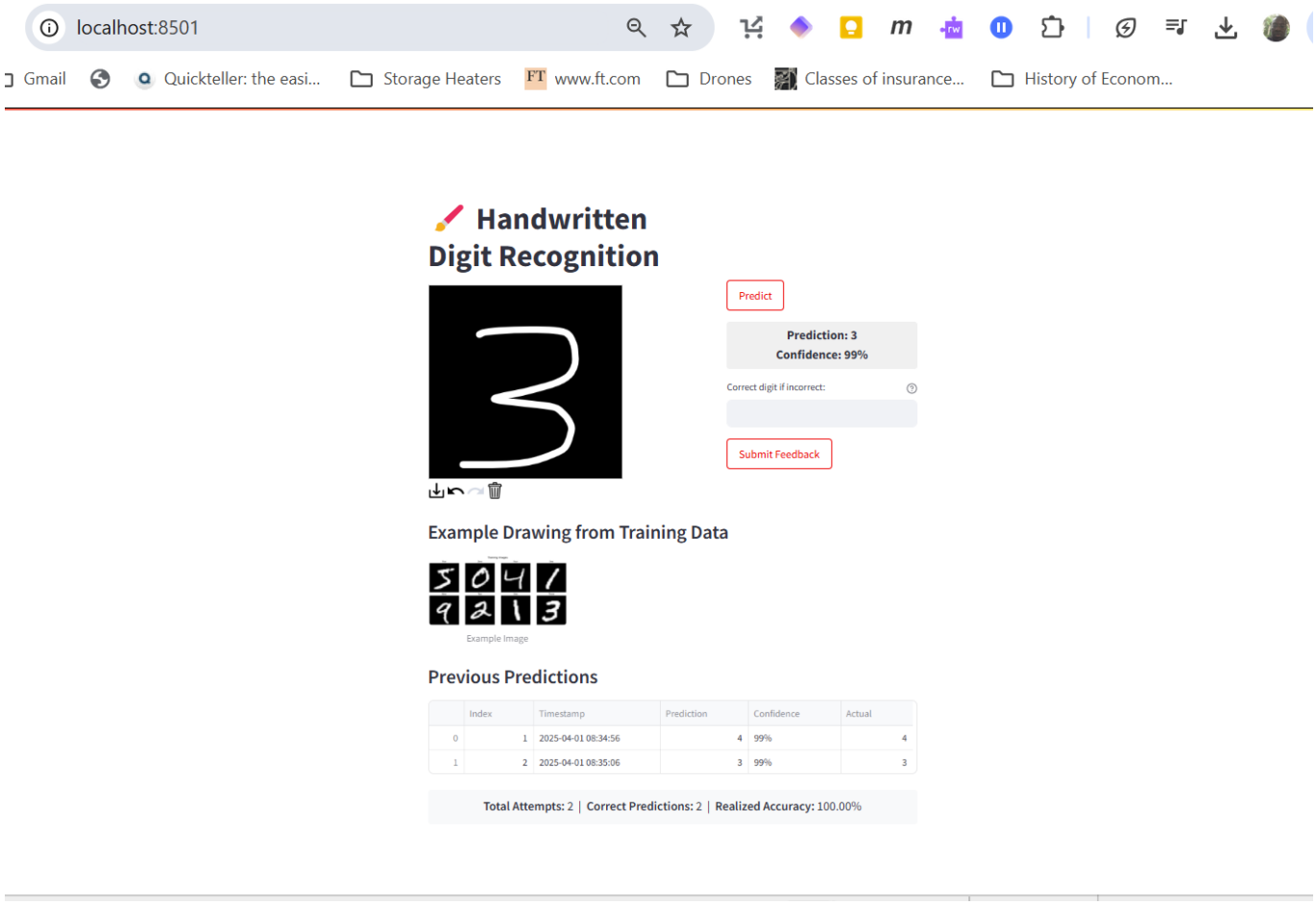
This is a web app that recognises handwritten digits between 0 and 9. The prediction is made using a Convolutional Neural Network (CNN) trained on the well-known MNIST database.

