MaCVi

October 25, 2022

Learning resource: Keras Official Website (https://keras.io/examples/vision/oxford_pets_image_s

0.1 Importing and loading the image sets

[3]:
```
import os
#importing all the functions from function.py
from function import *
from tensorflow import keras
import numpy as np
from tensorflow.keras.preprocessing.image import load_img
from IPython.display import Image, display
import numpy as np
from tensorflow.keras.preprocessing.image import load_img
from PIL import ImageOps
import random

input_dir = "MaSTr1325_images_512x384/"
target_dir = "MaSTr1325_masks_512x384/"
img_size = (512, 384)
num_classes = 4
batch_size = 32

input_img_paths = sorted(
    [
        os.path.join(input_dir, fname)
        for fname in os.listdir(input_dir)
        if fname.endswith(".jpg")
    ]
)
target_img_paths = sorted(
    [
        os.path.join(target_dir, fname)
        for fname in os.listdir(target_dir)
        if fname.endswith(".png")
        and not fname.startswith(".")
    ]
)
```

```
print("Number of samples:", len(input_img_paths))

for input_path, target_path in zip(input_img_paths[:10], target_img_paths[:10]):
    print(input_path, "|", target_path)
```

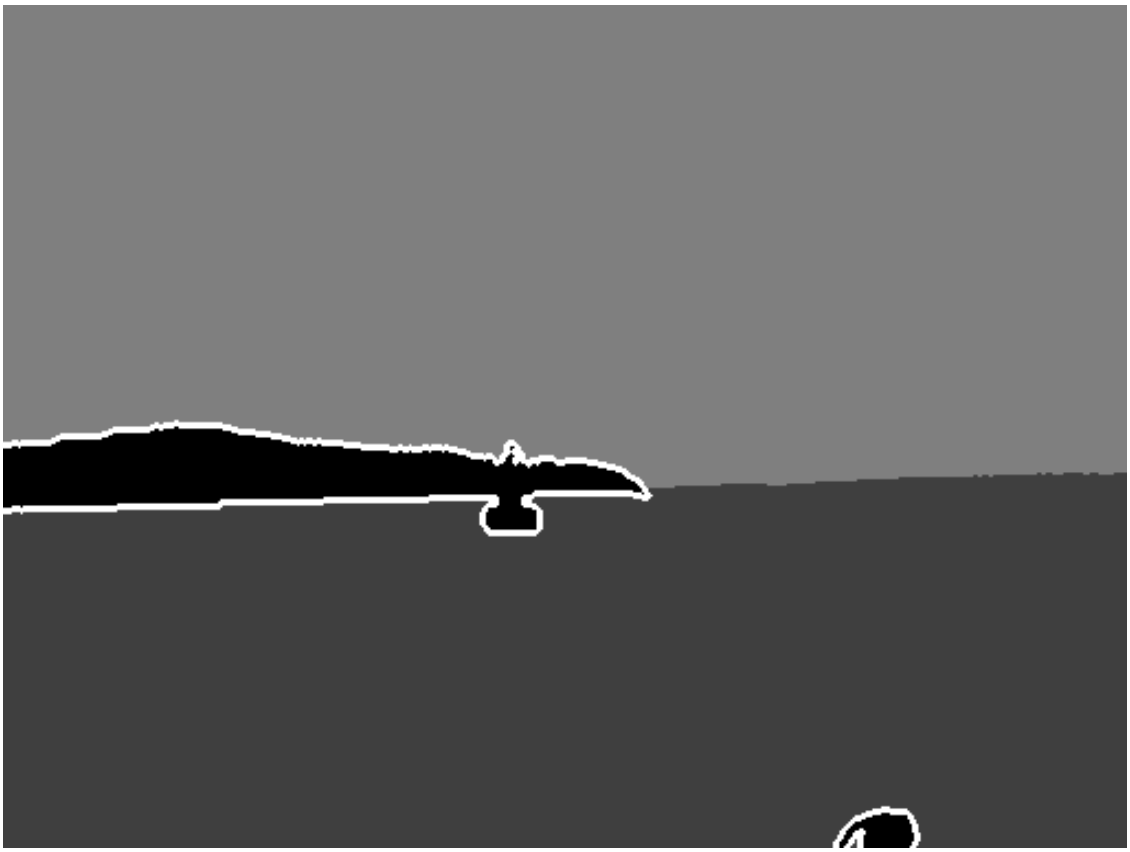
```
2022-10-25 22:45:55.835790: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```

