

Exp No. 8	Monitor and logging in Machine Learning with Python	REG. NO: URK23CS1197
Date:		

Objective:

The objective is to ensure the machine learning models are performing as expected by tracking their performance, logging relevant metrics, and diagnosing any issues that arise during training and deployment.

Tools/ Software Required:

- pandas
- scikit-learn
- joblib
- pytest
- fastapi
- uvicorn
- python-multipart

Job Role:

Machine Learning Engineer / Data Scientist

Skills Required:

- **Proficiency in Python:** Strong understanding of Python programming language.
- **Machine Learning Frameworks:** Familiarity with frameworks like TensorFlow, PyTorch, or Scikit-learn.
- **Logging Libraries:** Knowledge of Python logging libraries (e.g., logging module) and ML-specific logging tools (e.g., MLflow).
- **Monitoring Tools:** Experience with monitoring tools and platforms (e.g., Prometheus, Grafana).
- **Data Analysis:** Ability to analyze logged data to identify trends and issues.
- **Debugging:** Strong debugging skills to troubleshoot and resolve issues in ML models.

Prerequisites:

- **Basic Understanding of Machine Learning:** Knowledge of machine learning concepts and algorithms.
- **Experience with Python:** Comfortable working with Python for data manipulation and analysis.
- **Familiarity with ML Frameworks:** Prior experience with at least one machine learning framework.
- **Basic Knowledge of Logging and Monitoring:** Understanding of how to implement logging and monitoring in software development.

Description:

Logging is a method of tracking and storing data to ensure application availability and to assess the impact of state transformations on performance. The purpose of logging is to create an ongoing record of application events. Log files can be used to review any event within a system, including failures and state transformations. Consequently, log messages can provide valuable information to help pinpoint the cause of performance problems. Log data can help DevOps teams troubleshoot issues by identifying which changes resulted in error reporting, but is only as valuable as the information it contains.

Monitoring is a diagnostic tool used for alerting DevOps to system-related issues by analysing metrics. Monitoring metrics is like the security alarm that alerts you to a possible intrusion; log files act as the security camera footage that will provide clues to tell you what happened and how.

Commands:

Logging

```
import logging

# Create a custom logger
logger = logging.getLogger(__name__)

# Set the logging level
logger.setLevel(logging.INFO)

# Create handlers
c_handler = logging.StreamHandler()
f_handler = logging.FileHandler('file.log')

# Create formatters and add them to handlers
c_format = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
f_format = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')

c_handler.setFormatter(c_format)
f_handler.setFormatter(f_format)

# Add handlers to the logger
logger.addHandler(c_handler)
logger.addHandler(f_handler)

# Example logging
logger.info('This is an info message')
logger.error('This is an error message')
```

Monitoring

```
import tensorflow as tf
from tensorflow.keras.callbacks import TensorBoard

# Load your dataset
```

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

```
# Preprocess your data
```

```
x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
# Create your model
```

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
```

```
tf.keras.layers.Dense(10)
```

```
])
```

```
# Compile your model
```

```
model.compile(optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy'])
```

```
# Set up TensorBoard callback
```

```
tensorboard_callback = TensorBoard(log_dir="./logs")
```

```
# Train your model with TensorBoard callback
```

```
model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test),
    callbacks=[tensorboard_callback])
```

To visualize the logs, you can run TensorBoard in your terminal:

```
bash
```

```
tensorboard --logdir=./logs
```

Questions:

1. Write a Python script using the logging module to log the training loss and accuracy of a neural network model at each epoch.
2. Design and implement a custom monitoring and logging system for a machine learning project using Python, TensorBoard, and Prometheus.

Program:

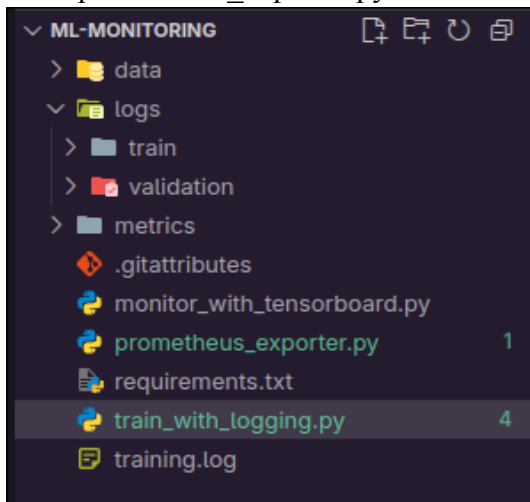
Project Structure:

Ex8_ML_Monitoring/

```

├── metrics/
├── logs/
├── train_with_logging.py
├── monitor_with_tensorboard.py
└── prometheus_exporter.py

```

**requirements.txt**

```

tensorflow
numpy
prometheus-client
tensorboard
protobuf
grpcio

```

monitor_with_tensorboard.py

```

import tensorflow as tf
from tensorflow.keras.callbacks import TensorBoard
from tensorflow.keras.datasets import mnist

```

```

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train / 255.0
x_test = x_test / 255.0

```

```

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28,28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10)
])

```

```

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),

```

```

metrics=['accuracy'])

tensorboard_callback = TensorBoard(log_dir="logs/")

model.fit(x_train, y_train, validation_data=(x_test, y_test),
          epochs=5, callbacks=[tensorboard_callback])

prometheus_exported.py
from prometheus_client import start_http_server, Gauge
import tensorflow as tf
from tensorflow.keras.datasets import mnist
import time

print("✅ Prometheus Metrics Exporter Running at http://127.0.0.1:8000/metrics")

loss_metric = Gauge("training_loss", "Training Loss")
accuracy_metric = Gauge("training_accuracy", "Training Accuracy")

start_http_server(8000)

(x_train, y_train), _ = mnist.load_data()
x_train = x_train / 255.0

model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(28,28)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

for epoch in range(5):
    history = model.fit(x_train, y_train, epochs=1, verbose=0)
    loss = history.history['loss'][0]
    accuracy = history.history['accuracy'][0]
    loss_metric.set(loss)
    accuracy_metric.set(accuracy)
    print(f'Epoch {epoch+1} | Loss: {loss:.4f}, Accuracy: {accuracy:.4f}')

# ✅ Keep server running forever
while True:
    time.sleep(1)

```

train_with_logging.py

```

import logging
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical

logging.basicConfig(filename='training.log', level=logging.INFO,
                    format='%(asctime)s - %(levelname)s - %(message)s')

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train / 255.0
x_test = x_test / 255.0

y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

class LogTrainingCallback:
    def __init__(self, model):
        self.model = model

    def on_epoch_end(self, epoch, logs):
        loss = logs.get('loss')
        acc = logs.get('accuracy')
        logging.info(f'Epoch {epoch+1}: Loss={loss:.4f}, Accuracy={acc:.4f}')

log_callback = LogTrainingCallback(model)

for epoch in range(5):
    history = model.fit(x_train, y_train, epochs=1, verbose=0)
    logs = history.history
    log_callback.on_epoch_end(epoch, {'loss': logs['loss'][0], 'accuracy': logs['accuracy'][0]})

```

Expected Output :

Q1.

python train_with_logging.py

The screenshot shows a Jupyter Notebook environment with the following components:

- File Explorer:** Shows the project structure with files like `train_with_logging.py`, `training.log`, `requirements.txt`, and `monitor_with_tensorboard.py`.
- Code Editor:** Displays the code for `train_with_logging.py`, which imports `logging`, `tensorflow.keras.datasets`, `tensorflow.keras.models`, `tensorflow.keras.layers`, and `tensorflow.keras.utils`. It defines a `model` with `Flatten` and `Dense` layers, and sets up logging to `training.log`.
- Terminal:** Shows the output of the command `python train_with_logging.py`. The output includes:
 - Initial log messages from `tensorflow` and `tensorflow.keras`.
 - Warning messages about GPU device initialization and memory allocation.
 - The training progress for 5 epochs, showing loss and accuracy values.

training.log