The user draws a digit on the slate and then presses the predict button. If the app predicts wrong, the user types in the correct number and presses the submit feedback button. To try another digit, click on the delete icon below the slate and draw another digit as before. A history of previous digits drawn by the user is shown in a table below the slate together with a running prediction accuracy of the app. However this table is lost once the app is refreshed. Persistent storage of past predictions is done in a Postgres database hosted on the same web server,

Try it out here: URL: http://138.199.200.113:8501



Project Structure

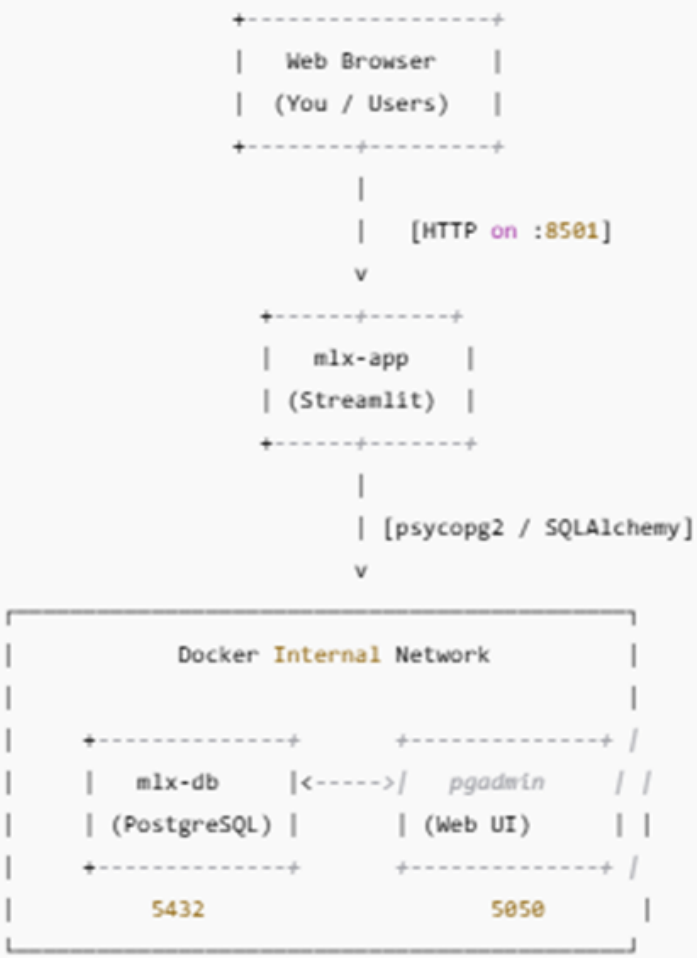
graphql Copy Edit

```
MLX_PROJECT/
├── app/                          # Streamlit + ML logic
│   ├── main.py                   # UI + prediction logic
│   ├── CNNModelMNIST.py          # PyTorch model
│   ├── model.pth                 # Trained model weights
│   ├── Dockerfile                # Docker config for Streamlit app
│   └── ...
│
├── db/                           # Postgres SQL init scripts
│   ├── init.sql                  # CREATE DATABASE script
│   └── create_table.sql          # CREATE TABLE IF NOT EXISTS ...
│
├── .env                          # App + DB + pgAdmin environment variables
├── docker-compose.yaml           # Defines services: app, db, pgadmin
├── requirements.txt              # Python dependencies
├── .gitignore
└── README.md
```

Container	Purpose	Port(s) Exposed	Volume Persistence
`mlx-app`	Your Streamlit app for digit recognition	8501 → host	Not needed
`mlx-db`	PostgreSQL database	5432 (internal only)	`pgdata:/var/lib/postgresql/data` ✅ persisted
`pgadmin`	pgAdmin UI to inspect the DB	5050 → host	`pgadmin-data:/var/lib/pgadmin` ✅ persisted

🧩 MLX Project – Container Interaction Diagram

pgsql Copy Edit



Data

The MNIST dataset was used

Preprocessing:

Model Architecture

-

```python
# Define the CNN model by subclassing nn.Module
class CNNModel(nn.Module):
    def __init__(self):
        super().__init__()  # Initialize the base class
        # Convolutional layer 1: Input channels = 1 (grayscale), Output channels = 16
        self.conv1 = nn.Conv2d(1, 16, kernel_size=3, padding=1)
        # Convolutional layer 2: Input channels = 16, Output channels = 32
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, padding=1)
        # Max pooling layer: Reduces spatial dimensions
        self.pool = nn.MaxPool2d(2, 2)
        # Fully connected layer 1
        self.fc1 = nn.Linear(32 * 7 * 7, 128)
        # Fully connected layer 2 (Output layer)
        self.fc2 = nn.Linear(128, 10)
        # Dropout layer to prevent overfitting
        self.dropout = nn.Dropout(0.25)

    def forward(self, x):
        # Convolutional layer 1 followed by ReLU activation and pooling
        x = self.pool(F.relu(self.conv1(x)))
        # Convolutional layer 2 followed by ReLU activation and pooling
        x = self.pool(F.relu(self.conv2(x)))
        # Flatten the output for the fully connected layers
        x = x.view(-1, 32 * 7 * 7)
        # Apply dropout
        x = self.dropout(x)
        # Fully connected layer 1 with ReLU activation
        x = F.relu(self.fc1(x))
        # Output layer
        x = self.fc2(x)
        return x
```

