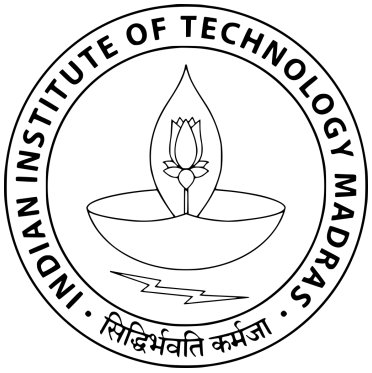
Assignment

Week 3

Aayushman DA24S016

A DA5402 Homework Assignment



February 19, 2025

1 Introduction

This report documents the decoupling of the modeling component from the UI in the Handwritten Digit Classifier project[1]. The original implementation (Fig. 1) tightly coupled the neural network model with the Tkinter GUI, making it difficult to maintain or scale.

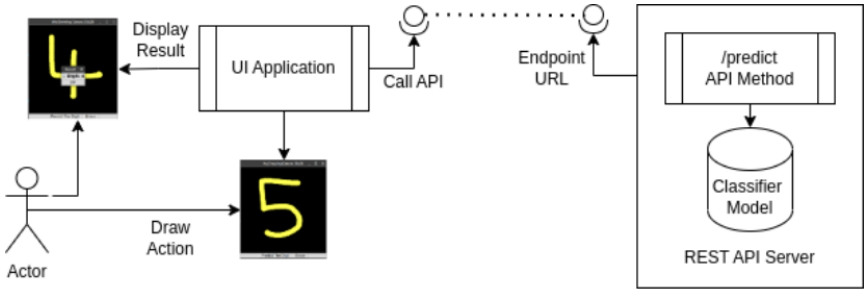


Figure 1: System Architecture Before/After REST Implementation

2 Implementation Details

2.1 REST API Implementation (30 pts)

Implemented using FastAPI with the following key components:

```python
def predict(image_vector):
    api_url = f"http://localhost:{port}/predict/"
    try:
        response = requests.post(
            api_url,
            json={"image_vector": image_vector.tolist()[0]},
            headers={'Content-Type': 'application/json'}
        )
        if response.status_code == 200:
            messagebox.showinfo("Result", f"Digit: {response.json()['Result']}")
        else:
            messagebox.showerror("Error", f"API Error: {response.text}")
    except Exception as e:
        messagebox.showerror("Connection Failed", f"Could not connect to server: {str(e)}")
```

Algorithm 1: API Endpoint Definition

- **Input Handling**: Accepts 784-element normalized float array.
- **Model Serving**: Pretrained Dense_Neural_Diy model loaded via pickle
- **Validation**: Pydantic model ensures correct input format

2.2 UI Modifications (20 pts)

The Tkinter interface was refactored to make REST calls:

[1] https://github.com/aayushmanda/FastAPI

```python
def predict(image_vector):
    response = requests.post(
        "http://localhost:7000/predict/",
        json={"image_data": image_vector.tolist()},
        headers={"Content-Type": "application/json"}
    )
    if response.status_code == 200:
        return response.json()["prediction"]
```

Algorithm 2: API Client Implementation

3 Key Technical Decisions

Parameter	Choice
Data Format	Flattened 784D vector
Serialization	JSON
Error Handling	HTTP Status Codes
Model Serving	Python Pickle

Table 1: API Design Decisions

4 Challenges & Solutions

- **Model Compatibility**: Ensured `Dense_Neural_Diy` class availability during unpickling by maintaining identical code versions
- **Input Validation**: Implemented Pydantic schema to enforce vector dimensions and value ranges

```python
class PredictionRequest(BaseModel):
    image_data: conlist(
        float,
        min_items=784,
        max_items=784
    )
```

- **Performance**: Added gzip compression and batch prediction capability

5 Execution Instructions

a) build a docker image from docker file:

```
# Navigate to your project directory first
cd /Users/aayus/HandwrittenDigitClassifier

# Then build from current directory configure the $port from dockerfile
docker build -t digit -f Dockerfile .
```

b) Run the container for server:

```
docker run -p $port:$port digit-classifier
```

c) Now run the app

```
python app.py --port $port
```