```
2025-11-06 19:22:26,402 - INFO - Epoch 1: Loss=0.2568, Accuracy=0.9264
2025-11-06 19:22:30,369 - INFO - Epoch 2: Loss=0.1102, Accuracy=0.9679
2025-11-06 19:22:35,079 - INFO - Epoch 3: Loss=0.0767, Accuracy=0.9770
2025-11-06 19:22:39,444 - INFO - Epoch 4: Loss=0.0576, Accuracy=0.9820
2025-11-06 19:22:43,699 - INFO - Epoch 5: Loss=0.0456, Accuracy=0.9861
```

Q2.

python monitor_with_tensorboard.py

```

1 from tensorflow.keras.callbacks import TensorBoard
2 from tensorflow.keras.datasets import mnist
3
4 (x_train, y_train), (x_test, y_test) = mnist.load_data()
5 x_train = x_train / 255.0
6 x_test = x_test / 255.0
7
8 model = tf.keras.models.Sequential([
9     tf.keras.layers.Flatten(input_shape=(28,28)),
10    tf.keras.layers.Dense(128, activation='relu'),
11    tf.keras.layers.Dropout(0.2),
12    tf.keras.layers.Dense(10)
13 ])
14
15 model.compile(optimizer='adam',
16               loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
17               metrics=['accuracy'])
18
19 ts = tf.keras.callbacks.TensorBoard(log_dir='./logs')
20 model.fit(x_train, y_train, validation_data=(x_test, y_test),
21          callbacks=[ts], epochs=10)

```

Terminal Output:

```

(mlops) ~/Documents/College_Work/MLOps/EXPS/ml-monitoring
└─(07:23:53 PM on main *)→ python monitor_with_tensorboard.py

2025-11-06 19:23:57.009053: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:31] Could not find cuda drivers on your machine, GPU will not be used.
2025-11-06 19:23:57.072128: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2_FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2025-11-06 19:23:58.946219: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:31] Could not find cuda drivers on your machine, GPU will not be used.
/home/jerine/anaconda3/envs/mlops/lib/python3.11/site-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an 'input_shape' / 'input_dim' argument to a layer. When using Sequential models, prefer using an 'Input(shape)' object as the first layer in the model instead.
super().__init__(**kwargs)
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1762437241.211468 106816 cuda_executor.cc:1309] INTERNAL: CUDA Runtime error: Failed call to cudaGetRuntimeVersion: Error loading CUDA libraries. GPU will not be used.: Error loading CUDA libraries. GPU will not be used.
W0000 00:00:1762437241.218198 106816 gpu_device.cc:2342] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above are installed properly if you would like to use GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required libraries for your platform.
Skipping registering GPU devices...
2025-11-06 19:24:01.403464: W external/local_xla/xla/tsl/framework/cpu_allocator_impl.cc:84] Allocation of 188160000 exceeds 10% of free system memory.
Epoch 1/5
1875/1875 ━━━━━━━━━━━ 65 3ms/step - accuracy: 0.9155 - loss: 0.2897 - val_accuracy: 0.9573 - val_loss: 0.1378
Epoch 2/5
1875/1875 ━━━━━━━━━━━ 55 3ms/step - accuracy: 0.9599 - loss: 0.1376 - val_accuracy: 0.9690 - val_loss: 0.0984
Epoch 3/5
1875/1875 ━━━━━━━━━━━ 65 3ms/step - accuracy: 0.9685 - loss: 0.1040 - val_accuracy: 0.9722 - val_loss: 0.0923
Epoch 4/5
1875/1875 ━━━━━━━━━━━ 65 3ms/step - accuracy: 0.9744 - loss: 0.0841 - val_accuracy: 0.9769 - val_loss: 0.0747
Epoch 5/5
1875/1875 ━━━━━━━━━━━ 65 3ms/step - accuracy: 0.9772 - loss: 0.0730 - val_accuracy: 0.9775 - val_loss: 0.0718
└─(07:24:47 PM on main *)→

```

Tensorboard
tensorboard --logdir=logs

