

BDL: A06
Nikhil Anand BE20B022

Task 1:

The following 3 images were created which are 28x28 images in the specific format that our model takes as input.



Thus, we may input this into the task1.py model server and get the predictions.
(Without reformatting images)

For eg, when the 5 is inputted:

Execute

Clear

Responses


Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/predict' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@resized5.jpg;type=image/jpeg'
```

Request URL

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

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{ "digit": "5" }</pre></div><div> Download</div></div>

Response headers

When the 9 is inputted:

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/predict' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@resized9.jpg;type=image/jpeg'
```

Request URL

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

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{ "digit": "9" }</pre></div><div> Download</div></div> <div><div>Response headers</div><div><pre>content-length: 13 content-type: application/json date: Thu, 23 Apr 2024 13:49:18 GMT server: uvicorn</pre></div></div>

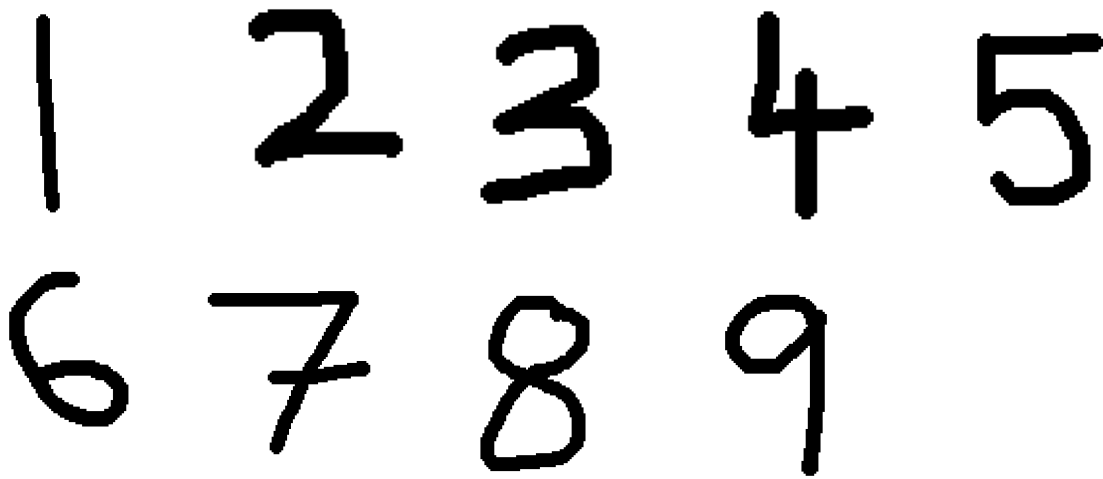
Responses

Code	Description	Links
200	Successful Response	No links

Task 2:

The code has been implemented and submitted as task2.py. In order to run this, we must run in the command line: `uvicorn task2:app --reload`

We can create the following digit images using an online paint program.



The code:

```
@app.post("/predict")
async def predict(file: UploadFile):
    request_object_content = await file.read()
    img = Image.open(io.BytesIO(request_object_content))

    resized_img = await format_image(img)

    arr = np.array(resized_img)

    print(arr, arr.shape)

    flattened_image = arr.reshape(-1)
    flattened_image_list = flattened_image.tolist()

    model = await load_model("/Users/nikhilanand/FastAPI_BDL/training_1/cp.weights.h5")
    digit = await predict_digit(model, flattened_image_list)

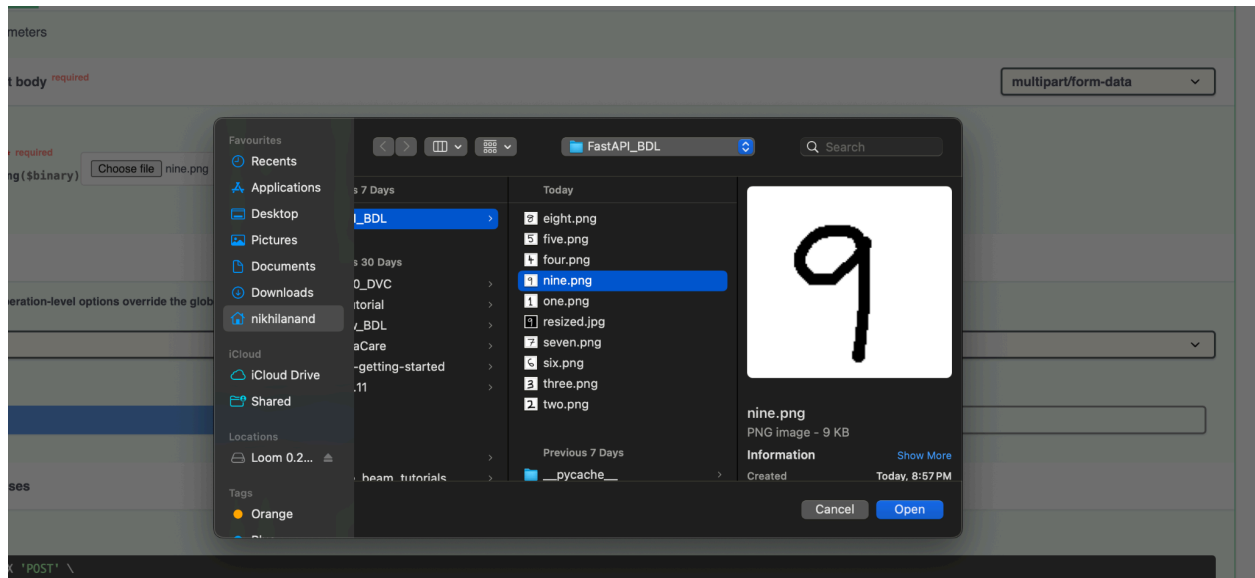
    return {"digit": digit}
```

