MLops and ML_flow task

Rana Hassan Hosny (gamma-2b model)
Zaid Mahmoud (Phi3.5)
Zaid Wael (Llama3.1-8b model)

Table of content:

Gamma model :

ML_Ops part:

Models links:

you can find the tuned model in : RanaHossny213/gamma_tuned-2b

the Quantized model in : https://huggingface.co/RanaHossny213/model

the GGUF format in : https://huggingface.co/RanaHossny213/gamma-ft-gguf

used approach:

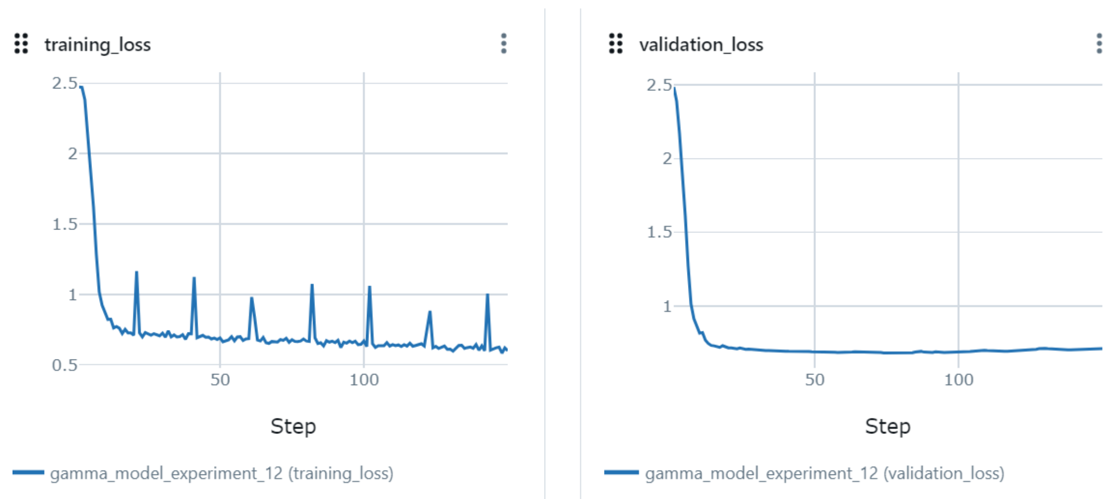
I have conducted over 13 experiments with various parameter configurations, focusing on tuning the learning rate and dropout values. Here's a summary of the setups I tested:

- **Learning rates**: $2e^{-4}$ and $2e^{-3}$
- **Dropout values**: 0, 0.001, and 0.1 (LoRA dropout)

Observations:

- Both the training loss and validation loss decreased together initially.
- However, they plateaued at a certain loss level without further improvement.

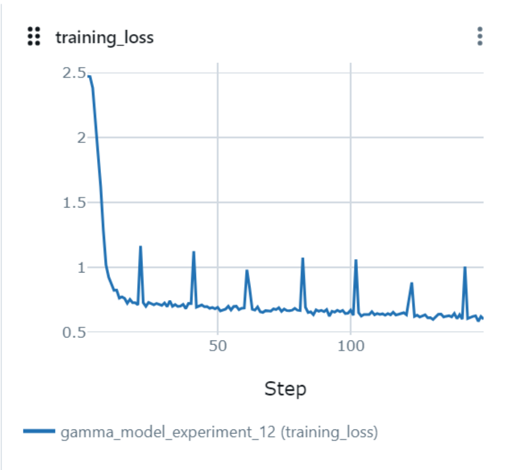
Below is a screenshot showing one of the results:

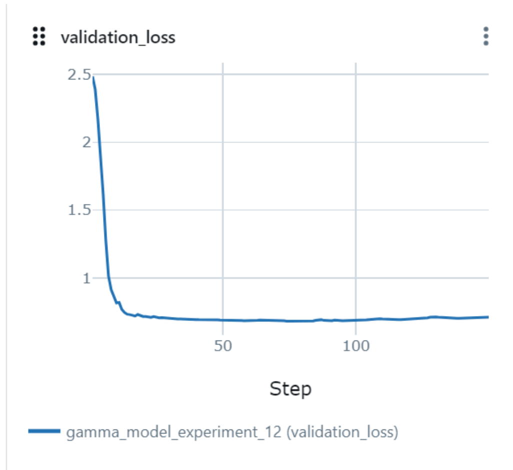


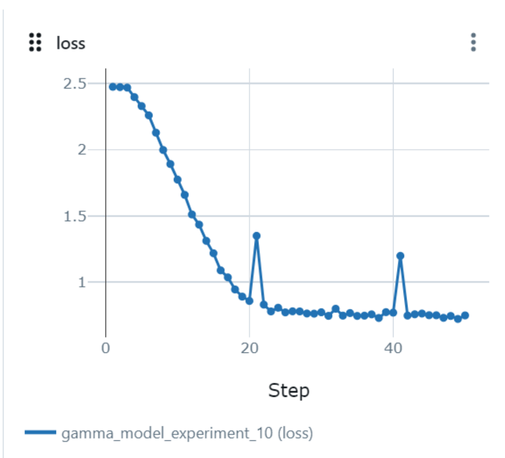
I also experimented with different epoch sizes (maximum steps) to analyze their impact:

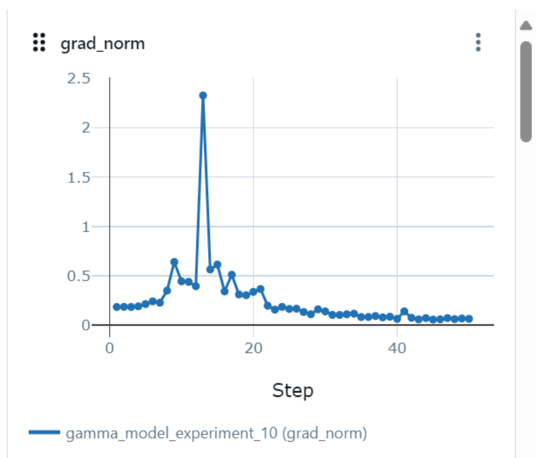
Max steps tried: 50, 150, 300

- when i did 150 is get a better result in comparison of 30 but still the stacking in certain loss for validation and training is exist:

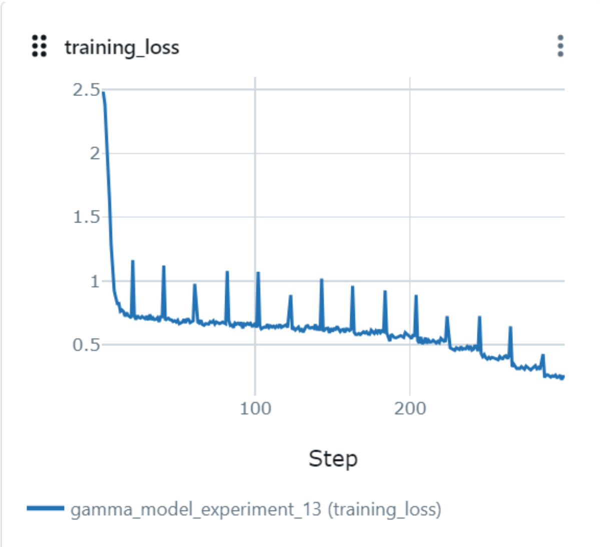


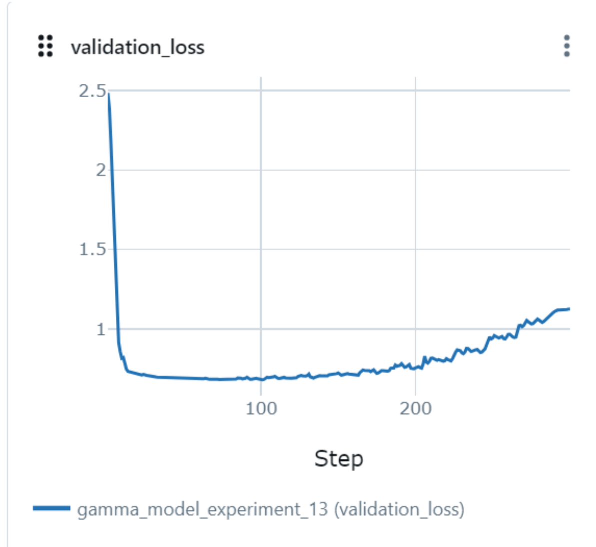






- Increasing the steps to 150 caused overfitting, as the model began to memorize the training data without further improvement in validation performance.





I also experimented with different prompts to evaluate their effect on performance. Here's the prompt I used:

Below is a description of a time series dataset. Your task is to identify the best-fitting machine learning algorithm based on the given search space.

Instructions:

- Return the name of the best algorithm from the search space.

- Provide the response in one word only, without any explanation or additional text.

Search Space Algorithms:

AdaboostRegressor, ElasticNetRegressor, ExtraTreesRegressor, GaussianProcessRegressor, LassoRegressor, LightgbmRegressor, RandomForestRegressor, SVR, XGBoostRegressor.