```

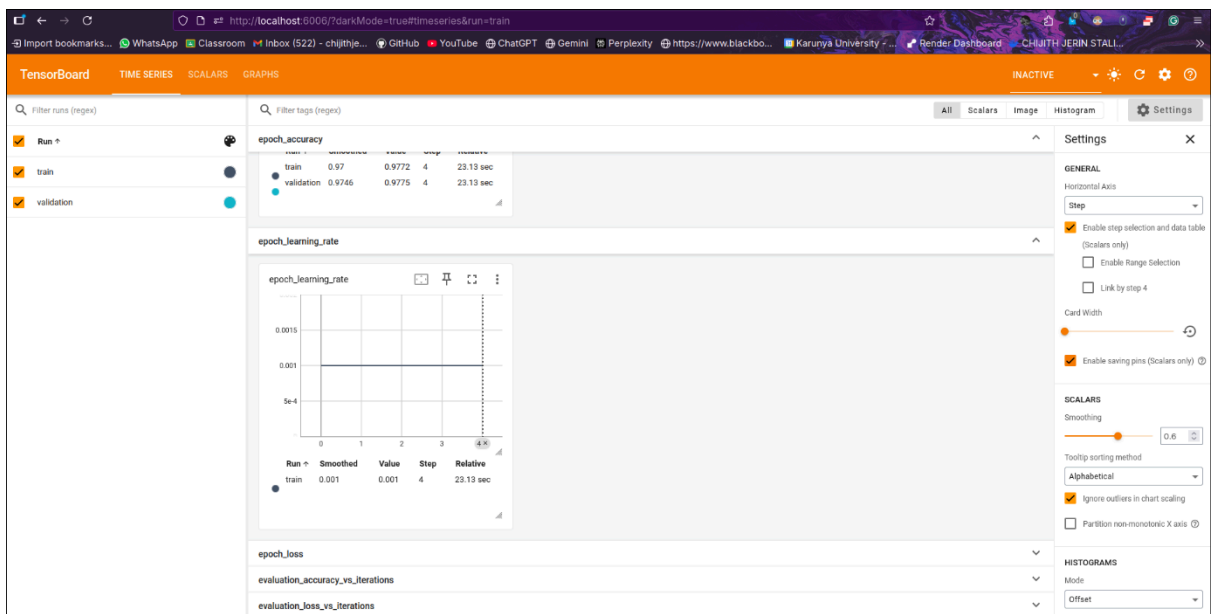
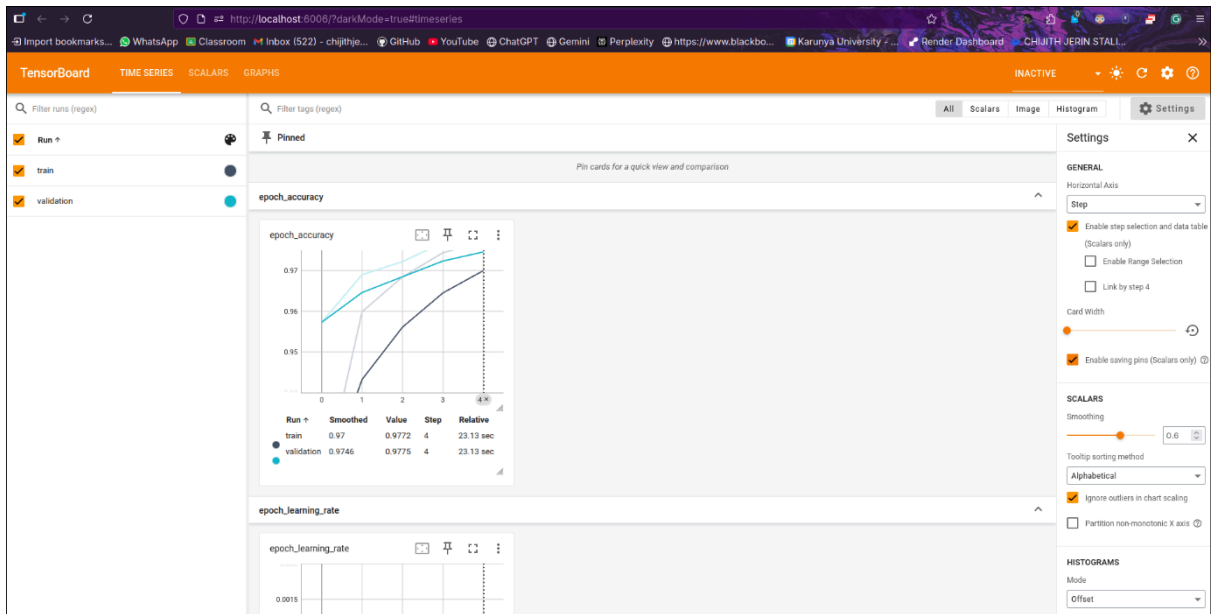
(mlops) ~/Documents/College_Work/MLOps/EXPS/ml-monitoring
└─(07:25:19 PM on main *)→ tensorboard --logdir=logs

