Managing the Machine Learning Lifecycle with MLflow: A Tech Preview Using PhytoOracle (and chest x-ray)

Artin Majdi  UArizona, ECE Dept., Data Science Institute (Data7)
Ariyan Zarei  UArizona, CS Dept., PhytoOracle

April 16, 2021
Personal experience

- Organization/Tracking
- Platform dependence
- Accessing old simulations
- Deployment
<table>
<thead>
<tr>
<th>What We’ll Cover Today</th>
</tr>
</thead>
<tbody>
<tr>
<td>Challenges in ML development</td>
</tr>
<tr>
<td>How MLflow can help</td>
</tr>
<tr>
<td>What is MLflow?</td>
</tr>
<tr>
<td>Tech Preview with Case Studies</td>
</tr>
</tbody>
</table>
Typical ML Project Requirements (MLOps)

**Data**
- Ethical fairness
- Pre and post processing
- Accessibility

**Development**
- Design
- Agnosticism & Reproducibility
- Versioning & Tracking
  (experiment, code, dependencies)

**Deployment**
- Continuous Monitoring
- Multiple access mode
- Visualization
## Challenges in Development

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data management</td>
<td>Pre-processing, Accessibility</td>
</tr>
<tr>
<td>Model optimization</td>
<td>Architecture design, Hyper parameter tuning</td>
</tr>
<tr>
<td>Experiment/Analysis</td>
<td>Convergence, Hyper parameter tuning</td>
</tr>
<tr>
<td>Collaborative development</td>
<td>Data/model accessibility, Permission assignments</td>
</tr>
<tr>
<td>Platform dependence</td>
<td>OS, Versioning conflicts</td>
</tr>
</tbody>
</table>
How MLflow Can Help with These Challenges

- Data management
- Model optimization
- Experiment/Analysis
- Collaborative development
- Platform independence
- Deployment

- Easy to integrate into existing projects
- Works with most ML libraries and languages
- Tracking
- Collaboration-friendly
- Platform-agnostic
- Enables reproducibility
- Modular design
- Model registry and versioning
MLflow Components

- **Tracking**: Record experiments config, results and sources code
- **Models**: Standardized format for saving models
- **Projects**: Reproducible packaging
- **Model registry**: Centralized model management review & sharing
- **Plugins**: Framework agnostic tool for ML
### ML Case Studies

<table>
<thead>
<tr>
<th>Pathology classification</th>
<th>Goal</th>
<th>Segmentation of plant disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIH and CheXpert</td>
<td>Data</td>
<td>Manually Collected/Labeled</td>
</tr>
<tr>
<td>X-ray</td>
<td>Data type</td>
<td>RGB Images</td>
</tr>
<tr>
<td>Python, Keras (TensorFlow)</td>
<td>Language</td>
<td>Python, Keras (TensorFlow)</td>
</tr>
<tr>
<td>Pandas, scikit-image,...</td>
<td>Dependencies</td>
<td>OpenCV, PIL, ...</td>
</tr>
<tr>
<td>Local (CPU) &amp; HPC (GPU)</td>
<td>Processing</td>
<td>HPC (GPU)</td>
</tr>
</tbody>
</table>

**Chest X-ray**: PhytoOracle
MLflow User Interface for Chest Classification
Tracking Key Features

- **Parameters**
- **Metrics**
- **Tags and notes**
- **Artifacts**
- **Source code**
How to Log Parameters/Metrics with MLflow

Automatic tracking of endless text/csv/pickle output files!

Logging parameters/metrics/artifacts

```
# %

"""" Saving MLflow parameters & metrics """
mlflow.log_param("epochs", epochs)
mlflow.log_param("batch_size", batch_size)
mlflow.log_metric("accuracy", test_acc)
mlflow.log_metric("test_loss", test_loss)

mlflow.keras.log_model(model, "my_model_log")
mlflow.keras.save_model(model, 'my_model')

with open('predictions.txt', 'w') as f:
    f.write("predicted_classes")

mlflow.log_artifact('predictions.txt')
```

Using mlflow built-in automatic logging

```
"""" Logging the parameters automatically """
mlflow.keras.autolog()
```
MLflow Project Structure

Environment

Entry point

```
name: My Project

conda_env: my_env.yaml

docker_env:
    image: mlflow-docker-example

entry_points:

    main:
        parameters:
            data_file: path
            regularization: {type: float, default: 0.1}

        command: "python train.py -r {regularization} {data_file}"

    validate:
        parameters:
            data_file: path

        command: "python validate.py {data_file}"```
Model Registry

Audience

Developer
Downstream User
Reviewer/Evaluator

mlflow

Registered Models

<table>
<thead>
<tr>
<th>Name</th>
<th>Latest Version</th>
<th>Staging</th>
<th>Production</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>classifier</td>
<td>Version 1</td>
<td>-</td>
<td>-</td>
<td>2021-02-14 21:58:05</td>
</tr>
<tr>
<td>model_A</td>
<td>Version 5</td>
<td>-</td>
<td>Version 4</td>
<td>2021-02-01 18:54:33</td>
</tr>
</tbody>
</table>
**MLflow on CyVerse**

### Tracking Server
- CyVerse Cloud
- Native Service

```python
def set_tracking_uri(server):
    mlflow.set_tracking_uri(server)
```

### Artifact Storage
- CyVerse Data Store

```python
def create_experiment(name, artifact_location):
    mlflow.create_experiment(name=name, artifact_location=artifact_location)
```

