# DA5402 - Assignment 8

Anirudh Rao be21b004

## Task 1

The provided code for handwriting recognition was modified to incorporate MLflow tracking. The modified code can be found in `mlflow_handwriting_recognition.py`. The steps to download the dataset (IAM Dataset) for this are mentioned in the 'Steps to run code' section of this report.

The IAM Dataset is loaded and preprocessed by filtering out comments and erroneous lines from the word file. Paths for the corresponding images are extracted, and labels are vectorized. The dataset is split into three sets: training, validation, and test. The file paths of images in the test set is logged as an MLflow artifact for future testing. The `prepare_dataset()` function is used to resize the images, apply padding, and create a batched dataset.

A CNN-RNN model is built for handwriting recognition. The model consists of convolutional layers followed by LSTM layers. A custom `CTCLayer` is implemented to calculate the CTC loss, which is used during training to compute the error between predicted and true labels. Hyperparameters are logged as MLflow parameters.

The model is trained on three different splits of the dataset. For each split, the model trains for 5 epochs. During training, the train loss, validation loss, and edit distance are computed and logged. MLflow logs these metrics for each epoch. The MLflow runs for these three different splits are run parallelly using `concurrent.futures`. The trained models are registered on MLflow. The vocabulary used by the model's `CTCLayer` is logged in JSON format as an artifact in MLflow.

After training, two graphs are generated and logged to MLflow as artifacts:

- Epochs vs. Training & Validation Losses: This plot tracks the changes in training and validation loss across epochs.
- Epochs vs. Average Edit Distance: This plot tracks the average edit distance across epochs.

The training process is wrapped in a `try-except` block to catch any exceptions and log the errors. All relevant actions, including dataset splitting and training progress, are logged to a file named `script.log`. The logs help in tracking the progress of each split and the overall training process.

## Task 2

It was observed that split #2 achieved the best performance in terms of all three metrics being tracked (see Figure 6 in the 'Sample outputs' section). This corresponded to the first registered version of the model and was used while creating the API. This was done in the `handwriting_recognition_api.py` script.

This code sets up an API web service for performing handwriting recognition using a pre-trained model loaded from MLflow. The application is designed to receive an image of handwritten text, preprocess it, and then predict the corresponding text using the pre-trained model. The model and its associated vocabulary are stored in MLflow.

Image preprocessing is handled by the `preprocess_image()` function, which resizes and normalizes the input image before passing it to the model for inference. The CTC output is decoded using greedy decoding in the `decode_ctc_output()` function, mapping the predicted indices to characters via the vocabulary file stored as a JSON artifact in MLflow.

When the API receives an image through the `/predict` POST endpoint, the image is processed and passed to the model. The model then predicts the text, which is returned in the response. The API functionality can be tested using `curl`. An example of this is shown in the 'Sample outputs' section. The steps to use `curl` are in the 'Steps to run code' section.

## Files submitted

- `requirements.txt` – Text file that contains the Python libraries to be installed prior to running the scripts.
- `mlflow_handwriting_recognition.py` – Trains three handwriting recognition models and logs them to MLflow, along with parameters and artifacts. Also creates a log file called `script.log`.
- `handwriting_recognition_api.py` – Exposes an MLflow model as an API service.

## Steps to run code

- Install the dependencies provided in `requirements.txt` using `pip`
- Run `mlflow server --host 127.0.0.1 --port 5000` in the command line. Access the MLflow UI at http://localhost:5000/.
- Download the IAM dataset by running these commands after replacing `[link]` with the URL of this.:

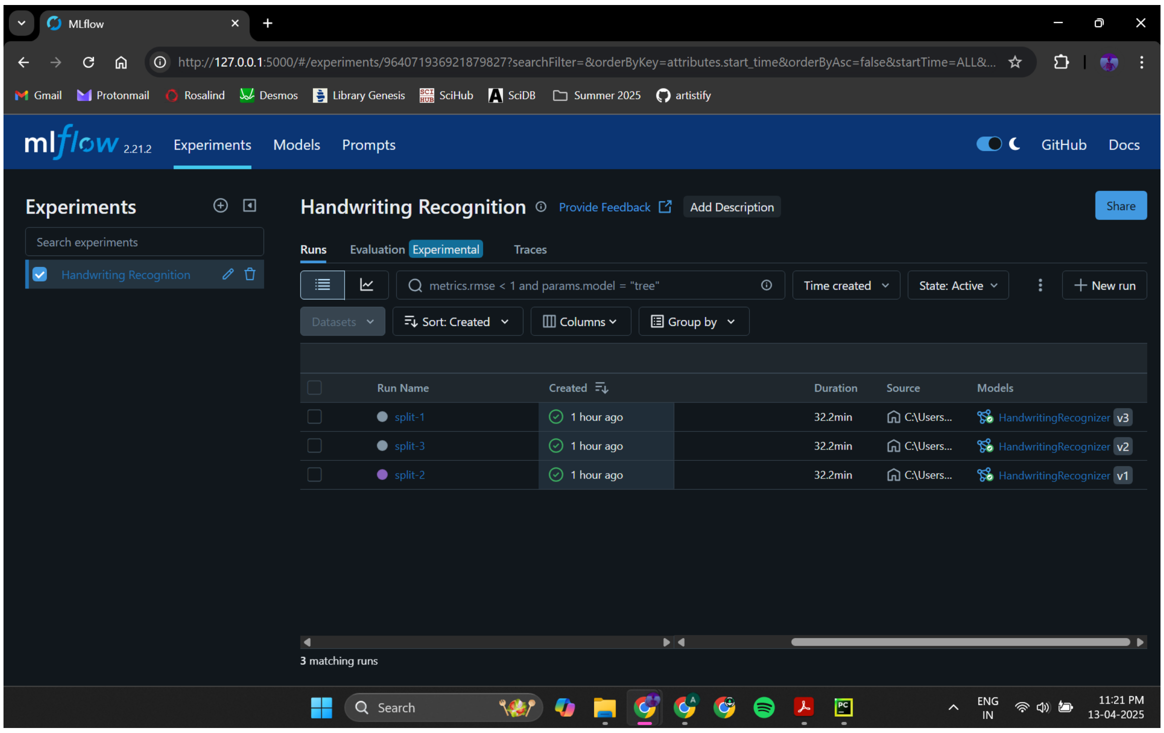
```
wget [link]
unzip -qq IAM_Words.zip
mkdir data
mkdir data/words
tar -xf IAM_Words/words.tgz -C data/words
mv IAM_Words/words.txt data
```

- Run `mlflow_handwriting_recognition.py`. Optionally, modify the number of splits, number of epochs, and other hyperparameters prior to running this script. These can be found under `CONFIGURATION` at the head of the script.
- Once complete, use the MLflow UI to identify the best performing model.
- Modify the head of the `handwriting_recognition_api.py` to include the best model, the ID of the run that created this best model, and the vocabulary JSON artifact name of this model.
- Run `handwriting_recognition_api.py` to start the API service at http://localhost:5050/.
- On the command line, run

  ```
  curl.exe -X POST "http://localhost:5050/predict" -F 'image=@"[filepath]"'
  ```

  Replace `[filepath]` with the actual path to the test image file. The API will return the recognised text in JSON format and display it in the command line. —

