PROJECT REPORT
ON
IMAGE SUPER RESOLUTION USING KERAS

Submitted in partial fulfillment of the requirement for the
Lab Oriented Project (CSP4401) of

COMPUTER SCIENCE AND ENGINEERING

B.E. Batch-2017

in

November-2020

Under the Guidance of:
Mr. Mukesh Kumar

Submitted By:
Himanshu Verkiya(1711981120)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
CHITKARA UNIVERSITY
HIMACHAL PRADESH

CERTIFICATE

This is to be certified that the project entitled “Image Super Resolution Using Keras” has been submitted for the Bachelor of Computer Science Engineering at Chitkara University, Himachal Pradesh during the academic semester July 2020-December 2020 is a Bonafide piece of project work carried out by “Himanshu Verkiya(1711981120)” towards the partial fulfillment for the award of the Lab Oriented Project (CSP4401) under the guidance of “Mr. Mukesh Kumar” and supervision.

Mr. Mukesh Kumar

(Assistant Professor)

CANDIDATE'S DECLARATION

I, **HIMANSHU VERKIYA (1711981120),** B.E.CSE-2017 of the Chitkara University, Himachal Pradesh hereby declare that the Lab Oriented Project Report entitled **"IMAGE SUPER RESOLUTION USING KERAS"** is an original work and data provided in the study is authentic to the best of my knowledge. This report has not been submitted to any other Institute for the award of any other course.

Himanshu Verkiya
ID No.1711981120

Place : Baddi
Date : 26/11/20

ACKNOWLEDGEMENT

It is my pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior and acts during the time of study.

I express my sincere gratitude to all for providing an opportunity to undergo Lab Oriented Project as the part of the curriculum.

I am thankful to “Mr. Mukesh Kumar ” for his support, cooperation, and motivation provided to me during the training for constant inspiration, presence and blessings.

I also extend my sincere appreciation to ***“Mr. Mukesh Kumar”*** who provided his valuable suggestions and precious time in accomplishing my project report.

Lastly, I would like to thank the almighty and my parents for their moral support and friends with whom I shared my day-to day experience and received lots of suggestions that improve my quality of work.

Himanshu Verkiya
ID No 1711981120

1. ABSTRACT

PROJECT TITLE : “IMAGE SUPER RESOLUTION USING KERAS”

PROBLEM STATEMENT :

The problem statement is quite familiar. You may all have faced problems with distorted images at some point and hence would have tried to enhance the image quality. Well, due to the advances in deep learning techniques, we’ll try to enhance the resolution of images by training a convolution neural network and using auto-encoders.

PROJECT DESCRIPTION:

In this project,we are going to implement an Auto-encoder, use Keras with Tensor-flow as its back-end to train Auto-encoder, and use this deep learning powered Auto-encoder to significantly enhance the quality of images. That is, our neural network will create high-resolution images from low-res source images.We will be using power of Google Cloud Platform to do heavy processing tasks rather than using our own system. With he help of resources provided by Google Collaboratory, we can use jupyter notebooks for extensive model training and processing of big data-sets.
We are going to train our model with the help of data-set containing 8000+ images of cars.

2. SPECIFICATIONS

2.1 HARDWARE REQUIREMENTS

To develop this application, we need a minimum hardware requirement that would help the performance of the

application better. This project has been developed with following hardware requirement:

- Processor : Dual core 2.0 GHz
- RAM : 2GB
- Hard Disk : 80 GB
- Network : 56 kilo-bits per second (Kbps)

2.2 SOFTWARE REQUIREMENTS:

- Technology : Google Cloud
- Operating System : Linux/Windows/Mac-OS
- Language : Python
- Tool : Keras,Tensor-flow,Jupyter Notebooks

Image_Super_Resolution

Image Super Resolution using Autoencoders

0.1 Project Overview and Import Libraries

[23]:
```
from tensorflow.keras.layers import Input, Dense, Conv2D, MaxPooling2D, Dropout
from tensorflow.keras.layers import Conv2DTranspose, UpSampling2D, add
from skimage.transform import resize, rescale
from tensorflow.keras.models import Model
from tensorflow.keras import regularizers
import matplotlib.pyplot as plt
from scipy import ndimage, misc
from matplotlib import pyplot
import tensorflow as tf
import numpy as np
np.random.seed(0)
import re
import os

print(tf.__version__)
```

```
2.3.0
```

0.2 What are Autoencoders?

Credit: Autoencoder Schema by Francois Chollet, 2016.

