Thread DL production notes

endlessforms 12/13/22

Design pain points

1. The scan image size
   a. 4k x 20k pixels ~ 60MB, this is a lot of data to pass to an ML model to infer
   b. Deep Learning model is large - artifacts are within the 100MB range and are not very nimble.
2. Imbedding ML model (Python, Tensorflow) into C#.NET project is challenging
      i. I haven't found a good example of it.
3. Model can’t be retrained as quickly as needed with current system.
4. No way to monitor model drift after deploying into production environment.
   a. Model could become progressively less accurate on new pipe species.
5. No way to centralize multiple scans from various clients.

It may be beneficial to host our Deep Learning model on a cloud-based server to alleviate these pain points.

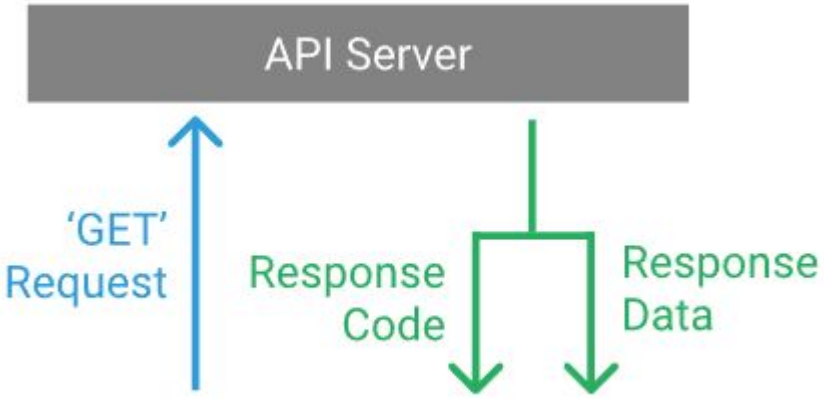
Creating Thread Defect API

What is an API?

An API, or Application Programming Interface, is a server that you can use to retrieve and send data to using code. APIs are most commonly used to retrieve data, and that will be the focus of this beginner tutorial.

When we want to receive data from an API, we need to make a request. Requests are used all over the web.

We will send thread scan images to an API and return our model's predictions.



https://www.dataquest.io/blog/python-api-tutorial/

Thread API

```python
from fastapi import FastAPI, File, UploadFile
import tensorflow as tf
import json
from model_definition import SegmentationModel

"""
Deep Learning Thread Defect Detector
Use this to create a route to send thread scan images for model prediction

"""

app = FastAPI()

model = SegmentationModel().model
model.load_weights('UNET_256x256_20nov_2022_final_weights.h5')

@app.post('/')

async def scoring_endpoint(data: UploadFile = File(...)):
    image_bytes = await data.read()
    image = tf.io.decode_image(image_bytes)
    yhat = model.predict(tf.expand_dims(image, axis=0))

    return {"prediction": json.dumps(yhat.tolist())}
```

Libraries used, including segmentation model.

Invoke FastAPI

Load model weights and class.

Define endpoint. Image is loaded, model makes prediction, returns JSON packet.

Testing Locally

In terminal:

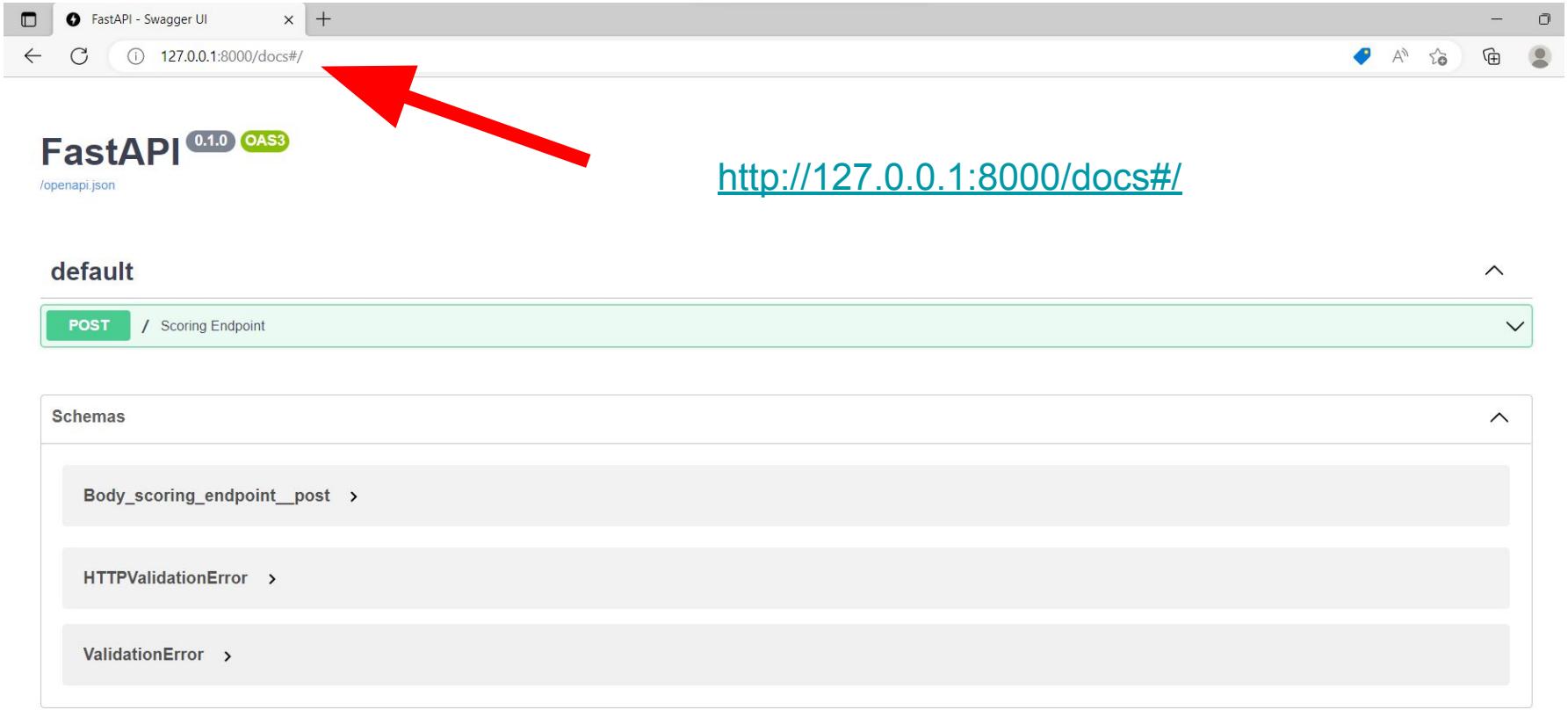
uvicorn api:app --reload

```
INFO:     Will watch for changes in these directories: ['C:\\Users\\sauce\\OneDrive\\Desktop\\ml thread segmentation']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [12468] using StatReload
2022-12-13 15:37:45.335673: 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:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
INFO:     Started server process [7736]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

No endpoint yet, so nothing is returned.

{"detail":"Method Not Allowed"}

Testing Locally - swagger UI to test endpoints



expand tab…

Try it out

Looking for form-data

FastAPI 0.1.0 OAS3

/openapi.json

default

POST / Scoring Endpoint

Parameters

Try it out

No parameters

Request body required

multipart/form-data

data * required

string($binary)

Responses

Code	Description	Links
200	Successful Response	*No links*

Testing Locally - swagger UI to test endpoints

data * required

string($binary)

Choose File 1694_test_image.bmp

Loading test image

Execute

Clear

execute

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'data=@1694_test_image.bmp;type=image/bmp'
```

