Human Segmentation

16 сентября 2019 г.

[1]:
```
import os
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import json
from glob import glob
import numpy as np
import matplotlib.pyplot as plt
import math
from keras import Model
from keras.callbacks import EarlyStopping, ModelCheckpoint,␣
 ↪ReduceLROnPlateau
from keras.models import load_model
from keras.optimizers import Adam
from keras.layers import Input, Conv2D, Conv2DTranspose, MaxPooling2D,␣
 ↪concatenate, Dropout
from keras.losses import binary_crossentropy
import tensorflow as tf
import keras as keras
from keras import backend as K
from tqdm import tqdm_notebook
from keras.utils.generic_utils import get_custom_objects

from build_madel import build_madel
from mymetrics import dice_coef, dice_loss, bce_dice_loss,␣
 ↪get_iou_vector, my_iou_metric
from code_rle import encode_rle
from create_html import generate_html

TRUSTVAL = 0.4

path = "data/train"
images = os.listdir(path)
```

0.1 Loss functions

Функции для подсчета ошибок лежат в файле mymetrics.py, пользуемся метрикой Sørensen–Dice_coefficient.

[4]:
```
get_custom_objects().update({'bce_dice_loss': bce_dice_loss })
get_custom_objects().update({'dice_loss': dice_loss })
get_custom_objects().update({'dice_coef': dice_coef })
get_custom_objects().update({'my_iou_metric': my_iou_metric })
```

0.2 Build model

[7]:
```
input_layer = Input((320, 240, 3))
output_layer = build_model(input_layer, 16)
model = Model(input_layer, output_layer)
model.compile(loss=bce_dice_loss, optimizer=Adam(lr=1e-3),␣
 ↪metrics=[my_iou_metric])
model.save_weights('./keras.weights')
```

[6]:
```
imagss = [i[:-4] for i in images]

train_x = np.array([np.array(Image.open(f"{path}/{ind}.jpg"))/255 for ind␣
 ↪in imagss])
train_y = np.array([[np.array(Image.open(f"{path}_mask/{ind}.png"))/255]␣
 ↪for ind in imagss]).reshape((1315, 320, 240, 1))
```

0.3 Train model

Я пользуюсь оптимизатором Adam, он проще в настройке и показал себя лучше, чем, например, градиентный спуск. Подбирая параметры я пришел к таким настройкам:

[7]:
```
counter = 0
while True:
    history = model.fit(train_x, train_y,
                        batch_size=32,
                        epochs=1,
                        verbose=1,
                        validation_split=0.1
                       )

    if (history.history['my_iou_metric'][0] > 0.75 or counter > 4):
        break
    counter+=1
```

```
Train on 1183 samples, validate on 132 samples
Epoch 1/1
```

```
1183/1183 [==============================] - 563s 476ms/step - loss: 1.0413␣
 ↪-
my_iou_metric: 0.2917 - val_loss: 1.0251 - val_my_iou_metric: 0.3159
Train on 1183 samples, validate on 132 samples
Epoch 1/1
1183/1183 [==============================] - 554s 469ms/step - loss: 0.8234␣
 ↪-
my_iou_metric: 0.3993 - val_loss: 0.7228 - val_my_iou_metric: 0.4879
Train on 1183 samples, validate on 132 samples
Epoch 1/1
1183/1183 [==============================] - 562s 475ms/step - loss: 0.6875␣
 ↪-
my_iou_metric: 0.4937 - val_loss: 0.6424 - val_my_iou_metric: 0.5462
Train on 1183 samples, validate on 132 samples
Epoch 1/1
1183/1183 [==============================] - 565s 478ms/step - loss: 0.6202␣
 ↪-
my_iou_metric: 0.5579 - val_loss: 0.6395 - val_my_iou_metric: 0.5697
Train on 1183 samples, validate on 132 samples
Epoch 1/1
1183/1183 [==============================] - 563s 476ms/step - loss: 0.5756␣
 ↪-
my_iou_metric: 0.5852 - val_loss: 0.5913 - val_my_iou_metric: 0.5886
Train on 1183 samples, validate on 132 samples
Epoch 1/1
1183/1183 [==============================] - 560s 473ms/step - loss: 0.5802␣
 ↪-
my_iou_metric: 0.5877 - val_loss: 0.6074 - val_my_iou_metric: 0.5871
```

0.4 Save as json

[ ]:
```
val_path = 'data/valid'
val_images = os.listdir(val_path)
val_x = np.array([np.array(Image.open(f"{val_path}/{ind}"))/255 for ind␣
 ↪in val_images])
pre = model.predict(val_x)

data = {val_images[i][:-4]:encode_rle(pre[i] > TRUSTVAL) for i in␣
 ↪range(len(val_x))}

json_data = json.dumps(data)
```

[ ]:
```
with open('data.json', 'w') as f:
    json.dump(json_data, f)
```

0.5 Save as html page

[ ]:
```
for i in range(len(val_x)):
    a = (pre[i] > TRUSTVAL).astype(np.uint8)*255
    j = Image.fromarray(a.reshape((320, 240)), mode = 'L')
    j.save("data/validMask/" + val_images[i])
```

[ ]:
```
from datetime import datetime

paths_to_imgs = sorted(glob("data/valid/*"))
pred_masks = [np.array(Image.open(path)) for path in sorted(glob("data/
 ↪validMask/*"))]

_ = get_html(paths_to_imgs, pred_masks, path_to_save="results/example")
```