Encoder Architecture

0.3 Build the Encoder

[24]:
```
input_img = Input(shape=(256, 256, 3))
```

[25]:
```
l1 = Conv2D(64, (3, 3), padding = 'same', activation = 'relu',
            activity_regularizer = regularizers.l1(10e-10))(input_img)
```

```
l2 = Conv2D(64, (3, 3), padding = 'same', activation = 'relu',
            activity_regularizer = regularizers.l1(10e-10))(l1)

l3 = MaxPooling2D(padding = 'same')(l2)
l3 = Dropout(0.3)(l3)

l4 = Conv2D(128, (3, 3),  padding = 'same', activation = 'relu',
            activity_regularizer = regularizers.l1(10e-10))(l3)

l5 = Conv2D(128, (3, 3), padding = 'same', activation = 'relu',
            activity_regularizer = regularizers.l1(10e-10))(l4)

l6 = MaxPooling2D(padding = 'same')(l5)

l7 = Conv2D(256, (3, 3), padding = 'same', activation = 'relu',
            activity_regularizer = regularizers.l1(10e-10))(l6)

encoder = Model(input_img, l7)
```

[26]:
```
encoder.summary()
```

```
Model: "functional_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 256, 256, 3)]     0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 256, 256, 64)      1792
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 256, 256, 64)      36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 128, 128, 64)      0
_________________________________________________________________
dropout_2 (Dropout)          (None, 128, 128, 64)      0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 128, 128, 128)     73856
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 128, 128, 128)     147584
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 64, 64, 256)       295168
=================================================================
Total params: 555,328
Trainable params: 555,328
Non-trainable params: 0
```

0.4 Build the Decoder to Complete the Network

[27]:
```
l1 = Conv2D(64, (3, 3), padding = 'same', activation = 'relu',
            activity_regularizer = regularizers.l1(10e-10))(input_img)

l2 = Conv2D(64, (3, 3), padding = 'same', activation = 'relu',
            activity_regularizer = regularizers.l1(10e-10))(l1)

l3 = MaxPooling2D(padding = 'same')(l2)
l3 = Dropout(0.3)(l3)

l4 = Conv2D(128, (3, 3),  padding = 'same', activation = 'relu',
            activity_regularizer = regularizers.l1(10e-10))(l3)

l5 = Conv2D(128, (3, 3), padding = 'same', activation = 'relu',
            activity_regularizer = regularizers.l1(10e-10))(l4)

l6 = MaxPooling2D(padding = 'same')(l5)

l7 = Conv2D(256, (3, 3), padding = 'same', activation = 'relu',
            activity_regularizer = regularizers.l1(10e-10))(l6)
```

[28]:
```
l8 = UpSampling2D()(l7)

l9 = Conv2D(128, (3, 3), padding = 'same', activation = 'relu',
            activity_regularizer = regularizers.l1(10e-10))(l8)

l10 = Conv2D(128, (3, 3), padding = 'same', activation = 'relu',
             activity_regularizer = regularizers.l1(10e-10))(l9)

l11 = add([l5, l10])
l12 = UpSampling2D()(l11)

l13 = Conv2D(64, (3, 3), padding = 'same', activation = 'relu',
             activity_regularizer = regularizers.l1(10e-10))(l12)

l14 = Conv2D(64, (3, 3), padding = 'same', activation = 'relu',
             activity_regularizer = regularizers.l1(10e-10))(l13)

l15 = add([l14, l2])

decoded = Conv2D(3, (3, 3), padding = 'same',
                 activation = 'relu', activity_regularizer = regularizers.
 ↪l1(10e-10))(l15)
```

```
autoencoder = Model(input_img, decoded)
autoencoder_hfenn = Model(input_img, decoded)
```

[29]:
```
autoencoder.summary()
```

```
Model: "functional_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 256, 256, 3)      0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 256, 256, 64)      1792
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 256, 256, 64)      36928
_________________________________________________________________
max_pooling2d_6 (MaxPooling2D (None, 128, 128, 64)     0
_________________________________________________________________
dropout_3 (Dropout)          (None, 128, 128, 64)      0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 128, 128, 128      73856
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 128, 128, 128      147584
_________________________________________________________________
max_pooling2d_7 (MaxPooling2D (None, 64, 64, 128)      0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 64, 64, 256)       295168
_________________________________________________________________
up_sampling2d_2 (UpSampling2D (None, 128, 128, 256     0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 128, 128, 128      295040
_________________________________________________________________
```

```
------------------
conv2d_26 (Conv2D)              (None, 128, 128, 128 147584      conv2d_25[0][0]
__________________________________________________________________________________
------------------
add_2 (Add)                     (None, 128, 128, 128 0           conv2d_23[0][0]
                                                                 conv2d_26[0][0]
__________________________________________________________________________________
------------------
up_sampling2d_3 (UpSampling2D)  (None, 256, 256, 128 0           add_2[0][0]
__________________________________________________________________________________
------------------
conv2d_27 (Conv2D)              (None, 256, 256, 64) 73792
up_sampling2d_3[0][0]
__________________________________________________________________________________
------------------
conv2d_28 (Conv2D)              (None, 256, 256, 64) 36928       conv2d_27[0][0]
__________________________________________________________________________________
------------------
add_3 (Add)                     (None, 256, 256, 64) 0           conv2d_28[0][0]
                                                                 conv2d_21[0][0]
__________________________________________________________________________________
------------------
conv2d_29 (Conv2D)              (None, 256, 256, 3)  1731        add_3[0][0]
==================================================================================
==================
Total params: 1,110,403
Trainable params: 1,110,403
Non-trainable params: 0
__________________________________________________________________________________
------------------
```