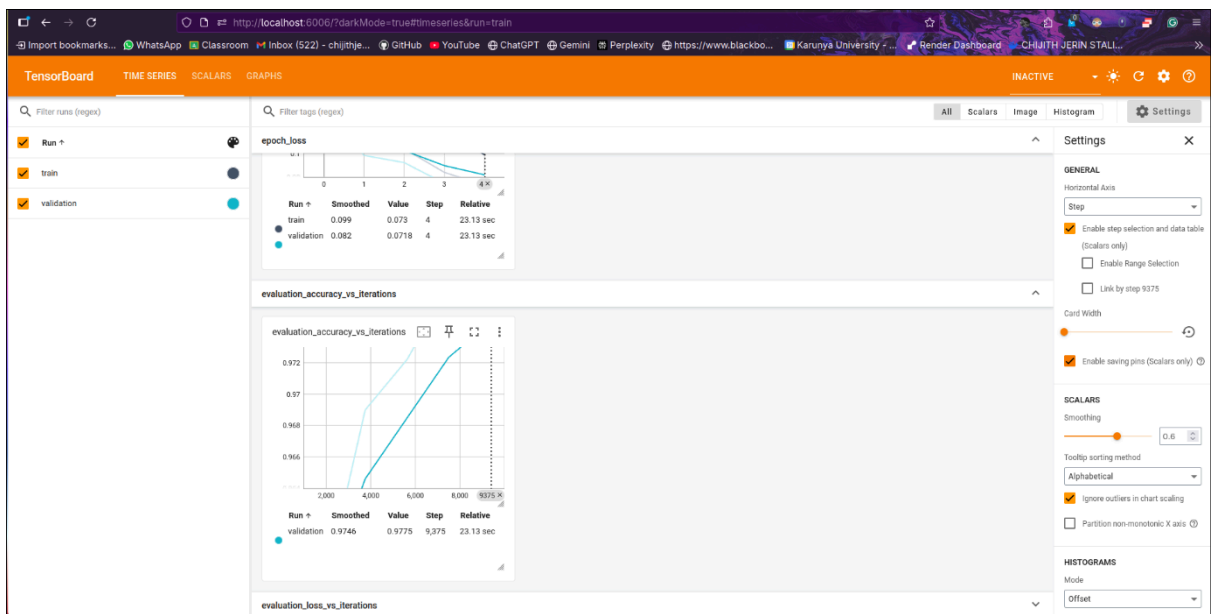
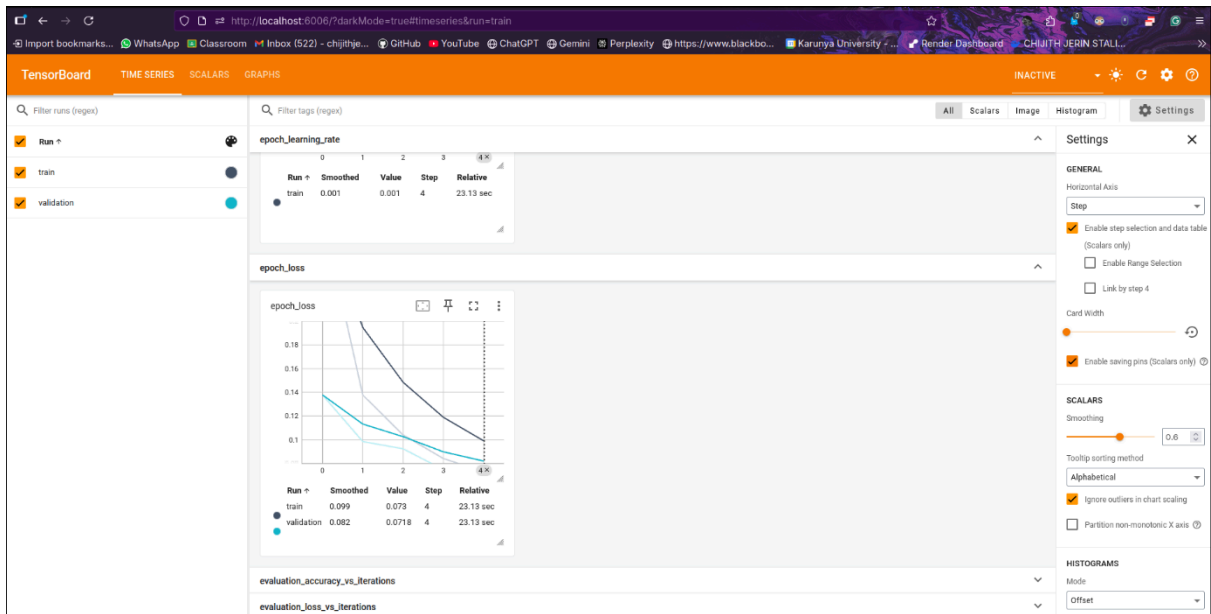
/home/jerine/anaconda3/envs/mlops/lib/python3.11/site-packages/tensorboard/default.py:30: UserWarning: pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain from using this package or pin to Setuptools<81.
import pkg_resources
2025-11-06 19:25:21.610063: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:31] Could not find cuda drivers on your machine, GPU will not be used.
2025-11-06 19:25:21.676983: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2_FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2025-11-06 19:25:23.99917: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:31] Could not find cuda drivers on your machine, GPU will not be used.
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1762437325.355485 109245 cuda_executor.cc:1309] INTERNAL: CUDA Runtime error: Failed call to cudaGetRuntimeVersion: Error loading CUDA libraries. GPU will not be used.: Error loading CUDA libraries. GPU will not be used.
W0000 00:00:1762437325.362195 109245 gpu_device.cc:2342] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above are installed properly if you would like to use GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required libraries for your platform.
Skipping registering GPU devices...