Request URL

```
http://127.0.0.1:8000/
```

Server response

Code | Details

200

Response body

Returns prediction JSON packet.


```
{
  "prediction": "[[[[0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1
0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0,
0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0,
0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0,
0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0,
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0,
0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0,
0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.
0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 6.838661061998643e-23, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0,
0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
```

Home Workspaces API Network Explore Search Postman Invite

y Workspace New Import Import Elements POST https://blooming-atoll GET Untitled Request

Untitled Request

My first collection
First folder inside collection
Second folder inside collection

Create a collection for your reques

A collection lets you group related reques
and easily set common authorization, tes
scripts, and variables for all requests in i

Create Collection

Import

File Folder Link Raw text Code repository New API Gateway New

Paste raw text

```
curl -X 'POST' \
  'http://127.0.0.1:8000/' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'data=@1694_test_image.bmp;type=image/bmp'
```

Use POSTMAN to format endpoint

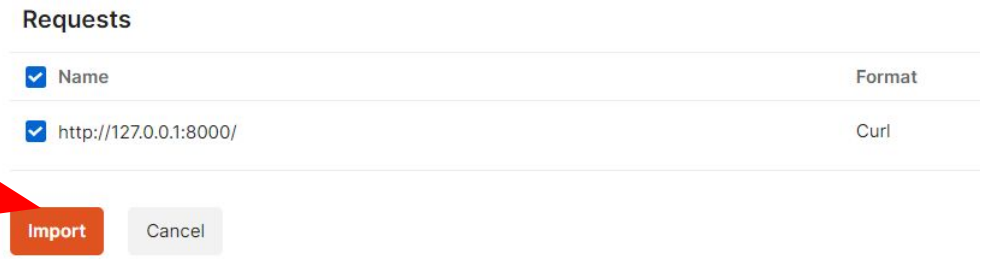
Continue

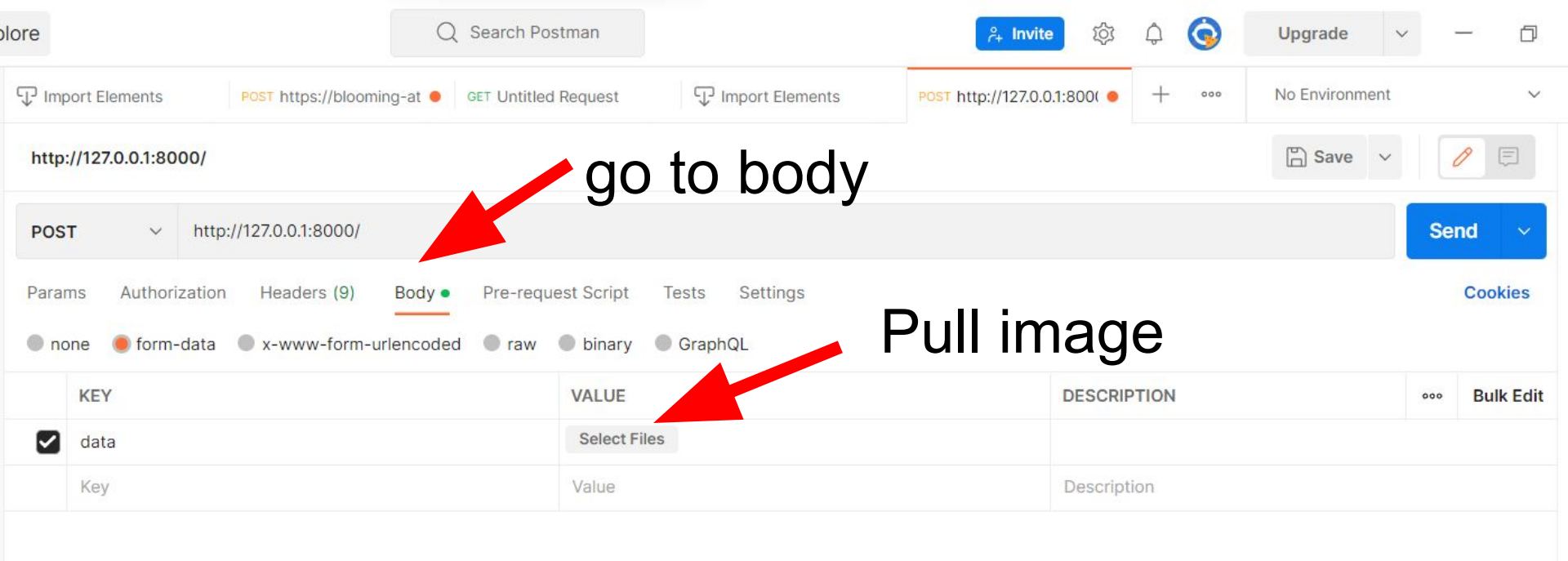
Postman

Software

Postman is an API platform for developers to design, build, test and iterate their APIs. As of April 2022, Postman reports having more than 20 million registered users and 75,000 open APIs, which it says constitutes the world's largest public API hub.