```python
def set_experiment(experiment_name):
    mlflow.set_experiment(experiment_name)
```

### Additional Tooling
- DE-VICE (Flask, Python-Dask, R-shiny...)

- Amazon S3 and S3-compatible storage
- Azure Blob Storage
- Google Cloud Storage
- MySQL
- SQLite
- PostgreSQL
- FTP server
- SFTP Server
- NFS
- HDFS
PhytoOracle

- Joint project
  - Danforth Center
  - School of Plant Science
  - Data Science Institute
  - CyVerse
- Funded by DOE
- Analyze plants in drought stress conditions
  - Genomics $\leftrightarrow$ Phenomics
  - Genomics $\leftrightarrow$ Disease
  - Disease detection
  - Predictive plant modeling
- 5+ cameras and sensors
- Previous CyVerse Webinar
  - [https://cyverse.org/webinar-PhytoOracle](https://cyverse.org/webinar-PhytoOracle)
Charcoal Dry Rot

- A fungal disease in water-stressed sorghum plants
  - Caused by *Macrophomina phaseolina*
  - Dead tissue
  - Light gray - yellow
  - Starts from tips of the leaves

- Ultimate Goal
  - Disease detection
  - Locate affected regions using drones
  - Apply fungicides

- Train Neural Networks
  - Semantic segmentation

- Labeled 1400 Images
  - [http://www.labelbox.com](http://www.labelbox.com)
Why Use MLflow?

- Cottage Industry → Collaborative/Distributed project
- MLflow helps with
  - Collaboration
  - Keeping track of experiments
  - Comparing the results
  - Designing new experiments
  - Storing the models
  - Deploying the models
  - Reproducibility and reusability
Charcoal_DryRot_Segmentation 

Date: 2021-04-06 10:59:41
User: arlyanzarei

Source: U-Net_model_mlflow.py
Duration: 40.2min

Git Commit: 4f8ec4343112706d82893231c746b5f02fedeb09
Status: FINISHED

- **Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>None</td>
</tr>
<tr>
<td>batch_size</td>
<td>8</td>
</tr>
<tr>
<td>class_weight</td>
<td>None</td>
</tr>
<tr>
<td>epochs</td>
<td>20</td>
</tr>
<tr>
<td>general_batch_size</td>
<td>8</td>
</tr>
<tr>
<td>general_epochs</td>
<td>20</td>
</tr>
<tr>
<td>general_loss</td>
<td>weighted_dice_coef</td>
</tr>
<tr>
<td>general_optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>general_patience</td>
<td>30</td>
</tr>
<tr>
<td>global_momentum</td>
<td>0.5</td>
</tr>
<tr>
<td>initial_epoch</td>
<td>0</td>
</tr>
<tr>
<td>learning_rate</td>
<td>0.0005</td>
</tr>
<tr>
<td>max_pooling_size</td>
<td>2</td>
</tr>
<tr>
<td>max_queue_size</td>
<td>10</td>
</tr>
<tr>
<td>min_delta</td>
<td>0</td>
</tr>
</tbody>
</table>
The image shows a web interface with a graph displaying model accuracy over epochs. The x-axis represents the epoch, ranging from 0 to 18, and the y-axis represents accuracy, ranging from 0.6 to 0.8. Two lines are plotted: one for 'train' and one for 'test'. The accuracy for both 'train' and 'test' increases as the number of epochs increases. The graph suggests that the model's accuracy improves with training, achieving a higher accuracy rate after 18 epochs.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run ID</td>
<td>a822dbb50dcf47cb84518e3a1eb9ce18</td>
<td>61cb6bb0e4e67583cfa2f2e1c1144a</td>
</tr>
<tr>
<td>Run Name:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start Time:</td>
<td>2021-04-06 10:50:41</td>
<td>2021-04-06 10:50:40</td>
</tr>
<tr>
<td>baseline</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>batch_size</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>class_weight</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>epochs</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>general_batch_size</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>general_epochs</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>general_loss</td>
<td>weighted_dice_coef</td>
<td>weighted_dice_coef</td>
</tr>
<tr>
<td>general_optimizer</td>
<td>Adam</td>
<td>Adam</td>
</tr>
<tr>
<td>general_patience</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>global_momentum</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>initial_epoch</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>learning_rate</td>
<td>0.0005</td>
<td>5e-05</td>
</tr>
<tr>
<td>max_pooling_size</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>max_queue_size</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>min_delta</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>monitor</td>
<td>val_mean Io_u</td>
<td>val_mean Io_u</td>
</tr>
<tr>
<td>opt_amsgd</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>opt_beta 1</td>
<td>0.9</td>
<td>0.9</td>
</tr>
</tbody>
</table>
Lessons Learned Using MLflow

01. Setting up the server and database
02. Comparing artifacts not possible
03. Autolog not working all the time

01. Prerequisites: Familiarity with Conda/Docker, SQL flavors, ssh-tunneling, ...
02. Dependency version mismatch
03. Doesn’t support singularity
Getting Started with MLflow on CyVerse

Via External Collaborative Partnerships (ECP), researchers are paired with an expert to address their project’s specific computational needs and more (postgresql . . )

Request an ECP: https://cyverse.org/ecp
Helpful Resources

Links

- MLflow
  [https://mlflow.org](https://mlflow.org)
- CyVerse
  [www.cyverse.org](http://www.cyverse.org)
- PhytoOracle Docs
  [https://tinyurl.com/phytooracle-rtd](https://tinyurl.com/phytooracle-rtd)
- CyVerse Webinar on PhytoOracle
  [https://cyverse.org/webinar-PhytoOracle](https://cyverse.org/webinar-PhytoOracle)
- MLflow use case with MNIST classification
  [https://github.com/artinmajdi/mlflow_workflow](https://github.com/artinmajdi/mlflow_workflow)
- MLflow use case with Chest X-Rays
  [https://github.com/artinmajdi/chest-x-ray-classification](https://github.com/artinmajdi/chest-x-ray-classification)