 - val_loss: 0.0125 - val_accuracy: 0.4785
Epoch 23/256
1/1 [==============================] - 0s 332ms/step - loss: 0.0135
- accuracy: 0.5461 - val_loss: 0.0117 - val_accuracy: 0.4521
Epoch 24/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0126
- accuracy: 0.5223 - val_loss: 0.0111 - val_accuracy: 0.3860
Epoch 25/256
1/1 [==============================] - 0s 334ms/step - loss: 0.0119
- accuracy: 0.4785 - val_loss: 0.0102 - val_accuracy: 0.3877
Epoch 26/256
1/1 [==============================] - 0s 336ms/step - loss: 0.0111
- accuracy: 0.4750 - val_loss: 0.0093 - val_accuracy: 0.4441
Epoch 27/256
1/1 [==============================] - 0s 334ms/step - loss: 0.0102
- accuracy: 0.4880 - val_loss: 0.0085 - val_accuracy: 0.4929
Epoch 28/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0094
- accuracy: 0.5021 - val_loss: 0.0074 - val_accuracy: 0.5217
Epoch 29/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0086
- accuracy: 0.5152 - val_loss: 0.0065 - val_accuracy: 0.5698
Epoch 30/256
1/1 [==============================] - 0s 334ms/step - loss: 0.0079
- accuracy: 0.5339 - val_loss: 0.0059 - val_accuracy: 0.6138
```

```
Epoch 31/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0073
- accuracy: 0.5491 - val_loss: 0.0056 - val_accuracy: 0.6296
Epoch 32/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0068
- accuracy: 0.5527 - val_loss: 0.0059 - val_accuracy: 0.5566
Epoch 33/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0068
- accuracy: 0.5243 - val_loss: 0.0059 - val_accuracy: 0.6985
Epoch 34/256
1/1 [==============================] - 0s 341ms/step - loss: 0.0073
- accuracy: 0.5950 - val_loss: 0.0050 - val_accuracy: 0.6978
Epoch 35/256
1/1 [==============================] - 0s 338ms/step - loss: 0.0060
- accuracy: 0.5881 - val_loss: 0.0057 - val_accuracy: 0.6379
Epoch 36/256
1/1 [==============================] - 0s 331ms/step - loss: 0.0064
- accuracy: 0.5712 - val_loss: 0.0051 - val_accuracy: 0.7119
Epoch 37/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0062
- accuracy: 0.6049 - val_loss: 0.0048 - val_accuracy: 0.6780
Epoch 38/256
1/1 [==============================] - 0s 339ms/step - loss: 0.0058
- accuracy: 0.5925 - val_loss: 0.0052 - val_accuracy: 0.5488
Epoch 39/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0058
- accuracy: 0.5678 - val_loss: 0.0047 - val_accuracy: 0.6365
Epoch 40/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0054
- accuracy: 0.5986 - val_loss: 0.0047 - val_accuracy: 0.7190
Epoch 41/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0054
- accuracy: 0.6285 - val_loss: 0.0047 - val_accuracy: 0.7109
Epoch 42/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0053
- accuracy: 0.6296 - val_loss: 0.0045 - val_accuracy: 0.6477
Epoch 43/256
1/1 [==============================] - 0s 336ms/step - loss: 0.0051
- accuracy: 0.6281 - val_loss: 0.0047 - val_accuracy: 0.5620
Epoch 44/256
1/1 [==============================] - 0s 337ms/step - loss: 0.0052
- accuracy: 0.6267 - val_loss: 0.0045 - val_accuracy: 0.5894
Epoch 45/256
1/1 [==============================] - 0s 338ms/step - loss: 0.0049
- accuracy: 0.6432 - val_loss: 0.0046 - val_accuracy: 0.6704
```

```
Epoch 46/256
1/1 [==============================] - 0s 381ms/step - loss: 0.0050
- accuracy: 0.6653 - val_loss: 0.0043 - val_accuracy: 0.7063
Epoch 47/256
1/1 [==============================] - 0s 338ms/step - loss: 0.0048
- accuracy: 0.6780 - val_loss: 0.0043 - val_accuracy: 0.6970
Epoch 48/256
1/1 [==============================] - 0s 336ms/step - loss: 0.0047
- accuracy: 0.6880 - val_loss: 0.0043 - val_accuracy: 0.6350
Epoch 49/256
1/1 [==============================] - 0s 334ms/step - loss: 0.0046
- accuracy: 0.6982 - val_loss: 0.0043 - val_accuracy: 0.5803
Epoch 50/256
1/1 [==============================] - 0s 337ms/step - loss: 0.0046
- accuracy: 0.6943 - val_loss: 0.0042 - val_accuracy: 0.6528
Epoch 51/256
1/1 [==============================] - 0s 334ms/step - loss: 0.0045
- accuracy: 0.7189 - val_loss: 0.0041 - val_accuracy: 0.7188
Epoch 52/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0044
- accuracy: 0.7158 - val_loss: 0.0040 - val_accuracy: 0.7253
Epoch 53/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0043
- accuracy: 0.7310 - val_loss: 0.0040 - val_accuracy: 0.6611
Epoch 54/256
1/1 [==============================] - 0s 331ms/step - loss: 0.0042
- accuracy: 0.7432 - val_loss: 0.0041 - val_accuracy: 0.6228
Epoch 55/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0042
- accuracy: 0.7355 - val_loss: 0.0039 - val_accuracy: 0.6990
Epoch 56/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0041
- accuracy: 0.7606 - val_loss: 0.0039 - val_accuracy: 0.7405
Epoch 57/256
1/1 [==============================] - 0s 334ms/step - loss: 0.0041
- accuracy: 0.7505 - val_loss: 0.0038 - val_accuracy: 0.7305
Epoch 58/256
1/1 [==============================] - 0s 334ms/step - loss: 0.0040
- accuracy: 0.7729 - val_loss: 0.0039 - val_accuracy: 0.6655
Epoch 59/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0040
- accuracy: 0.7598 - val_loss: 0.0038 - val_accuracy: 0.7219
Epoch 60/256
1/1 [==============================] - 0s 331ms/step - loss: 0.0038
- accuracy: 0.7805 - val_loss: 0.0038 - val_accuracy: 0.7646
```