Generates Core to test endpoint locally





http://127.0.0.1:8000/

Save

POST http://127.0.0.1:8000/ Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
☑	data	1694_test_image.bmp			
	Key	Value	Description		

returns prediction.

Body Cookies Headers (4) Test Results Status: 200 OK Time: 556 ms Size: 2.32 MB Save Response

Pretty Raw Preview Visualize JSON


```
{
    "prediction": "[[[[0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.
        0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.
        0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.
        0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]
        , [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.
        0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.
        0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.
        0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.
        0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.
        0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.
        0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.
```

Test this request format. We need to make sure image is properly loaded and prediction is properly formatted.

nport Elements | POST http://127.0.0.1:800(| No Environment

Save | Send | Cookies

CRIPTION | Bulk Edit

cription

s 2.32 MB | Save Response

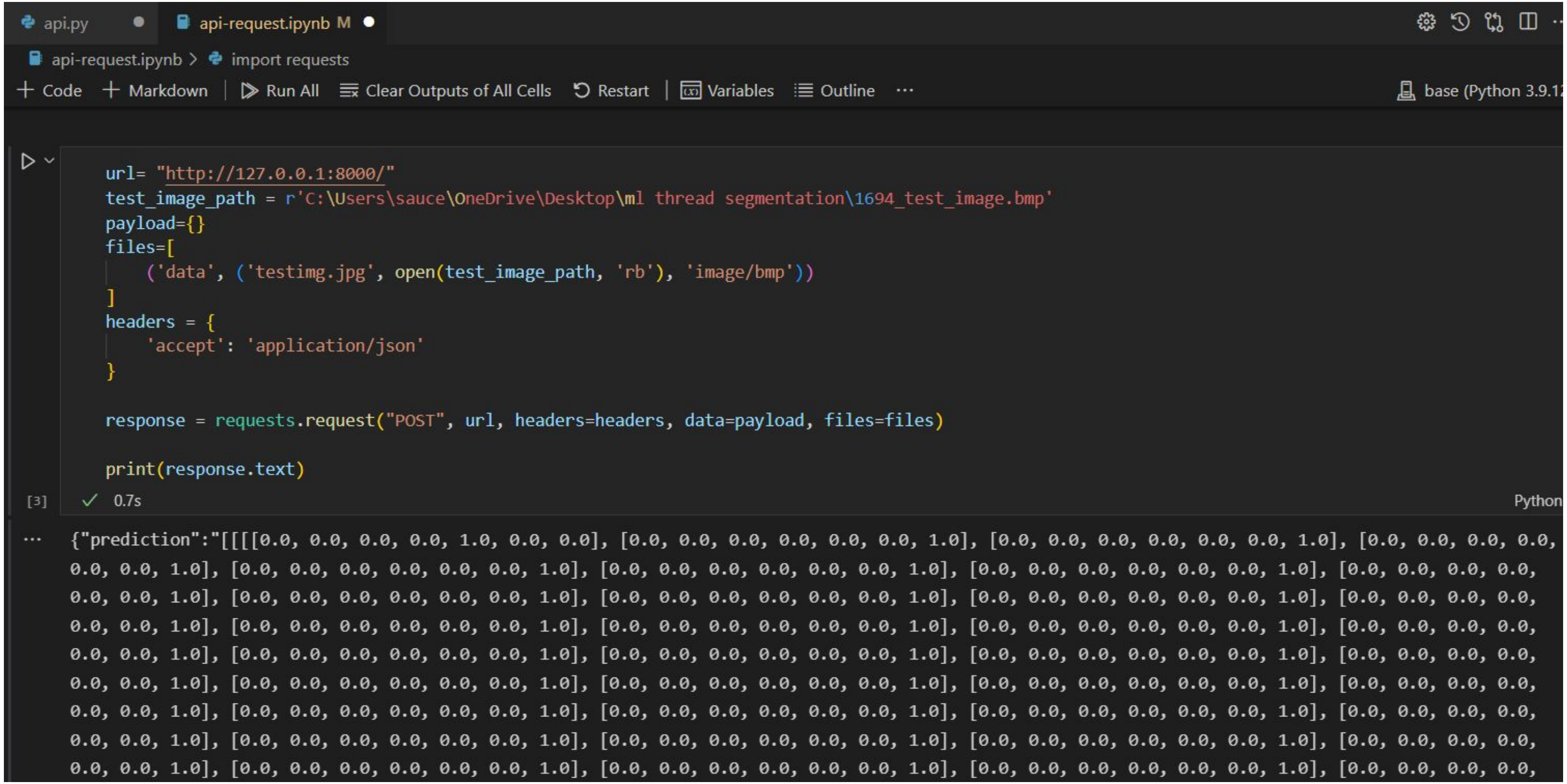
, 0.0, 0.0, 0.0, 0.0, 0.
, 0.0, 0.0, 0.0, 0.0, 1.
0.0, 0.0, 0.0, 1.0]

Code snippet

Python - Requests

```python
import requests

url = "http://127.0.0.1:8000/"

payload={}
files=[
  ('data',('1694_test_image.bmp',open('/C:/Users/sauce/OneDrive/Desktop/ml thread segmentation/1694_test_image.bmp','rb'),'image/bmp'))
]
headers = {
  'accept': 'application/json'
}

response = requests.request("POST", url, headers=headers, data=payload, files=files)

print(response.text)
```