```
Number of samples: 1325
MaSTr1325_images_512x384/0001.jpg | MaSTr1325_masks_512x384/0001m.png
MaSTr1325_images_512x384/0002.jpg | MaSTr1325_masks_512x384/0002m.png
MaSTr1325_images_512x384/0003.jpg | MaSTr1325_masks_512x384/0003m.png
MaSTr1325_images_512x384/0004.jpg | MaSTr1325_masks_512x384/0004m.png
MaSTr1325_images_512x384/0005.jpg | MaSTr1325_masks_512x384/0005m.png
MaSTr1325_images_512x384/0006.jpg | MaSTr1325_masks_512x384/0006m.png
MaSTr1325_images_512x384/0007.jpg | MaSTr1325_masks_512x384/0007m.png
MaSTr1325_images_512x384/0008.jpg | MaSTr1325_masks_512x384/0008m.png
MaSTr1325_images_512x384/0009.jpg | MaSTr1325_masks_512x384/0009m.png
MaSTr1325_images_512x384/0010.jpg | MaSTr1325_masks_512x384/0010m.png
```

0.2 Checking a image and its segmentation mask

[4]:
```
# Display input image #7
display(Image(filename=input_img_paths[9]))

# Display auto-contrast version of corresponding target (per-pixel categories)
img = ImageOps.autocontrast(load_img(target_img_paths[9]))
display(img)
```



0.3 Getting the ML model architecture, using UNet

[6]:
```
from tensorflow.keras import layers
# Build model
model = get_model(img_size, num_classes)
model.summary()
```

```
Model: "model_1"
__________________________________________________________________________________
__________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================
==================
 input_2 (InputLayer)           [(None, 512, 384, 3  0           []
                                )]

 conv2d_9 (Conv2D)              (None, 256, 192, 4)  112
['input_2[0][0]']

 batch_normalization_15 (BatchN  (None, 256, 192, 4)  16
['conv2d_9[0][0]']
```

```
 ormalization)

 activation_15 (Activation)     (None, 256, 192, 4)  0
['batch_normalization_15[0][0]']

 activation_16 (Activation)     (None, 256, 192, 4)  0
['activation_15[0][0]']

 separable_conv2d_6 (SeparableC  (None, 256, 192, 8)  76
['activation_16[0][0]']
 onv2D)

 batch_normalization_16 (BatchN  (None, 256, 192, 8)  32
['separable_conv2d_6[0][0]']
 ormalization)

 activation_17 (Activation)     (None, 256, 192, 8)  0
['batch_normalization_16[0][0]']

 separable_conv2d_7 (SeparableC  (None, 256, 192, 8)  144
['activation_17[0][0]']
 onv2D)

 batch_normalization_17 (BatchN  (None, 256, 192, 8)  32
['separable_conv2d_7[0][0]']
 ormalization)

 max_pooling2d_3 (MaxPooling2D)  (None, 128, 96, 8)  0
['batch_normalization_17[0][0]']

 conv2d_10 (Conv2D)             (None, 128, 96, 8)   40
['activation_15[0][0]']

 add_7 (Add)                    (None, 128, 96, 8)   0
['max_pooling2d_3[0][0]',
'conv2d_10[0][0]']

 activation_18 (Activation)     (None, 128, 96, 8)   0           ['add_7[0][0]']

 separable_conv2d_8 (SeparableC  (None, 128, 96, 16)  216
['activation_18[0][0]']
 onv2D)

 batch_normalization_18 (BatchN  (None, 128, 96, 16)  64
['separable_conv2d_8[0][0]']
 ormalization)

 activation_19 (Activation)     (None, 128, 96, 16)  0
```

```
['batch_normalization_18[0][0]']

