OriginalCNN

May 15, 2022

[ ]:
```
!wget http://data.csail.mit.edu/places/places205/testSetPlaces205_resize.tar.gz
```

```
--2022-05-15 15:37:59--
http://data.csail.mit.edu/places/places205/testSetPlaces205_resize.tar.gz
Resolving data.csail.mit.edu (data.csail.mit.edu)… 128.52.129.40
Connecting to data.csail.mit.edu (data.csail.mit.edu)|128.52.129.40|:80…
connected.
HTTP request sent, awaiting response… 200 OK
Length: 2341250899 (2.2G) [application/octet-stream]
Saving to: 'testSetPlaces205_resize.tar.gz'

testSetPlaces205_re 100%[===================>]   2.18G  35.4MB/s    in 66s

2022-05-15 15:39:05 (33.7 MB/s) - 'testSetPlaces205_resize.tar.gz' saved
[2341250899/2341250899]
```

[ ]:
```
!tar -xzf testSetPlaces205_resize.tar.gz
```

[ ]:
```
import tensorflow
from tensorflow.keras.layers import Conv2D, UpSampling2D
from tensorflow.keras.layers import Activation, Dense, Dropout, Flatten
from tensorflow.keras.layers import Normalization
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import InputLayer
from tensorflow.keras.preprocessing.image import ImageDataGenerator,␣
↪array_to_img, img_to_array, load_img
from skimage.color import rgb2lab, lab2rgb, rgb2gray
from skimage.io import imsave
import numpy as np
import random
import PIL
from PIL import Image
```

[ ]:
```
import shutil
```

[ ]:
```
import os
```

[ ]:
```
os.makedirs('images/blackval/class/', exist_ok=True)
os.makedirs('images/train/class/', exist_ok=True) # 40,000 images
os.makedirs('images/val/class/', exist_ok=True)   #  1,000 images
for i, file in enumerate(os.listdir('testSet_resize')):
  if(i<1000):
    img = Image.open('testSet_resize/'+file)
    imgGray = img.convert('L')
    imgGray.save(str(i)+'.jpg')
    shutil.copyfile(str(i)+'.jpg', 'images/blackval/class/'+str(i)+'.jpg')
    os.rename('testSet_resize/' + file, 'images/val/class/' + file)
  elif(i > 1000 and i<6000):
    os.rename('testSet_resize/' + file, 'images/train/class/' + file)
```

[ ]:
```
import os, os.path
a=os.listdir('images/train/class/')
number_files=len(a)
print(number_files)
```

```
4999
```

[ ]:
```
```

[ ]:
```
model = Sequential()
model.add(InputLayer(input_shape=(256, 256, 1)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', strides=2))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', strides=2))
model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(256, (3, 3), activation='relu', padding='same', strides=2))
model.add(Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(2, (3, 3), activation='tanh', padding='same'))
model.add(UpSampling2D((2, 2)))
opt = tensorflow.keras.optimizers.Adam(learning_rate=0.001)
model.compile(loss='MSE', optimizer=opt)
```

[ ]:
```
import os
Y=[]
for filename in os.listdir('images/val/class/'):
    Y.append(img_to_array(load_img('images/val/class/'+filename)))
```

```
Y = np.array(Y, dtype=float)
```

[ ]:
```
X=[]
i=0
for filename in os.listdir('images/train/class/'):
    X.append(img_to_array(load_img('images/train/class/'+filename)))
X = np.array(X, dtype=float)
```

[ ]:
```
# Image transformer
datagen = ImageDataGenerator(
        shear_range=0.2,
        zoom_range=0.2,
        rotation_range=20,
        horizontal_flip=True)

split = int(0.90*len(X))
Xtrain = X[:split]
Xtrain = 1.0/255*Xtrain
```

[ ]:
```

```

[ ]:
```
batch_size = 50
def image_a_b_gen(batch_size):
    for batch in datagen.flow(Xtrain, batch_size=batch_size):
        lab_batch = rgb2lab(batch)
        X_batch = lab_batch[:,:,:,0]
        Y_batch = lab_batch[:,:,:,1:] / 128
        yield (X_batch.reshape(X_batch.shape+(1,)), Y_batch)
```

[ ]:
```
loss=model.fit(image_a_b_gen(batch_size), steps_per_epoch=100, epochs=10)
```

```
Epoch 1/10
100/100 [==============================] - 114s 1s/step - loss: 0.0356
Epoch 2/10
100/100 [==============================] - 114s 1s/step - loss: 0.0123
Epoch 3/10
100/100 [==============================] - 114s 1s/step - loss: 0.0120
Epoch 4/10
100/100 [==============================] - 114s 1s/step - loss: 0.0122
Epoch 5/10
100/100 [==============================] - 113s 1s/step - loss: 0.0122
Epoch 6/10
100/100 [==============================] - 114s 1s/step - loss: 0.0122
Epoch 7/10
100/100 [==============================] - 113s 1s/step - loss: 0.0120
Epoch 8/10
100/100 [==============================] - 113s 1s/step - loss: 0.0121
Epoch 9/10
```

```
100/100 [==============================] - 113s 1s/step - loss: 0.0123
Epoch 10/10
100/100 [==============================] - 113s 1s/step - loss: 0.0121
```

