A05 - Tracking a Model building experiment in MLFlow Report

Vinayak Gupta
EE20B152

Introduction

This report details the exploration of MLflow for experiment tracking and comparison in the context of neural network model building for MNIST digit classification. The provided Jupyter notebook (MNIST.ipynb) serves as the foundation, containing code for building classification models with various configuration settings. Our objective is to leverage MLflow to automate the tracking of model performance metrics and configuration parameters, enabling a comprehensive analysis of the impact of these settings on model performance.

This report outlines the implementation steps undertaken to achieve this objective. We will explore the functionalities of mlflow.autolog() and mlflow.log_metrics() for logging metrics like loss and accuracy. Similarly, we will investigate the use of mlflow.autolog() and mlflow.log_param() for logging model configuration parameters. We will then employ the with mlflow.start_run() construct to manage individual experiment runs for each configuration within the notebook.

Furthermore, we will establish a hierarchical experiment structure in MLflow. The overall MNIST experiment will encompass various sub-experiments, each representing a specific neural network configuration. This structure will facilitate the comparison of performance trends across different configurations. Finally, we will utilise the comparison functionalities within MLflow to analyse and visualise the performance metrics of the models. The report concludes with snapshots capturing these comparisons, providing insights into the impact of configuration choices on model performance.

Task #1 - Logging Metrics

Metrics (8)

Search metrics

Metric	Value
val_loss	0.14420431852340698
loss	0.08864030987024307
log_metrics_loss	0.14538928717374802
validation_accuracy	0.9567999839782715
validation_loss	0.14420431852340698
accuracy	0.9736166596412659
log_metrics_accuracy	0.9571900010108948
val_accuracy	0.9567999839782715

In the above table, the log_metrics_loss and log_metrics_accuracy have been logged using the mlflow.log_metrics() whereas the other metrics have been logged directly from mlflow.autolog(). Both have been logged together just for comparison purposes.

We observe that both log the same the loss and accuracy values proving no point for manual logging. However, manual logging is especially useful while logging something useful information which the mlflow might not log automatically.
For example - For a task the total loss function maybe

$$L_{total} = L_1 + L_2 + L_3$$

Now, MLFlow might just log L_{total} but it would be useful to log L_1, L_2 and L_3 as well which can be done using manual logging through mlflow.log_metrics()

Task #2 - Logging Parameters

Parameters (26)

Search parameters

Parameter	Value
optimizer_weight_decay	None
validation_freq	1
optimizer_beta_1	0.9
validation_steps	None
shuffle	True
optimizer_global_clipnorm	None
sample_weight	None
optimizer_epsilon	1e-07
initial_epoch	0
validation_batch_size	None
class_weight	None
optimizer_clipnorm	None
optimizer_ema_overwrite_frequency	None
validation_split	0.0
optimizer_ema_momentum	0.99
optimizer_gradient_accumulation_steps	None
steps_per_epoch	None
optimizer_use_ema	False
optimizer_amsgrad	False

Parameter	Value
optimizer	<keras.src.optimizers.adam.Adam object at 0x29a648a90>
config	basic
learning_rate	0.001
hidden_layer_size	20

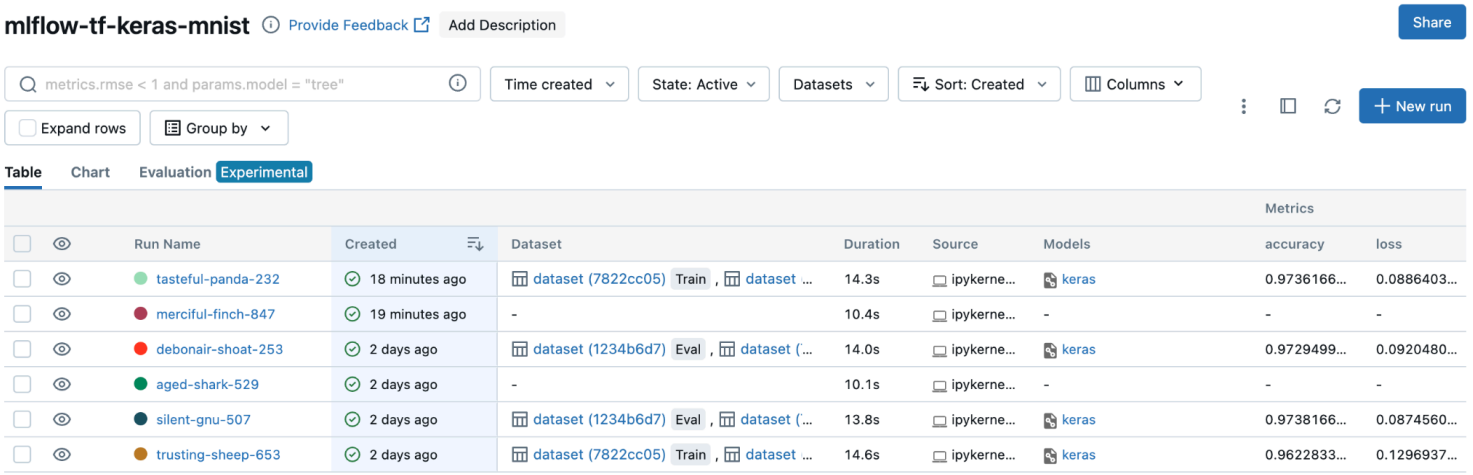
In the above 2 tables, the first table is the parameters logged using autolog and the second table are the parameters logged using mlflow.log_parameters.
We see that that autolog captures a variety of parameters useful to log. Whereas in contrast, the log_parameters only logs those specific parameters which are required to be logged by the user.