 separable_conv2d_9 (SeparableC  (None, 128, 96, 16)  416
['activation_19[0][0]']
 onv2D)

 batch_normalization_19 (BatchN  (None, 128, 96, 16)  64
['separable_conv2d_9[0][0]']
 ormalization)

 max_pooling2d_4 (MaxPooling2D)  (None, 64, 48, 16)  0
['batch_normalization_19[0][0]']

 conv2d_11 (Conv2D)             (None, 64, 48, 16)   144          ['add_7[0][0]']

 add_8 (Add)                    (None, 64, 48, 16)   0
['max_pooling2d_4[0][0]',
'conv2d_11[0][0]']

 activation_20 (Activation)     (None, 64, 48, 16)   0            ['add_8[0][0]']

 separable_conv2d_10 (Separable  (None, 64, 48, 32)  688
['activation_20[0][0]']
 Conv2D)

 batch_normalization_20 (BatchN  (None, 64, 48, 32)  128
['separable_conv2d_10[0][0]']
 ormalization)

 activation_21 (Activation)     (None, 64, 48, 32)   0
['batch_normalization_20[0][0]']

 separable_conv2d_11 (Separable  (None, 64, 48, 32)  1344
['activation_21[0][0]']
 Conv2D)

 batch_normalization_21 (BatchN  (None, 64, 48, 32)  128
['separable_conv2d_11[0][0]']
 ormalization)

 max_pooling2d_5 (MaxPooling2D)  (None, 32, 24, 32)  0
['batch_normalization_21[0][0]']

 conv2d_12 (Conv2D)             (None, 32, 24, 32)   544          ['add_8[0][0]']

 add_9 (Add)                    (None, 32, 24, 32)   0
['max_pooling2d_5[0][0]',
'conv2d_12[0][0]']
```

```
 activation_22 (Activation)     (None, 32, 24, 32)   0           ['add_9[0][0]']

 conv2d_transpose_8 (Conv2DTran  (None, 32, 24, 32)  9248
['activation_22[0][0]']
 spose)

 batch_normalization_22 (BatchN  (None, 32, 24, 32)  128
['conv2d_transpose_8[0][0]']
 ormalization)

 activation_23 (Activation)     (None, 32, 24, 32)   0
['batch_normalization_22[0][0]']

 conv2d_transpose_9 (Conv2DTran  (None, 32, 24, 32)  9248
['activation_23[0][0]']
 spose)

 batch_normalization_23 (BatchN  (None, 32, 24, 32)  128
['conv2d_transpose_9[0][0]']
 ormalization)

 up_sampling2d_9 (UpSampling2D)  (None, 64, 48, 32)  0           ['add_9[0][0]']

 up_sampling2d_8 (UpSampling2D)  (None, 64, 48, 32)  0
['batch_normalization_23[0][0]']

 conv2d_13 (Conv2D)             (None, 64, 48, 32)   1056
['up_sampling2d_9[0][0]']

 add_10 (Add)                   (None, 64, 48, 32)   0
['up_sampling2d_8[0][0]',
'conv2d_13[0][0]']

 activation_24 (Activation)     (None, 64, 48, 32)   0
['add_10[0][0]']

 conv2d_transpose_10 (Conv2DTra  (None, 64, 48, 16)  4624
['activation_24[0][0]']
 nspose)

 batch_normalization_24 (BatchN  (None, 64, 48, 16)  64
['conv2d_transpose_10[0][0]']
 ormalization)

 activation_25 (Activation)     (None, 64, 48, 16)   0
['batch_normalization_24[0][0]']
```

