CS5830 Image Classification

Team 09
R Prithiviraj (ME20B136)
B Srivathsan (ME20B176)
R S Harish Krishnan (ME20B081)

Overview

FastAPI

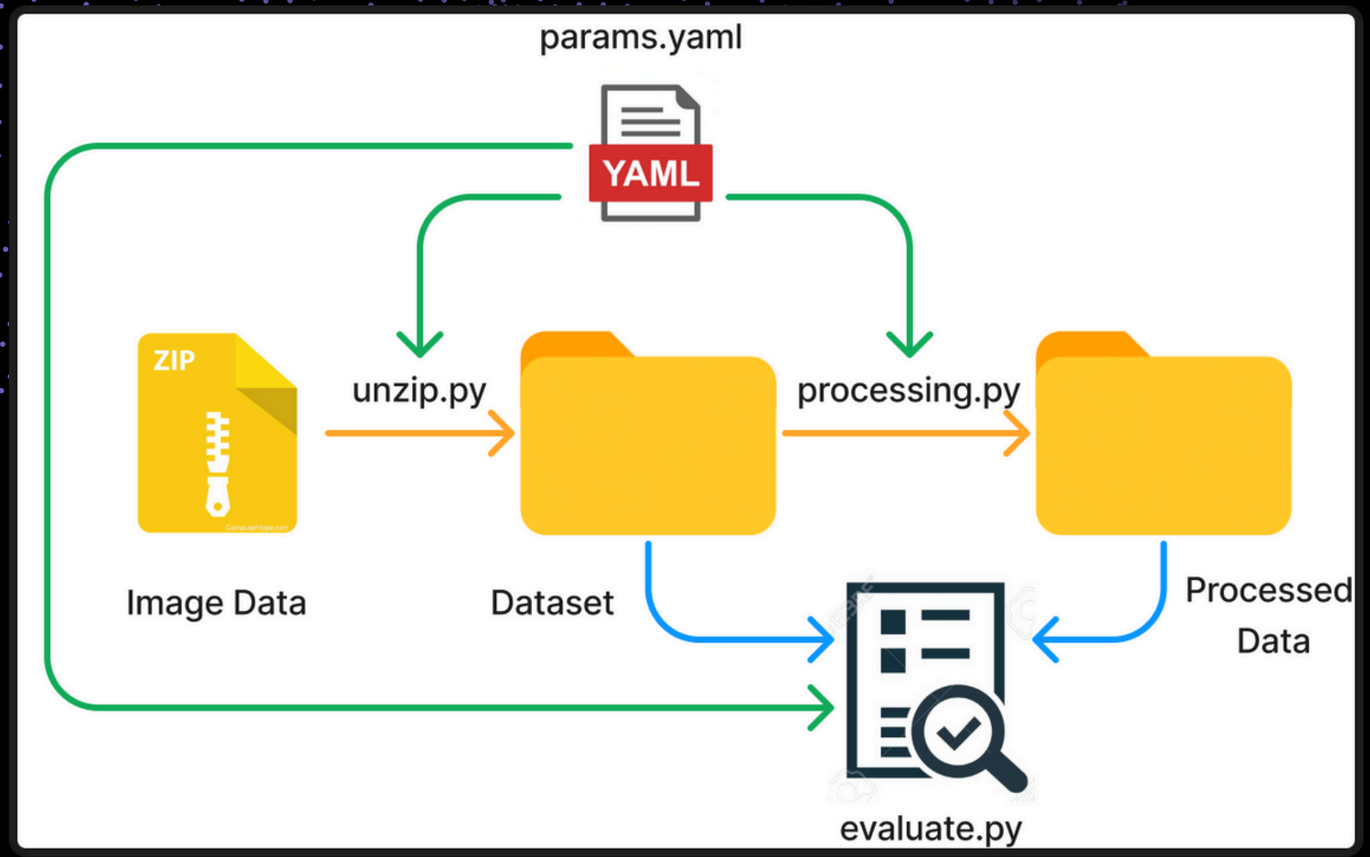
git

01 02 03 04 05 06 07 08

DVC

mlflow

Data Processing

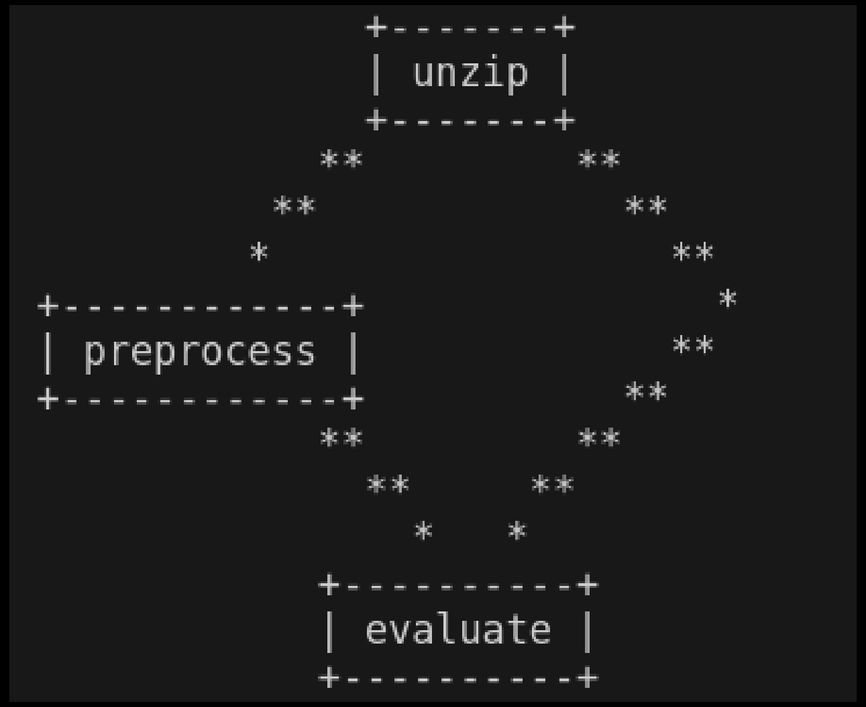


- 5 class dataset is available as a .zip file
- 3 stages were used for processing the data

- Stages Involved:
  - Unzipping the Dataset
  - Processing the file using **multiprocessing** as an alternative for **Spark**
    - Image Resizing
    - Mean Variance Normalization
  - Evaluate:
    - Check if the unzipped file and processed file have similar content structure

- Git was used for tracking the files and version control.
- DVC was used to pipeline the stages.