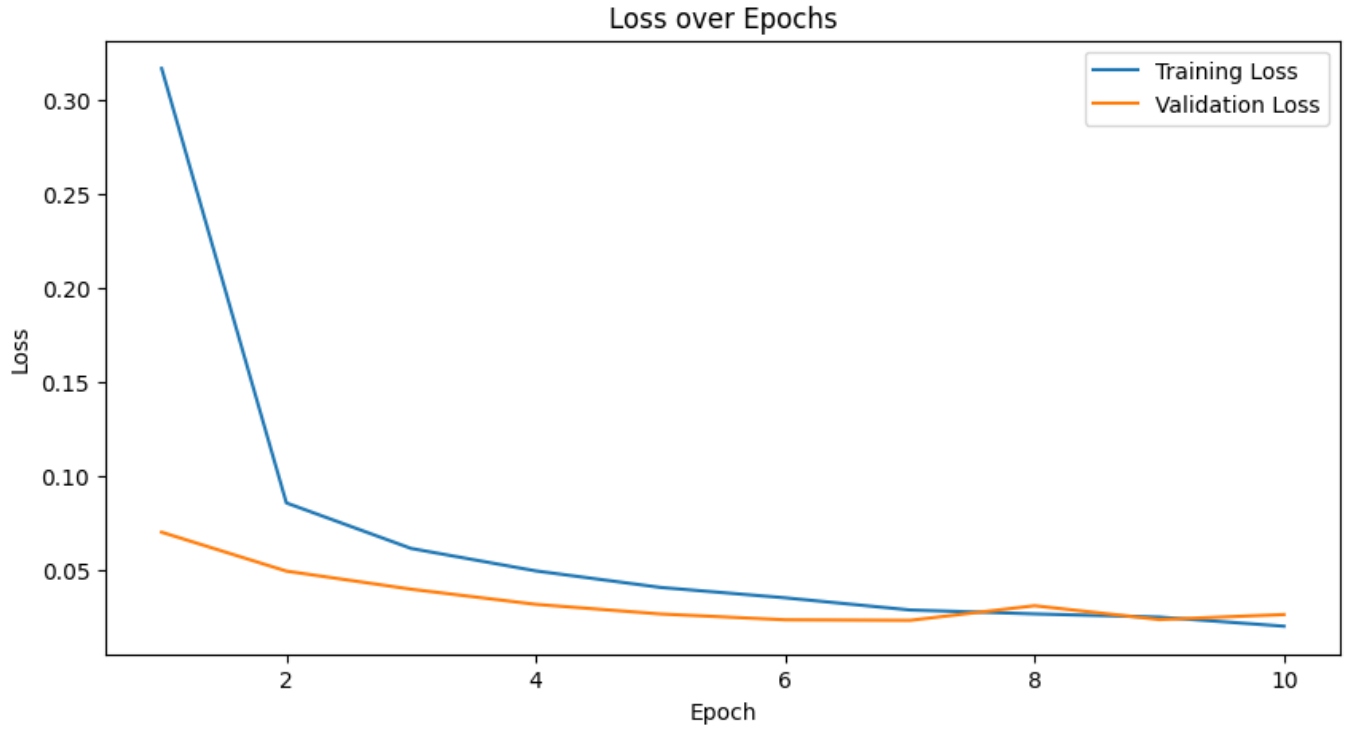
Training

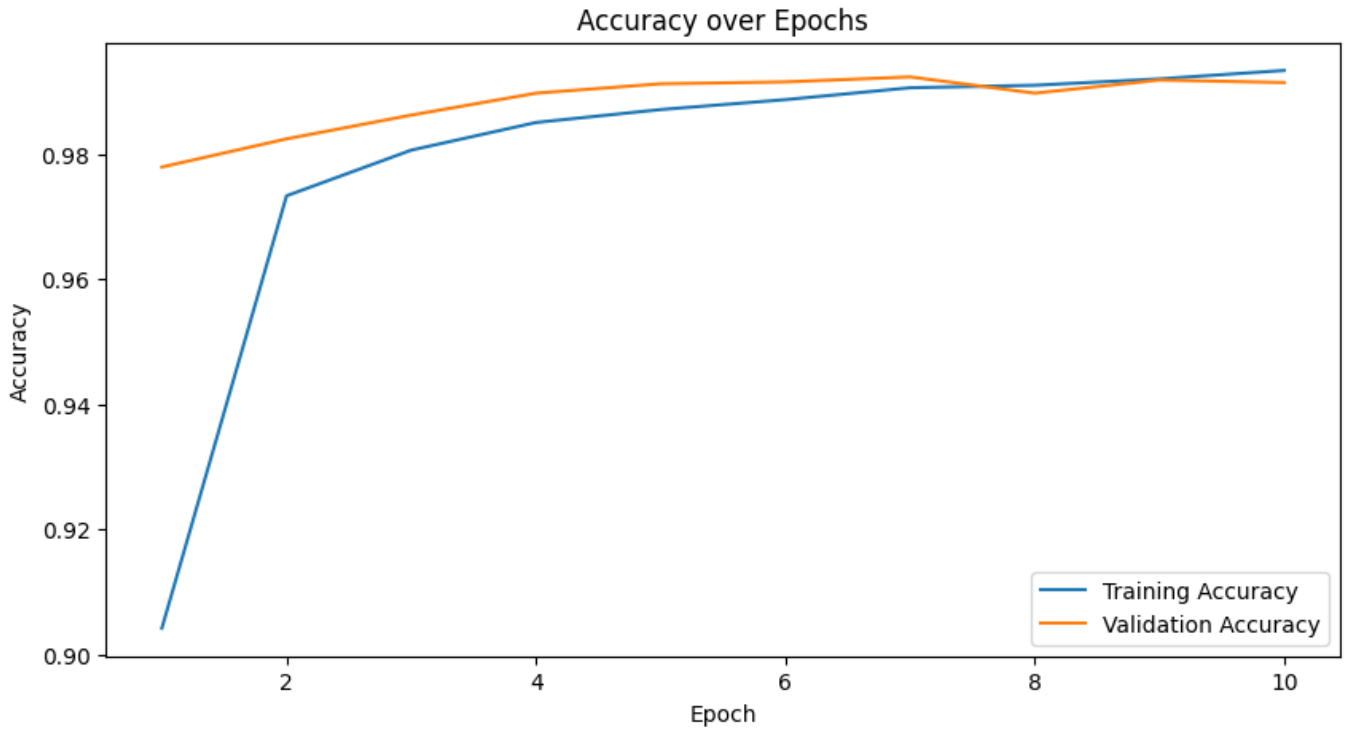
- In order to make use of GPUs and so speed up training, model was run in a Google Colab Notebook, Model was run for 10 epochs. Othe training details including hyper-parameters are as per the screenshot below:

```python
# Instantiate the model and move it to the device (CPU or GPU)
model = CNNModel().to(device)

# Define the loss function (Cross-Entropy Loss for multi-class classification)
criterion = nn.CrossEntropyLoss()

# Define the optimizer (Adam optimizer with a learning rate of 0.001)
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Number of epochs to train
num_epochs = 10
```

Model Evaluation

Model accuracy of over 90% was achieved on the MNIST test dataset.





Inference

File

🧠 Model Weights

This repo includes a pre-trained CNN for MNIST digit recognition:

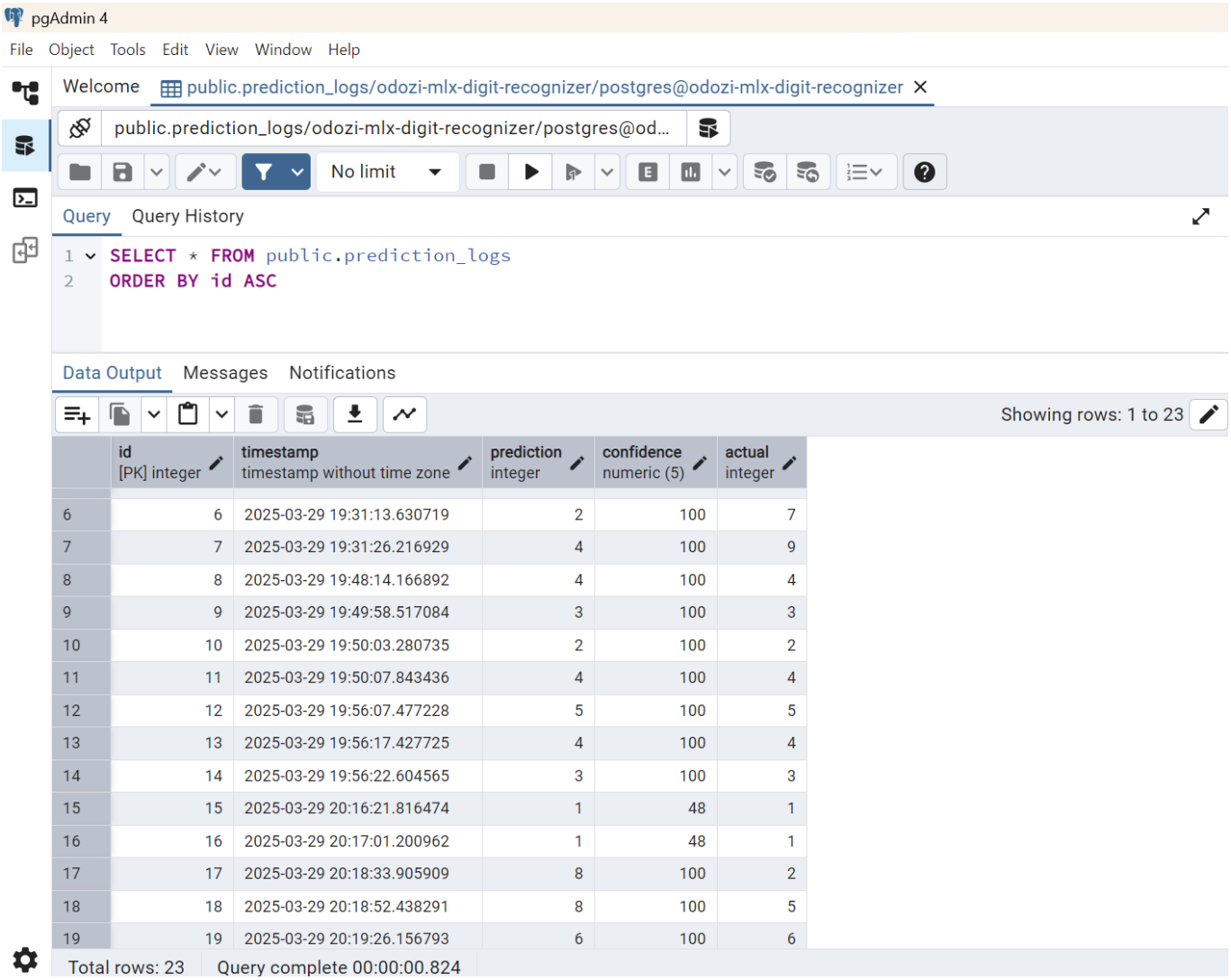
- `app/mnist_cnn.pth` — weights for a small CNN trained on the MNIST dataset

You can retrain your own using `CNNModelMNIST.py` or swap in a different `.pth`.

Deployment

Three Docker containers were used as per the project structure. One for the PyTorch Model and Streamlit App the second for the initialisiation of the Postgres DB and Table and the 3rd for a PGAdmin tool to be used for viewing the prediciton logs ona browser The Containers were built and hosted on Hetzner VPS Instance Git Hub actions were used to push any changes to the app folder automatically to the VPS using ssh login and there to re-build the container Docker-compose was used to automatically restart the App anytime the VPS was rebooted On first run, the `init.sql` script (in `db/init.sql`) sets up the PostgreSQL schema for logging predictions. Docker Compose handles this automatically.

The Postgres database table is shown below after over 20 attemopts had been logged



	id [PK] integer	timestamp timestamp without time zone	prediction integer	confidence numeric (5)	actual integer
6	6	2025-03-29 19:31:13.630719	2	100	7
7	7	2025-03-29 19:31:26.216929	4	100	9
8	8	2025-03-29 19:48:14.166892	4	100	4
9	9	2025-03-29 19:49:58.517084	3	100	3
10	10	2025-03-29 19:50:03.280735	2	100	2
11	11	2025-03-29 19:50:07.843436	4	100	4
12	12	2025-03-29 19:56:07.477228	5	100	5
13	13	2025-03-29 19:56:17.427725	4	100	4
14	14	2025-03-29 19:56:22.604565	3	100	3
15	15	2025-03-29 20:16:21.816474	1	48	1
16	16	2025-03-29 20:17:01.200962	1	48	1
17	17	2025-03-29 20:18:33.905909	8	100	2
18	18	2025-03-29 20:18:52.438291	8	100	5
19	19	2025-03-29 20:19:26.156793	6	100	6

Ideas for future

- Trying out different models eg LeNet-5, Vision Transformers, Capsule Networks, Google Vision API, LLMs like ChatGPT
- Splitting out the model from the Streamlit app so that there are 4 containers instead of 3