Test in Jupyter Notebook - request looks good, need to format



```python
url= "http://127.0.0.1:8000/"
test_image_path = r'C:\Users\sauce\OneDrive\Desktop\ml thread segmentation\1694_test_image.bmp'
payload={}
files=[
    ('data', ('testimg.jpg', open(test_image_path, 'rb'), 'image/bmp'))
]
headers = {
    'accept': 'application/json'
}

response = requests.request("POST", url, headers=headers, data=payload, files=files)

print(response.text)
```

```
{"prediction":"[[[[0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0], [0.0, 0.0, 0.0, 0.0,
```

Format output as numpy array

```
"""add np array wrapper, add thresholding """

import numpy as np
yhat = np.array(json.loads(prediction['prediction']))
yhat = np.where(yhat > 0.3, 1.0, 0.0)

yhat
```

[6] ✓ 0.2s

```
Output exceeds the size limit. Open the full output data in a text editor
array([[[[0., 0., 0., ..., 1., 0., 0.],
         [0., 0., 0., ..., 0., 0., 1.],
         [0., 0., 0., ..., 0., 0., 1.],
         ...,
         [0., 0., 0., ..., 0., 0., 0.],
         [0., 0., 0., ..., 0., 0., 0.],
         [0., 0., 0., ..., 0., 0., 1.]],
```

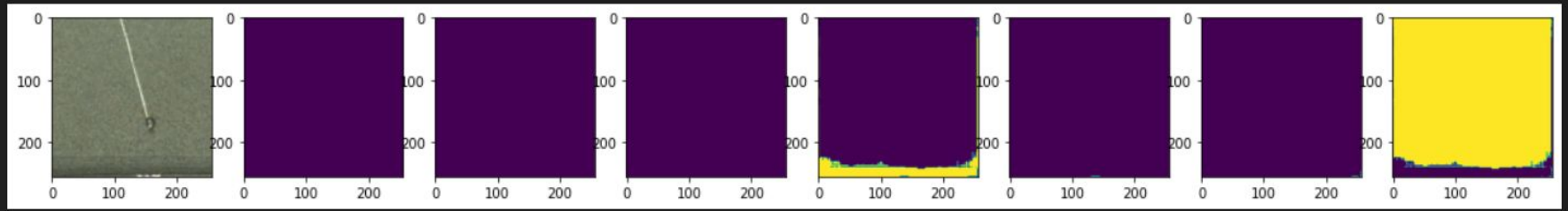
Predictions are *interpretable*. Need to update model weights. Note, model performs poorly here, but we're just making the pipeline right now… (we will use ML Ops to optimize our model once the pipeline is working.)

```python
x = cv2.imread(test_image_path)
# squeeze prediction
yhat = np.array(json.loads(prediction['prediction']))

#yhat = np.squeeze(np.where(yhat > 0.3, 1.0, 0.0))
yhat = np.squeeze(yhat)
```

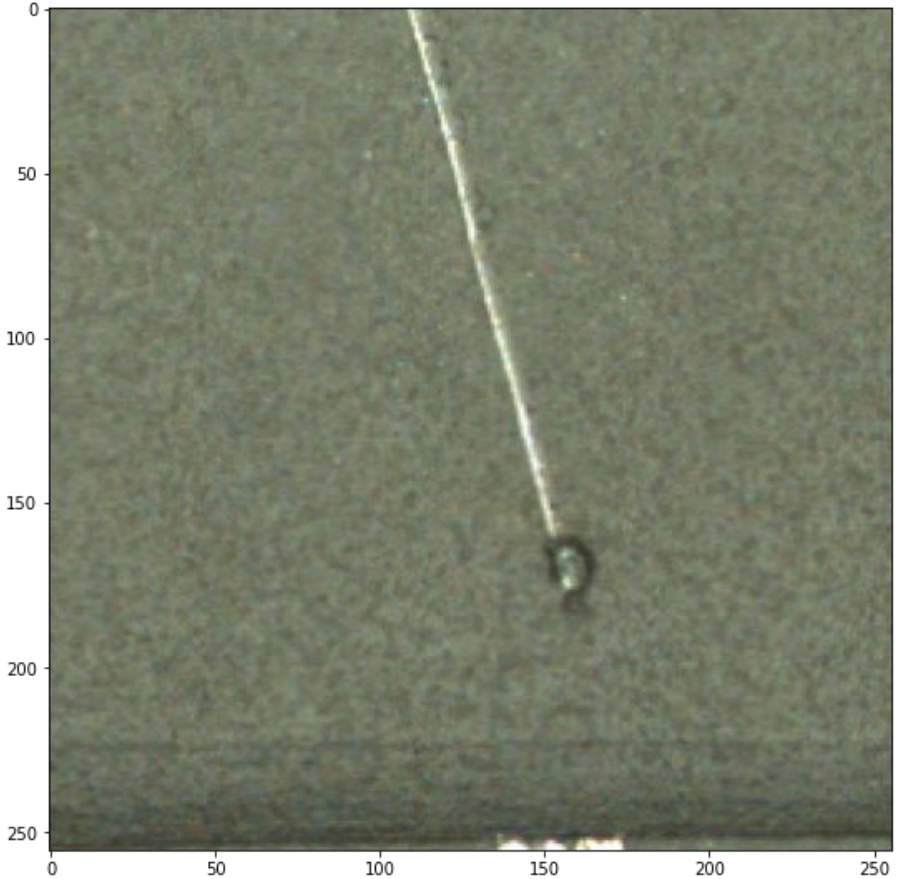
[7] ✓ 0.1s Python

```python
fig, ax = plt.subplots(1,8, figsize=(20,10))
ax[0].imshow(x)
for i in range(7):
    ax[i+1].imshow(yhat[:,:,i])
```