[ ]:
```
Xtest = rgb2lab(1.0/255*X[split:])[:,:,:,0]
Xtest = Xtest.reshape(Xtest.shape+(1,))
Ytest = rgb2lab(1.0/255*X[split:])[:,:,:,1:]
Ytest = Ytest / 128
print(model.evaluate(Xtest, Ytest, batch_size=batch_size))
```

```
10/10 [==============================] - 1s 113ms/step - loss: 0.0121
0.012103937566280365
```

[ ]:
```
color_me = []
for filename in os.listdir('images/val/class/'):
        color_me.append(img_to_array(load_img('images/val/class/'+filename)))
color_me = np.array(color_me, dtype=float)
color_me = rgb2lab(1.0/255*color_me)[:,:,:,0]
color_me = color_me.reshape(color_me.shape+(1,))# Test model
output = model.predict(color_me)
output = output * 128# Output colorizations
```

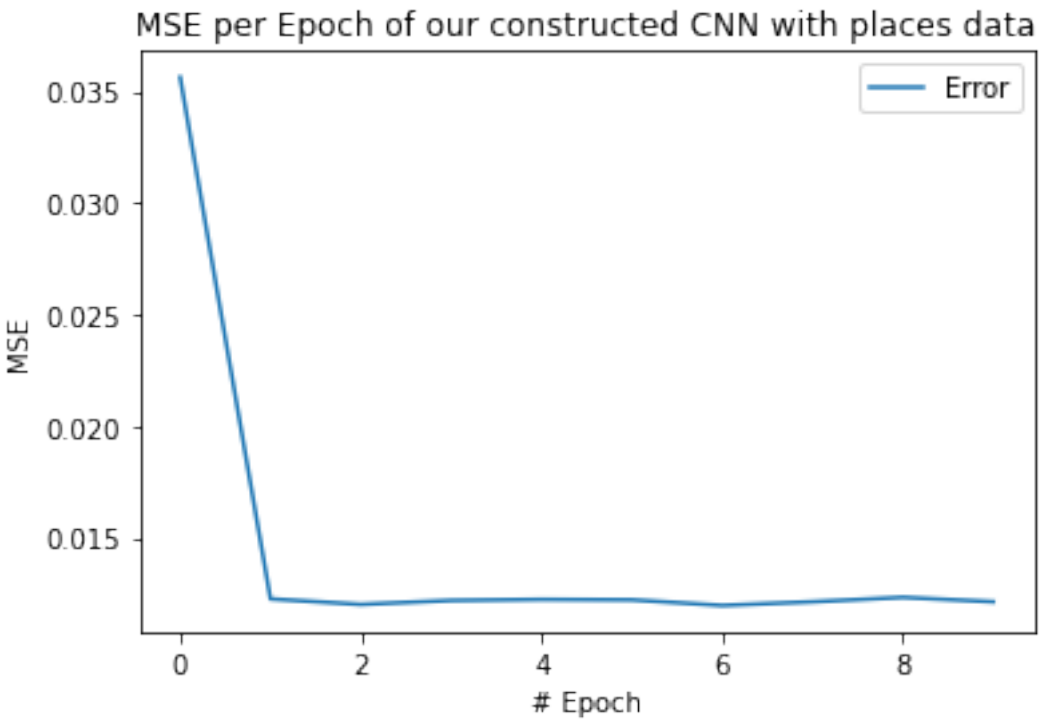
[ ]:
```

```

[ ]:
```
os.makedirs('images/output', exist_ok=True)
```

[ ]:
```
for i in range(len(output)):
        cur = np.zeros((256, 256, 3))
        cur[:,:,0] = color_me[i][:,:,0]
        cur[:,:,1:] = output[i]
        imsave("images/output"+str(i)+".png", lab2rgb(cur))
```

[ ]:
```
import matplotlib.pyplot as plt
for key in loss.history.keys():
    plt.plot(loss.history[key],label="Error")
plt.xlabel("# Epoch")
plt.ylabel("MSE")
plt.title("MSE per Epoch of our constructed CNN with places data")
plt.legend()
plt.show()
```


MSE per Epoch of our constructed CNN with places data
Error
0.035
0.030
0.025
0.020
0.015
MSE
0
2
4
6
8
Epoch

[ ]:
```
x
```