INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 6.078
(Volume 6, Issue 3)
Available online at: www.ijariit.com

End to end CI/CD pipeline for Machine Learning

Ram Mohan Vadavalasa
mohan8400@gmail.com

ABSTRACT

In every industry, machine learning applications are becoming popular, however, compared to traditional software applications to the process of developing, deploying, and continuously improving machine learning applications are more complex. In industry practice continuous integration, delivery, and deployment enable organizations to release new features in their products frequently. For engineering processes of developing and designing secure pipeline, to support continuous practices, how machine learning systems should be architected to gain a deep understanding in the process, and how to capture, improve and report data into different aspects of continuous integration, delivery, and deployment. Without proper pipeline for machine learning, it is hard to predict, test, explain, and improve data workflow behavior. Pipelining in machine learning brings different principles and practices to machine learning applications to work in a proper manner. In the industrial sector consequences of an irregular pipeline can cause financial, resource, and time will get wasted and some times it can indirectly influence companies personal reputation in the market. This paper discusses the problems which have been experiencing while building a machine learning pipeline and ultimately describe the framework to implement the problems in workflow. Methodically reviewing the state of the art of continuous execution to organize approaches and tools, recognize challenges and practices. As a result, the machine learning pipeline reduces the gaps and increases the speed of experimentation in the workflow.

Keywords— *Machine Learning*

1. INTRODUCTION

For transforming industries, solving complex real-world problems, and delivering value in all domains, a predictive and analytical way, data science and machine learning is becoming a core capability. Because we have well-developed resources like on-demand inexpensive computing resources on various cloud platforms, specialized accelerators for ML and advanced research in natural language process, computer vision and AI systems will help to accelerate the industry transformation.

Delivering business value to their users and developing predictive models to their businesses, many companies are investing huge amounts of money in data science and Machine learning. Machine learning system is composed of data collection, automation, resource management, process and metadata management, model analysis and data verification, testing and debugging, integration and deployment, serving, monitoring and infrastructure management. It is required to set up a machine learning environment for data science practices in order to develop and operate complex ML systems i.e CI, CD pipeline.

For building secure pipeline for machine learning many tools exist such as docker[13] for containerization, Scikit-Learn[11] or MLlib[12] for classification, Kubernetes[14] for container-orchestration, regression and clustering algorithms, tensorflow[15] for dataflow control, keras[16] python library for neural-network, airflow[17] for workflow management platform, ML flow[18] for end-to-end machine learning lifecycle and kubeflow[19] for making machine learning deployments, GIT[20] or SVN[21] for version control. These tools are essential for building models for constant and stable pipeline maintenance and automating an ML project.

In order to build an end to end machine learning pipeline, is required to do research for finalizing algorithm and hyperparameters, code and programming languages to generating those features, process used to obtain the data from different sources like log files and sensor output, the hardware used to run the model and other required specifications are necessary. When using deep learning or representative learning on unstructured data or semi-structured data, model understanding has become a critical issue because the challenges range from the understanding of different layers in deep architectures, analyzing results, planning better architecture without reducing and disturbing the model accuracy.

Initially discussed main problems encountering during building machine learning models. Later building a pipeline accentuate the pieces which ensure us that we can entirely reproduce any model using it.

2. RELATED WORK

Multiple people involved in different roles in the development and maintenance of a larger-scale machine learning pipeline.

- In an ML project, the team usually involves ML researchers, or data scientists, they mainly focus on model development, data analysis, and experimentation.
- ML in nature is experimental. We should implement different features, modeling techniques, algorithms, and parameter configurations in order to find which suits the best and works for the problem as quickly as possible. The main challenge is tracking which is working perfectly and which does not work. Mainly maximizing code reusability while maintaining reproducibility.
- In ML system testing is more involved than general software applications. In machine learning we need to train model quality evaluation, data validation, and model validation.
- In ML systems deployment plays a huge role in prediction service. For automatically retraining and deploying models in ML systems require a multi-step pipeline. The pipeline has to automate many steps implemented in model building and it reduces complexity and increases productivity.
- Due to constantly evolving new data into the system and due to sub-optimal coding ML models can have reduced and decreased performance in the production. Models can decay its performance and lead to degradation in the overall system output. Therefore, we need to track our data and continuous monitoring which is required; the model needs to send notifications to the user whenever it finds abnormality in the model; model possible to roll back if, in case values deviate from our expectations.
- In ML system Continous integration is no longer only about validating and testing code but also testing and validating data including data schemas and model validation.
- Continous delivery is no longer is about single service or package but it is a system automatically deploy another service for prediction and it concerned with automatically training and serving data in the models when new data arrive at the model.