## Sample outputs



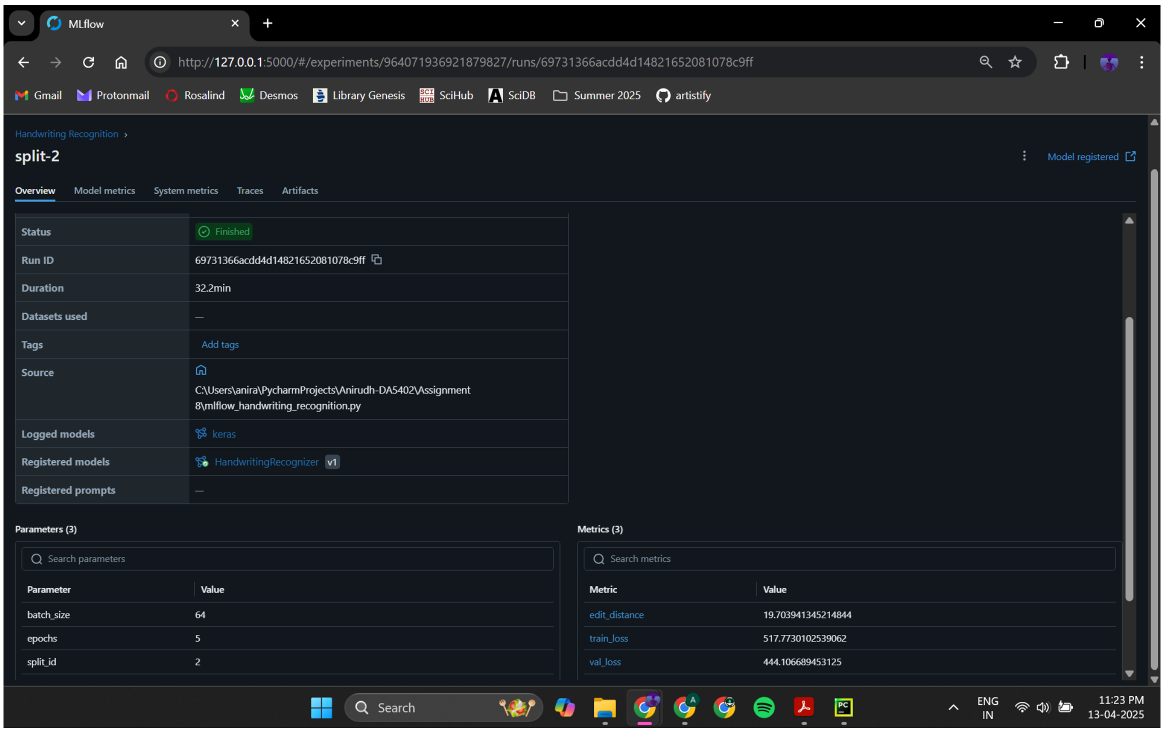
Handwriting Recognition

metrics.rmse < 1 and params.model = "tree"

| Run Name | Created | Duration | Source | Models |
|---|---|---|---|---|
| split-1 | 1 hour ago | 32.2min | C:\Users... | HandwritingRecognizer v3 |
| split-3 | 1 hour ago | 32.2min | C:\Users... | HandwritingRecognizer v2 |
| split-2 | 1 hour ago | 32.2min | C:\Users... | HandwritingRecognizer v1 |

3 matching runs

Figure 1: The three runs in MLflow


MLflow
http://127.0.0.1:5000/#/experiments/964071936921879827/runs/69731366acdd4d14821652081078c9ff
Handwriting Recognition ›
split-2
Model registered
Overview
Model metrics
System metrics
Traces
Artifacts

| | |
|---|---|
| Status | Finished |
| Run ID | 69731366acdd4d14821652081078c9ff |
| Duration | 32.2min |
| Datasets used | — |
| Tags | Add tags |
| Source | C:\Users\anira\PycharmProjects\Anirudh-DA5402\Assignment 8\mlflow_handwriting_recognition.py |
| Logged models | keras |
| Registered models | HandwritingRecognizer v1 |
| Registered prompts | — |

Parameters (3)

| Parameter | Value |
|---|---|
| batch_size | 64 |
| epochs | 5 |
| split_id | 2 |

Metrics (3)

| Metric | Value |
|---|---|
| edit_distance | 19.703941345214844 |
| train_loss | 517.7730102539062 |
| val_loss | 444.106689453125 |