The autolog can be useful for those who are running basic experiments and don't want much customization during logging. Whereas the log_parameters() function can be useful for custom logging of some useful parameters which wouldn't have been logged during the autolog or it can be used to limit the logging only to few 4-5 parameters which are required for the experiment

Task #3 - Create start_run() for each configuration

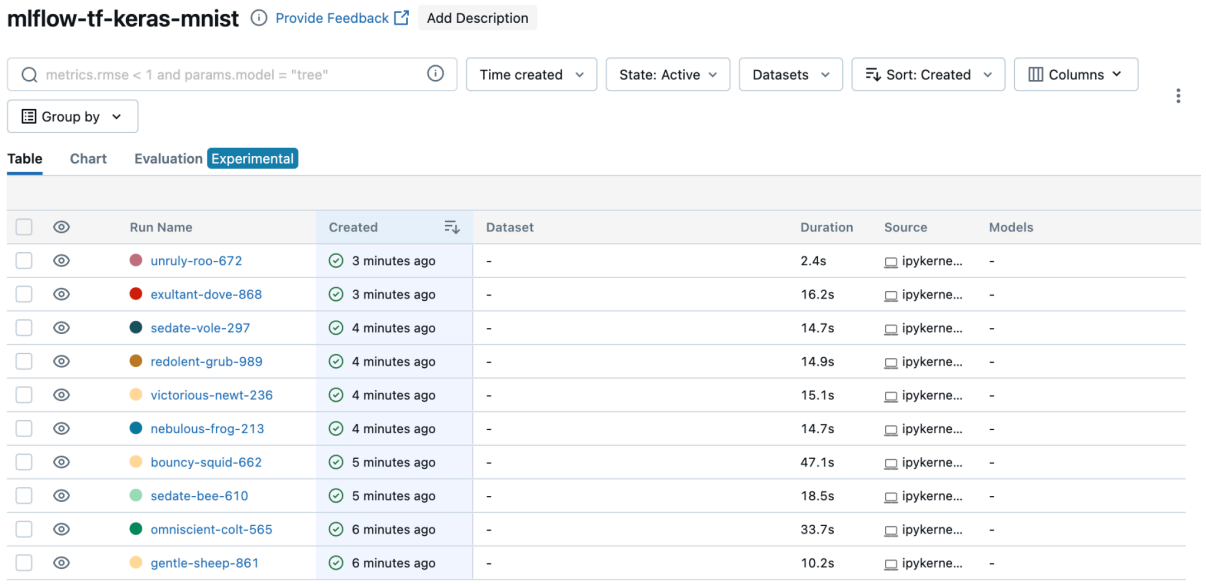
```python
## One Single Configuration Run
mlflow.set_experiment("mlflow-tf-keras-mnist")
track = "custom_logger"
if track == "autolog":
    mlflow.tensorflow.autolog()
    mlflow.autolog()
    history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))
else:
    mlflow.tensorflow.autolog()
    mlflow.autolog()
    with mlflow.start_run():
        history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))
        mlflow.log_metrics({"log_metrics_loss": sum(history.history["val_loss"])/len(history.history["val_loss"]), "log_m
        mlflow.log_param("learning_rate", learning_rate)
        mlflow.log_param("optimizer", optimizer)
        mlflow.log_param("config", "basic")
        mlflow.log_param("hidden_layer_size", 20)
```

The above is a code snippet usage of mlflow.start_run() for a particular configuration.



