7.1 Web与模型服务对接逻辑

7.1 Web与模型服务对接逻辑

1、将训练好的模型，进行导出，tf.export工具

2、使用TensorFlow serving 开启模型服务

3、编写模型客户端程序，在web后台调用

7.2 模型导出

- 应用tf.saved_model.simple_save完成模型导出
- tensorflow serving开启服务：需要固定文件夹格式，模型格式pb
  - 导出模型过程简单
  - 进行热更新，关注模型迭代，新的模型直接上传到原来的路径，指定版本号即可，serving自动更新到新的模型，预测就使用新的模型
    - 解耦合

7.2.1 keras 模型进行TensorFlow导出

2、导出模型过程

- 1、路径+模型名字："./model/commodity/",以及版本
  - 以bytes形式指定
- 2、调用模型，指定读取训练好的物体检测的模型h5文件
  - model SSD300
- 3、tf.saved_model.simple_save导出
  - tf.saved_model.simple_save(sess, # 会话，keras.get_session() export_path, inputs={'images': model.input}, # 模型输入，tensor outputs={t.name: t for t in model.outputs} # 模型输出：tensor)

7.4 TF Serving

7.4.2 TensorFlow Serving Docker

- source=/home/ubuntu/detectedmodel/commodity：tf.saved_model导出的模型pb路径，最后一个文件夹名称是模型名字
- target: /models/我的模型名称， 比如：/models/commodity

- -e MODEL_NAME=commodity：模型名称
- -t tensorflow/serving: serving取运行

```
docker run -p 8501:8501 -p 8500:8500 --mount
type=bind,source=/Users/huxinghui/workspace/ml/detection/ssd_detection/ssd/ser
ving_model/commodity,target=/models/commodity -e MODEL_NAME=commodity -t
tensorflow/serving
```

7.5 TensorFlow Client对接模型服务

- result_img = make_prediction(image.read())

7.5.1.1 Client端代码

下载：tensorflow-serving-api 1.10.1

下载：grpcio 1.17.1

步骤：

- 1、获取读取后台读取的图片图片大小处理，转换数组

```
def resize_img(image, input_size):
    img = io.BytesIO()
    img.write(image)
    # 使用pillow image 接收这个图片
    rgb = Image.open(img).convert('RGB')

    # 换砖大小
    if input_size:
        rgb = rgb.resize((input_size[0], input_size[1]))

    return rgb

# - 2、图片大小处理，转换数组
resize = resize_img(image, (300, 300))
image_array = img_to_array(resize)

image_tensor = preprocess_input(np.array([image_array]))
```

- 2、打开通道channel,构建stub，预测结果
  - 对于模型服务：传给模型服务：一张图片数据4维数组）
    - 拿到预测结果：{'concat_3:0': <tf.Tensor 'concat_3:0' shape=(?, 7308, 21) dtype=float32>}
  - grpc.insecure_channel("127.0.0.1:8500")
  - stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
  - results = stub.Predict(request)

        - 构建请求：模型名字，模型签名（默认）， 模型输入
        - request = predict_pb2.PredictRequest()
        - request.model_spec.name = 'commodity'
        - request.model_spec.signature_name = signature_constants.DEFAULT_SERVING_SIGNATURE_DEF_KEY
        - request.inputs['images'].CopyFrom(tf.contrib.util.make_tensor_proto(image_tensor, [1, 300, 300, 3]))
- 4、predict_pb2进行预测请求创建
    - session运行预测结果，将tensor转换成数组

Web Server开启

5.8.2 Docker管理运行Web部分

- dockerfile
- requeirments.txt:
    - 必须包含预测的时候tensorflow serving clientAPI用到的环境

```
grpcio==1.12.0
matplotlib==2.2.2
numpy==1.14.2
pandas==0.20.3
Pillow==4.3.0
tensorflow==1.12.0
flask==1.0.2
gunicorn==19.7.1
tensorflow-serving-api==1.10.1
keras==1.2.2
```

- start.sh

使用这三个文件制作一个docker的web镜像

- 写好web程序
- docker使用tensorflow-web镜像开启web程序

```
docker run -t -p 80:5000 -v
/Users/huxinghui/workspace/ml/detection/ssd_detection/web_code:/app --
name=web tf-serving-web
```

https://github.com/grpc/grpc/issues/12116 python3.6版本

7.7 项目接口与百度机器人对接

7.7.1 百度服务机器人介绍

7.7.2 接口对接百度修改

- 修改web业务逻辑：业务逻辑跟模型服务解耦合

```
# 1、接到请求数据，然后进行数据解析
request.json
百度：requestId，  image
requestId = req['requestId']
image = req['image']

# 2、定义一个新的预测接口，返回预测结果
输入:图片二进制，输出：y_predict，经过NMS结果

# 3、得到预测结果，按照百度物体检测协议说明，返回指定格式
resp = {
    "result":[]
}
    for i in range(y_predict[0][:, 1].shape[0]):
        resp['result'].append({"score": y_predict[0][:, 1][i], "root":" ",
                               "keyword": VOC_LABELS[str(y_predict[0][:, 0]
[i])]})
    resp['extInfos'] = {}
    resp['filterThreshold'] = 0.7
    resp['resultNum'] = y_predict[0][:, 1].shape[0]
    resp['requestId'] = requestId
```