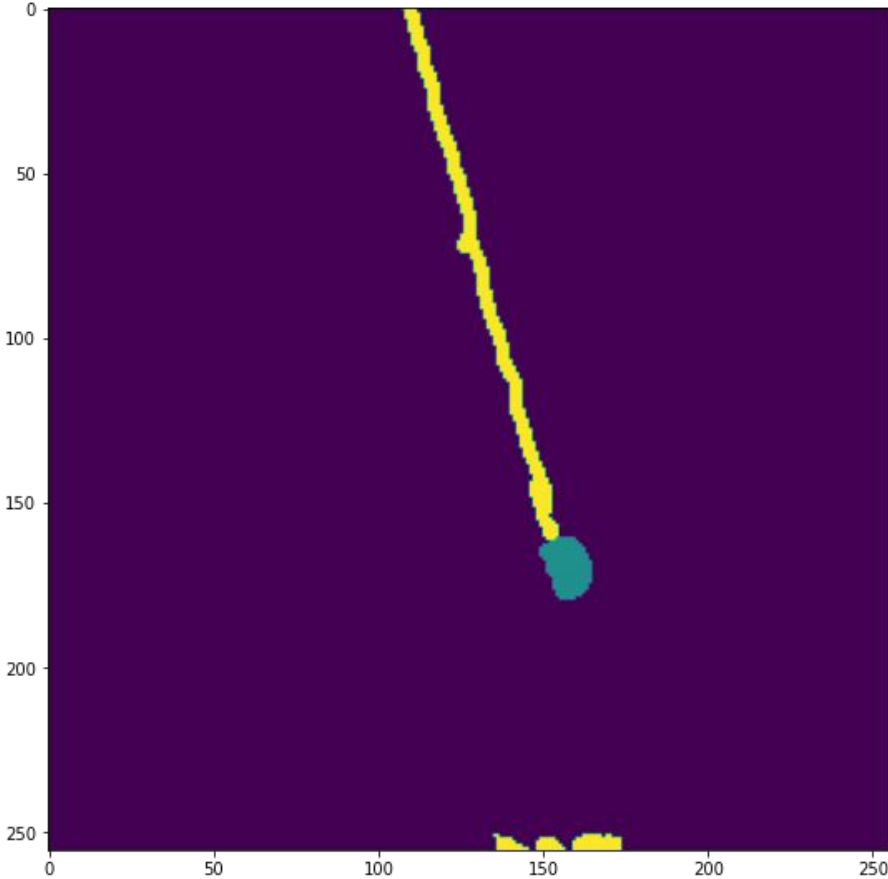
[8] ✓ 0.6s Python



Test Image



Correct Label



Deploying to Heroku

Heroku

Platform as a service company

heroku.com

Heroku is a cloud platform as a service supporting several programming languages. One of the first cloud platforms, Heroku has been in development since June 2007, when it supported only the Ruby programming language, but now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go. Wikipedia

Parent organization: Salesforce Inc

Founders: James Lindenbaum, Adam Wiggins, Orion Henry

Founded: 2007

Headquarters: San Francisco, CA

Files needed for Deployment

runtime.txt

You, 4 hours ago | 1 author (You)

```
python-3.9.16
```

Procfile

You, 3 hours ago | 1 author (You)

```
web: gunicorn -w 2 -k uvicorn.workers.UvicornWorker api:app
```

requirements.txt

You, 4 hours ago | 1 author (You)

```
fastapi==0.85.1
python-multipart==0.0.5
tensorflow-cpu==2.10.0
uvicorn==0.18.3
gunicorn==20.1.0
```

.gitignore

You, 3 hours ago | 1 author (You)

```
__pycache__
.ipynb_checkpoints
1694_test_image.bmp
1694_test_mask.bmp
api-request.ipynb
DL - testbed.ipynb
unet_multiclass.ipynb
```

To deploy to Heroku:

```
(base) C:\Users\sauce\OneDrive\Desktop\ml thread segmentation>git init
Reinitialized existing Git repository in C:/Users/sauce/OneDrive/Desktop/ml thread segmentation/.git/

(base) C:\Users\sauce\OneDrive\Desktop\ml thread segmentation>git add .
warning: in the working copy of 'api-request.ipynb', LF will be replaced by CRLF the next time Git touches it

(base) C:\Users\sauce\OneDrive\Desktop\ml thread segmentation>
```

```
(base) C:\Users\sauce\OneDrive\Desktop\ml thread segmentation>git commit -m "deploying to heroku"
[master 74376fc] deploying to heroku
 3 files changed, 24 insertions(+), 14 deletions(-)

(base) C:\Users\sauce\OneDrive\Desktop\ml thread segmentation>git status
On branch master
nothing to commit, working tree clean
```