DESCRIPTION:

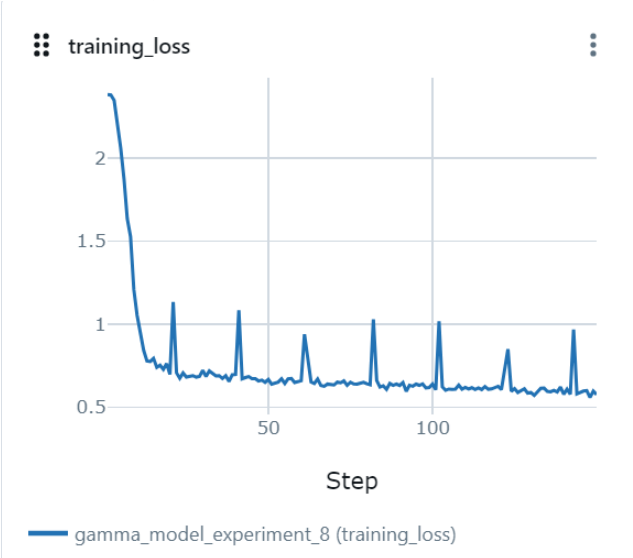
{}

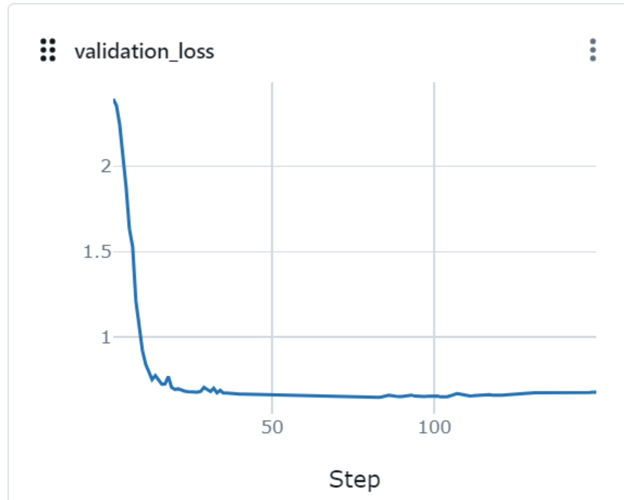
RESPONSE:

{}

Observations:

- Testing with this prompt did not result in any enhancement or noticeable improvement in performance.





Result:

The final model accuracy is not optimal. I believe this is due to the small size of the training data. With such limited data, traditional machine learning algorithms might perform better compared to more complex models.

Ml_flow part:

Setting Up MLflow Server with Ngrok for Colab Access:

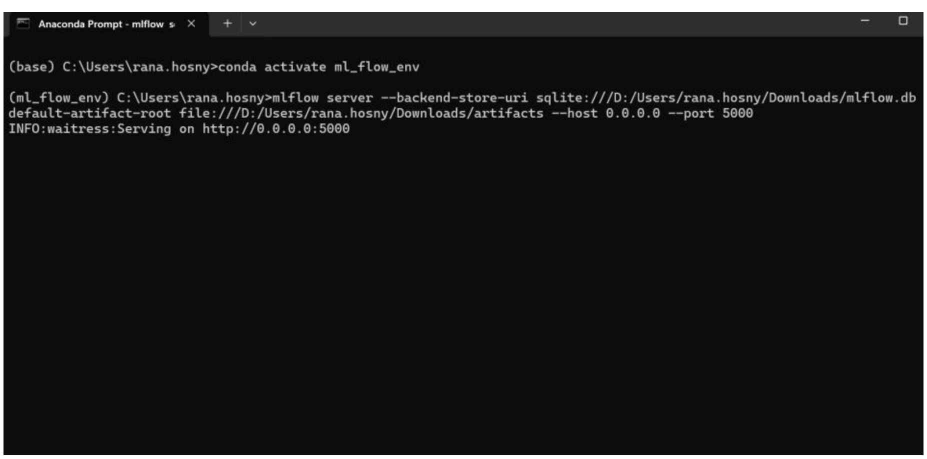
This document outlines the process undertaken to set up and configure the MLflow server for external access using Ngrok.

Conda Environment Setup:

Activated the Conda environment ml_flow_env using:
conda activate ml_flow_env

- Launched the MLflow server with the following configuration:
    - Backend Store URI: A SQLite database located at D:/Users/rana.hosny/Downloads/mlflow.db.
    - Artifact Root: A local directory D:/Users/rana.hosny/Downloads/artifacts for storing artifacts.
    - The server was hosted on all interfaces (0.0.0.0) and set to run on port 5000.

mlflow server --backend-store-uri sqlite:///D:/Users/rana.hosny/Downloads/mlflow.db --default-artifact-root file:///D:/Users/rana.hosny/Downloads/artifacts --host 0.0.0.0 --port 5000