In ML architecture training and evaluating an ML model to serve as a prediction service or to handle emergencies with the production system people will approach the pipeline with different priorities. ML experts have an enormous knowledge of ML, they can build models, use statistics, and able to handle pipelines. Other experts in the team can understand the problem domain for the specific product and can simultaneously maintain the health of the pipeline. If the pipeline experience new errors due to an out-of-range feature value example the parameter values are other than expected than these individuals in the company handle different roles in order to bring a pipeline to a normal position.

These experts could fix the model training problems like the quantization of the parameters. In an ML pipeline's lifecycle different individuals play different roles throughout the phase of the experiment, mostly ML and data science experts will involve in the pipeline. Many people with different technical backgrounds will have to handle a variety of tasks to keep the pipeline running smoothly and efficiently. It is the main takeaway in the pipeline life cycle.

2.1 Stairs for the machine learning process in data science

In any machine learning project after we define the business use cases and establish the business success criteria, the process of delivering an ML model to production involves different stages: It is possible to follow these steps completely manual and pipeline can be automated. Initially, we select and integrate the relevant data from various data sources[3] for ML task, later we perform exploratory data analysis(EDA) to understand the available data for building ML model then we implement different algorithms to train and preparing data for various ML models later we have to test the model quality and then model is confirmed to be adequate for deployment, in next step validated model is deployed to a target environment to serve predictions. For example microservices with a REST API to serve online predictions, an embedded model to an edge or a mobile device, and a part of a batch prediction system. The model predictive enactment is monitored to potentially entreat another new iteration in the ML process.

The level of mechanization of these steps defines the quality of the ML process, which reverberates the velocity of training models on given data.

2.2 Data lifecycle in machine learning

The ML pipeline lifecycle starts with preparing data for training. Depending on the available data and based on the type of problem encountering data can be unstructured, semi-structured, structured correspond to a different degree of curation. Suppose a team is developed a machine learning model and the goal is to predict application usage based on user visits and time investing in the application features. Here the input data is generated by heterogeneous data sources with different forms and quantities. In recent history first-class priority for many companies is data cleaning, data orchestration, and versioning. The big data brought a dramatic change in many companies for emerging new operations. For true, versioned, and reproducible data pipelines are still at the initial phase of development. But, for large scale production systems there are very powerful extract, transform and load tools such as kafka[22], spark[23],cassandra[24], and hadoop[25]. Machine learning contains additional constraints and a higher standard for understanding data.

There are still many pathholes in data provenance and governance. If a self-driving car is involved with an accident, it is required to know what expectations and assumptions in the data lead and contributed to that error condition. In many ways data can embed bias, so it is necessary for deep learning models, how data is extracted, prepared, and ultimately delivered as an input to the prediction engines in a machine learning pipeline.

2.3 Optimizing suitable processor for Machine learning process

Modern software applications are huge beneficiaries [4] of the fact that almost everything able to run on traditional and regular CPUs. Day to Day hardware landscape is becoming heterogeneous more complex and rich. There is some interesting work around

ARM processor for certain applications, but for the most part this resemblance at the hardware level has allowed a general maneuverability was unquestionably a big part of the overall cloud transformation. Modern machine learning and artificial intelligence owes much of its recent progress works tremendously good on the GPU(Graphics processing unit). Because GPUs for a few years have an enormous transformation in the development and possible to use in general computing devices.

The same GPUs used to power deep learning and video game applications, for accelerating deep learning training and inference hardware vendors are bolting up speed to develop custom silicon. To develop custom chips that are purpose-built for modern machine learning most companies like Graphcore with their "intelligence Processing Unit", Google with the Tensor Processing Unit, and Intel with the Nervana platform, are all racing in this development battle.

3. IMPLEMENTATION PROCESS

One of the key problem we are trying to solve in this paper is building end to end CI/CD pipeline for data processing in machine learning. tooling problem is one of the largest barriers to productive and pervasive artificial intelligence is an infrastructural because solid requirement of these intelligent systems is determinism, scrutability and reproducibility. Kubernetes, Mesos like container orchestration tools are an essential part of modern Ml but they are one of the small part of CI/CD system for deep learning. Additionally, conventional CI/CD and a similar system for ML/AI have different constraints,parameters, and goals. Generally these pipelines look quite commensurate. They take data as an input, build a model, deploy this model and then serve the model. The details, however, are prodigiously enthralling and benefit diving deeper into.