DAG



Prithiviraj7R › BDL-Project Private

+ Add a model | Help

Search experiments | Plots | Trends | Compare

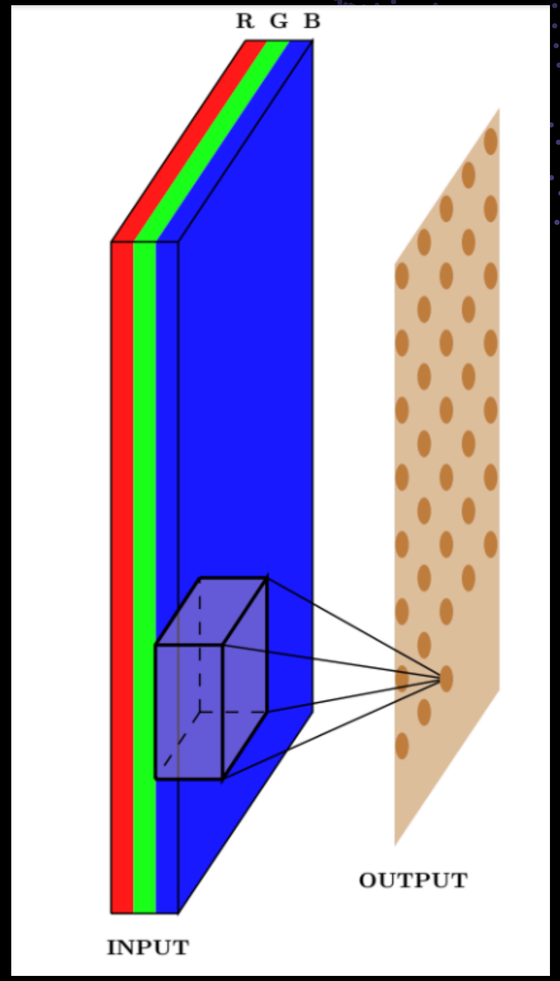
Experiment	Created	Message	CML	...s/live/metrics.json Matching	params.yaml: base zip_file_path	evaluate ORIGINAL_DAT...	evaluate OUTPUT_PATH	evaluate PROCESSED_D...	preprocess mean	preprocess output_folder	preprocess std	preprocess test_folder	preprocess train_folder	preprocess validation_folder
main														10 of 10 experiments (8 hidden)
BASELINE HEAD, main	04:59 PM	Update README.md		true	datasets.zip	datasets	eval_results	processed_data	[0.485,0.456,...	processed_data	[0.229,0.224,...	datasets/test	datasets/train	—
bc868f3	04:50 PM	Update README.md		true	datasets.zip	datasets	eval_results	processed_data	[0.485,0.456,...	processed_data	[0.229,0.224,...	datasets/test	datasets/train	—
a38df1c	04:50 PM	Update README.md		true	datasets.zip	datasets	eval_results	processed_data	[0.485,0.456,...	processed_data	[0.229,0.224,...	datasets/test	datasets/train	—
83b85a2	04:33 PM	Update README.md		true	datasets.zip	datasets	eval_results	processed_data	[0.485,0.456,...	processed_data	[0.229,0.224,...	datasets/test	datasets/train	—
3745f32	04:06 PM	Added Prometheus and Gr...		true	datasets.zip	datasets	eval_results	processed_data	[0.485,0.456,...	processed_data	[0.229,0.224,...	datasets/test	datasets/train	—
e495821	04:00 PM	Model Selection using MLfl...		true	datasets.zip	datasets	eval_results	processed_data	[0.485,0.456,...	processed_data	[0.229,0.224,...	datasets/test	datasets/train	—
96158b2	03:25 PM	Experiment run in DVC		true	datasets.zip	datasets	eval_results	processed_data	[0.485,0.456,...	processed_data	[0.229,0.224,...	datasets/test	datasets/train	—

Modelling

- Evaluated five CNN models with varying
  - Filter sizes
  - Number of filters
  - Batch normalization
  - Learning rate
  - Number of epochs

- Tracked performance metrics using mlflow, highlighting optimal parameter combinations for best accuracy and lowest loss.