```
 conv2d_transpose_11 (Conv2DTra  (None, 64, 48, 16)  2320
['activation_25[0][0]']
 nspose)

 batch_normalization_25 (BatchN  (None, 64, 48, 16)  64
['conv2d_transpose_11[0][0]']
 ormalization)

 up_sampling2d_11 (UpSampling2D  (None, 128, 96, 32)  0
['add_10[0][0]']
 )

 up_sampling2d_10 (UpSampling2D  (None, 128, 96, 16)  0
['batch_normalization_25[0][0]']
 )

 conv2d_14 (Conv2D)             (None, 128, 96, 16)  528
['up_sampling2d_11[0][0]']

 add_11 (Add)                   (None, 128, 96, 16)  0
['up_sampling2d_10[0][0]',
'conv2d_14[0][0]']

 activation_26 (Activation)     (None, 128, 96, 16)  0
['add_11[0][0]']

 conv2d_transpose_12 (Conv2DTra  (None, 128, 96, 8)  1160
['activation_26[0][0]']
 nspose)

 batch_normalization_26 (BatchN  (None, 128, 96, 8)  32
['conv2d_transpose_12[0][0]']
 ormalization)

 activation_27 (Activation)     (None, 128, 96, 8)   0
['batch_normalization_26[0][0]']

 conv2d_transpose_13 (Conv2DTra  (None, 128, 96, 8)  584
['activation_27[0][0]']
 nspose)

 batch_normalization_27 (BatchN  (None, 128, 96, 8)  32
['conv2d_transpose_13[0][0]']
 ormalization)

 up_sampling2d_13 (UpSampling2D  (None, 256, 192, 16  0
['add_11[0][0]']
 )                              )
```

```
 up_sampling2d_12 (UpSampling2D  (None, 256, 192, 8)  0
['batch_normalization_27[0][0]']
 )

 conv2d_15 (Conv2D)             (None, 256, 192, 8)  136
['up_sampling2d_13[0][0]']

 add_12 (Add)                   (None, 256, 192, 8)  0
['up_sampling2d_12[0][0]',
'conv2d_15[0][0]']

 activation_28 (Activation)     (None, 256, 192, 8)  0
['add_12[0][0]']

 conv2d_transpose_14 (Conv2DTra  (None, 256, 192, 4)  292
['activation_28[0][0]']
 nspose)

 batch_normalization_28 (BatchN  (None, 256, 192, 4)  16
['conv2d_transpose_14[0][0]']
 ormalization)

 activation_29 (Activation)     (None, 256, 192, 4)  0
['batch_normalization_28[0][0]']

 conv2d_transpose_15 (Conv2DTra  (None, 256, 192, 4)  148
['activation_29[0][0]']
 nspose)

 batch_normalization_29 (BatchN  (None, 256, 192, 4)  16
['conv2d_transpose_15[0][0]']
 ormalization)

 up_sampling2d_15 (UpSampling2D  (None, 512, 384, 8)  0
['add_12[0][0]']
 )

 up_sampling2d_14 (UpSampling2D  (None, 512, 384, 4)  0
['batch_normalization_29[0][0]']
 )

 conv2d_16 (Conv2D)             (None, 512, 384, 4)  36
['up_sampling2d_15[0][0]']

 add_13 (Add)                   (None, 512, 384, 4)  0
['up_sampling2d_14[0][0]',
'conv2d_16[0][0]']
```

```
 conv2d_17 (Conv2D)             (None, 512, 384, 4)  148
['add_13[0][0]']

==================================================================================
==================
Total params: 34,196
Trainable params: 33,724
Non-trainable params: 472
----------------------------------------------------------------------------------
------------------
```

0.4 Splitting the dataset into Training set, Cross Validation and Test set

[7]:
```
train_samples = int (1325 * 0.7)
val_samples = train_samples + int (1325 * 0.2)

random.Random(1337).shuffle(input_img_paths)
random.Random(1337).shuffle(target_img_paths)
train_input_img_paths = input_img_paths[:train_samples]
train_target_img_paths = target_img_paths[:train_samples]
val_input_img_paths = input_img_paths[train_samples:val_samples]
val_target_img_paths = target_img_paths[train_samples:val_samples]
test_input_img_paths = input_img_paths[val_samples:]
test_target_img_paths = target_img_paths[val_samples:]

train_gen = SeaKing(
    batch_size, img_size, train_input_img_paths, train_target_img_paths
)
val_gen = SeaKing(batch_size, img_size, val_input_img_paths,␣
 ↪val_target_img_paths)
```

0.5 Training the ML model

[6]:
```
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy")

callbacks = [
    keras.callbacks.ModelCheckpoint("SeaKing.h5", save_best_only=True)
]

