S8_Main_Project

March 11, 2018

1 Colurization

In this Project we are gonna courize an black and white image using Convolution Neural Network.

1.1 Training And Testing

First, We are gonna create an model and test the performance and predictive power of our model.

Imports

```
In [1]: import numpy as np
        from keras.models import Sequential
        from keras.layers import InputLayer , Conv2D ,MaxPooling2D ,Dense , UpSampling2D
        from keras.preprocessing.image import img_to_array, load_img, ImageDataGenerator
        from IPython.display import display, Image
        from skimage.color import rgb2lab , lab2rgb ,rgb2gray
```

```
/usr/lib/python3/dist-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second ar
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

Loading the Dataset

```
In [2]: from keras.datasets import cifar10
        (x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

Changing RGB to Lab 24-bit RGB has 16.7 million color combinations of which human eye is capable of percieving 2-10 million colors. So we're scaling down the colorspace dividing it by 255. Then we convert image from RGB space to LAB space. After that we extracted grayscale layer(First coloum of every row) into X. After that we extracted grayscale layer(Second and third coloum of every row) into Y.

We have to extract the b&w and color from the images where x is b&w values and y is color values.

```
In [3]: train_data = np.array(x_train,dtype=float)
        test_data = np.array(x_test,dtype=float)

        train_data = train_data * 1.0 / 255
        test_data = test_data * 1.0 / 255
```

Creating a Model

```
In [4]: model = Sequential()
        model.add(InputLayer(input_shape=(None,None,1)))
        model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
        model.add(MaxPooling2D(pool_size=(2, 2), strides=2, padding='same'))
        model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
        model.add(Dense(32))
        model.add(UpSampling2D((2, 2)))
        model.add(Conv2D(2, (3, 3), activation='tanh', padding='same'))
```

Compile the Model

```
In [5]: model.compile(optimizer='rmsprop', loss='mse')
```

Fiting the model

```
In [6]: #Todo Compile and fit the model

        #Image Transformer
        datagen = ImageDataGenerator(shear_range = 0.2, rotation_range = 20, horizontal_flip =

        #Generating Traing Data
        batch_size = 10
        def generate_image(batch_size):
            for batch in datagen.flow(train_data, batch_size = batch_size):
                batch = rgb2lab(batch)
                x_batch = batch[:,:,:,0]
                y_batch = batch[:,:,:,1:]/128
                yield (x_batch.reshape(batch_size,32,32,1), y_batch.reshape(batch_size,32,32,2
        model.fit_generator(generate_image(batch_size), epochs=10, steps_per_epoch=10)
```

```
Epoch 1/10
10/10 [==============================] - 2s 179ms/step - loss: 0.9672
Epoch 2/10
10/10 [==============================] - 0s 43ms/step - loss: 0.9770
Epoch 3/10
10/10 [==============================] - 0s 44ms/step - loss: 0.9774
Epoch 4/10
10/10 [==============================] - 0s 44ms/step - loss: 0.9633
Epoch 5/10
10/10 [==============================] - 0s 43ms/step - loss: 0.9806
```

```
Epoch 6/10
10/10 [==============================] - 0s 43ms/step - loss: 1.0373
Epoch 7/10
10/10 [==============================] - 0s 42ms/step - loss: 1.0101
Epoch 8/10
10/10 [==============================] - 0s 44ms/step - loss: 0.9708
Epoch 9/10
10/10 [==============================] - 1s 56ms/step - loss: 0.9720
Epoch 10/10
10/10 [==============================] - 0s 45ms/step - loss: 0.9445
```

```
Out[6]: <keras.callbacks.History at 0x7f6dce28fc50>
```

Evaluating the model

```
In [7]: x_test = rgb2lab(test_data*1.0/255)[:,:,:,0]
        y_test = rgb2lab(test_data*1.0/255)[:,:,:,1:]/128

        x_test = x_test.reshape(len(test_data),32,32,1)
        y_test = y_test.reshape(len(test_data),32,32,2)
```

```
In [8]: #Todo write code to evaluate the accuracy of the model
        print(model.evaluate(x_test, y_test, batch_size = batch_size))
```

```
10000/10000 [==============================] - 10s 1ms/step
0.003499813812552O2
```

Save The model

```
In [9]: #Todo Save the model to a json file
        model_json = model.to_json()
        with open("model.json","w") as file:
            file.write(model_json)
        model.save_weights("model.h5")
```

1.2 Colouring an Image

Load the Model

```
In [10]: #TODO Code to load the model
```

Input the image

```
In [11]: image = img_to_array(load_img('img1.jpg'))
         image = np.array(image, dtype=float)
```

```
In [12]: X = rgb2lab(1.0/255*image)[:,:,0]
         Y = rgb2lab(1.0/255*image)[:,:,1:]
         Y /= 128
         X = X.reshape(1, len(image), len(image[0]), 1)
         Y = Y.reshape(1, len(image), len(image[0]), 2)
```

Predict

```
In [13]: output = model.predict(X)
```

Save To File

```
In [14]: from skimage.io import imsave
         output *= 128
         cur = np.zeros((len(image), len(image[0]), 3))
         cur[:,:,0] = X[0][:,:,0]
         cur[:,:,1:] = output[0]
         imsave("img_result.png", lab2rgb(cur))
         imsave("img_gray_version.png", rgb2gray(lab2rgb(cur)))
```

```
/usr/local/lib/python3.6/dist-packages/skimage/color/colorconv.py:985: UserWarning: Color data
  warn('Color data out of range: Z < 0 in %s pixels' % invalid[0].size)
/usr/local/lib/python3.6/dist-packages/skimage/util/dtype.py:122: UserWarning: Possible precis
  .format(dtypeobj_in, dtypeobj_out))
/usr/local/lib/python3.6/dist-packages/skimage/util/dtype.py:122: UserWarning: Possible precis
  .format(dtypeobj_in, dtypeobj_out))
```

Output the Image

```
In [15]: from IPython.display import display,Image
```

```
In [16]: display(Image(filename="img_result.png"))
```

1.3 Video Colurization

Black and White video is courized using the trained model.

Load the model

```
In [17]: #Todo Load the saved model
```

Read the Input File

```
In [18]: #Todo
```

Fit the model

```
In [19]: #Todo
```

Save the output to File

```
In [20]: #Todo
```