- Results can be visualized through the mlflow UI



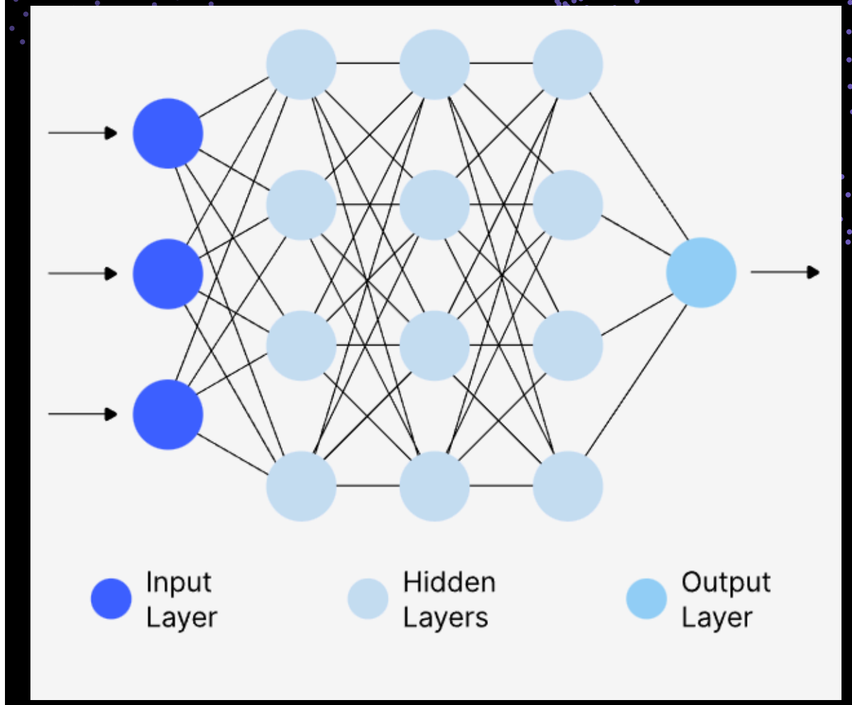


Image classification

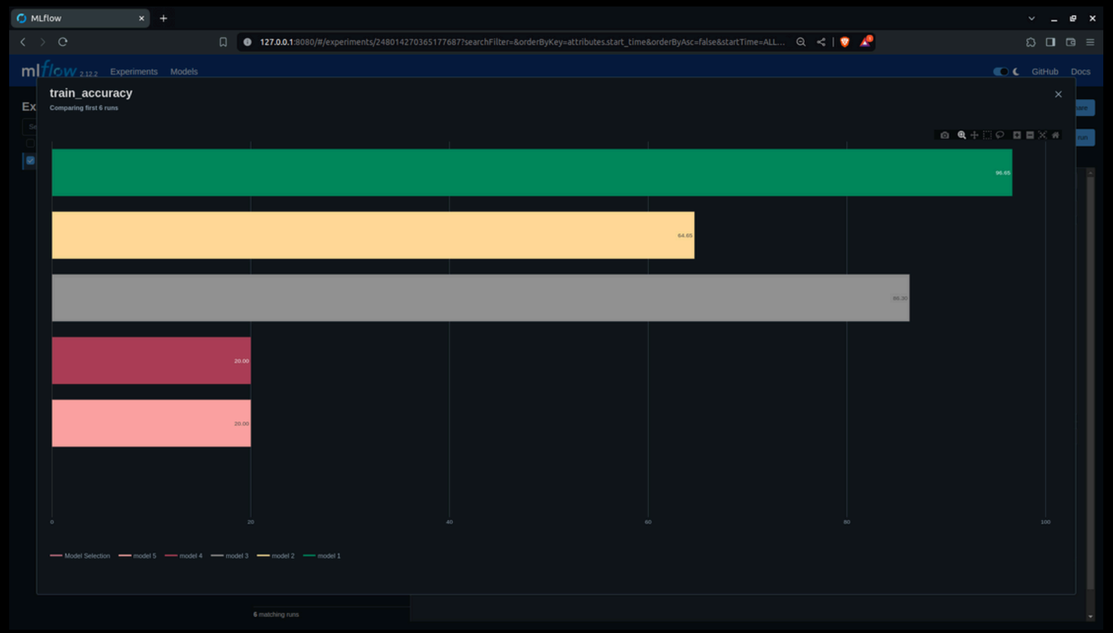
Provide Feedback | Add Description

