Edit article
View stats
View post

https://towardsdatascience.com/machine-learning-adventures-with-mlflow-64127713b0a1

Building a Workflow for Object Detection Model Training and Deployment using TensorFlow Object Detection API and MLflow

Wut Hmone Hnin Hlaing
M.Sc Data Analytics | Founder | Ex. Lead Back End Engineer

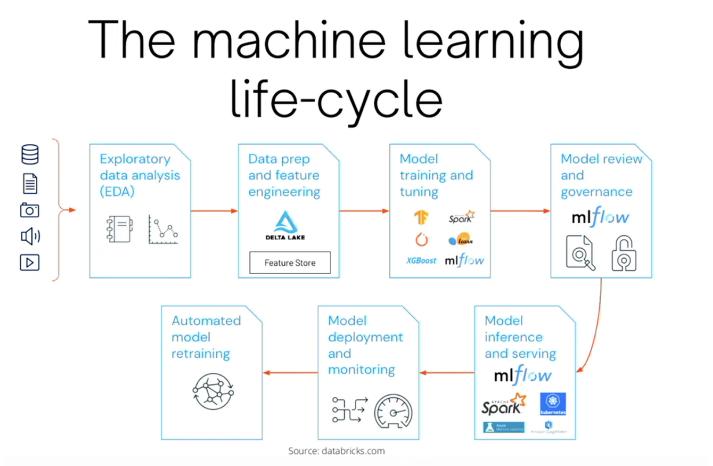
3 articles

April 9, 2023

Object detection is a computer vision task that involves identifying and locating objects of interest within an image or video stream. This task has many practical applications, such as surveillance, robotics, and self-driving cars. In recent years, deep learning has emerged as a powerful tool for solving object detection problems, with models like Faster R-CNN, SSD, and YOLO achieving state-of-the-art performance on benchmark datasets.

In this article, we'll explore a workflow for training and deploying an object detection model using the TensorFlow Object Detection API. This API provides a range of pre-trained models and tools for customizing, training, and exporting object detection models.

Workflow Overview


The machine learning life-cycle
Exploratory data analysis (EDA)
Data prep and feature engineering
DELTA LAKE
Feature Store
Model training and tuning
Spark
XGBoost
mlflow
Model review and governance
mlflow
Automated model retraining
Model deployment and monitoring
Model inference and serving
mlflow
Spark
Source: databricks.com

Here's an overview of the workflow we'll be following:

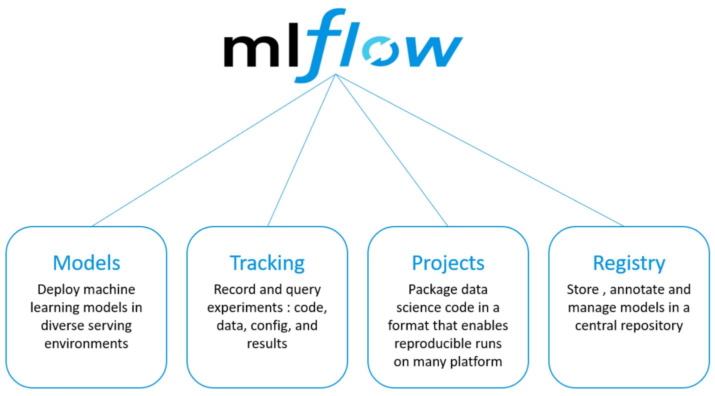
1. Data collection and preparation/pre-processing
2. Model selection and customization
3. Training
4. Evaluation
5. Fine-tuning
6. Exporting
7. Deployment
8. Testing and optimization

Let's dive into each of these steps in more detail.

1. Data Collection and Preparation/Pre-processing

The first step in any object detection project is to collect and prepare the data. This involves gathering images and annotations, labeling the objects of interest, and creating a training and validation set using tools such as **LabelImg.** The annotation format should be compatible with the TensorFlow Object Detection API, which supports popular formats such as COCO, PASCAL VOC, and TFRecord.

To streamline the data collection and preparation process, we can use the MLflow framework. MLflow is an open-source platform for managing the end-to-end machine learning lifecycle. It provides tools for tracking experiments, packaging code into reproducible runs, and sharing and deploying models.


mlflow
Models
Deploy machine learning models in diverse serving environments
Tracking
Record and query experiments : code, data, config, and results
Projects
Package data science code in a format that enables reproducible runs on many platform
Registry
Store , annotate and manage models in a central repository

Source: MLflow.org

Using MLflow, we can track information such as the number of images, the annotation format, and the training/validation split. This information will be useful for reproducibility and for keeping track of the data used in the training process.

2. Model Selection and Customization

The TensorFlow Object Detection API provides a range of pre-trained models to choose from, such as Faster R-CNN, SSD, and YOLO. These models are trained on large datasets and have been shown to achieve state-of-the-art performance on object detection tasks.