Log in to Heroku

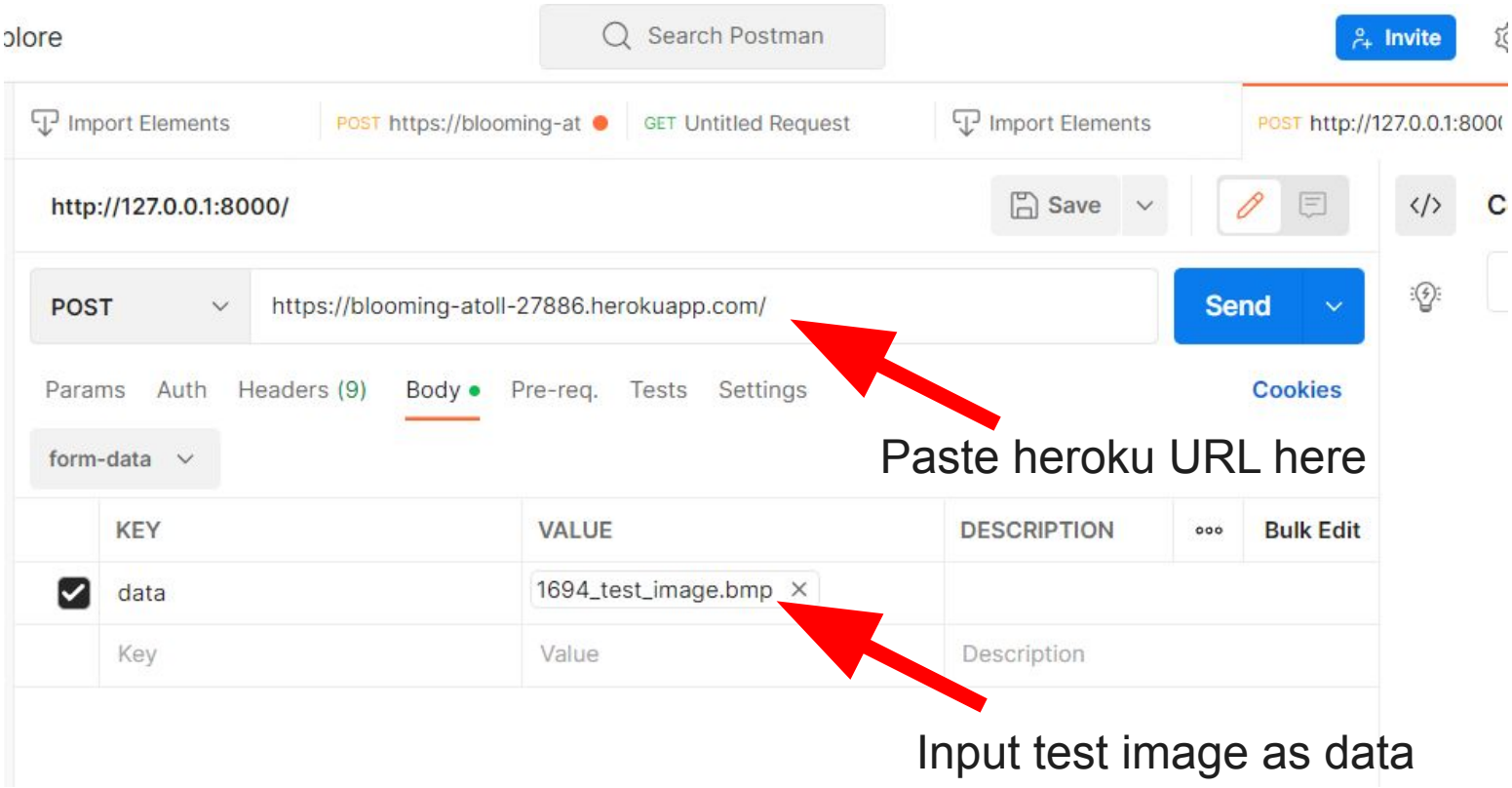
```
(base) C:\Users\sauce\OneDrive\Desktop\ml thread segmentation>heroku login
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/2010fc3a-8fff-4192-a1a9-6f655abcdf70
APAJiw2FAWIAAVGA.7-F-DUkeRpZjYBnGzH4zRUfahDR2I521PqwTFpT1VeE
Logging in... done
Logged in as skyler.saucedo@gmail.com
```

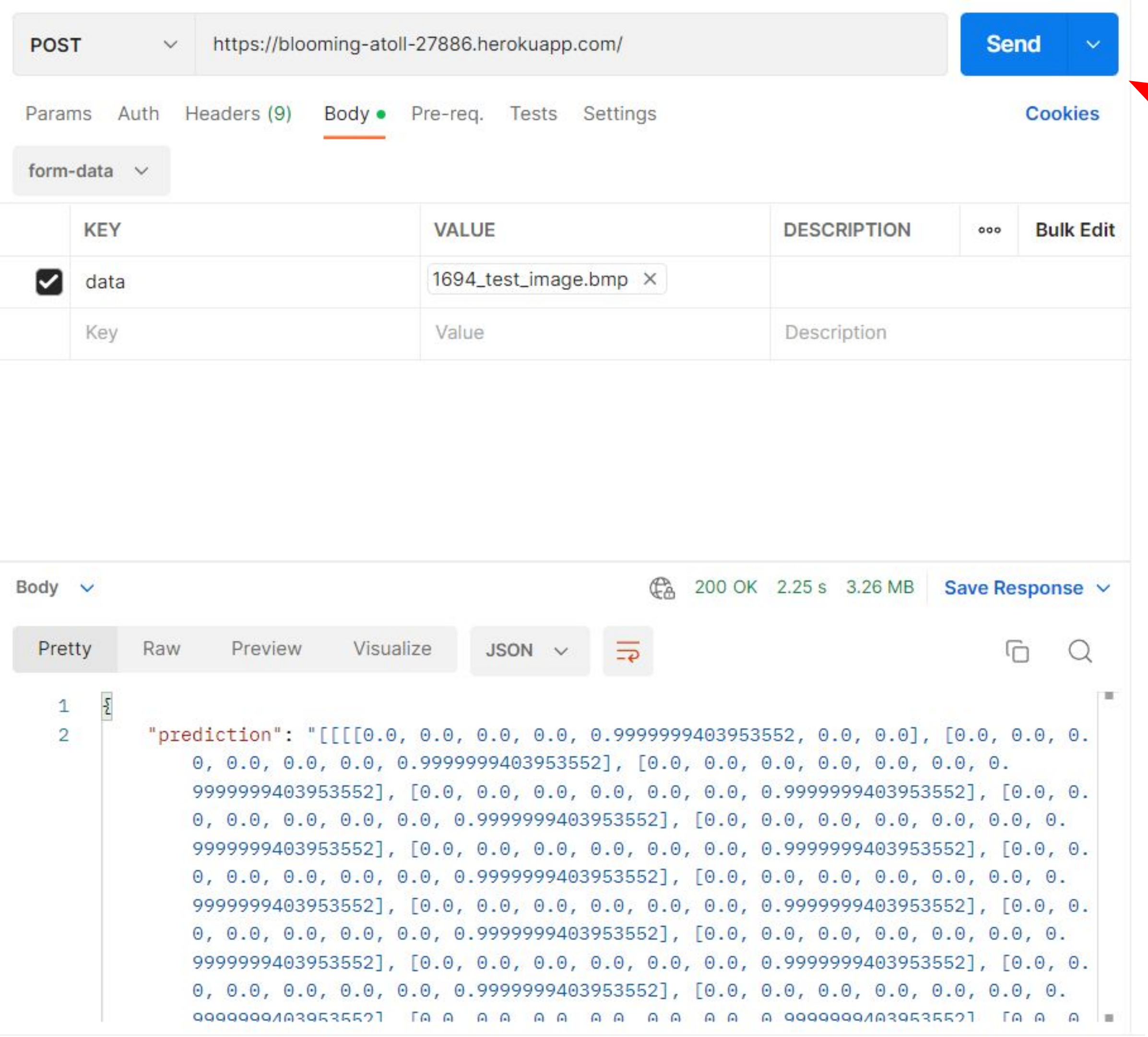
```
(base) C:\Users\sauce\OneDrive\Desktop\ml thread segmentation>heroku create
Creating app... done, ⬢ mighty-falls-82306
https://mighty-falls-82306.herokuapp.com/ | https://git.heroku.com/mighty-falls-82306.git

(base) C:\Users\sauce\OneDrive\Desktop\ml thread segmentation>git push heroku master
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 982 bytes | 982.00 KiB/s, done.
Total 6 (delta 4), reused 0 (delta 0), pack-reused 0
```

```
remote:        Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:        Done: 318.4M
remote: -----> Launching...
remote:  !     Warning: Your slug size (318 MB) exceeds our soft limit (300 MB) which may affect boot time.
remote:        Released v6
remote:        https://blooming-atoll-27886.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/blooming-atoll-27886.git
   53cfbaa..74376fc  master -> master

(base) C:\Users\sauce\OneDrive\Desktop\ml thread segmentation>
```