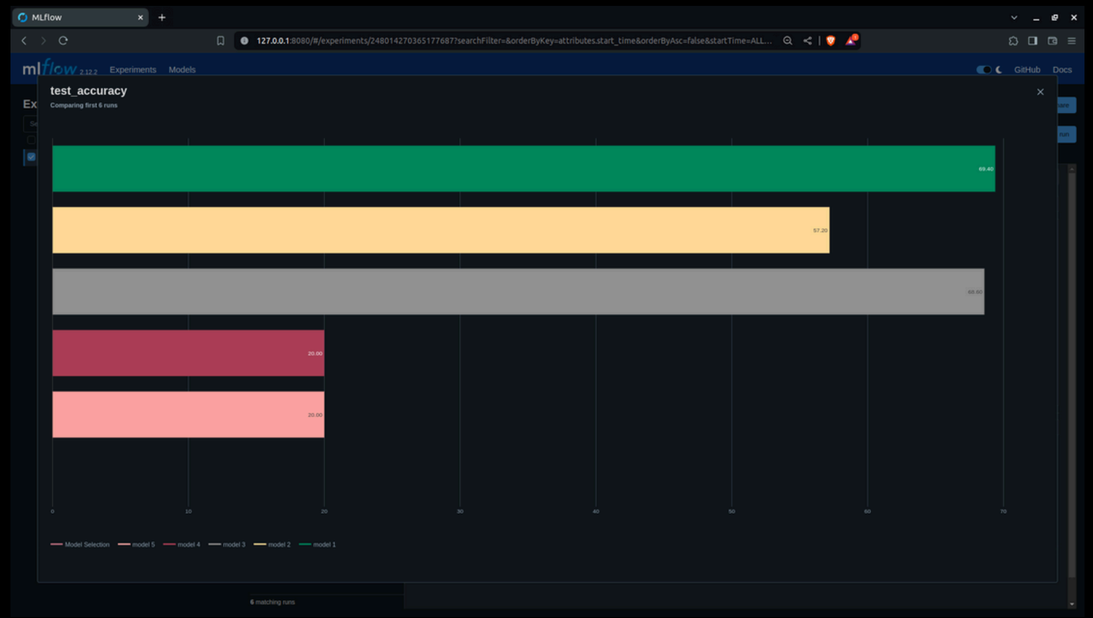
Share

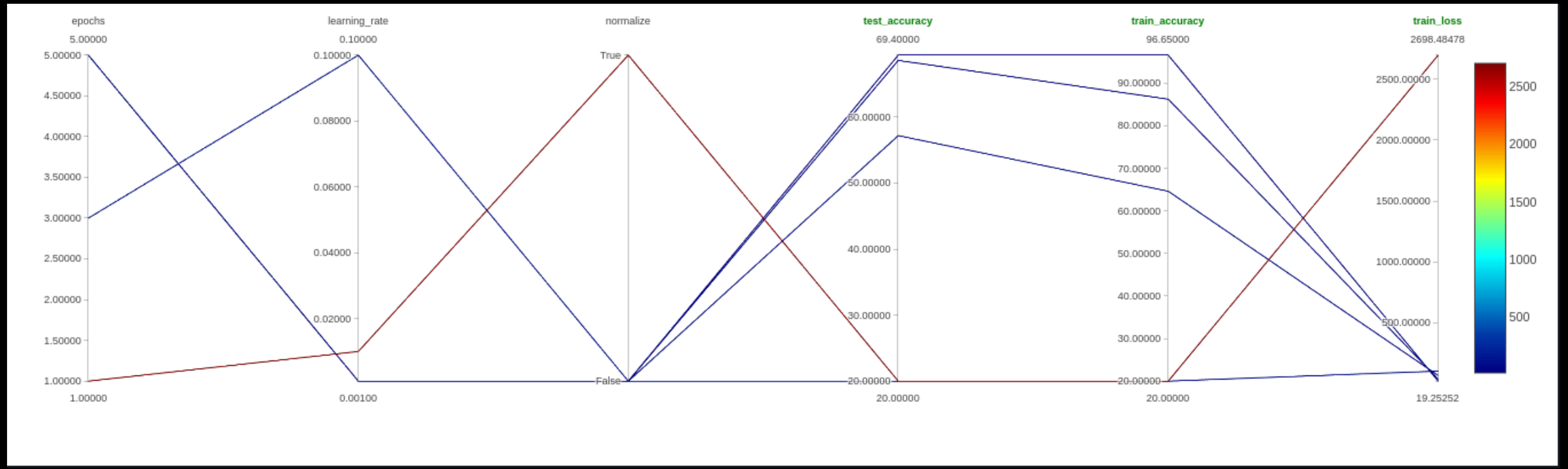
metrics.rmse < 1 and params.model = "tree" | Time created | State: Active | Datasets | Sort: Created | Columns | Group by | New run