# Train the model, doing validation at the end of each epoch.
epochs = 40
history = model.fit(train_gen, epochs=epochs, validation_data=val_gen,␣
 ↪callbacks=callbacks)
```

```
Epoch 1/40
28/28 [==============================] - 110s 4s/step - loss: 0.5022 - val_loss:
```

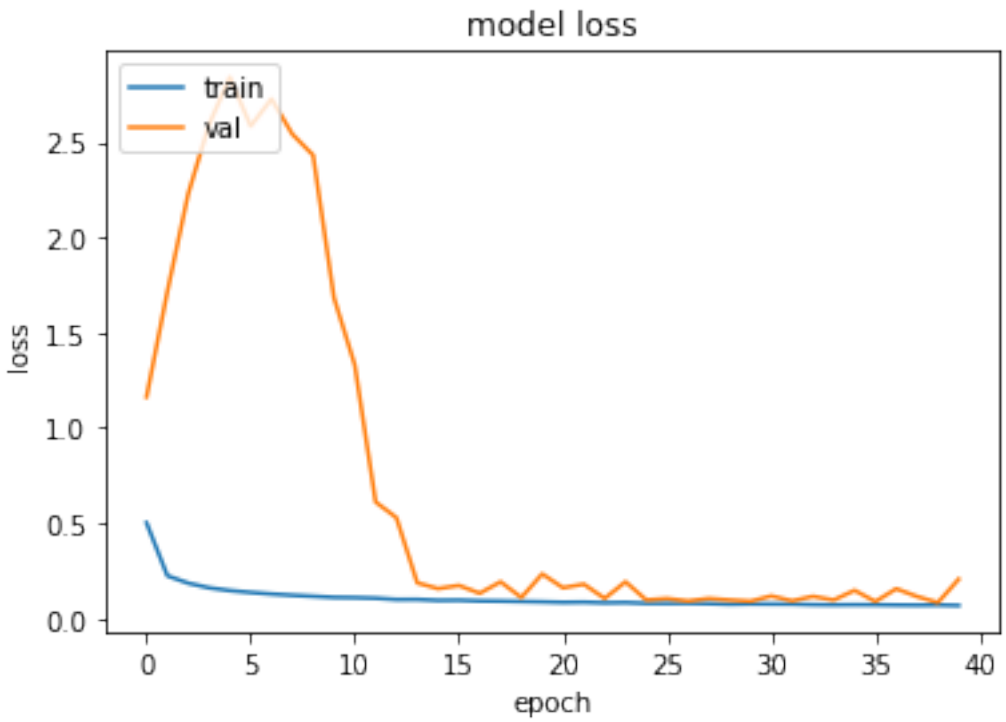
```
1.1625
Epoch 2/40
28/28 [==============================] - 107s 4s/step - loss: 0.2243 - val_loss:
1.7156
Epoch 3/40
28/28 [==============================] - 103s 4s/step - loss: 0.1855 - val_loss:
2.2282
Epoch 4/40
28/28 [==============================] - 101s 4s/step - loss: 0.1612 - val_loss:
2.5915
Epoch 5/40
28/28 [==============================] - 105s 4s/step - loss: 0.1467 - val_loss:
2.8407
Epoch 6/40
28/28 [==============================] - 108s 4s/step - loss: 0.1369 - val_loss:
2.5838
Epoch 7/40
28/28 [==============================] - 102s 4s/step - loss: 0.1286 - val_loss:
2.7267
Epoch 8/40
28/28 [==============================] - 103s 4s/step - loss: 0.1220 - val_loss:
2.5417
Epoch 9/40
28/28 [==============================] - 113s 4s/step - loss: 0.1166 - val_loss:
2.4312
Epoch 10/40
28/28 [==============================] - 104s 4s/step - loss: 0.1109 - val_loss:
1.6837
Epoch 11/40
28/28 [==============================] - 102s 4s/step - loss: 0.1096 - val_loss:
1.3304
Epoch 12/40
28/28 [==============================] - 103s 4s/step - loss: 0.1067 - val_loss:
0.6116
Epoch 13/40
28/28 [==============================] - 103s 4s/step - loss: 0.0998 - val_loss:
0.5292
Epoch 14/40
28/28 [==============================] - 106s 4s/step - loss: 0.1001 - val_loss:
0.1870
Epoch 15/40
28/28 [==============================] - 104s 4s/step - loss: 0.0953 - val_loss:
0.1570
Epoch 16/40
28/28 [==============================] - 105s 4s/step - loss: 0.0961 - val_loss:
0.1734
Epoch 17/40
28/28 [==============================] - 100s 4s/step - loss: 0.0923 - val_loss:
```