Model now lives here in the Heroku Cloud

Test new Endpoint in Postman





https://blooming-atoll-27886.herokuapp.com/

	KEY	VALUE	DESCRIPTION
☑	data	1694_test_image.bmp	
	Key	Value	Description

200 OK 2.25 s 3.26 MB

```
{
    "prediction": "[[[[0.0, 0.0, 0.0, 0.0, 0.9999999403953552, 0.0, 0.0], [0.0, 0.0, 0.
        0, 0.0, 0.0, 0.0, 0.9999999403953552], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
        9999999403953552], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.9999999403953552], [0.0, 0.
        0, 0.0, 0.0, 0.0, 0.0, 0.9999999403953552], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
        9999999403953552], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.9999999403953552], [0.0, 0.
        0, 0.0, 0.0, 0.0, 0.0, 0.9999999403953552], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
        9999999403953552], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.9999999403953552], [0.0, 0.
        0, 0.0, 0.0, 0.0, 0.0, 0.9999999403953552], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
        9999999403953552], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.9999999403953552], [0.0, 0.
        0, 0.0, 0.0, 0.0, 0.0, 0.9999999403953552], [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
```

Sending data packet returns

Prediction packet from new endpoint, hosted in the Heroku Cloud.

Next Steps:

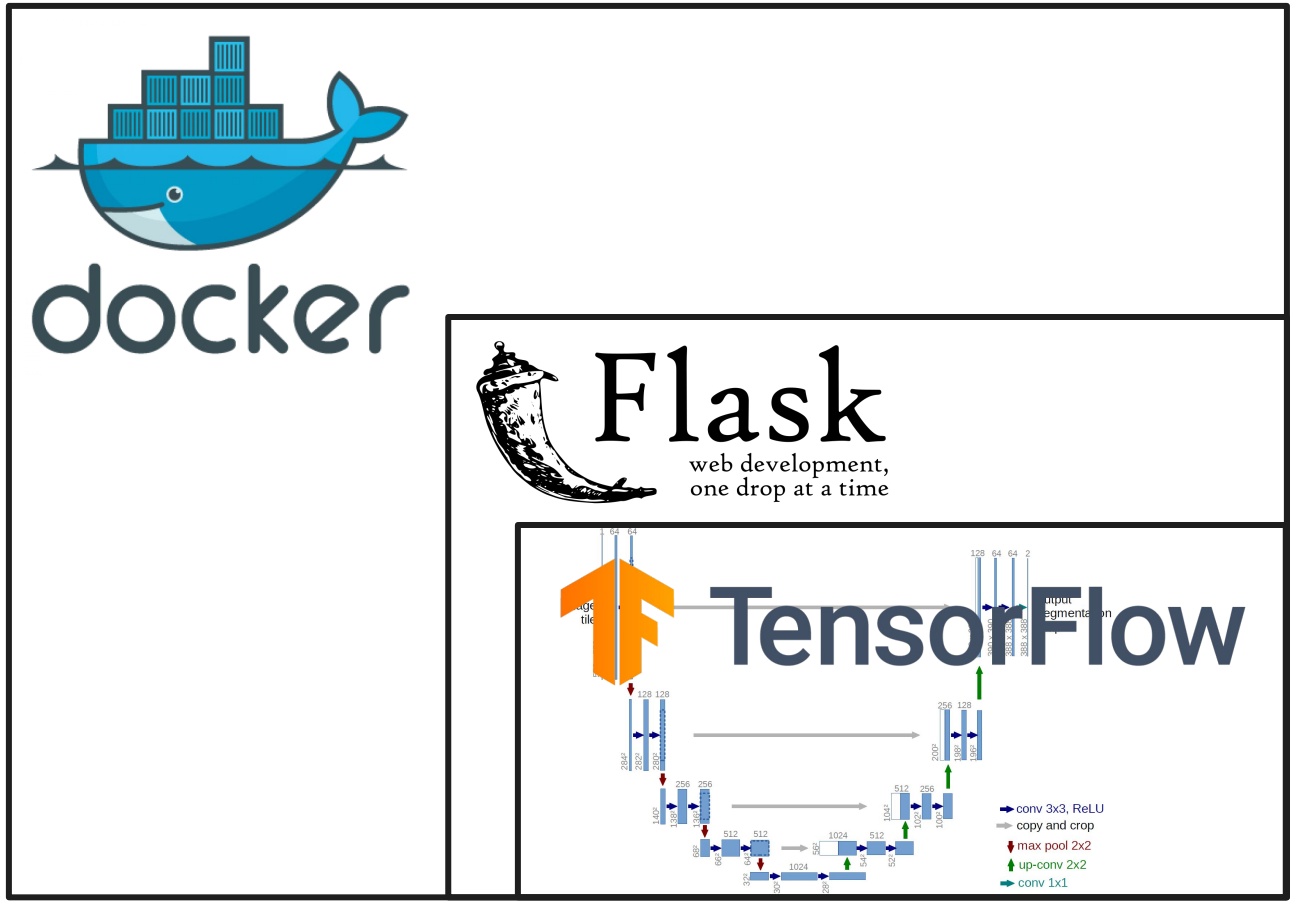
In UI.APP:

1. After scan is created, break 4k x 20k image into 256x256 tiles.
2. Save each tile as a .bmp, send to Heroku endpoint.
3. Return JSON packet of predictions, interpret and find defects.
4. Output defect object to be used in production for end-user.