Table | Chart | Evaluation | Experimental

				Metrics			Parameters				
Run Name	Created	Duration	test_accuracy	train_accuracy	train_loss	conv1	conv2	epochs	learning_rate	normalize	
Model Selection	19 minutes ago	19.2min	-	-	-	-	-	-	-	-	Show more columns (7 total)
model 5	1 minute ago	1.5min	20	20	101.820723...	{'in_channe...	{'in_channe...	3	0.1	False	
model 4	3 minutes ago	1.5min	20	20	2698.48478...	{'in_channe...	{'in_channe...	1	0.01	True	
model 3	11 minutes ago	7.2min	68.6000000...	86.3	35.8205042...	{'in_channe...	{'in_channe...	5	0.001	False	
model 2	17 minutes ago	4.6min	57.1999999...	64.6499999...	65.8338003...	{'in_channe...	{'in_channe...	5	0.001	False	
model 1	19 minutes ago	2.1min	69.3999999...	96.65	19.2525206...	{'in_channe...	{'in_channe...	5	0.001	False	



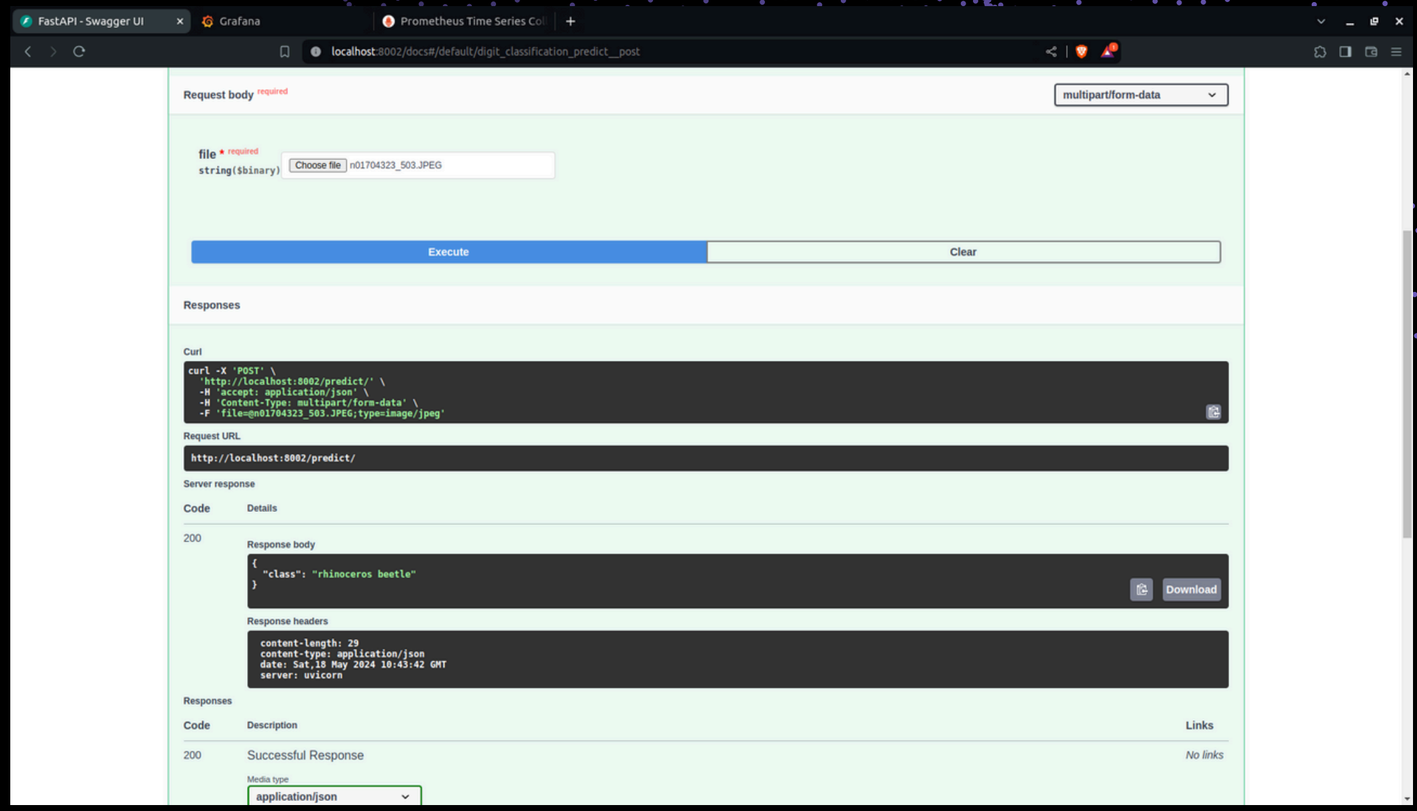



epochs
5.00000
5.00000
4.50000
4.00000
3.50000
3.00000
2.50000
2.00000
1.50000
