restoreimg-using-deepfillv2

December 27, 2023

#Load my model

[ ]:
```
import json, cv2, glob, os
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import keras
from keras import backend as K
import tensorflow as tf
from tensorflow.keras.models import Model
```

[ ]:
```
def mean_iou(y_true, y_pred):
    yt0 = y_true[:,:,:,0]
    yp0 = K.cast(y_pred[:,:,:,0] > 0.5, 'float32')
    inter = tf.compat.v1.count_nonzero(tf.logical_and(tf.equal(yt0, 1), tf.
 ↪equal(yp0, 1)))
    union = tf.compat.v1.count_nonzero(tf.add(yt0, yp0))
    iou = tf.where(tf.equal(union, 0), 1., tf.cast(inter/union, 'float32'))
    return iou
```

[ ]:
```
from tensorflow.keras.models import load_model

model1 = load_model('/content/model1.h5', custom_objects={'mean_iou':mean_iou})
```

[ ]:
```
test1_img = cv2.cvtColor(cv2.imread('/content/ctn_28_jpg.rf.
 ↪784d2f649da829c425ced45034dfc46d.jpg'), cv2.COLOR_BGR2RGB)
use = test1_img.copy()
use = cv2.resize(use, (160, 160))/255.
use = use[:,:,0:3]
print(use.shape)
pre = model1.predict(np.expand_dims(use, 0))
```

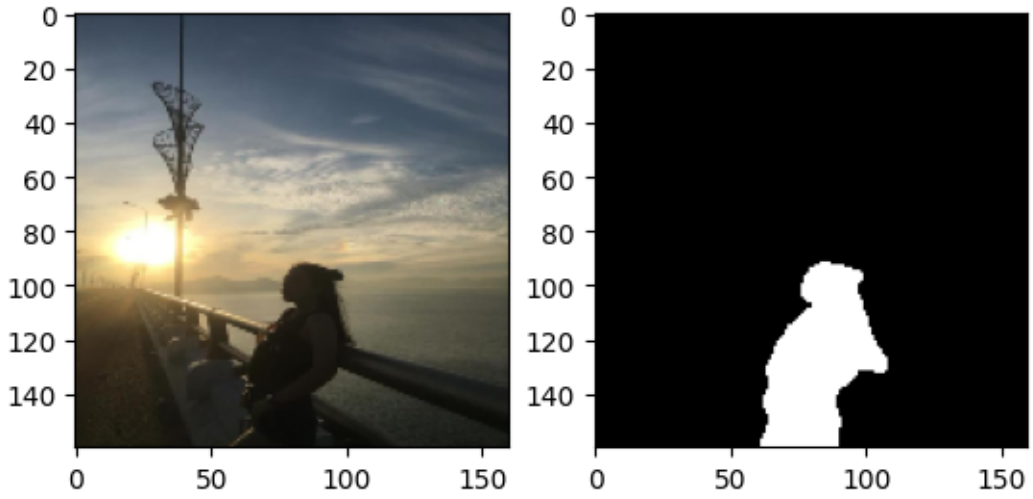
```
(160, 160, 3)
1/1 [==============================] - 0s 75ms/step
```

[ ]:
```
pre = pre.reshape((pre.shape[1],pre.shape[2]))

pre[pre >= 0.2]=1
```

```
pre[pre < 0.2] =0
```

[ ]:
```
plt.subplot(1, 2, 1)
plt.imshow(use)

plt.subplot(1, 2, 2)
plt.imshow(pre, 'gray')
```

[ ]:
```
<matplotlib.image.AxesImage at 0x78f0a74ca050>
```



[ ]:
```
rs = cv2.resize(pre, (640, 640))
plt.imsave('/content/input.png', cv2.resize(test1_img, (640, 640)))
plt.imsave('/content/mask.png', cv2.cvtColor(pre, cv2.COLOR_GRAY2RGB))
```

1 DEEPFILL-V2 DEMONSTRATION

Colab code for image inpainting.