3.1 Conventional Continuous Integration/Continuous Deployment workflow

For application development pipelines Continuous Integration/Continuous Deployment (CI/CD) describes a set of best practices because they are largely administered by development and operations teams to permit software developers to reliably and quickly introduce updates to production applications.Modern CI/CD pipeline includes reproducibility,reliability,safety,speed and version control.

Tools like circleci[26], Bamboo[27], TC[28], jenkins[29], codeship[30], Travis CI[31] attempt to formalize and corroborate a workflow for build, test and deploy. Building infrastructure with these tools quickly push new features into production.

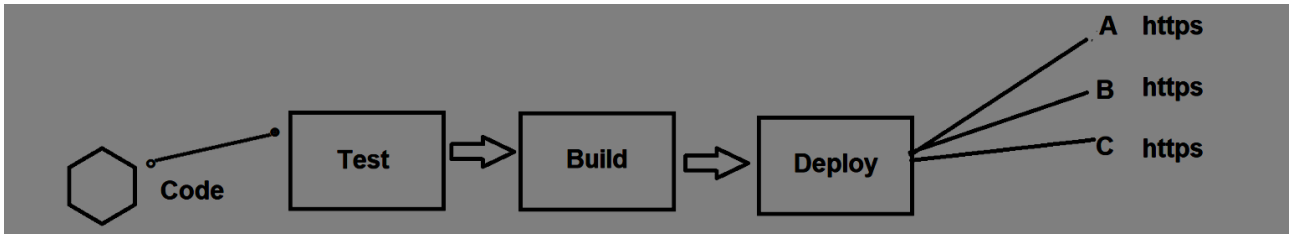

Fig. 1: Conventional Web application workflow

CI/CD are often lumped together as shown in figure 1, but they describe two separate concepts, Continuous Integration (CI) is primarily concerned with testing code as it is pushed i.e making sure that new features application are automatically tested using unit tests. By contrast, Continuous Deployment (CD) represents the actual release/delivery of the tested code. For example, a CD system could illustrate how distinct feature/release branches are deployed, or even how new features are carefully rolled out to new customers.i.e. test feature A on 30% of the new customer base. Eminently, these concepts have been largely inadequate for the modern deep-learning and machine learning pipelines.

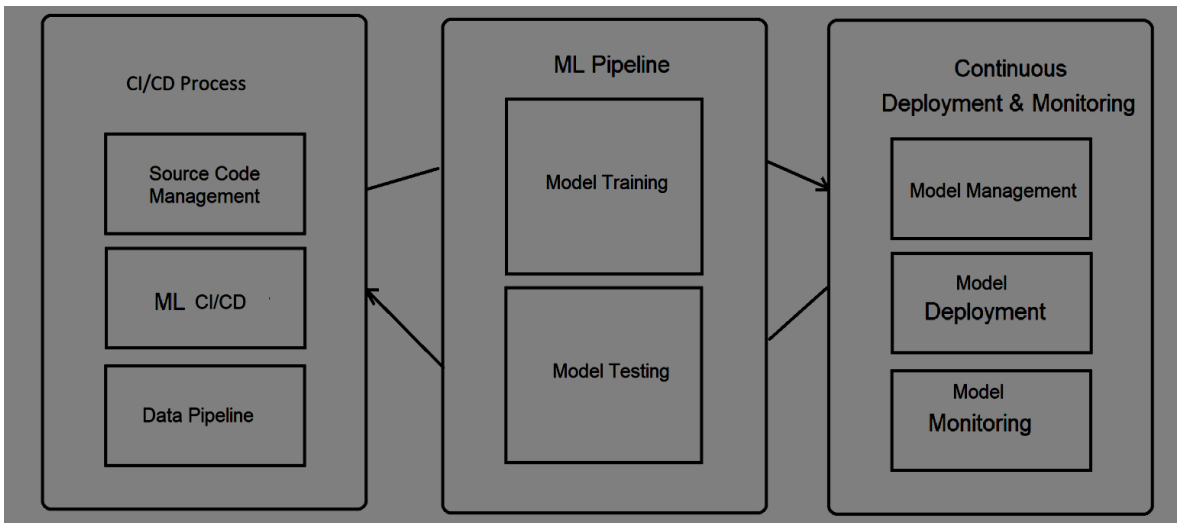

Fig. 2: End to End machine learning pipeline

3.2 Continuous Integration/Continuous Deployment workflow for ML

Ideal CI/CD system for machine learning there are number of key differences as shown in figure 2 and these differences are significantly enough to authorize completely development paradigms, tool stacks and new workflows.