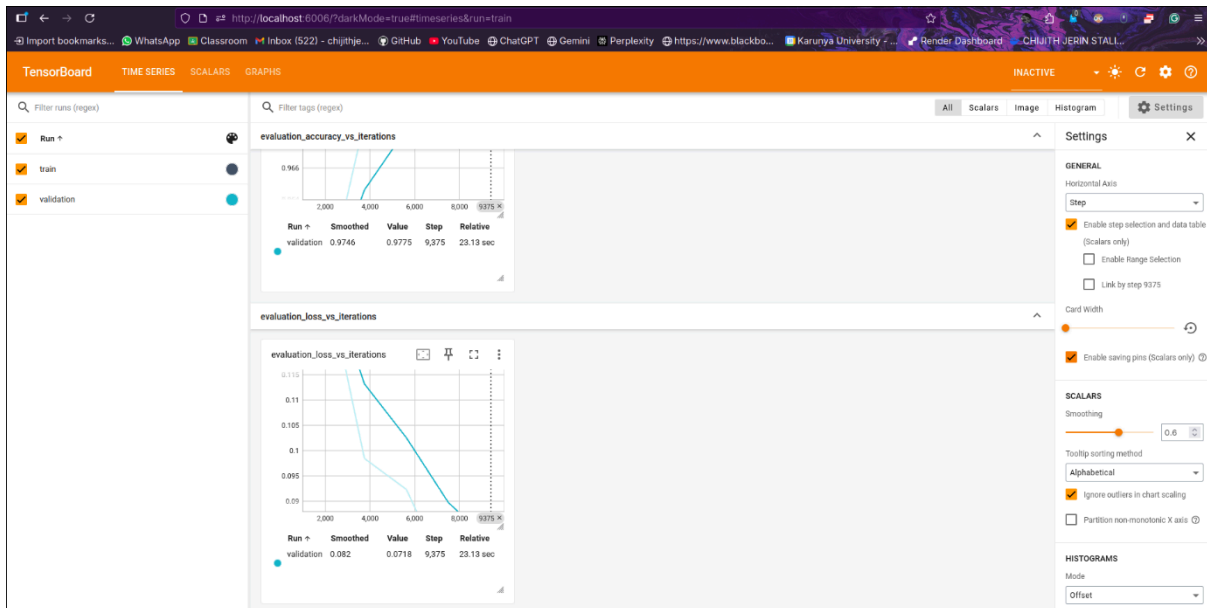
NOTE: Using experimental fast data loading logic. To disable, pass
"--load_fast=false" and report issues on GitHub. More details:
https://github.com/tensorflow/tensorboard/issues/4784

Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all
TensorBoard 2.20.0 at http://localhost:6006/ (Press CTRL+C to quit)

```





python prometheus_exporter.py

```

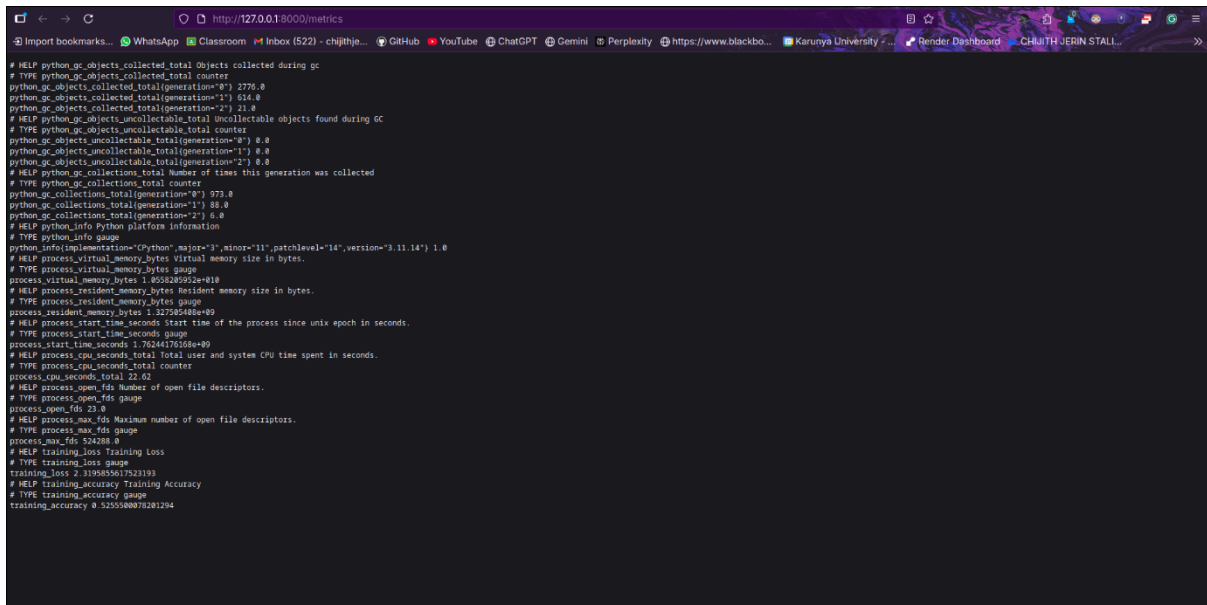
1 from prometheus_client import start_http_server, Gauge
2 import tensorflow as tf
3 from tensorflow.keras.datasets import mnist
4 import time
5
6 print("Prometheus Metrics Exporter Running at http://127.0.0.1:8000/metrics")
7
8 loss_metric = Gauge("training_loss", "Training Loss")
9 accuracy_metric = Gauge("training_accuracy", "Training Accuracy")
10
11 start_http_server(8000)
12
13 (x_train, y_train), _ = mnist.load_data()
14 x_train = x_train / 255.0
15
16 model = tf.keras.models.Sequential([
17     tf.keras.layers.Input(shape=(28, 28)),
18     tf.keras.layers.Flatten(),
19     tf.keras.layers.Dense(128, activation='relu'),
20     tf.keras.layers.Dense(10)
21 ])
22
23 # Training loop
24 for epoch in range(10):
25     # Training step
26     x_batch, y_batch = next(mnist.train_data_iterator(x_train, y_train, 128))
27     model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
28     model.fit(x_batch, y_batch, epochs=1, verbose=0)
29     # Update metrics
30     loss_metric.set(value=model.get_loss())
31     accuracy_metric.set(value=model.get_accuracy())
32     # Log metrics
33     print(f"Epoch {epoch+1}: Loss={loss_metric.get_value()}, Accuracy={accuracy_metric.get_value()}")
34
35 # End of training loop
36 
```

Terminal output:

```

(mlops) ~/Documents/College_Work/MLOps/EXP8/ml-monitoring
$ python prometheus_exporter.py
Prometheus Metrics Exporter Running at http://127.0.0.1:8000/metrics
Epoch 1: Loss=2.3196, Accuracy=0.5256
Epoch 2: Loss=2.3004, Accuracy=0.4737