```
0.1324
Epoch 18/40
28/28 [==============================] - 98s 4s/step - loss: 0.0913 - val_loss:
0.1925
Epoch 19/40
28/28 [==============================] - 99s 4s/step - loss: 0.0892 - val_loss:
0.1102
Epoch 20/40
28/28 [==============================] - 98s 4s/step - loss: 0.0874 - val_loss:
0.2341
Epoch 21/40
28/28 [==============================] - 98s 3s/step - loss: 0.0847 - val_loss:
0.1613
Epoch 22/40
28/28 [==============================] - 98s 3s/step - loss: 0.0859 - val_loss:
0.1802
Epoch 23/40
28/28 [==============================] - 98s 4s/step - loss: 0.0825 - val_loss:
0.1054
Epoch 24/40
28/28 [==============================] - 101s 4s/step - loss: 0.0838 - val_loss:
0.1927
Epoch 25/40
28/28 [==============================] - 101s 4s/step - loss: 0.0799 - val_loss:
0.0973
Epoch 26/40
28/28 [==============================] - 99s 4s/step - loss: 0.0796 - val_loss:
0.1045
Epoch 27/40
28/28 [==============================] - 102s 4s/step - loss: 0.0801 - val_loss:
0.0918
Epoch 28/40
28/28 [==============================] - 99s 4s/step - loss: 0.0783 - val_loss:
0.1049
Epoch 29/40
28/28 [==============================] - 98s 4s/step - loss: 0.0752 - val_loss:
0.0970
Epoch 30/40
28/28 [==============================] - 98s 4s/step - loss: 0.0766 - val_loss:
0.0905
Epoch 31/40
28/28 [==============================] - 97s 3s/step - loss: 0.0761 - val_loss:
0.1182
Epoch 32/40
28/28 [==============================] - 103s 4s/step - loss: 0.0756 - val_loss:
0.0927
Epoch 33/40
28/28 [==============================] - 97s 3s/step - loss: 0.0729 - val_loss:
```

```
0.1162
Epoch 34/40
28/28 [==============================] - 98s 3s/step - loss: 0.0719 - val_loss:
0.0972
Epoch 35/40
28/28 [==============================] - 98s 4s/step - loss: 0.0725 - val_loss:
0.1478
Epoch 36/40
28/28 [==============================] - 99s 4s/step - loss: 0.0721 - val_loss:
0.0890
Epoch 37/40
28/28 [==============================] - 98s 4s/step - loss: 0.0707 - val_loss:
0.1563
Epoch 38/40
28/28 [==============================] - 98s 3s/step - loss: 0.0701 - val_loss:
0.1156
Epoch 39/40
28/28 [==============================] - 98s 3s/step - loss: 0.0708 - val_loss:
0.0835
Epoch 40/40
28/28 [==============================] - 98s 4s/step - loss: 0.0684 - val_loss:
0.2066
```

0.6 Plotting the Train and Cross-validation loss graph

[8]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



0.7 Comparing Groundtruth segmentation mask and Nural Networks's mask

[12]:
```
val_gen = SeaKing(batch_size, img_size, val_input_img_paths,␣
 ↪val_target_img_paths)
val_preds = model.predict(val_gen)

i = 11

display(Image(filename=val_input_img_paths[i]))

# Display ground-truth target mask
img = ImageOps.autocontrast(load_img(val_target_img_paths[i]))
display(img)

# Display mask predicted by our model
display_mask(i)
```

```
8/8 [==============================] - 8s 1s/step
```