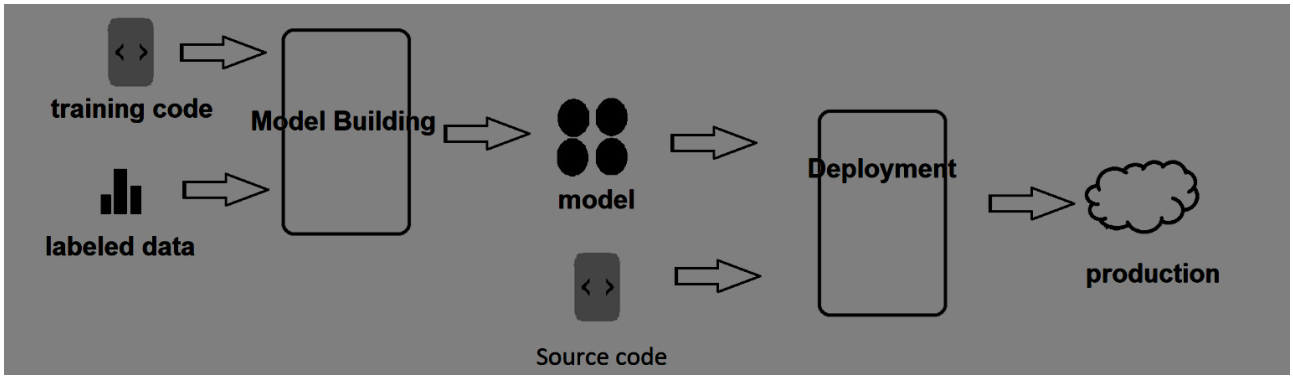

Fig. 3: ContinuousIntegration/ContinuousDeployment workflow in ML

Modern development practices CI and CD are two acronyms as shown in figure 3 and continuous integration is a practice that focuses on making preparing a releases to the system but continuous delivery or continuous deployment have a lot of familiarities, also have a crucial variance that can have critical reverberations for a business. This phase is managed and developed by Data Engineers or Machine Learning Engineers where code meets data and is enabled by absolute capabilities of Continuous Integration and Continuous Deployment which has managed and smoothen this phase.

Basically all our source code resides in source code management setup. Using this source code management(SCM) system similar to git or SVM where we can manage and integrate seamlessly with CI, CD and data pipelines together. CI/CD pipeline assists us to imbrute steps in our delivery process in the software, such as initiating with code builds[5,7], running automated tests, and deploying to a production environment or staging. CI/CD connect everything from commit to deploy. CI/CD activates providing standardised development feedback loops, enable fast product iterations and remove manual errors.

3.2.1 Continuous integration: In continuous integration merging changes back to the main branch as often as possible. These changes are validated by creating a build and running automated tests against the every single branch. Continuous integration puts a great prominence on testing automation to inspect that the application is not crushed whenever new commits are joined into the main branch. For writing tests, new features will help improvement and bug fix; after every new commit pushed into the main repository, we need to run the test data automatically for continuous monitoring. Especially at least once a day we need to merge the changes.

We can gain from this process, less bugs get dispatched to production and regressions are captured quickly by the automated tests. Less context switching because Engineers are awarded as soon as they disturb the process and work on fixing it before they move to another functionality. Here, continuous integration server can run lot of tests within less span of time and it reduces testing costs drastically. We can spend less time on testing and able to focus on crucial development to the quality culture.

3.2.2 Continuous Delivery: Using continuous delivery, we can automate our release process at any time and possible to release new changes to our customers quickly in a sustainable way. In continuous delivery, we can determine to release our updates based on our business requirements. In order to get benefits from continuous delivery we should deploy to production as early as possible to make sure that our release small batches that are easy to troubleshoot in case of a problem.

We need to embrace feature flags so that incomplete features do not affect customers in production. We can release new features more often, thus accelerating the feedback loop with your customers and there will be much less pressure on decisions for small changes, hence encouraging iterating faster.