However, these pre-trained models may not be suitable for all use cases. For example, if we're interested in detecting objects in a specific domain or with specific attributes, we may need to customize the pre-trained model.

To customize the model, we can modify its architecture, hyperparameters, and other aspects of the model. For instance, we may need to adjust the size of the model, change the number of layers, or add new object categories. Using MLflow, we can track the model selection and customization process, including the model architecture, hyperparameters, and any modifications made to the pre-trained model.

3. Training

Once the data is prepared and the model is customized, it's time to train the model. The TensorFlow Object Detection API provides tools for training object detection models, including data loading, training, and evaluation scripts.

Training an object detection model can be computationally intensive and time-consuming. To speed up the training process, we can use a GPU. We can also start by training the model on a small subset of the data to verify that it's working as expected. Then gradually increase the size of the training set and the number of iterations until we achieve the desired accuracy.

Using MLflow, we can track the training process, including the training/validation loss, learning rate, and number of iterations. This information will be useful for analyzing the training process and for identifying any issues that may arise.

4. Evaluation

After training, we need to evaluate the performance of the model on a separate test set. This is important to ensure that the model can generalize to new data and to estimate its performance in a real-world setting.

The TensorFlow Object Detection API provides tools for evaluating object detection models, including metrics such as mean average precision (mAP) and mean intersection over union (mIoU). These metrics measure the accuracy and overlap of the predicted bounding boxes with the ground truth boxes.

Using MLflow, we can track the evaluation process, including the evaluation metrics, and compare the performance of different models and hyperparameters. This information will be useful for selecting the best model and for tuning the hyperparameters.

5. Fine-tuning

After evaluating the model, we may find that its performance is not satisfactory. In that case, we can fine-tune the model by adjusting the hyperparameters or by using a larger or more diverse dataset.

Fine-tuning is a common practice in deep learning and can significantly improve the performance of the model. We can start by using the pre-trained weights of a model that is similar to our problem domain and then fine-tune the model on our specific dataset.

Using MLflow, we can track the fine-tuning process, including the fine-tuning hyperparameters, and compare the performance of different fine-tuned models. This information will be useful for optimizing the model for our specific problem domain.

6. Exporting

Once we have trained and fine-tuned the model, we need to export it in a format that can be used for inference. The TensorFlow Object Detection API supports exporting the model in various formats, such as TensorFlow SavedModel, TensorFlow Lite, and TensorFlow.js.

The exported model can be used for object detection in a variety of applications, such as mobile devices, web applications, and embedded systems.

Using MLflow, we can track the export process, including the export format, and ensure that the exported model is consistent with the training and fine-tuning settings.

7. Deployment

After exporting the model, we need to deploy it in a production environment. The deployment process may vary depending on the application, but it typically involves creating an inference pipeline that reads the input data, applies the object detection model, and outputs the detected objects.

The deployment process can be complex and may involve considerations such as scalability, latency, and security. Using MLflow, we can track the deployment process, including the deployment environment and any issues that arise during deployment.

8. Testing and Optimization

Finally, we need to test the deployed model and optimize its performance. This may involve collecting feedback from users, monitoring the model's performance, and updating the model as needed.

Using MLflow, we can track the testing and optimization process, including the user feedback and any updates made to the model. This information will be useful for maintaining and improving the model over time.

Conclusion

In this article, we explored a workflow for training and deploying an object detection model using the TensorFlow Object Detection API and the MLflow framework. This workflow involves data collection and preparation, model selection and customization, training, evaluation, fine-tuning, exporting, deployment, testing, and optimization.

By using MLflow to track the various stages of the workflow, we can ensure that the process is reproducible, scalable, and optimized for our specific problem domain. With this workflow, we can develop and deploy high-performance object detection models that can be used in a variety of real-world applications.

Reference:

1. Machine Learning adventures with MLFlow
2. Adventures in MLFlow
3. Introduction to MLOps with MLflow
4. Introduction to MLflow for MLOps Part 2: Docker Environment

---

Published by

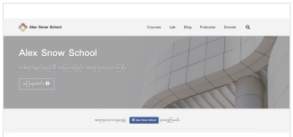
Wut Hmone Hnin Hlaing
M.Sc Data Analytics | Founder | Ex. Lead Back End Engineer
Published • 3d

3 articles

Let's learn about MLOps with MLflow!
#dataengineering #dataanalytics #machinelearning #mlflow #tensorflow

Wut Hmone Hnin Hlaing
M.Sc Data Analytics | Founder | Ex. Lead Back End Engineer

More from Wut Hmone Hnin Hlaing



The story of Alex Snow School

Wut Hmone Hnin Hlaing on ...

Everyone has some savings in the pocket.
Everyone loves to grow it, but don't know how.
Seeking advice from professionals is not cheap and easy.
So, why not learn-by-yourself?

FiniMini: The new way to becoming your own financial adviser

Wut Hmone Hnin Hlaing on ...