GROUP-3

MASKANI NAVEEN YADAV - 20201CEI0025

KATIPALLY YASHWANTH REDDY - 202021CEI0039

PERAM MAHENDRA REDDY - 20201CEI0112

INDLA MADHANKUMAR - 20201CEI0136

```
from tensorflow import keras
from keras import layers
```

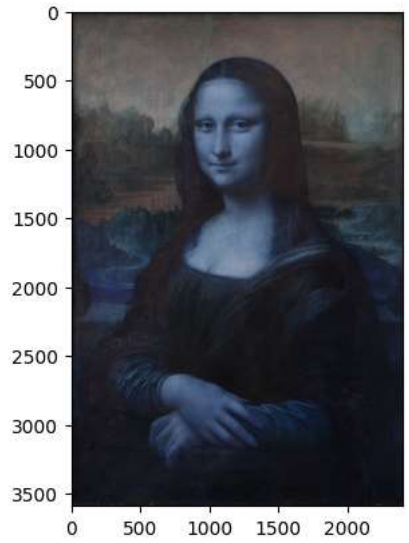
```
import matplotlib.pyplot as plt
import numpy as np
import cv2
from keras.preprocessing.image import img_to_array
from tensorflow.keras.layers import Conv2D,MaxPooling2D,UpSampling2D
from tensorflow.keras.models import Sequential
```

```
np.random.seed(42)
```

```
img_data=[]
SIZE=256
```

```
img=cv2.imread("Mona_Lisa.png",1)
plt.imshow(img,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7c2069dcde10>
```




```
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
img=cv2.resize(img,(SIZE,SIZE))
img_data.append(img_to_array(img))
img_array=np.reshape(img_data,(len(img_data),SIZE,SIZE,3))
img_array=img_array.astype('float32')/255.
```

```
model=Sequential()
model.add(Conv2D(32,(3,3),activation='relu',padding='same',input_shape=(SIZE,SIZE,3)))
model.add(MaxPooling2D((2,2),padding='same'))
model.add(Conv2D(8,(3,3),activation='relu',padding='same'))
model.add(MaxPooling2D((2,2),padding='same'))
model.add(Conv2D(8,(3,3),activation='relu',padding='same'))

model.add(MaxPooling2D((2, 2), padding='same'))

model.add(Conv2D(8, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(8, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(3, (3, 3), activation='relu', padding='same'))


model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])
model.summary()
```

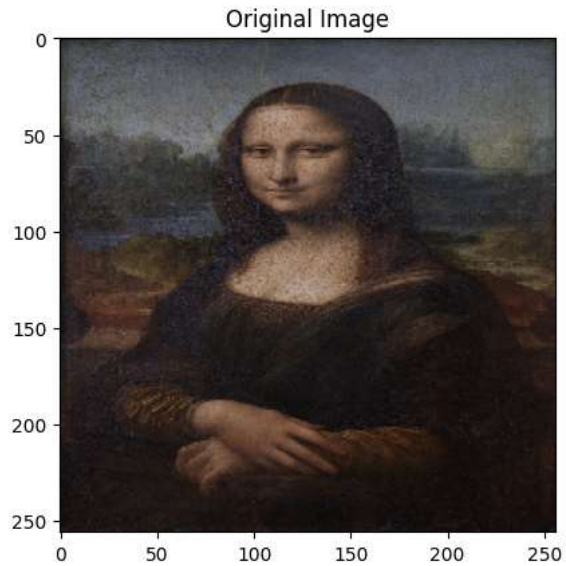
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 256, 256, 32)      896

 max_pooling2d (MaxPooling2  (None, 128, 128, 32)      0
 D)

 conv2d_1 (Conv2D)           (None, 128, 128, 8)       2312

 max_pooling2d_1 (MaxPoolin  (None, 64, 64, 8)         0
 g2D)

 conv2d_2 (Conv2D)           (None, 64, 64, 8)         584

 max_pooling2d_2 (MaxPoolin  (None, 32, 32, 8)         0
 g2D)

 conv2d_3 (Conv2D)           (None, 32, 32, 8)         584

 up_sampling2d (UpSampling2  (None, 64, 64, 8)         0
 D)

 conv2d_4 (Conv2D)           (None, 64, 64, 8)         584

 up_sampling2d_1 (UpSamplin  (None, 128, 128, 8)       0
 g2D)

 conv2d_5 (Conv2D)           (None, 128, 128, 32)      2336

 up_sampling2d_2 (UpSamplin  (None, 256, 256, 32)      0
 g2D)

 conv2d_6 (Conv2D)           (None, 256, 256, 3)       867

=================================================================
Total params: 8163 (31.89 KB)
Trainable params: 8163 (31.89 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
model.fit(img_array, img_array,epochs=5000,shuffle=True)
```