```
Epoch 61/256
1/1 [==============================] - 0s 337ms/step - loss: 0.0038
- accuracy: 0.7836 - val_loss: 0.0038 - val_accuracy: 0.7698
Epoch 62/256
1/1 [==============================] - 0s 331ms/step - loss: 0.0038
- accuracy: 0.7958 - val_loss: 0.0038 - val_accuracy: 0.7314
Epoch 63/256
1/1 [==============================] - 0s 339ms/step - loss: 0.0038
- accuracy: 0.7844 - val_loss: 0.0038 - val_accuracy: 0.8025
Epoch 64/256
1/1 [==============================] - 0s 346ms/step - loss: 0.0037
- accuracy: 0.8094 - val_loss: 0.0037 - val_accuracy: 0.8040
Epoch 65/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0037
- accuracy: 0.8049 - val_loss: 0.0037 - val_accuracy: 0.7686
Epoch 66/256
1/1 [==============================] - 0s 332ms/step - loss: 0.0037
- accuracy: 0.7899 - val_loss: 0.0037 - val_accuracy: 0.7817
Epoch 67/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0036
- accuracy: 0.7948 - val_loss: 0.0037 - val_accuracy: 0.8113
Epoch 68/256
1/1 [==============================] - 0s 337ms/step - loss: 0.0036
- accuracy: 0.8069 - val_loss: 0.0037 - val_accuracy: 0.8123
Epoch 69/256
1/1 [==============================] - 0s 331ms/step - loss: 0.0036
- accuracy: 0.8096 - val_loss: 0.0036 - val_accuracy: 0.7820
Epoch 70/256
1/1 [==============================] - 0s 334ms/step - loss: 0.0036
- accuracy: 0.8033 - val_loss: 0.0036 - val_accuracy: 0.7717
Epoch 71/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0035
- accuracy: 0.8011 - val_loss: 0.0036 - val_accuracy: 0.7888
Epoch 72/256
1/1 [==============================] - 0s 334ms/step - loss: 0.0035
- accuracy: 0.8110 - val_loss: 0.0036 - val_accuracy: 0.7954
Epoch 73/256
1/1 [==============================] - 0s 339ms/step - loss: 0.0035
- accuracy: 0.8140 - val_loss: 0.0036 - val_accuracy: 0.7791
Epoch 74/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0035
- accuracy: 0.8094 - val_loss: 0.0036 - val_accuracy: 0.7554
Epoch 75/256
1/1 [==============================] - 0s 332ms/step - loss: 0.0035
- accuracy: 0.8044 - val_loss: 0.0035 - val_accuracy: 0.7815
```

```
Epoch 76/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0034
- accuracy: 0.8096 - val_loss: 0.0035 - val_accuracy: 0.7986
Epoch 77/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0034
- accuracy: 0.8141 - val_loss: 0.0035 - val_accuracy: 0.7837
Epoch 78/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0034
- accuracy: 0.8115 - val_loss: 0.0035 - val_accuracy: 0.7712
Epoch 79/256
1/1 [==============================] - 0s 394ms/step - loss: 0.0034
- accuracy: 0.8088 - val_loss: 0.0035 - val_accuracy: 0.7927
Epoch 80/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0033
- accuracy: 0.8152 - val_loss: 0.0035 - val_accuracy: 0.8062
Epoch 81/256
1/1 [==============================] - 0s 330ms/step - loss: 0.0033
- accuracy: 0.8239 - val_loss: 0.0035 - val_accuracy: 0.7981
Epoch 82/256
1/1 [==============================] - 0s 364ms/step - loss: 0.0033
- accuracy: 0.8149 - val_loss: 0.0035 - val_accuracy: 0.7903
Epoch 83/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0033
- accuracy: 0.8110 - val_loss: 0.0035 - val_accuracy: 0.8022
Epoch 84/256
1/1 [==============================] - 0s 332ms/step - loss: 0.0033
- accuracy: 0.8178 - val_loss: 0.0035 - val_accuracy: 0.8096
Epoch 85/256
1/1 [==============================] - 0s 335ms/step - loss: 0.0033
- accuracy: 0.8262 - val_loss: 0.0034 - val_accuracy: 0.8110
Epoch 86/256
1/1 [==============================] - 0s 333ms/step - loss: 0.0032
- accuracy: 0.8148 - val_loss: 0.0034 - val_accuracy: 0.8013
Epoch 87/256
1/1 [==============================] - 0s 332ms/step - loss: 0.0032
- accuracy: 0.8125 - val_loss: 0.0034 - val_accuracy: 0.8159
Epoch 88/256
1/1 [==============================] - 0s 338ms/step - loss: 0.0032
- accuracy: 0.8238 - val_loss: 0.0034 - val_accuracy: 0.8159
Epoch 00088: early stopping
1/1 [==============================] - 0s 59ms/step - loss: 0.0034
- accuracy: 0.8159
val_loss = 0.0034120238851755857, val_accuracy = 0.81591796875
```


Pixelated Image
Pixelated Image
Pixelated Image
Pixelated Image
Predicted Image
Predicted Image
Predicted Image
Predicted Image