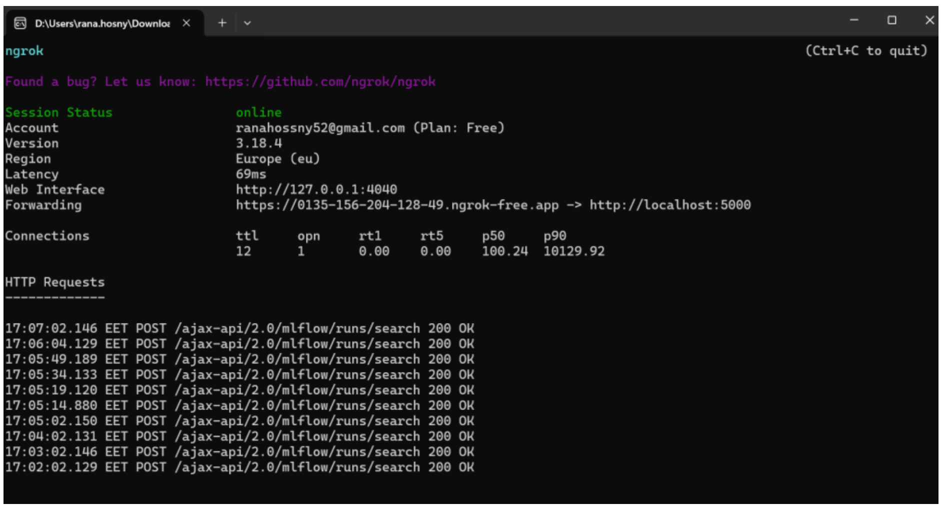
Ngrok Configuration:

Configured Ngrok to expose the local MLflow server to the internet for remote access:

Authenticated Ngrok with the provided authtoken:
ngrok authtoken <your-auth-token>

- Created an HTTP tunnel for port 5000:

ngrok http 5000



Logged Parameters and Metrics

The following describes the parameters and metrics logged during the training process using MLflow, along with their sources and significance.

1- Training Arguments

Logged directly from the TrainingArguments instance:

- per_device_train_batch_size: Batch size per device.
- gradient_accumulation_steps: Number of gradient accumulation steps.
- warmup_steps: Number of warmup steps for the learning rate scheduler.
- max_steps: Maximum number of training steps.
- learning_rate: Learning rate for optimization.
- fp16: Whether to use FP16 precision (if BF16 is unsupported).

- bf16: Whether to use BF16 precision (if supported).
- logging_steps: Number of steps between logging updates.
- optim: Optimization method used (adamw_8bit).
- weight_decay: Weight decay applied to model parameters.
- lr_scheduler_type: Type of learning rate scheduler used.
- seed: Random seed for reproducibility.
- output_dir: Directory for saving outputs.
- eval_strategy: Strategy for evaluation during training.
- save_strategy: Strategy for saving checkpoints during training.

2- Tokenizer Configuration

- Tokenizer: The tokenizer class name (e.g., PreTrainedTokenizerFast).
- Padding Side: Indicates whether padding is applied on the left or right.
- EOS Token: Specifies if an End-Of-Sequence token is added.
- Pad Token: The token used for padding sequences (e.g., [PAD]).

3- LoRA Configuration:

- r: Rank of the low-rank adaptation.
- lora_alpha: LoRA alpha parameter for scaling.
- lora_dropout: Dropout probability applied to LoRA layers.
- use_gradient_checkpointing: Whether gradient checkpointing is enabled.
- random_state: Random seed for LoRA-related stochastic operations.
- use_rslora: Whether the RS-LoRA technique is used.
- loftq_config: Configuration for quantization (if applicable).

4- Model Metadata

- model_name: The name of the model being trained (tagged with gamma-2b).

5- Metrics Logged

The following metrics are tracked during the training process via the MLFlowLoggingCallback:

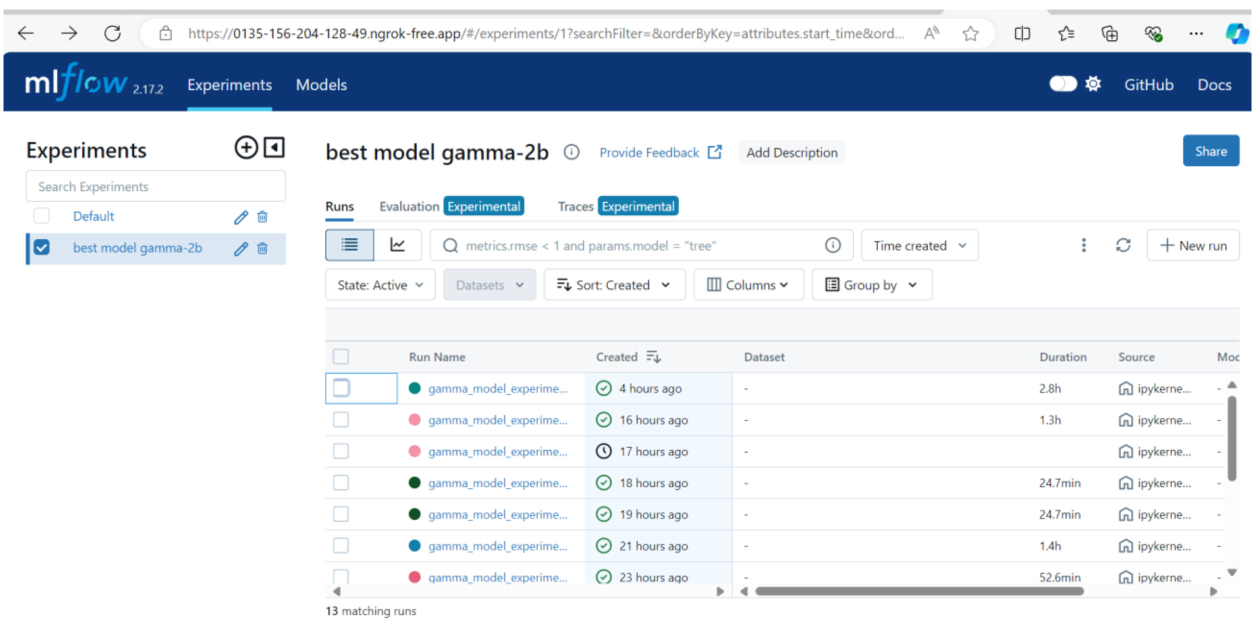
Training Metrics

- training_loss: Logged at every training step, tracking the loss from the training loop.

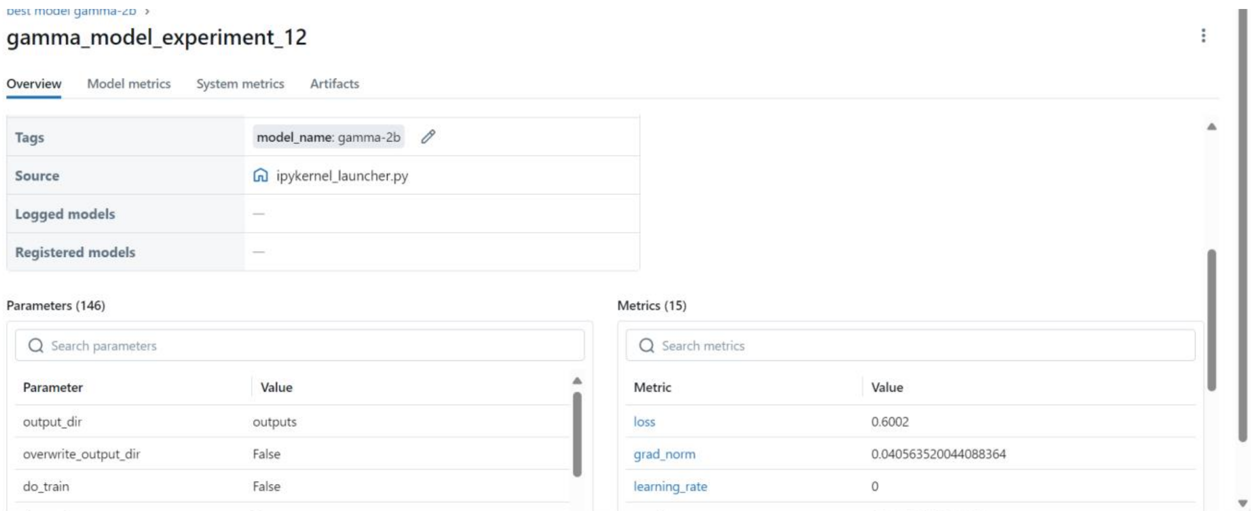
Validation Metrics

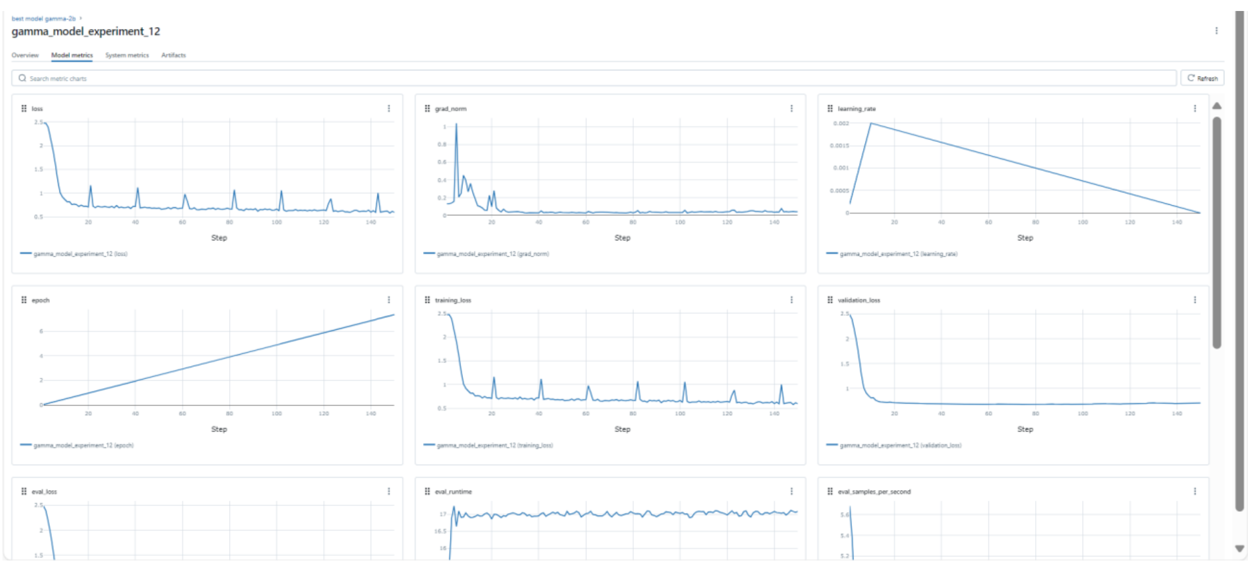
- validation_loss: Loss calculated on the validation dataset during evaluation.