3.3 Pipelining in Machine Learning

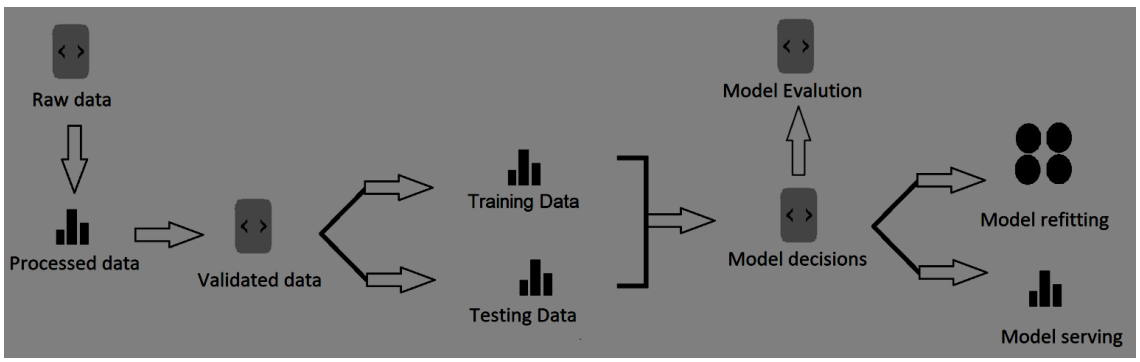

Fig. 4: Data Pipelining in Machine Learning

In traditional software applications data tends to be transactional in nature but in machine learning [9,44] data tends to be structured or unstructured. Unstructured data can further come in multiple forms such as video,audio, images and text. In addition, machine learning pipeline has multiple stages for data management such as data preparation,data annotation,data acquisition,data augmentation, data sampling, and data quality checking; all of these steps involve their own life cycle thereby necessitating a new

set of tools and processes. Machine learning pipeline contains different stages in order to process data with differet machine learning models as shown in figure 4.

Modern containerization such as Docker or similar technology pushed infrastructure forward in many ways but in machine learning containerization still relatively new and substantial amount of production work should take place in machine learning environments. Most of the modern ML is still being done on powerful desktop computers. Lot of big companies working towards building fast processors and big GPU‘s. For speeding up machine learning platforms big companies like Google[32] and IBM[33] investing large amount of money because one who successful with those operations will get long term competitive advantages. Some companies already developed end to end machine learning platforms such as Uber[34] has their version called Michaelangelo[35]. Airbnb[40] has BigHead[41] and Facebook[36] has FBLearner[37] and Kubeflow[38] are open source end to end machine learning platforms.

3.3.1 Data Understanding: When we start a machine learning project, we will get data from heterogeneous sources compiled and make it into single source. Better understanding of all the features and variables easily able to predict a target variable. Once we have the available target variable from our observations based on the data we collected, then we go to use supervised learning, in case if we don't have the target variable available from observations we need to use unsupervised learning.

Data estimation is the first step in data understanding. This should be undertaken before the kick-off a project as it is an predominant step to acquiesce its practicability. This task estimates what data is available and how it situate to the business problem. Collected data should be able to answer the following questions. What type of data is available? Do we have access to the floor truth and the values we are trying to predict? What is data format? From where data should be accessed? Which fields are most important? How do we join the multiple data sources? Which metrics should be considered and reported? How data is related to the end solution? For making predictions using machine learning required a labeled data set, which can contain appropriate values and category that the machine learning model able to learn from data and predict; this is called a floor Truth or floor principle.

All the different data points are available for a machine learning model able to predict the end result and should check that there is a proven evidence, these available data points what we are wishing to achieve. If we add unrelated features to the algorithm indirectly we are adding noise to the system and it will decrease the model performance and it leads to biased model result. So better to use initially obvious features later we can add less obvious features and this will allow us to evaluate if the additional features worth using or not.

It is important to consider potential harms caused by insufficient representation, generalizing patterns could replicate bias model in machine learning and it effects reduced model accuracy. Necessary precautions should be followed in order to avoid risks during model building.

3.3.2 Data Preparation: Similar to other data management challenges [42] in machine learning arises challenges in different forms in data preparation. Once we access the data from different sources, we have to create meaningful insights in the data for preparing model training. We have to check- how many number of records data contains? How many missing values and how we have to deal with those missing values? Check if our data is balanced or not?

The easy way to deal with missing values is removing all the rows that have a missing value in the data but there is a possibility that valuable information can get lost and it could lead to bias model. So it is important to understand, there is any hidden pattern or reason for the missing values. For example certain group of people does not like to vote any political party in the coming election or certain group of people have no interest to talk about particular product in a public survey, removing those rows will prevent learning trends within these groups. At those situations we can replace missing value with an appropriate substitute for example the mean, the medium, mode or continuous variables can be used. For categorical data „NA“ can be used instead of missing values.

Finding outliers are important in data preparation, outliers could indicate because of bad data, at this situation we may wish to remove these data points otherwise replace them with other values just like missing value. Some times outliers are sensitive to linear regression and able to work efficiently with models like random forest or gradient boosted trees.

3.3.3 Data Validation: ML largely depends on its data, and it requires data validation to perform well. Data is the substance that keeps machine learning working. No matter how powerful and consistent a machine learning or deep learning model is, but it can on no account do what we want it to do with inadequate data. Sometimes random noise makes difficult to find patterns in the data points, certain categorical variable having low frequency in the data points and incorrect numeric values, causes model to fail otherwise reduces accuracy. Validation process can only identify stability problem in the model but it cannot directly point out what is going wrong in the model. Once the erroneous code rolls out to model fabrication, it causes the feature to acquire imprecise values.