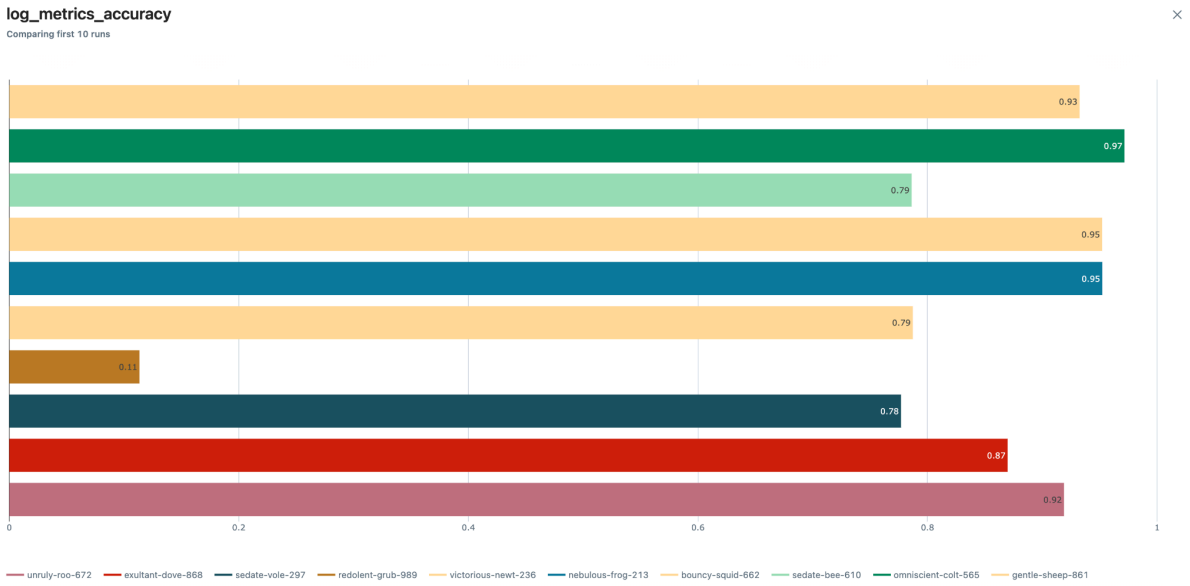
Each time the cell is run, a new entry in the MLFlow dashboard is created.

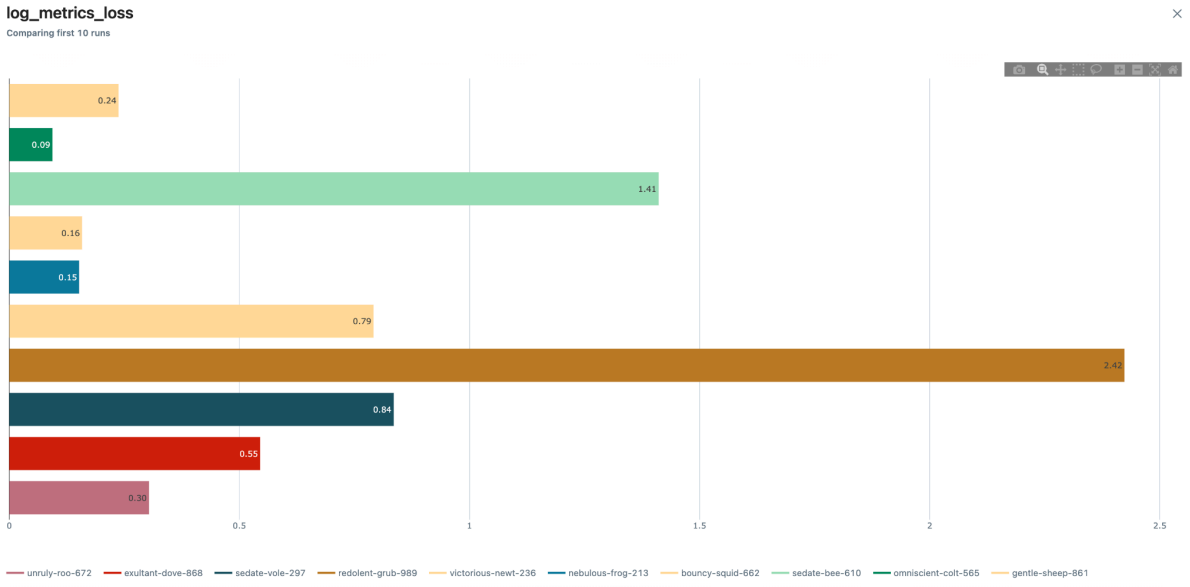
Task #5 - All 10 Sub-Experiments in One Experiment