DeepFillv2 Pytorch Repo

Original Paper

```
@article{yu2018generative,
  title={Generative Image Inpainting with Contextual Attention},
  author={Yu, Jiahui and Lin, Zhe and Yang, Jimei and Shen, Xiaohui and Lu, Xin and Huang, Thor
  journal={arXiv preprint arXiv:1801.07892},
  year={2018}
}

@article{yu2018free,
```

```
  title={Free-Form Image Inpainting with Gated Convolution},
  author={Yu, Jiahui and Lin, Zhe and Yang, Jimei and Shen, Xiaohui and Lu, Xin and Huang, Thor
  journal={arXiv preprint arXiv:1806.03589},
  year={2018}
}
```

NOTE

- The current colab code **DOES NOT** run on GPU. Has to be updated.
- The inpainting is being done after resizing the image to 512x512. This can be changed in the RESIZE_TO parameter in the *config.py* file.

1.1 SETUP

The below cell does the following-

Clone github repo: https://github.com/vrindaprabhu/deepfillv2_colab.git.
Download the model file

[ ]:
```
#@title Run this cell for setup { display-mode: "form"}
!git clone https://github.com/vrindaprabhu/deepfillv2_colab.git
!gdown "https://drive.google.com/u/0/uc?
  ↪id=1uMghKl883-9hDLhSiI8lRbHCzCmmRwV-&export=download"
!mv /content/deepfillv2_WGAN_G_epoch40_batchsize4.pth deepfillv2_colab/model/
  ↪deepfillv2_WGAN.pth
```

```
Cloning into 'deepfillv2_colab'…
remote: Enumerating objects: 99, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 99 (delta 2), reused 1 (delta 1), pack-reused 96
Receiving objects: 100% (99/99), 571.56 KiB | 5.06 MiB/s, done.
Resolving deltas: 100% (44/44), done.
Downloading…
From: https://drive.google.com/u/0/uc?id=1uMghKl883-9hDLhSiI8lRbHCzCmmRwV-&expor
t=download
To: /content/deepfillv2_WGAN_G_epoch40_batchsize4.pth
100% 64.8M/64.8M [00:01<00:00, 64.8MB/s]
```

Change to the code directory

[ ]:
```
cd deepfillv2_colab
```

```
/content/deepfillv2_colab
```

1.2 INPUTS AND MASKS

The below cell is used to obtain the input images and create/upload masks.
Please make sure that the right input and mask are correctly given, else the result may not be on the expected lines!

Example image and mask is present in `examples` folder.

```
[ ]: #@title Run to upload the input image and generate/upload masks{ display-mode:␣
      ↪"form" }
     from google.colab import files
     from ipywidgets import Button, HBox, VBox, widgets
     from IPython.display import display, clear_output
     import shutil


     from create_mask import create_bbox_mask, create_ff_mask


     class StopExecution(Exception):
         def _render_traceback_(self):
             pass


     def upload_file():
         uploaded = files.upload()
         try:
             fn = list(uploaded.keys())[0]
         except:
             print ("Please upload a valid image file!")
             raise StopExecution
         print('Uploaded file "{name}" of {length} bytes'.
      ↪format(name=fn,length=len(uploaded[fn])))
         return fn


     def on_button_clicked(b):
         with output:
             if b.description == "upload":
                 clear_output()
                 fn = upload_file()
                 shutil.move(fn, "./input/mask.png")

             if b.description == 'random free-form':
                 create_ff_mask()
                 clear_output()
                 print("random free form mask created and saved in input folder")

             if b.description == 'random bbox':
                 create_bbox_mask()
                 clear_output()
                 print("random bounding box mask created and saved in input folder")
```

```
    print ("")
    print ("")
    print ("PLEASE RUN THE NEXT CELL")


print ("UPLOAD INPUT FILE")
fn = upload_file()
shutil.move(fn, "./input/input_img.png")

output = widgets.Output()
print ("")
print ("")
print ("SELECT MASK TYPE TO INPAINT")
words = ['random free-form', 'random bbox', 'upload']
items = [Button(description=w) for w in words]
display(HBox([items[0], items[1], items[2]]), output)

items[0].on_click(on_button_clicked)
items[1].on_click(on_button_clicked)
items[2].on_click(on_button_clicked)
```

```
UPLOAD INPUT FILE
```

```
<IPython.core.display.HTML object>
```

```
Saving input (5).png to input (5).png
Uploaded file "input (5).png" of 308841 bytes
```

```
SELECT MASK TYPE TO INPAINT
```

```
HBox(children=(Button(description='random free-form', style=ButtonStyle()),␣
 ↪Button(description='random bbox', …
```

```
Output()
```

```
PLEASE RUN THE NEXT CELL
```