The bad feature values influence the model and decreases its performance and result in consequential production problems in the course of model serving where the model is executed. In the production ML infrastructure data validation is a key element, by detecting an issue during model building will help us to rollout of a broken model and prevent a significant financial, resource loss and negative impact on the business process.

Having appropriate data there is a possibility that we can avoid Overfitting and underfitting problems during model building. Validation is the gateway to our model being streamlined for performance and being consistent for a period of time before the model is going to be retrained. The Validate module is mandatory to make sure the training data does not contain flaws that may promulgate down to the model training.

3.3.4 Data Cleaning: Based on the problem and the data type data cleaning involve in different techniques. Different methods can be applied and each has its own measurements. Irrelevant data should be filtered from dataset. For example, if we are trying to analyzing data from the general health of the population, we don't need their contact details, similarly when we are interested in only one continent attributes, we don't need to include other continents attributes. Checking number format and data type used, removing unwanted white spaces at the beginning or at the end of the string, making sure all the string values are in certain format. i.e lower or upper case, for numerical values making sure all the values have a certain measurement unit. Checking scaling and normalization of the data in all formats.

The user have to decide to fix the patches in the data based on the notifications, after understanding whether cleaning the data will improve the data quality or not and which part of the data still required to fix.

3.3.5 Model Training: In a ML pipeline the training step is the sector that ends up usually utilizing the most compute time. There is a big difference between modern Deep Learning and traditional data science applications because we can do a data science application on an inexpensive laptop computer but getting start with modern deep learning requires huge investment in expensive Graphics processing units otherwise sophisticated cloud tools.

Depending on the complexity of the model the training task can take anywhere from a couple hours to a couple weeks or more, the type of accelerator being used and the amount of data especially particular dimensions, features to be processed. Tools like TensorFlow Distributed, Uber's Horovod are made huge progress in Distributed training and these tools increases efficiency, reduces complexity and minimizes patches in the pipeline.

Once the data is validated, it will fed into the train module using frameworks like Sklearn[38], Keras[16], Pytorch[39], and TensorFlow[15] . In case neural nets are involved in the model, we have to specify how many hidden layers involved and the specific activation functions at each layer should be used. Later this training model can be evaluated by an evaluate module, there it will check the model has an accurate and acceptable accuracy, wheater the data should contain more features and data or not.

Note that an ML pipeline may train a group of models using either the same or different input data where the predictions will increase accuracy and efficiency of the model. Decisions are made based on what algorithms experiment with the prepared data and what output extract from prioritized features. ML Engineers or Data Scientists will train the models, after making the training and test set splits into labeled data and run multiple experiments before finalizing a suitable model for the pipeline. Throughout the process, many decisions are made depends on the hyperparameters, at the same time endeavor to optimize the architecture of the training algorithm to accomplish the best results.

3.3.6 Model Testing: The finalized model is tested against multiple datasets that are collected and using various competitor services, if they are accessible. All of this is facilitated using compute units like GPUs, TPUs and CPUs and using various storage resources.

3.3.7 Data Serving: No matter how well you build a model, it is most important taking enough care when we ship the model to production and we have to focus enough on data ingestion too. After the model is trained and deployed, the serve module is responsible for receiving the serving input data and preparing it as serving data that can be processed through the model.

The serving input data is usually process a single example at a time. The serving input data needs the same preparation as applied to the raw training data as earlier. It is often the case that serving data is channelled and logged back as training data for the next drilling epoch through some bulk data processing phases, thereby accomplishing the data lifecycle.

3.3.8 Model Refitting: Realtime, predictive engines in machine learning pipeline [46] is not only deploy forward and also, update continuously. When new data enters into the model system will update to these new observations which is called refitting the model. Traditional software application are mostly unidirectional, it means process flows from code to deployment. Usually there are only static releases in software development. i.e, stages like development and production environments but newer technologies such as Jenkins support for each developer distinct deployments processes completely looks static from outside but there is customer feedback and bug fixes that influence the development life cycle and also, this is mostly depends upon organization process and development model usage in the company. With sophisticated teams with substantial resources still it is hard to put static ML pipelines into practice.

By contrast, modern ML and AI doesn't yet have that same ecosystem and there is expected possibility to build same ecosystem for both ML and AI self automated and corrected systems for the both in the future and research would involve in order to build self automated systems. It makes sense for a number of reasons because still there is no lamp and wamp stack for ML pipelines but tools are changing quickly and modern deep learning has regularly bringing grand scheme of things.