First we import the image through the post http function. Then the image is formatted. Then we flatten it and load the model and predict the digit. The digit is shown in the Swagger UI as a dictionary `{"digit": digit}`

You can see the remaining functions in the code file which is submitted along with this report.

Results

Loading the 9:



It outputs 9:

<p>Response body</p> <pre>{ "digit": "9" }</pre>
<p>Response headers</p> <pre>content-length: 13 content-type: application/json date: Thu, 25 Apr 2024 15:34:48 GMT server: uvicorn</pre>
<p>inses</p> <p>Description</p> <p>Successful Response</p>

Loading the 5,

Curl

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

Request URL

http://127.0.0.1:8000/predict

Server response

Code	Details
200	<p>Response body</p> <pre>{ "digit": "5" }</pre> <p>Download</p>

Response headers

Loading the 2,

/

Execute Clear

Responses

Curl

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

Request URL

http://127.0.0.1:8000/predict

Server response

Code	Details
200	<p>Response body</p> <pre>{ "digit": "2" }</pre> <p>Download</p>

Response headers

```
content-length: 13
content-type: application/json
date: Thu, 25 Apr 2024 15:36:03 GMT
server: uvicorn
```

Loading the 3,

Curl

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

Request URL

http://127.0.0.1:8000/predict

Server response

Code	Details
200	<p>Response body</p> <pre>{ "digit": "3" }</pre> <p>Download</p>

Response headers

```
content-length: 13
content-type: application/json
date: Thu, 25 Apr 2024 15:36:28 GMT
server: uvicorn
```

Loading the 4,

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/predict' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=four.png;type=image/png'
```

Request URL

http://127.0.0.1:8000/predict

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{ "digit": "4" }</pre></div><div><div>Download</div></div></div> <div><div>Response headers</div><div><pre>content-length: 13 content-type: application/json date: Thu, 25 Apr 2024 15:36:49 GMT server: uvicorn</pre></div></div>

Responses

Code	Description	Links
------	-------------	-------

Loading the 8,

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/predict' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=eight.png;type=image/png'
```

Request URL

http://127.0.0.1:8000/predict

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{ "digit": "8" }</pre></div><div><div>Download</div></div></div> <div><div>Response headers</div><div><pre>content-length: 13 content-type: application/json date: Thu, 25 Apr 2024 15:40:25 GMT server: uvicorn</pre></div></div>

Loading the 7,

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/predict' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=seven.png;type=image/png'
```

Request URL

http://127.0.0.1:8000/predict

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{ "digit": "7" }</pre></div><div><div>Download</div></div></div> <div><div>Response headers</div><div><pre>content-length: 13 content-type: application/json date: Thu, 25 Apr 2024 15:40:05 GMT server: uvicorn</pre></div></div>

Responses

Code	Description	Links
------	-------------	-------

Loading the 1,



Thus, **our model worked for all 10 digits!**

Points allocation:

10% for a clean coding style: It has been implemented in clean concise code

30% for the correctness of the implementation : It is correct and works appropriately

10% for github project : The github link is here

https://github.com/nikhilanand03/FastAPI_BDL_Assignment

20% for readable comments : Comments have been added in the code

10% for input arguments' validation/boundary check: assert statements have been added

20% for unit test modules: assert statements are added