Epoch 3: Loss=2.3026, Accuracy=0.4403
Epoch 4: Loss=2.3026, Accuracy=0.4403
Epoch 5: Loss=2.3026, Accuracy=0.4403

```



```

# HELP python_gc_objects_collected_total Objects collected during GC
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 2770.0
python_gc_objects_collected_total{generation="1"} 514.0
python_gc_objects_collected_total{generation="2"} 21.0
# HELP python_gc_objects_uncollectable_total Uncollectable objects found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 972.0
python_gc_collections_total{generation="1"} 88.0
python_gc_collections_total{generation="2"} 6.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="11",patchlevel="14",version="3.11.14"} 1.0
# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 1.055828952e+08
# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 1.327585488e+09
# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.76244176168e+09
# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 22.62
# HELP process_open_fds Number of open file descriptors.
# TYPE process_open_fds gauge
process_open_fds 23.0
# HELP process_max_fds Maximum number of open file descriptors.
# TYPE process_max_fds gauge
process_max_fds 524288.0
# HELP training_loss Training Loss
# TYPE training_loss gauge
training_loss 2.3195853617521193
# HELP training_accuracy Training Accuracy
# TYPE training_accuracy gauge
training_accuracy 0.5255580078201294

```

Result :

The machine learning model's training loss and accuracy were successfully logged and monitored using Python logging, TensorBoard, and Prometheus. This enabled real-time observation of model performance and ensured correct behavior during the training process.