Ml_flow in action:



Screenshot from one of the experiments :





Llama3.1-8B model :

Llama Model:

Model Name: Llama 8B
Purpose: Fine-tune the Llama model to predict the best-fitting machine learning algorithm based on dataset descriptions.

Ml_flow part:

MLflow for Experiment Tracking:

- MLflow was configured with a tracking server exposed using Ngrok for remote access in Colab.
- Metrics, parameters, and losses were logged during training for analysis and reproducibility.

Models links:

you can find the tuned model in: https://huggingface.co/ZiadWael/unsloth-Llama3.1-tuned

The quantized model in: https://huggingface.co/ZiadWael/unsloth-Llama-tuned

the GGUF format in: https://huggingface.co/ZiadWael/unsloth-llama-ft-gguf

ML_Ops part:

Used Approach:

Experiments Conducted:

1. **Parameter Variations**:
    - Learning Rates: 2e-4, 2e-3.
    - LoRA Dropout Values: 0, 0.001, 0.1.
    - Batch Sizes: 2, 4.

2. **Key Observations**:

   - Training and validation losses decreased together initially but plateaued at certain levels.
   - Increasing the number of steps leading to overfitting without improving validation performance.

3. **Prompts Used**:

Example Prompt:
Below is a description of a time series dataset. Your task is to identify the best-fitting machine learning algorithm based on the given search space.

Search Space Algorithms:

AdaboostRegressor, ElasticNetRegressor, ExtraTreesRegressor, GaussianProcessRegressor, LassoRegressor, LightgbmRegressor, RandomForestRegressor, SVR, XGBoostRegressor.

DESCRIPTION:

{Dataset Description}

RESPONSE:

{Algorithm Name}

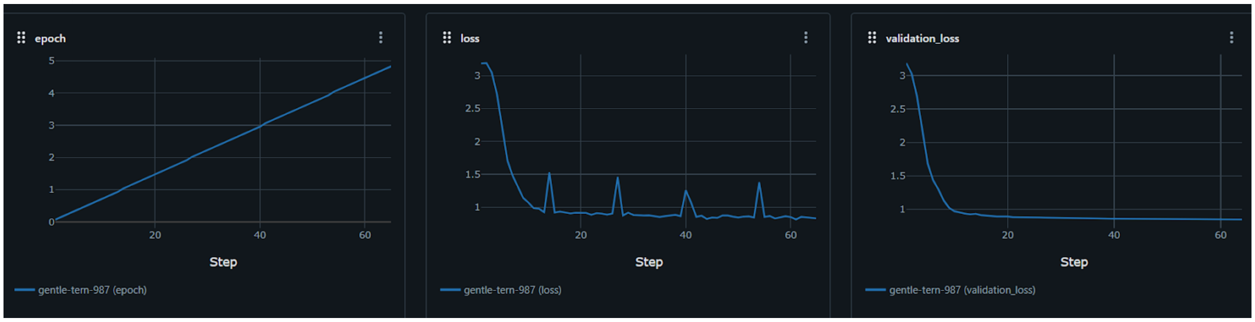
4. **Findings**:

   - Refining the prompt and adding detailed dataset descriptions slightly improved model performance.
   - Validation performance plateaued after ~150 steps, with limited gains from additional tuning.