Figure 2: Overview of the run `split-2`

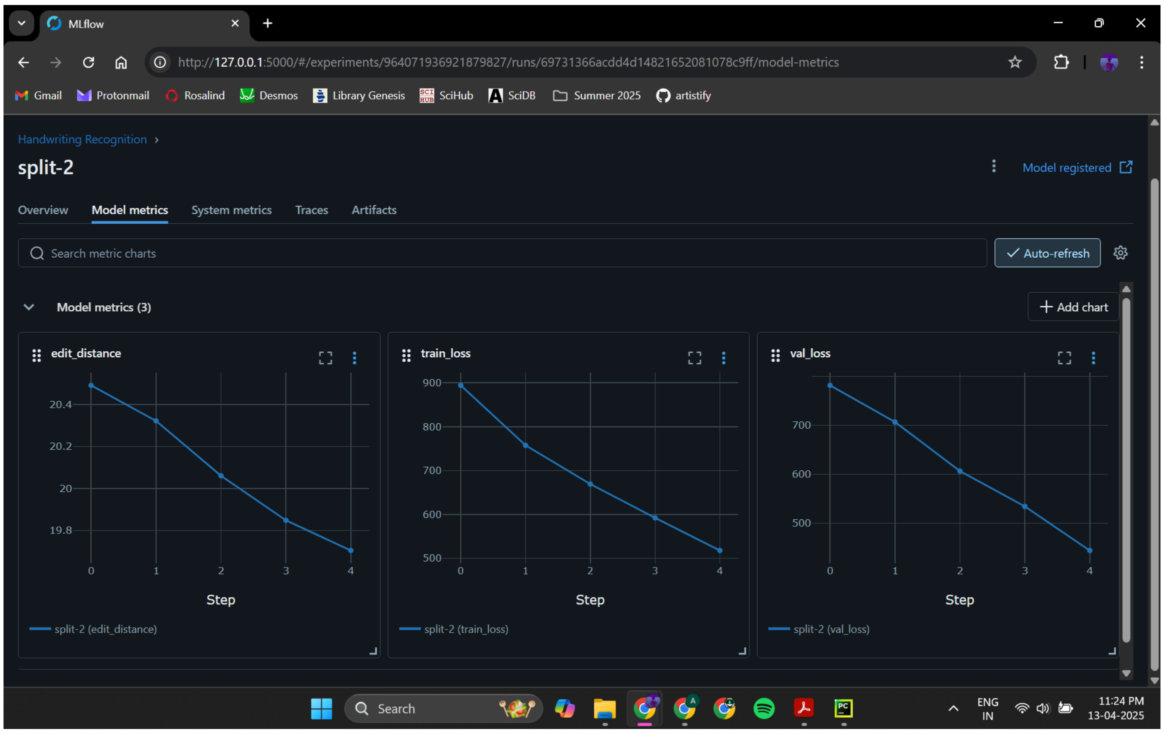
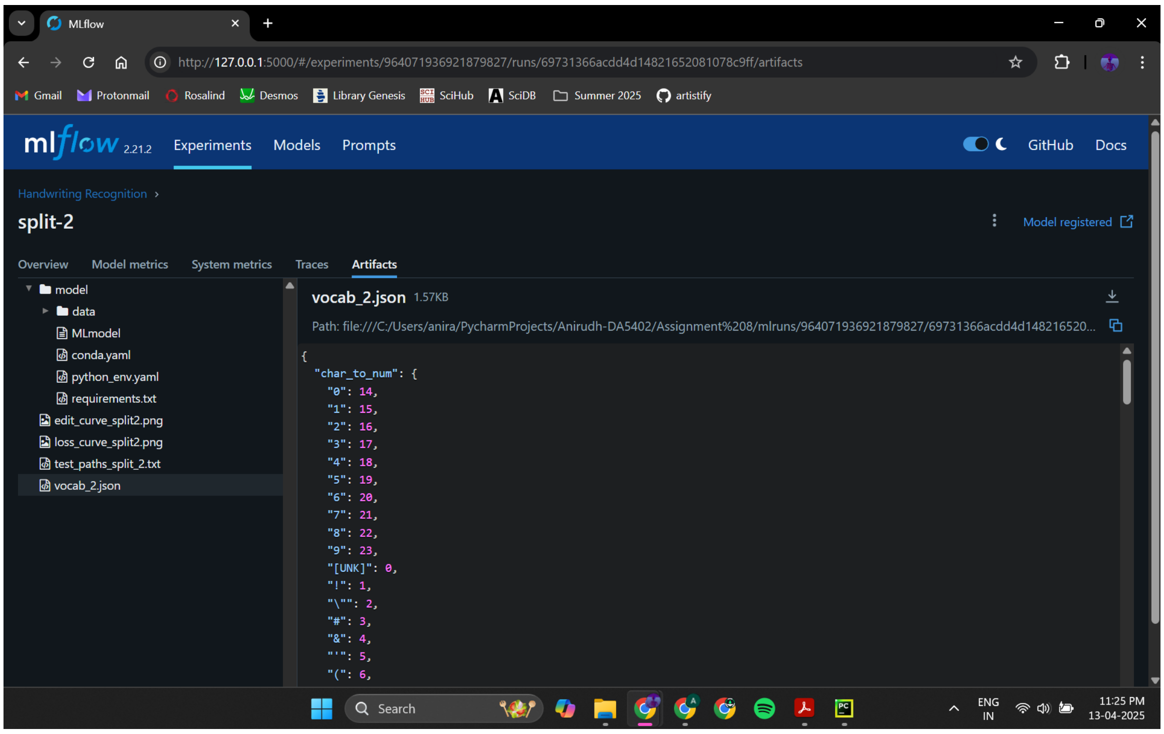


