Tensorflow Deployment.

Serving Tensorflow model with TF Serving.

① Prepare your model:

↳ Suppose you have an ArcFace model with the input tensors named
- input_1. (None x 170x170 x 3). → Image dimensions.
- input_2. (None x 93742). → Number of classes.

↳ The model needs to be saved in tf SavedModel format:

```
>>> arcface.load_weights("model.weights.h5")
>>> arcface.summary()  # To see the input tensors' names.
>>> tf.keras.models.save_model(arcface, "<abs_path>/model_name/<version #>")
```

- save_model → saves models in tf SavedModel format.
- arcface → ③ The model.
- ④ Model path:
  - +1. Abs path to model directory
  - +2. Model dir name.
  - +3. Version number.

↳ Example:

```
tf.keras.models.save_model(arcface, "/home/hieu/arcface/1")
```

```
/home/hieu
  └ /arcface
      └ /1
          ├ /assets
          ├ saved_model.pb
          └ /variables.
```

② Installing docker tf serving.

2.1. Installing Docker.

↳ Follow docs.docker.com/engine/install/ubuntu.

2.2. Installing TF serving.

↳
```
$ docker pull tensorflow/serving
```

⑤ Serving the model.

↳
```
$ sudo docker run -t --rm -p {LOCAL_PORT}:8501 \
-v "<abs_path>/model_name:/models/model_name" \
-e "MODEL_NAME=model_name" \
tensorflow/serving
```

↳ For example: model_name is arcface_keras.

```
<abs_path>
  └ /arcface_keras
      └ /1
          ├ /assets , /variables
          └ saved_model.pb
```

Scanned with CamScanner

=> docker command:

```
$ docker run -t --rm -p 8502:8501
                -v "<abs_path>/arcface_keras:/models/arcface_keras"
                -e "MODEL_NAME=arcface_keras"
                tensorflow/serving
```

↳ Now the model can be access for prediction at:

* "http://localhost:8502/v1/models/arcface_keras:predict"
  - 8502 → port #
  - v1 → ver #
  - arcface_keras → model name.

④ Using the model.

↳ You can send data for prediction using python requests. Eg:

```
import cv2
import json
import time
import requests
import numpy as np

url = "http://localhost:8502/v1/models/arcface_keras:predict"
test_img = cv2.imread("2.jpg")

def make_request(model_url, img):
    img = cv2.resize(img, (170,170))
    labl = np.zeros(93742)

    data = json.dumps({
        "signature_name": "serving_default",
        "instances": [{
            "input_1": img.tolist(),
            "input_2": labl.tolist()
        }]
    })
    headers = {"content-type": "application/json"}
    r = requests.post(model_url, data=data, headers=headers)

    predictions = np.array(json.loads(r.text)["predictions"])
    return predictions.
```

① Input tensors names.

② inputs have to be plain lists.

Scanned with CamScanner