All 10 sub-experiments under 1 main experiment named **mlflow-tf-keras-mnist**

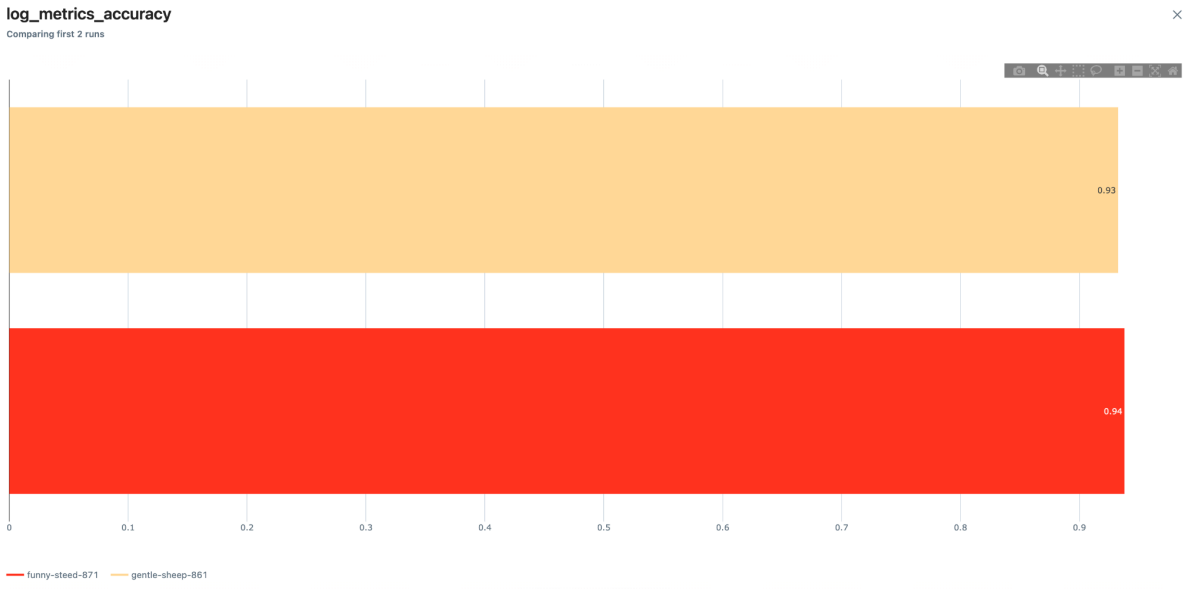
Task #6 - Compare within 1 Experiments



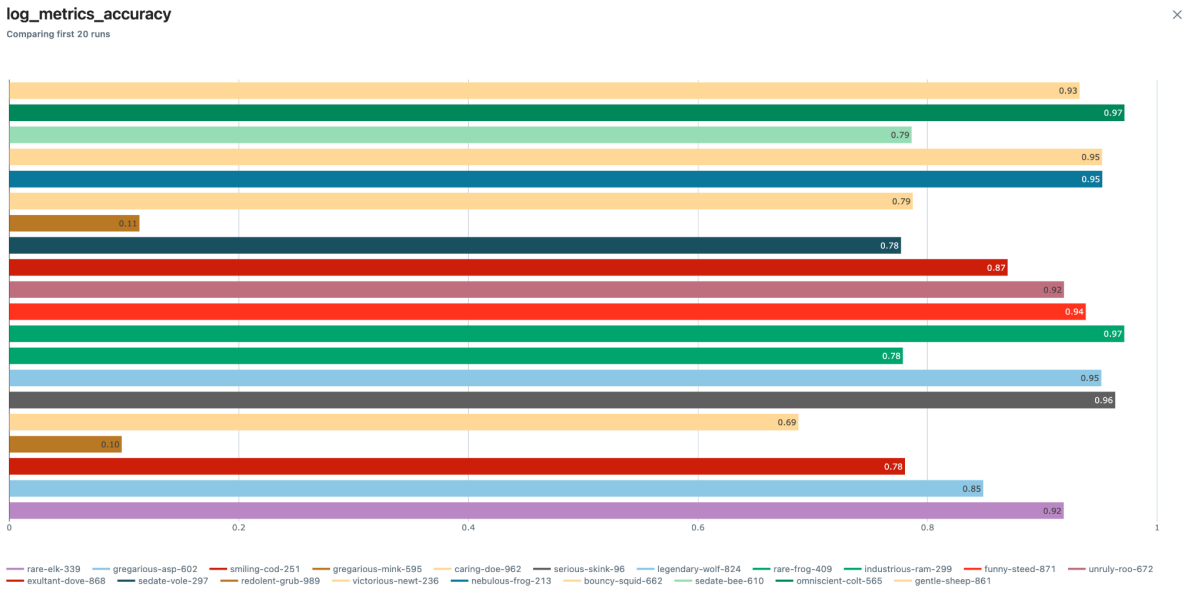


We observe that we obtain the best results in the second experiment which is the experiment with a larger neural network with Adam optimizer(lr = 1e-3)

Task #7 - Compare across several Experiments



These 2 above plots are comparisons of the same configuration compared against different experiments. We observe that the accuracy obtained is more or else similar. The first row corresponds to the result from 1st experiment and the 2nd row from 2nd experiment.



The above is a graph comparing models from 2 different experiments as asked in Task 7. We select all the configurations and compare them across experiments. We observe that similar configurations across experiments achieve similar accuracy.