1.3 INPAINT!!

[ ]:
```
#@title Run to trigger inpainting. { display-mode: "form" }
!python inpaint.py
```

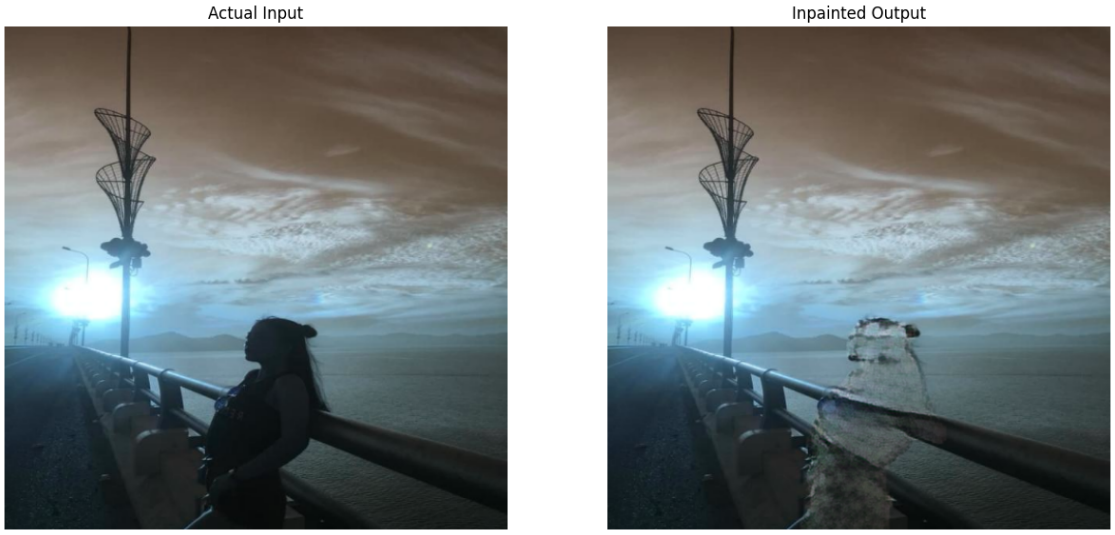
```
-- Generator is created! --
-- Initialized generator with xavier type --
-- INPAINT: Loading Pretrained Model --
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557:
UserWarning: This DataLoader will create 8 worker processes in total. Our
```

```
suggested max number of worker in current system is 2, which is smaller than
what this DataLoader is going to create. Please be aware that excessive worker
creation might get DataLoader running slow or even freeze, lower the worker
number to avoid potential slowness/freeze if necessary.
  warnings.warn(_create_warning_msg(
-- Inpainting is finished --
```

1.4 OUTPUT COMPARISION

[ ]:
```
#@title Run to check the output.{ display-mode: "form" }
import cv2
import matplotlib.pyplot as plt

resize_size = (512,512)

input_image = cv2.imread("input/input_img.png")
output_image = cv2.imread("output/inpainted_img.png")

f, axarr = plt.subplots(1,2, figsize=(15,15))
axarr[0].imshow(cv2.resize(input_image, resize_size))
axarr[0].title.set_text('Actual Input')
axarr[0].axis('off')

axarr[1].imshow(cv2.resize(output_image, resize_size))
axarr[1].title.set_text('Inpainted Output')
axarr[1].axis('off')
```

[ ]:
```
(-0.5, 511.5, 511.5, -0.5)
```



Upload new images and run the trigger cell to observe outputs on different images.

[ ]:
```
img = cv2.imread('/content/deepfillv2_colab/input/mask.png')
plt.imshow(img)
print(img.shape)
```

```
(160, 160, 3)
```