1.00000
1.00000
learning_rate
0.10000
0.10000
0.08000
0.06000
0.04000
0.02000
0.00100
normalize
True
False
test_accuracy
69.40000
90.00000
80.00000
70.00000
60.00000
50.00000
40.00000
30.00000
20.00000
20.00000
train_accuracy
96.65000
90.00000
80.00000
70.00000
60.00000
50.00000
40.00000
30.00000
20.00000
20.00000
train_loss
2698.48478
2500.00000
2000.00000
1500.00000
1000.00000
500.00000
19.25252
2500
2000
1500
1000
500

FastAPI

Model Deployment

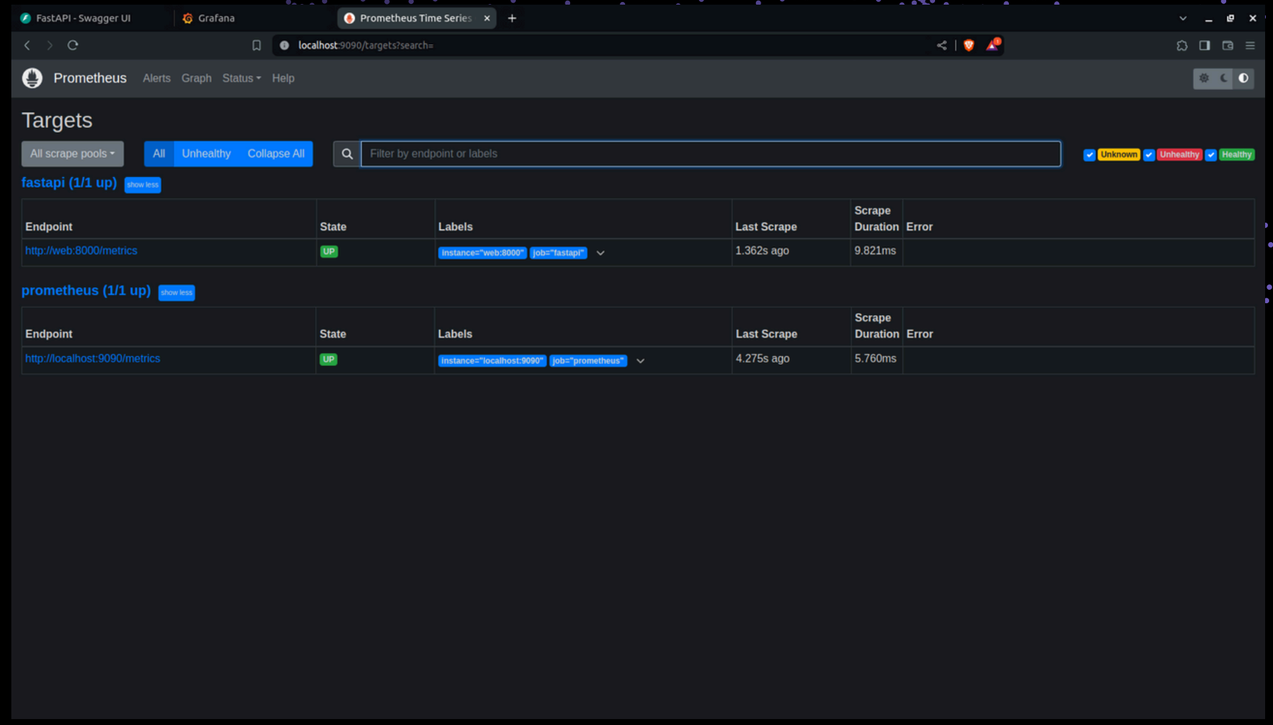
- The best model based on performance on the test data was chosen to be deployed.
- A FastAPI application is designed to host the model's predictive capabilities. The API endpoint accepts image uploads, then processes the image, before passing it through a pretrained CNN model and returns the predicted class label.
- The application is finally hosted using Uvicorn and accessible for SwaggerUI documentation.