3.3.9 Data improvement: Data improvement refers to the enhancement of the training and serving data with information from outward data sources in order to progress the efficiency of the developed model. A common form of advancement is to join in a new data source in order to amplify the current features with new signals. Another form is using the same signals with distinct transformations, for example using a new embedding for log data .

3.4 Model Deployment

After training and testing, models need to be deployed in production[45] for business decisions and continuous monitoring. Continuous deployment is an wonderful way to accelerate the feedback loop in business and it accelerated business process. When we are deploying our pipeline we have to decide on which platform we are going to deploying our pipeline.

Depending upon our business requirement and based on our user traffic we have to choose right platform such as infrastructure as a service, software as a service, platform as a service otherwise machine learning as a service. If our application is containerized then deployment is much easier and containerization also supports containers orchestration(such as Kubernetes).

3.4.1 Model Monitoring: Once model is deployed in the production environment, it is essential to monitor continuously for better and quality business decisions.

3.4.2 Model Management: Model management helps companies to organize, develop,train, validate and monitor models in machine learning pipeline. In model management, trained models are maintained in the model registry and these models will be traced continuously.

4. CONCLUSION

The process from data preparation to deploying a machine learning model life cycle includes several steps, all of those steps need to be controlled and automated in order to improve machine learning model velocity and quality in a pipeline.

This paper explained different practices and challenges facing in CI/CD pipeline in machine learning. CI/CD process in machine learning helps better and faster pipeline development and reduces cost and increases productivity. Future research required to produce coherent mechanisms to handle data in machine learning models and require research on self-updating and self monitoring machine learning pipelines because they will reduce lof of constraints in the pipeline life process.

5. REFERENCES

[1] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman. Project adam: building an efficient and scalable deep learning training system. In OSDI, 2014.

[2] Mu Li, David G. Anderson, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. 2014.

[3] Scaling Distributed Machine Learning with the Parameter Server. In Operating Systems Design and Implementation (OSDI). 583–598.

[4] M. Li, D. G. Andersen, J. W. Park, A. J. Smola et al.,"Scaling Distributed Machine Learning with the Parameter Server." OSDI, 2014.

[5] Anna Wróblewska, Tomasz Stanisławek, Bartłomiej Prus-Zaja˛czkowski, Łukasz Garncarek; Robotic Process Automation of Unstructured Data with Machine Learning

[6] Danilo Sato, Arif Wider, Christoph Windheuser; (martinfowler.com); Continuous Delivery for Machine Learning - Automating the end-to-end lifecycle of Machine Learning applications

[7] Rolando Garcia, Vikram Sreekanti, Neeraja Yadwadkar, Daniel Crankshaw, Joseph E. Gonzalez, Joseph M. Hellerstein; Context: The Missing Piece in the Machine Learning Lifecycle

[8] Peter Sugimura, Florian Hartl Building a Reproducible Machine Learning Pipeline

[9] Hui Miao, Amit Chavan, Amol Deshpande;(University of Maryland, College Park, MD, USA) ProvDB: A System for Lifecycle Management of Collaborative Analysis Workflows

[10] Tom van der Weide, Dimitris Papadopoulos, Oleg Smirnov, Michal Zielinski, Tim van Kasteren Versioning for End-to- End Machine Learning Pipelines;(conference paper)May 2017

[11] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith et al., "MLbase: A Distributed Machine-learning System," CIDR, 2013.

[12] Scikit-Learn; Licence:New BSD License; Available online link:„https://scikit-learn.org/stable/"; Original author: David Cournapeau; stable release data: 0.22 / 3 December 2019

[13] Mllib; Licence:Binaries: software as a service; Available online link: https://spark.apache.org/mllib/

[14] Docker; Licence:Binaries:Freemium software as a service; Available online link: „https://www.docker.com/"; Initial release data: March 20, 2013

[15] Kubernetes; Licence:Apache License 2.0 ; Available online link: „https:// kubernetes.io/"; Developed by: Cloud Native Computing Foundation; Stable release: 1.18 / March 25, 2020;

[16] Tensorflow; Licence:Apache License 2.0; Available online link: „https:// www.tensorflow.org/"; Stable release: 2.1.0 / January 8, 2020;

[17] Keras; Platform: Cross-platform software; Available online link: „https:// keras.io/"; Initial release date: March 27, 2015

[18] Airflow; Licence:Binaries:Freemium software as a service; Available online link: „https://airflow.apache.org/"; Original author(s): Maxime Beauchemin Stable release: 1.10.10 / April 9, 2020;

[19] ML flow; Licence:Binaries:Open source platform; Available online link: https://mlflow.org/"; stable release data: MLflow 1.8 Released! (21 Apr 2020)

[20] kubeflow; Available online link:„https://www.kubeflow.org/"; Stable release: 0.7 / May 7, 2019;

[21] GIT; License: GNU General Public License; Available online link: https://git-scm.com/"

[22] SVN; License: Apache License 2.0; Available online link: https://subversion.apache.org/; Initial release date: October 20, 2000

[23] kafka; Licence:Binaries:Freemium software as a service; Available online link: „https://kafka.apache.org/"; Initial release date: January 2011.