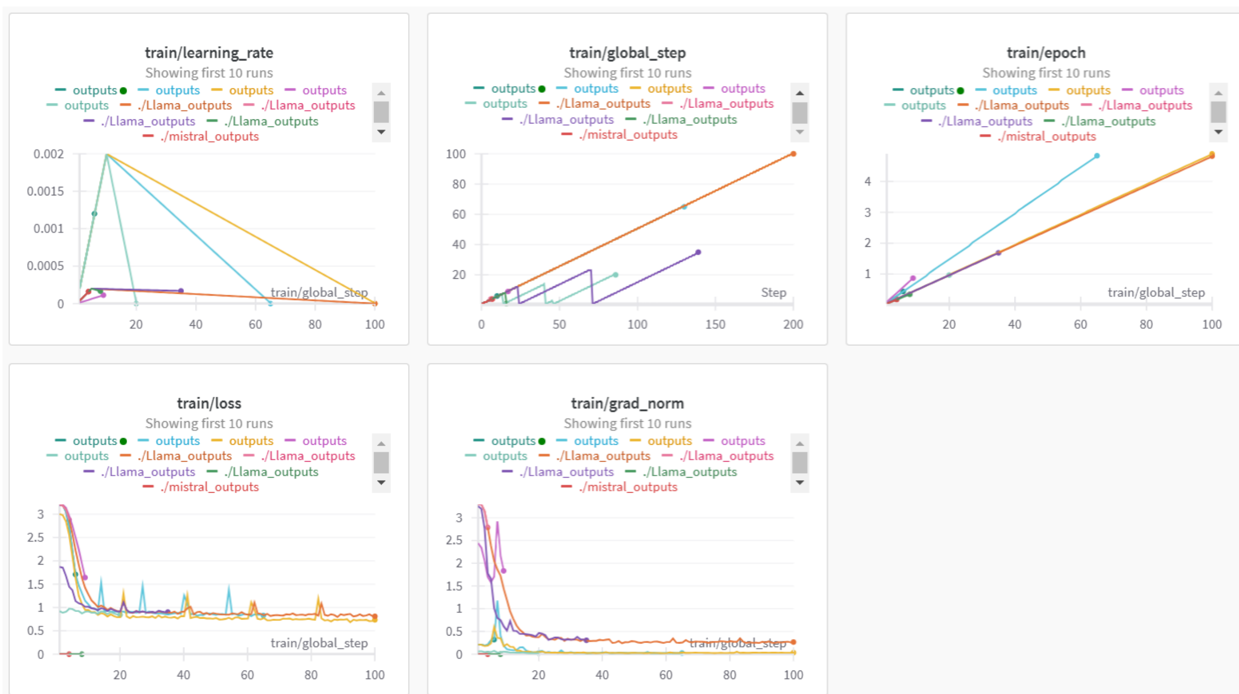
Results:

- **Training Loss**: 0.82
- **Validation Loss**: 0.848

- **Accuracy**: ~15%



epoch
loss
validation_loss
Step
gentle-tern-987 (epoch)
gentle-tern-987 (loss)
gentle-tern-987 (validation_loss)



train/learning_rate
Showing first 10 runs
outputs
./Llama_outputs
./mistral_outputs
train/global_step
Step
train/epoch
train/loss
train/grad_norm

Phi3.5: 3.8B

Ml_ops part

the results were conducted from using various combinations of hyperparameters and different warm up steps (5,10,50) and batch sizes(4,6,8)

The links for the runs:

https://drive.google.com/drive/folders/1eErWKEOb5l1TruINlYOyFNecMNjX5dFF?usp=sharing

https://drive.google.com/drive/folders/1nFWNa41vTQZaNHiHoO8fD_zf1GdlHi8l?usp=sharing

the last run:

https://drive.google.com/drive/folders/1y6fQFfagPWIEWJ5QnT9m51jwocaa7vEg?usp=sharing

The combinations of hyper parameters:

combinations = [

{"num_epochs": 1, "learning_rate": 2e-5, "batch_size": 4, "gradient_accumulation_steps": 8},

{"num_epochs": 2, "learning_rate": 5e-5, "batch_size": 4, "gradient_accumulation_steps": 4},

{"num_epochs": 5, "learning_rate": 1e-4, "batch_size": 8, "gradient_accumulation_steps": 2},

{"num_epochs": 3, "learning_rate": 3e-5, "batch_size": 6, "gradient_accumulation_steps": 6},

{"num_epochs": 6, "learning_rate": 5e-5, "batch_size": 8, "gradient_accumulation_steps": 6},]

Observations:

The train and validation loss decreased as the epochs number increased till they reached 0.64 for train and 0.61 for validation after 4 epochs using the last combination:

{"num_epochs": 6, "learning_rate": 5e-5, "batch_size": 8, "gradient_accumulation_steps": 6}

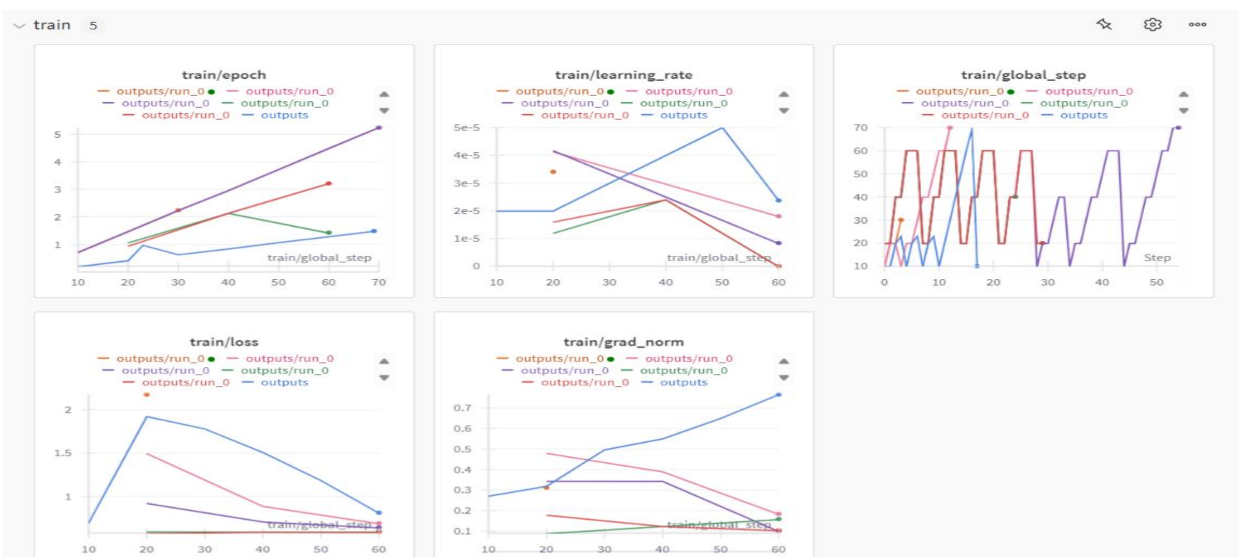
Also due to the small number of the train set the accuracy wasn't the

