17. Semantic segmentation: Github resources

Whenever one desires to try out some advanced technique not yet available as a nicely packaged tool like scikit-image, the best solution is to first search for open-source code that approximates what one wants to do. One of the main repositories of such code is Github (https://github.com/). As an examples, we will here do semantic segmentation, i.e. segmenting objects in an image.

In [1]:
```
import sys
import numpy as np
import skimage
import skimage.io
import skimage.transform
from matplotlib import pyplot as plt
```

17.1 Finding and exploring a repository

Let's have a look at this repository (https://github.com/bonlime/keras-deeplab-v3-plus).

17.2 Installing

We follow the instructions as given. We first check what version of tensorflow we have:

In [2]:
```
import tensorflow
```

In [3]:
```
tensorflow.__version__
```

Out[3]:
```
'1.14.0'
```

So we have to follow the second set of instructions. These are unix type commands that we would normally type in a terminal. As Jupyter support bash commands we can also do it right here:

In [4]:
```
%%bash
git clone https://github.com/bonlime/keras-deeplab-v3-plus/
cd keras-deeplab-v3-plus/
git checkout 714a6b7d1a069a07547c5c08282f1a706db92e20
```

```
fatal: destination path 'keras-deeplab-v3-plus' already exists and is not
an empty directory.
HEAD is now at 714a6b7... Merge branch 'master' of https://github.com/bon
lime/keras-deeplab-v3-plus
```

17.3 Making the package accessible

Since we only want to try out the package, we will simply add it's path to our current path. If we try multiple packages, this avoid over-crowding the conda environement with useless code. If we want to use it "in production" we can always install it later.

In [5]:
```
sys.path.append('keras-deeplab-v3-plus')
```

Now we can finally import the package:

In [6]:
```
from model import Deeplabv3
```

```
Using TensorFlow backend.
```

17.4 Using the network

We simply follow the instructions given in the repository to run the code. We only modify the image importation as we use a different package (skimage). As always there are some parameters set for pre-processing:

In [7]:
```
trained_image_width=512
mean_subtraction_value=127.5
```

Then we can pick the image of our choice:

In [8]:
```
image = skimage.io.imread('https://upload.wikimedia.org/wikipedia/common
s/thumb/0/0c/Cow_female_black_white.jpg/1920px-Cow_female_black_white.jp
g')
#image = skimage.io.imread('https://upload.wikimedia.org/wikipedia/commo
ns/3/33/Chat-affut.JPG')
#image = skimage.io.imread('https://upload.wikimedia.org/wikipedia/commo
ns/1/18/TrailKitty.jpg')
image = image.astype('float')
```

And run the remaining of the proposed code:

In [9]:
```
# resize to max dimension of images from training dataset
w, h, _ = image.shape
ratio = float(trained_image_width) / np.max([w, h])
resized_image = skimage.transform.resize(image,(int(ratio * w),int(ratio
* h)))
#resized_image = np.array(Image.fromarray(image.astype('uint8')).resize
((int(ratio * h), int(ratio * w))))

# apply normalization for trained dataset images
resized_image = (resized_image / mean_subtraction_value) - 1.

# pad array to square image to match training images
pad_x = int(trained_image_width - resized_image.shape[0])
pad_y = int(trained_image_width - resized_image.shape[1])
resized_image = np.pad(resized_image, ((0, pad_x), (0, pad_y), (0, 0)),
mode='constant')

# make prediction
deeplab_model = Deeplabv3()
res = deeplab_model.predict(np.expand_dims(resized_image,0))
labels = np.argmax(res.squeeze(), -1)
```

```
/usr/local/lib/python3.5/dist-packages/skimage/transform/_warps.py:105: U
serWarning: The default mode, 'constant', will be changed to 'reflect' in
skimage 0.15.
  warn("The default mode, 'constant', will be changed to 'reflect' in "
/usr/local/lib/python3.5/dist-packages/skimage/transform/_warps.py:110: U
serWarning: Anti-aliasing will be enabled by default in skimage 0.15 to a
void aliasing artifacts when down-sampling images.
  warn("Anti-aliasing will be enabled by default in skimage 0.15 to "
WARNING: Logging before flag parsing goes to stderr.
W0123 11:16:21.968451 139872335447808 deprecation_wrapper.py:119] From /u
sr/local/lib/python3.5/dist-packages/keras/backend/tensorflow_backend.py:
4074: The name tf.nn.avg_pool is deprecated. Please use tf.nn.avg_pool2d
instead.


---------------------------------------------------------------------------
--
AttributeError                            Traceback (most recent call las
t)
<ipython-input-9-a13c11e8142d> in <module>()
     14
     15 # make prediction
---> 16 deeplab_model = Deeplabv3()
     17 res = deeplab_model.predict(np.expand_dims(resized_image,0))
     18 labels = np.argmax(res.squeeze(), -1)

~/Documents/CAS_data_science/CAS_21.01.2020_Python_Image_Processing/PyIma
geCourse-master/keras-deeplab-v3-plus/model.py in Deeplabv3(weights, inpu
t_tensor, input_shape, classes, backbone, OS, alpha)
    441     b4 = BatchNormalization(name='image_pooling_BN', epsilon=1e-
5)(b4)
    442     b4 = Activation('relu')(b4)
--> 443     b4 = BilinearUpsampling((int(np.ceil(input_shape[0] / OS)), i
nt(np.ceil(input_shape[1] / OS))))(b4)
    444
    445     # simple 1x1

/usr/local/lib/python3.5/dist-packages/keras/engine/base_layer.py in __ca
ll__(self, inputs, **kwargs)
    487             # Actually call the layer,
    488             # collecting output(s), mask(s), and shape(s).
--> 489             output = self.call(inputs, **kwargs)
    490             output_mask = self.compute_mask(inputs, previous_mas
k)
    491

~/Documents/CAS_data_science/CAS_21.01.2020_Python_Image_Processing/PyIma
geCourse-master/keras-deeplab-v3-plus/model.py in call(self, inputs)
     91     def call(self, inputs):
     92         if self.upsampling:
---> 93             return K.tf.image.resize_bilinear(inputs, (inputs.sha
pe[1] * self.upsampling[0],
     94                                                        inputs.sha
pe[2] * self.upsampling[1]),
     95                                                align_corners=True)

AttributeError: module 'keras.backend' has no attribute 'tf'
```

Since we padded and reshaped the image in the pre-processing step, we have now to correct the size of the output labels:

In [ ]:
```
if pad_x > 0:
    labels = labels[:-pad_x,:]
if pad_y > 0:
    labels = labels[:, :-pad_y]
labels = skimage.transform.resize(labels,(w, h),preserve_range=True, ord
er=0)
```

17.5 Checking the output

In [ ]:
```
plt.imshow(labels)
plt.show()
plt.imshow(image[:,:,0])
plt.show()
```

In [ ]:
```
class_names = np.array(['background','aeroplane', 'bicycle', 'bird', 'bo
at',
                        'bottle', 'bus', 'car', 'cat', 'chair',
                        'cow', 'diningtable', 'dog', 'horse',
                        'motorbike', 'person', 'pottedplant',
                        'sheep', 'sofa', 'train', 'tvmonitor'])
```

In [ ]:
```
class_names[np.unique(labels).astype(int)]
```