Figure 3: Metrics of the run `split-2`



split-2

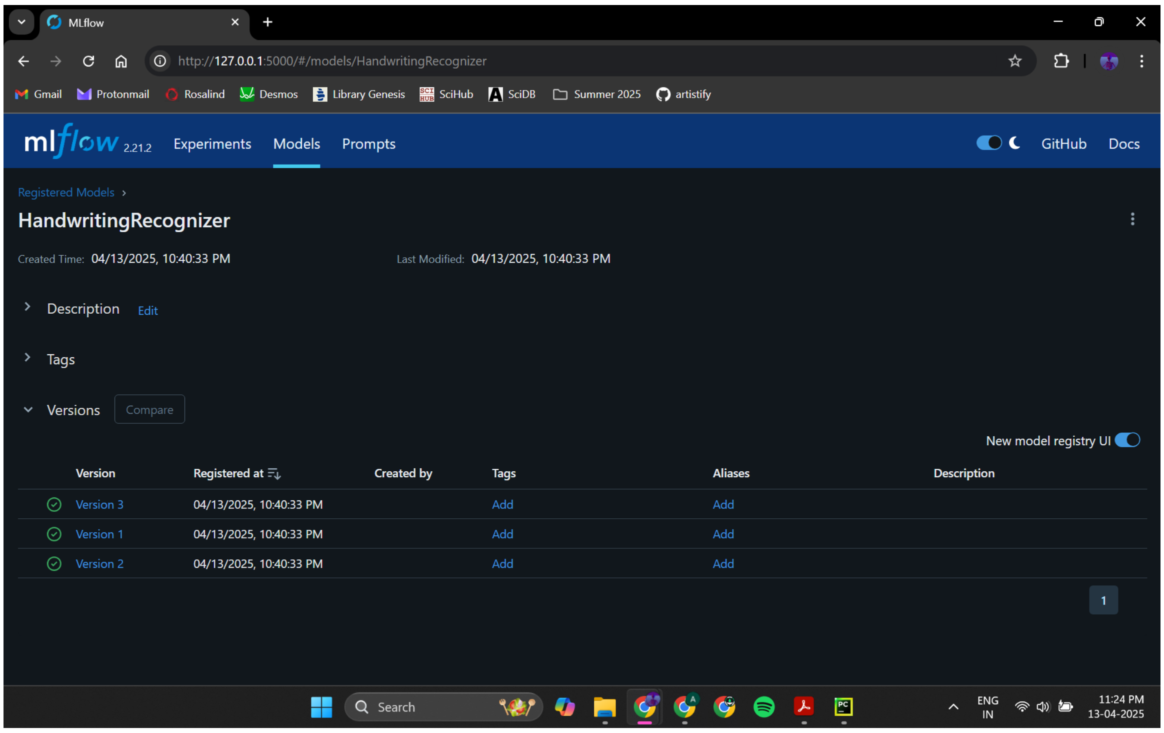
```
model
  data
  MLmodel
  conda.yaml
  python_env.yaml
  requirements.txt
edit_curve_split2.png
loss_curve_split2.png
test_paths_split_2.txt
vocab_2.json
```

vocab_2.json 1.57KB

Path: file:///C:/Users/anira/PycharmProjects/Anirudh-DA5402/Assignment%208/mlruns/964071936921879827/69731366acdd4d148216520...

```
{
  "char_to_num": {
    "0": 14,
    "1": 15,
    "2": 16,
    "3": 17,
    "4": 18,
    "5": 19,
    "6": 20,
    "7": 21,
    "8": 22,
    "9": 23,
    "[UNK]": 0,
    "!": 1,
    "\"": 2,
    "#": 3,
    "&": 4,
    "'": 5,
    "(": 6,
```