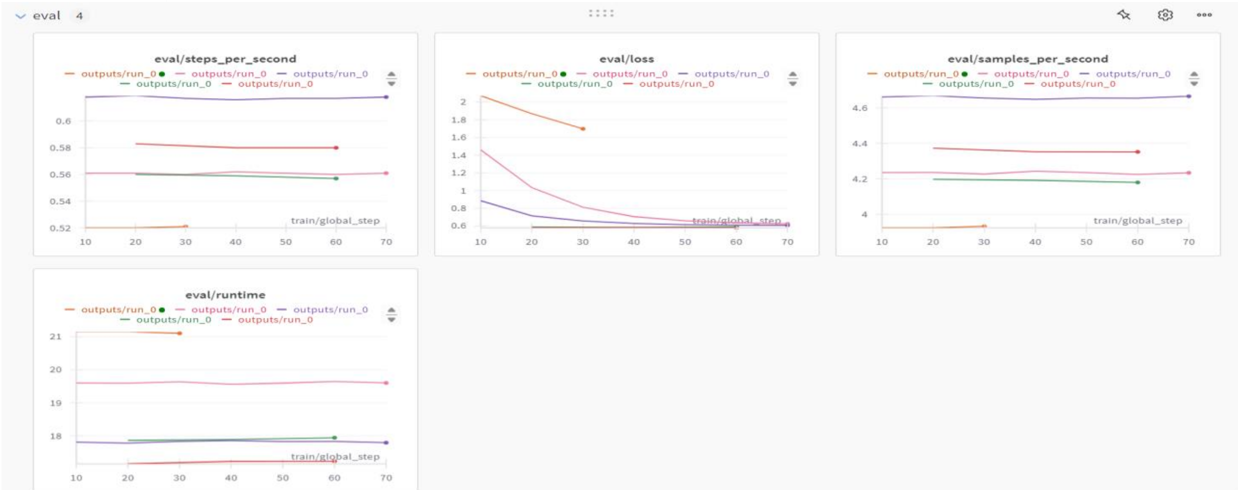
```
max_steps is given, it will override any value given in num_train_epochs
==((====))==  Unsloth - 2x faster free finetuning | Num GPUs = 1
   \\   /|    Num examples = 662 | Num Epochs = 6
O^O/ \_/ \    Batch size per device = 8 | Gradient Accumulation steps = 6
\        /    Total batch size = 48 | Total steps = 70
 "-____-"     Number of trainable parameters = 29,884,416
```

[70/70 39:44, Epoch 5/6]

Step	Training Loss	Validation Loss
10	No log	0.884496
20	0.924400	0.713580
30	0.924400	0.655516
40	0.711600	0.627706
50	0.711600	0.613718
60	0.643600	0.608566
70	0.643600	0.606980

Below is some snippets from Wandb showing the train loss and validation loss for different runs:





Models links:

the Quantized models:

Ziad177/model_checkpoint_0

Ziad177/model_checkpoint_1

Ziad177/model_checkpoint_2

Ziad177/model_checkpoint_3

the GGUF checkpoints:

Ziad177/model_checkpoint_0_gguf

Ziad177/model_checkpoint_1_gguf

Ziad177/model_checkpoint_2_gguf

Ziad177/model_checkpoint_3_gguf

ML_flow part:

You can see the results after running the model on mlflow using ngrok

```
!ngrok authtoken 2othLgNKEnV2xjbWRJ1CMaERYh0_4vEPnCMuk2Ncv72626hqt

```

```
Authtoken saved to configuration file: /root/.config/ngrok/ngrok.yml
```

```python
from pyngrok import ngrok
import mlflow

# Expose the MLflow UI
public_url = ngrok.connect(5000)
print(f"MLflow UI is available at {public_url}")

# Run MLflow UI in the background
!mlflow ui --port 5000


```