Health Monitoring

- The application uses the Instrumentator to instrument and expose metrics. Various Prometheus metrics are set up to monitor API usage, run time, memory and CPU usage, as well as network I/O.

- Then customized dashboards have been created using Grafana to visualize and analyze Prometheus metrics for the Rest API. The dashboards empower proactive monitoring and help troubleshooting issues, setting alerts, thus ensuring SLOs are met.



Dashboard


Search or jump to...
ctrl+k
Home › Dashboards › New dashboard
Add
Share
Last 5 minutes
API requests by IP
12
API runtime (milli seconds)
125
API T/L time (micro seconds)
2.49
Memory Usage (GB)
6.79
CPU Utilisation rate (%)
12.8
Network I/O bytes (KB)
540
520
500
16:19:00
16:19:30
16:20:00
16:20:30
16:21:00
16:21:30
16:22:00
16:22:30
16:23:00
16:23:30
Network I/O bytes rate (KB/s)
6
5.5
5
4.5
16:19:00
16:19:30
16:20:00
16:20:30
16:21:00
16:21:30
16:22:00
16:22:30
16:23:00
16:23:30

Dockerization

- Docker was used to containerize the FastAPI application, ensuring a consistent and reproducible environment across different stages of development and deployment.
- Alongside FastAPI, Prometheus and Grafana was integrated within the Docker container to collect and store real-time metrics, visualize these metrics, providing a comprehensive dashboard for real-time analytics and insights into the application's performance.
- This containerized setup streamlined the deployment process, enhanced scalability, and ensured that monitoring and visualization tools were readily available and consistently configured across all environments.

Version Control

- Implemented version control using Git with a well structured repository to include data processing scripts, model training code, the FastAPI application, and monitoring setups with Docker configurations.
- Proper documentation detailing methods and usage steps, ensuring each part of the project, from setup to execution has been done using github.

Github repo : https://github.com/Prithiviraj7R/BDL-Project

Future Improvements

- Usage of pretrained models like ResNet or GoogLeNet can improve the model performance.
- Train on a larger dataset to have better performance, also with larger dataset, processing would take more time, and hence usage of Spark would become justifiable.

Contributions

R S Harish Krishnan (ME20B081): Integrated Prometheus and Grafana into a FastAPI application and containerized the application using Docker.

R Prithiviraj (ME20B136): Implemented the model and tracked model tuning using MLflow, and developed the FastAPI application.

B Srivathsan (ME20B176): Developed the data processing pipeline using DVC and containerized the application using Docker.

THANK YOU SO MUCH!