Figure 4: Artifacts of the run `split-2`

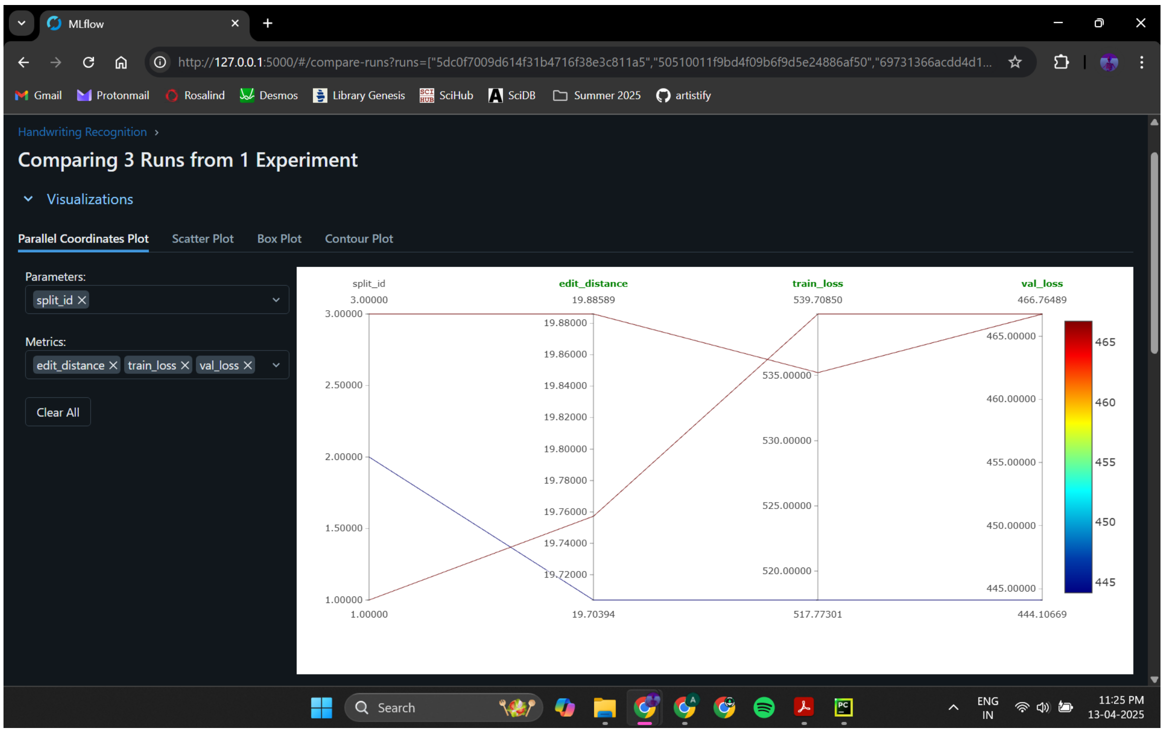


HandwritingRecognizer

Created Time: 04/13/2025, 10:40:33 PM    Last Modified: 04/13/2025, 10:40:33 PM

| Version | Registered at | Created by | Tags | Aliases | Description |
|---|---|---|---|---|---|
| Version 3 | 04/13/2025, 10:40:33 PM | | Add | Add | |
| Version 1 | 04/13/2025, 10:40:33 PM | | Add | Add | |
| Version 2 | 04/13/2025, 10:40:33 PM | | Add | Add | |

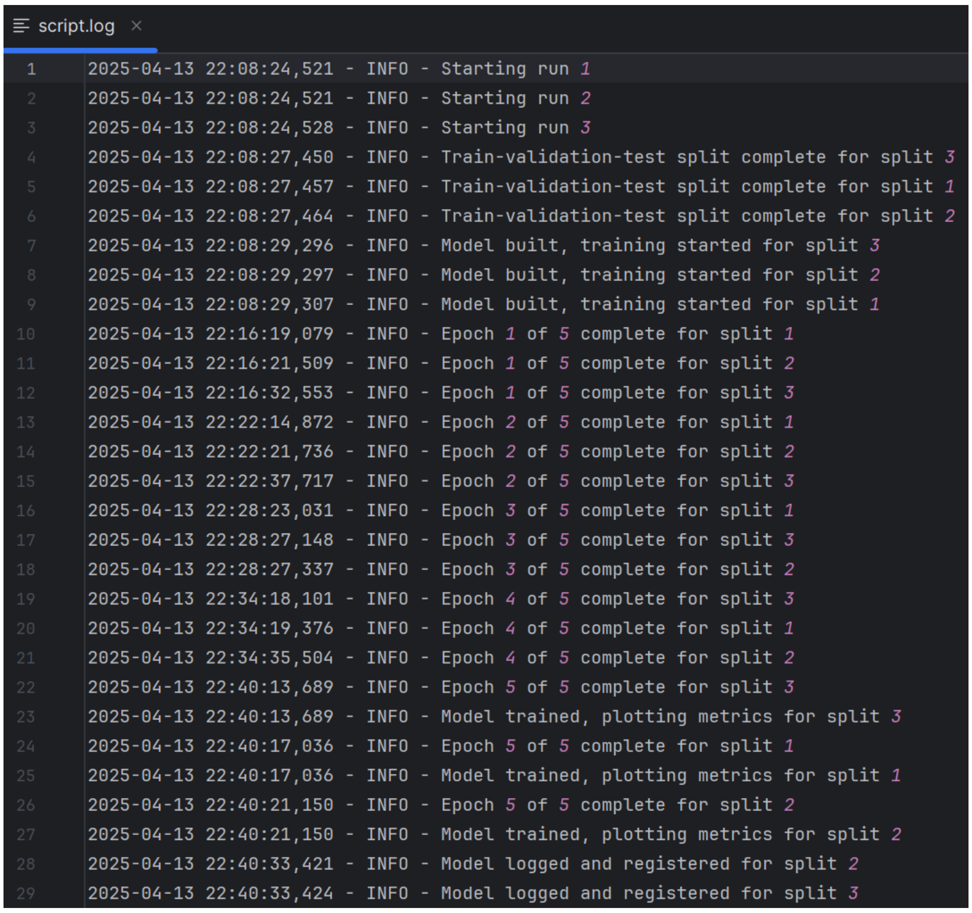
Figure 5: The three models registered in MLflow