Later:

5. Integrate ML OPS (Weights & Biases) to optimize model params.
6. Find way to monitor Model performance over time.

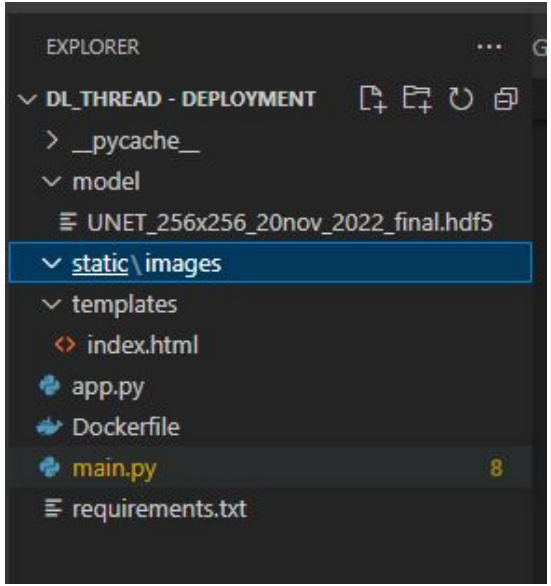
Previous approach - Docker Container



Encapsulate main.py, model, dependencies, and Flask app in Docker container.

Host docker container in Heroku, a server-based software for API development.

Image requirements



Dockerfile

```
You, 1 hour ago | 1 author (You)
1  FROM python:3.9.12
2  WORKDIR /app
3  COPY requirements.txt .
4  RUN pip install --upgrade pip
5  RUN pip install -r requirements.txt
6  COPY . .
7  CMD ["python", "app.py"]        You, 5
```

```
You, 9 minutes ago | 1 author (You)
1  Flask==2.1.0
2  Werkzeug==2.0.1
3  requests==2.24.0
4  gunicorn==20.1.0
5  scikit-learn==0.23.0
6  Pillow
7  tensorflow-cpu
8  matplotlib
9  opencv-python-headless
10 pandas
11 numpy        You, now • Uncom
```

requirements.txt

Creating Docker Image

Command Prompt

```
C:\Users\endle\Desktop\dl_thread - deployment>docker image build -t dl_thread_dec_11-app .
```

Command Prompt

```
#9 703.1 note: This is an issue with the package mentioned above, not pip.
#9 703.1 hint: See above for output from the failure.
------
executor failed running [/bin/sh -c pip install -r requirements.txt]: exit code: 1

C:\Users\sauce\OneDrive\Desktop\dl_thread - deployment>docker image build -t dl-thread-dec-12.app .
[+] Building 460.9s (11/11) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 32B                                                                     0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load metadata for docker.io/library/python:3.9.12                                        0.6s
 => [1/6] FROM docker.io/library/python:3.9.12@sha256:4200eda05642eabf85b70648e17e4b433f2d7608c0130ab5dcb56cce6bc35364   0.0s
 => [internal] load build context                                                                       0.0s
 => => transferring context: 5.09kB                                                                     0.0s
 => CACHED [2/6] WORKDIR /app                                                                           0.0s
 => [3/6] COPY requirements.txt .                                                                       0.0s
 => [4/6] RUN pip install --upgrade pip                                                                 3.8s
 => [5/6] RUN pip install -r requirements.txt                                                         448.8s
 => [6/6] COPY . .                                                                                      0.2s
 => exporting to image                                                                                  7.3s
 => => exporting layers                                                                                 7.3s
 => => writing image sha256:e1a87f275f313216d944ac10205e6c0847e00e1412d24274ecacdfe56b45951f             0.0s
 => => naming to docker.io/library/dl-thread-dec-12.app                                                 0.0s

C:\Users\sauce\OneDrive\Desktop\dl_thread - deployment>
```

Command Prompt

```
=> [internal] load metadata for docker.io/library/python:3.7.10                                   0.4s
=> [1/5] FROM docker.io/library/python:3.7.10@sha256:c9b155d64106b7c939cb82396251401418dbdd6ccf51a90c9ea4ede1316  0.0s
=> [internal] load build context                                                                  0.0s
=> => transferring context: 852B                                                                  0.0s
=> CACHED [2/5] WORKDIR /app                                                                      0.0s
=> [3/5] COPY requirements.txt .                                                                  0.0s
=> [4/5] RUN pip install -r requirements.txt                                                     87.5s
=> [5/5] COPY . .                                                                                 0.1s
=> exporting to image                                                                             9.2s
=> => exporting layers                                                                            9.2s
=> => writing image sha256:293b247f92c8ad2dea7eb303e759c92663fa9ab22cff59537a344d4e806e2287       0.0s
=> => naming to docker.io/library/dl_thread_dec_11-app                                            0.0s

C:\Users\endle\Desktop\dl_thread - deployment>docker image ls
REPOSITORY                                  TAG                                 IMAGE ID       CREATED
SIZE
dl_thread_dec_11-app                        latest                              293b247f92c8   29 minutes ago
2.98GB
```

Docker image too large to host on the cloud… too clunky

```
C:\Users\endle\Desktop\dl_thread - deployment>docker run -p 5000:5000 -d dl_thread_dec_11-app
6f1c6f7cac66e85d529cd1bcfcc11313c36e63c0933331f87c74cf68ba1cee30

C:\Users\endle\Desktop\dl_thread - deployment>
```

Previous test. HTML app

Testing locally

Results - this approach takes ~2 mins to make but works very well

Results