[24] Spark; License: Apache License 2.0; Available online link: https:// spark.apache.org/"; Original author: Matei Zaharia; Stable release: 2.4.5 / February 8, 2020;

[25] Cassandra;License: Apache License 2.0; Available online link: https://cassandra.apache.org/"; Original author(s): Avinash Lakshman, Prashant Malik; Stable release: 3.11.6 / February 14, 2020

[26] Hadoop; Software genre: Distributed computing; Available online link: „https://hadoop.apache.org/"; Initial release date: April 1, 2006.

[27] Circleci; Available online link: „https://circleci.com/
[28] Bamboo; License: Proprietary; free for open-source projects; Availableonline link: https://www.atlassian.com/software/bamboo“; Initial release: 20 February 2007
[29] TC(Team city); License: Proprietary commercial software, Freeware for teams meeting supplier conditions; Available online : link:„https://www.jetbrains.com/teamcity/“; Stable release: 2019.2 / December 9, 2019
[30] Jenkins;License: Proprietary; Available online link: „https://www.atlassian.com/software/jira“; Initial release date: January 2011
[31] Ccodeship ; vailable online link: „https://codeship.com/“
[32] Travis CI; License: MIT License; Available online link: „https://travis-ci.org/
[33] Google; Licence:Binaries:Freemium software as a service; Available online link: „https://www.google.com/ ; Initial release date: January 2011
[34] IBM; Available online link: „www.ibm.com
[35] Uber; Available online link: „www.uber.com
[36] Michaelangelo;Available online link: „https://eng.uber.com/michelangelo-machine-learning-platform/
[37] Facebook Available online link: „www.facebook.com
[38] FBLearner; Available online link: https://engineering.fb.com/core-data/introducing-fblearner-flow-facebook-s-ai- backbone/
[39] Sklearn; Available online link: „https://scikit-learn.org/stable/
[40] Pytorch; Original author(s): Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan;License: BSD; Available online link: https://pytorch.org/ ; Stable release: 1.4.0 / 15 January 2020;
[41] Airbnb ; Available online link: „ww.airbnb.com
[42] BigHead; Available online link: „https://databricks.com/session/bighead-airbnbs-end-to-end-machine-learning-platform
[43] J. M. Hellerstein, V. Sreekanti, J. E. Gonzalez, J. Dalton, A. Dey, S. Nag,K. Ramachandran, S. Arora, A. Bhattacharyya, S. Das, et al. Ground: A data context service. In CIDR, 2017.
[44] L. Berti-Equille, T. Dasu, and D. Srivastava, Discovery of complex glitch patterns: A novel approach to quantitative data cleaning, in ICDE, 2011, pp. 733–744.
[45] Mojtaba Shahin, Muhammad Ali Babar, Liming Zhu; Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices; IEEE Access, 2017.
[46] J.Madhavan, S. Balakrishnan, K. Brisbin, H. Gonzalez, N. Gupta, A. Y. Halevy,K. Jacqmin-Adams, H. Lam, A. Langen, H. Lee, R. McChesney, R. Shapley, and W. Shen. Big Data Storytelling Through Interactive Maps. IEEE Data Eng.
[47] Mitar Milutinovic; Towards Automatic Machine Learning Pipeline Design; Electrical Engineering and Computer Sciences University of California at Berkeley; Technical Report No. UCB/EECS-2019-123; August 16, 2019
[48] Santosh Rao, NetApp; White Paper ; Building a Data Pipeline for Deep Learning Take your AI project from pilot to production; March 2019
[49] Behrouz Derakhshan, Alireza Rezaei Mahdiraji, Tilmann Rabl, and Volker Markl, DFKI GmbH Technische Universität Berlin, Continuous Deployment of Machine Learning Pipelines
[50] Evan R. Sparks, Shivaram Venkataraman, Tomer Kaftanz, Michael J. Frankliny, Benjamin Recht. KeystoneML: Optimizing Pipelines for Large-Scale Advanced Analytics.