Parameters: split_id

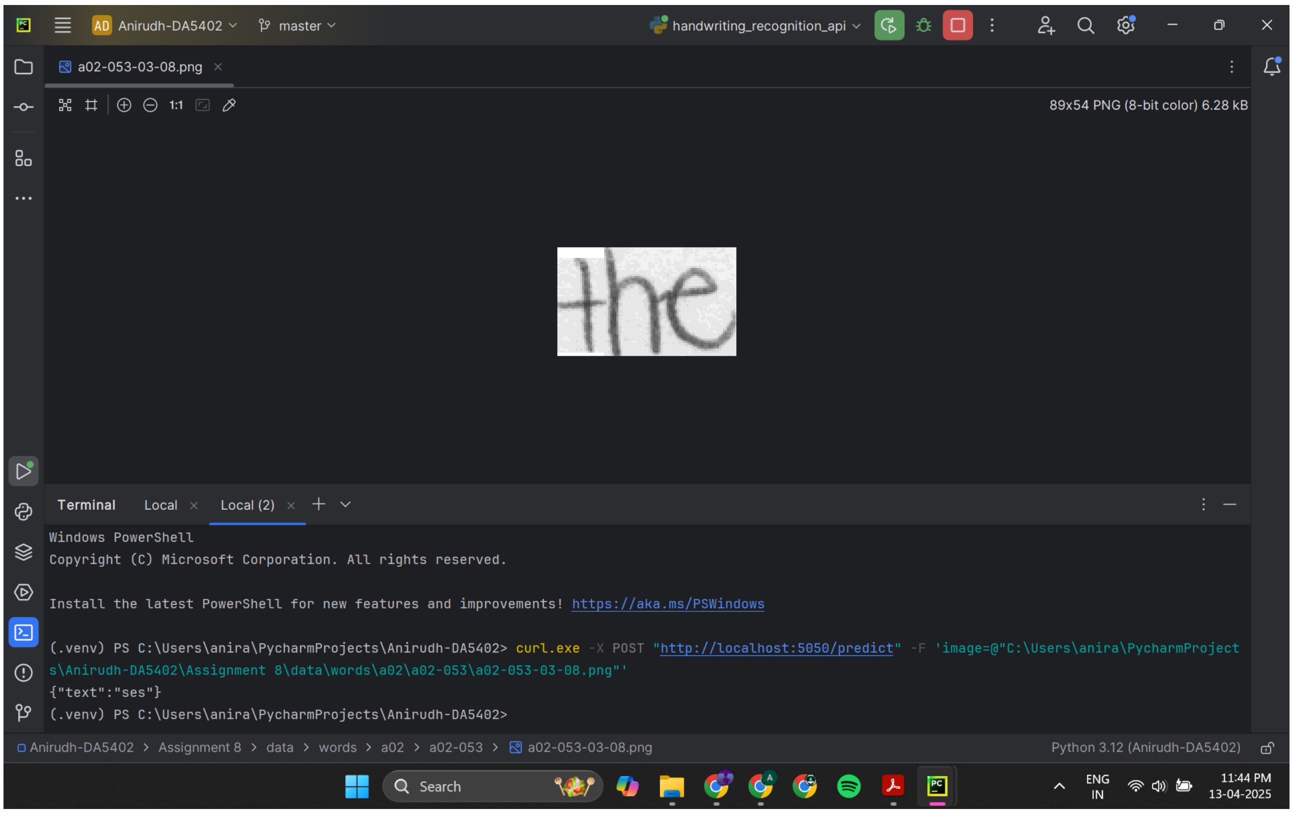
Metrics: edit_distance, train_loss, val_loss