[30]:
```
autoencoder.compile(optimizer = 'adadelta', loss = 'mean_squared_error')
```

0.5 Create Dataset and Specify Training Routine

[31]:
```
def train_batches(just_load_dataset=False):

    batches = 256

    batch = 0
    batch_nb = 0
    max_batches = -1

    ep = 4
```

```
    images = []
    x_train_n = []
    x_train_down = []

    x_train_n2 = []
    x_train_down2 = []

    for root, dirnames, filenames in os.walk("data/cars_train"):
        for filename in filenames:
            if re.search("\.(jpg|jpeg|JPEG|png|bmp|tiff)$", filename):
                if batch_nb == max_batches:
                    return x_train_n2, x_train_down2
                filepath = os.path.join(root, filename)
                image = pyplot.imread(filepath)
                if len(image.shape) > 2:

                    image_resized = resize(image, (256, 256)) # Resize the␣
↪image so that every image is the same size
                    x_train_n.append(image_resized) # Add this image to the␣
↪high res dataset
                    dwn1 = rescale(image_resized, 2)
                    x_train_down.append(rescale(dwn1,0.5))
                    batch += 1
                    if batch == batches:
                        batch_nb += 1

                        x_train_n2 = np.array(x_train_n)
                        x_train_down2 = np.array(x_train_down)

                        if just_load_dataset:
                            return x_train_n2, x_train_down2

                        print('Training batch', batch_nb, '(', batches, ')')

                        autoencoder.fit(x_train_down2, x_train_n2,
                            epochs=ep,
                            batch_size=10,
                            shuffle=True,
                            validation_split=0.15)

                        x_train_n = []
                        x_train_down = []

                        batch = 0

    return x_train_n2, x_train_down2
```

0.6 Load the Dataset and Pre-trained Model

[32]:
```
x_train_n, x_train_down = train_batches(just_load_dataset = True)
```

[33]:
```
autoencoder.load_weights('data/sr.img_net.mse.final_model5.no_patch.weights.
↪best.hdf5')
```

0.7 Model Predictions and Visualizing the Results

[34]:
```
encoder.load_weights('data/encoder_weights.hdf5')
```

[35]:
```
encoded_imgs = encoder.predict(x_train_down)
```

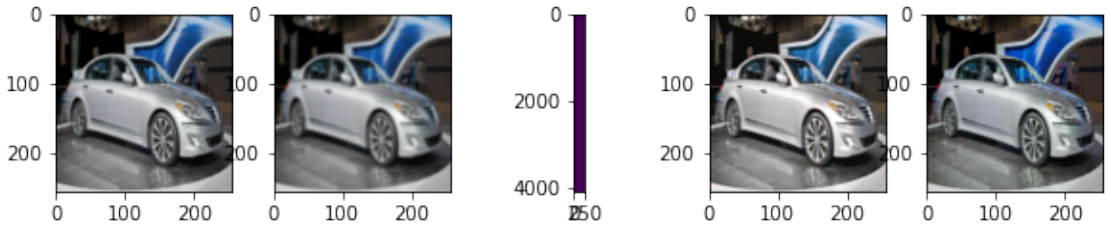
[36]:
```
encoded_imgs.shape
```

[36]:
```
(256, 64, 64, 256)
```

[37]:
```
sr1 = np.clip(autoencoder.predict(x_train_down), 0.0, 1.0)
```

[38]:
```
image_index = 251
```

[39]:
```
plt.figure(figsize = (20, 20))
i = 1
ax = plt.subplot(10, 10, i)
plt.imshow(x_train_down[image_index])
i += 1
ax = plt.subplot(10, 10, i)
plt.imshow(x_train_down[image_index], interpolation = "bicubic")
i += 1
ax = plt.subplot(10, 10, i)
plt.imshow(encoded_imgs[image_index].reshape((64*64, 256)))
i += 1
ax = plt.subplot(10, 10, i)
plt.imshow(sr1[image_index])
i += 1
ax = plt.subplot(10, 10, i)
plt.imshow(x_train_n[image_index])
plt.show()
```


0
100
200
0
100
200
0
100
200
0
2000
4000
0
100
200
0
100
200
0
100
200
0
100
200

CONCLUSION

We learned about basic functionality of auto-encoders and implemented an Image Super-Resolution enhancement task. This task could have multiple use cases in daily lifestyles. For example, we can use this technique to enhance the quality of low-resolution videos as well. So, even without labels, we can work with the image data and solve several real-world problems.

We will be using power of Google Cloud Platform to do heavy processing tasks rather than using our own system. With he help of resources provided by Google Collaboratory, we can use jupyter notebooks for extensive model training and processing of big data-sets.We are going to train our model with the help of data-set containing 8000+ images of cars.

Applications of artificial intelligence has improved quality of life in many ways whether it's healthcare or education.Hence, it can be expected to see artificial intelligence to be one of the most demanding skill in near future.People should be ready to invest their time in researching solutions to real life problems which will impact upcoming future generations.