```
Epoch 4981/5000
1/1 [==============================] - 0s 314ms/step - loss: 0.0010 - accuracy: 0.8591
Epoch 4982/5000
1/1 [==============================] - 0s 307ms/step - loss: 0.0010 - accuracy: 0.8583
Epoch 4983/5000
1/1 [==============================] - 0s 330ms/step - loss: 0.0010 - accuracy: 0.8578
Epoch 4984/5000
1/1 [==============================] - 0s 335ms/step - loss: 0.0010 - accuracy: 0.8585
Epoch 4985/5000
1/1 [==============================] - 0s 313ms/step - loss: 9.9747e-04 - accuracy: 0.8591
Epoch 4986/5000
1/1 [==============================] - 0s 307ms/step - loss: 9.9422e-04 - accuracy: 0.8592
Epoch 4987/5000
1/1 [==============================] - 0s 332ms/step - loss: 9.9502e-04 - accuracy: 0.8590
Epoch 4988/5000
1/1 [==============================] - 0s 302ms/step - loss: 9.9753e-04 - accuracy: 0.8587
Epoch 4989/5000
1/1 [==============================] - 0s 381ms/step - loss: 0.0010 - accuracy: 0.8580
Epoch 4990/5000
1/1 [==============================] - 0s 451ms/step - loss: 0.0010 - accuracy: 0.8578
Epoch 4991/5000
1/1 [==============================] - 0s 412ms/step - loss: 0.0010 - accuracy: 0.8587
Epoch 4992/5000
1/1 [==============================] - 0s 451ms/step - loss: 9.9961e-04 - accuracy: 0.8592
Epoch 4993/5000
1/1 [==============================] - 0s 452ms/step - loss: 0.0010 - accuracy: 0.8586
Epoch 4994/5000
1/1 [==============================] - 0s 415ms/step - loss: 0.0010 - accuracy: 0.8582
Epoch 4995/5000
1/1 [==============================] - 0s 448ms/step - loss: 0.0010 - accuracy: 0.8576
Epoch 4996/5000
1/1 [==============================] - 0s 413ms/step - loss: 0.0010 - accuracy: 0.8575
Epoch 4997/5000
1/1 [==============================] - 0s 417ms/step - loss: 0.0010 - accuracy: 0.8580
Epoch 4998/5000
1/1 [==============================] - 0s 406ms/step - loss: 9.9963e-04 - accuracy: 0.8586
Epoch 4999/5000
1/1 [==============================] - 0s 298ms/step - loss: 9.9779e-04 - accuracy: 0.8589
Epoch 5000/5000
1/1 [==============================] - 0s 315ms/step - loss: 9.9801e-04 - accuracy: 0.8586
<keras.src.callbacks.History at 0x7c2069d1efe0>
```

```
plt.imshow(img_array[0].reshape(SIZE,SIZE,3),cmap='gray')
plt.title("Original Image")
```

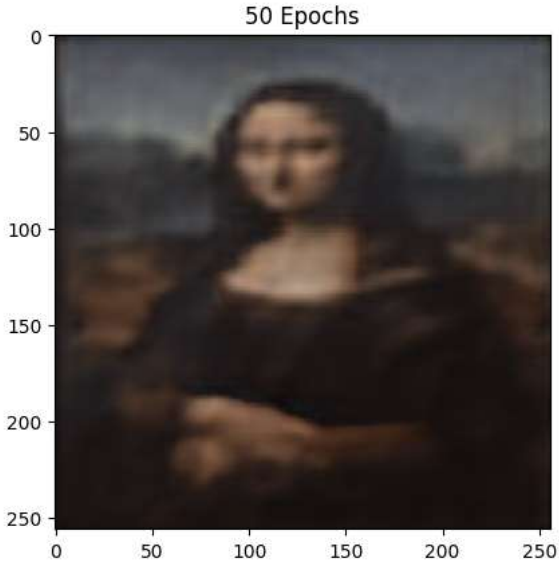
```
Text(0.5, 1.0, 'Original Image')
```



Generate Using ... randomly select 5 items from a list Close

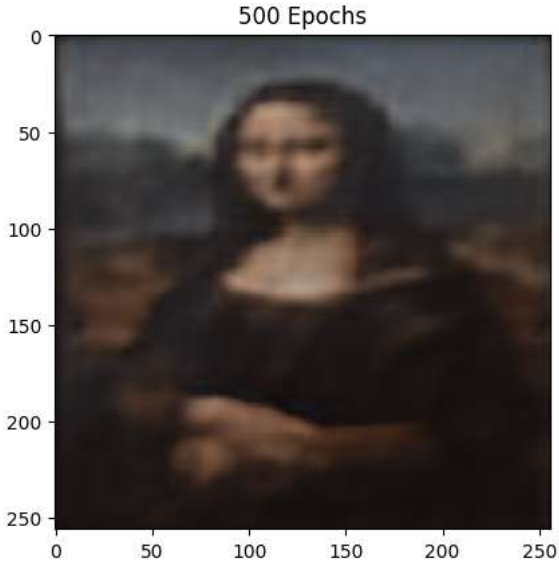
```
print("Neural network output")
pred = model.predict(img_array)

import matplotlib.pyplot as plt

plt.imshow(pred[0].reshape(SIZE,SIZE,3), cmap="gray")
plt.title("50 Epochs")
```

```
Neural network output
1/1 [==============================] - 0s 127ms/step
Text(0.5, 1.0, '50 Epochs')
```




```
print("Neural network output")
pred = model.predict(img_array)

import matplotlib.pyplot as plt

plt.imshow(pred[0].reshape(SIZE,SIZE,3), cmap="gray")
plt.title("500 Epochs")
```

```
Neural network output
1/1 [==============================] - 0s 61ms/step
Text(0.5, 1.0, '500 Epochs')
```




```
print("Neural network output")
pred = model.predict(img_array)

import matplotlib.pyplot as plt

plt.imshow(pred[0].reshape(SIZE,SIZE,3), cmap="gray")
plt.title("5000 Epochs")
```

```
Neural network output
1/1 [==============================] - 0s 60ms/step
Text(0.5, 1.0, '5000 Epochs')
```