Figure 6: Comparison of the three models registered in MLflow



```
2025-04-13 22:08:24,521 - INFO - Starting run 1
2025-04-13 22:08:24,521 - INFO - Starting run 2
2025-04-13 22:08:24,528 - INFO - Starting run 3
2025-04-13 22:08:27,450 - INFO - Train-validation-test split complete for split 3
2025-04-13 22:08:27,457 - INFO - Train-validation-test split complete for split 1
2025-04-13 22:08:27,464 - INFO - Train-validation-test split complete for split 2
2025-04-13 22:08:29,296 - INFO - Model built, training started for split 3
2025-04-13 22:08:29,297 - INFO - Model built, training started for split 2
2025-04-13 22:08:29,307 - INFO - Model built, training started for split 1
2025-04-13 22:16:19,079 - INFO - Epoch 1 of 5 complete for split 1
2025-04-13 22:16:21,509 - INFO - Epoch 1 of 5 complete for split 2
2025-04-13 22:16:32,553 - INFO - Epoch 1 of 5 complete for split 3
2025-04-13 22:22:14,872 - INFO - Epoch 2 of 5 complete for split 1
2025-04-13 22:22:21,736 - INFO - Epoch 2 of 5 complete for split 2
2025-04-13 22:22:37,717 - INFO - Epoch 2 of 5 complete for split 3
2025-04-13 22:28:23,031 - INFO - Epoch 3 of 5 complete for split 1
2025-04-13 22:28:27,148 - INFO - Epoch 3 of 5 complete for split 3
2025-04-13 22:28:27,337 - INFO - Epoch 3 of 5 complete for split 2
2025-04-13 22:34:18,101 - INFO - Epoch 4 of 5 complete for split 3
2025-04-13 22:34:19,376 - INFO - Epoch 4 of 5 complete for split 1
2025-04-13 22:34:35,504 - INFO - Epoch 4 of 5 complete for split 2
2025-04-13 22:40:13,689 - INFO - Epoch 5 of 5 complete for split 3
2025-04-13 22:40:13,689 - INFO - Model trained, plotting metrics for split 3
2025-04-13 22:40:17,036 - INFO - Epoch 5 of 5 complete for split 1
2025-04-13 22:40:17,036 - INFO - Model trained, plotting metrics for split 1
2025-04-13 22:40:21,150 - INFO - Epoch 5 of 5 complete for split 2
2025-04-13 22:40:21,150 - INFO - Model trained, plotting metrics for split 2
2025-04-13 22:40:33,421 - INFO - Model logged and registered for split 2
2025-04-13 22:40:33,424 - INFO - Model logged and registered for split 3
```

Figure 7: The `script.log` file


Anirudh-DA5402
master
handwriting_recognition_api
a02-053-03-08.png
89x54 PNG (8-bit color) 6.28 kB
Terminal
Local
Local (2)
Anirudh-DA5402 > Assignment 8 > data > words > a02 > a02-053 > a02-053-03-08.png
Python 3.12 (Anirudh-DA5402)

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

(.venv) PS C:\Users\anira\PycharmProjects\Anirudh-DA5402> curl.exe -X POST "http://localhost:5050/predict" -F 'image=@"C:\Users\anira\PycharmProject
s\Anirudh-DA5402\Assignment 8\data\words\a02\a02-053\a02-053-03-08.png"'
{"text":"ses"}
(.venv) PS C:\Users\anira\PycharmProjects\Anirudh-DA5402>
```

Figure 8: The model API being tested using `curl`

The image provided is in the test set of the model (determined using the test set artifact logged in MLflow). Since the model was only trained for 5 epochs, it is not very accurate.