2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)
Karnataka, India. Dec 29-31, 2023

2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON) | 979-8-3503-1912-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/SMARTGENCON60755.2023.10442069

Utilization of Kubeflow for Deploying Machine Learning Models across Several Cloud Providers

Kanwarpartap Singh Gill [1]
Chitkara University Institute of Engineering and Technology, Chitkara University,
Punjab, India
kanwarpartap.gill@chitkara.edu.in

Vatsala Anand [2]
Chitkara University Institute of Engineering and Technology, Chitkara University,
Punjab, India
vatsala.anand@chitkara.edu.in

Rahul Chauhan [3]
Computer Science and Engineering, Graphic Era Hill University,
Dehradun, Uttarakhand, India
chauhan14853@gmail.com

Ruchira Rawat [4]
Computer Science & Engineering, Graphic Era Deemed to be University,
Dehradun, Uttarakhand, India
ruchira.rawat.cse@geu.ac.in

Pao-Ann Hsiung [5]
Dept. of Computer Science and Information Engineering,
National Chung Cheng university,
Chiayi County, Taiwan R.O.C.
pahsiung@cs.ccu.edu.tw

***Abstract*— Kubeflow is a global machine learning platform that is open-source in nature. Its primary objective is to streamline the process of deploying, managing, and scaling machine learning (ML) and deep learning (DL) workflows. This platform is specifically built to operate on Kubernetes, which is a widely used container orchestration platform. Kubeflow offers a comprehensive platform for managing machine learning operations (MLOps), facilitating efficient collaboration across data scientists, ML engineers, and DevOps teams at every stage of the machine learning lifecycle. The use of Kubernetes' containerization capabilities facilitates the establishment of a portable and scalable environment specifically designed for innovative machine learning applications. The objective of this study is to investigate the procedure of installing Machine Learning models on Kubernetes by using Kubeflow, an open-source technology recognised as an end-to-end ML Stack orchestration toolkit. In this study, development of comprehensive Machine Learning models is done using Kubeflow in the form of pipelines. Then many aspects of the tool are analysed, including the simplicity of its setup, the deployment of models, their performance, restrictions, and features. This project aims to serve as a seminar or introduction report, providing assistance to vanilla cloud/Kubernetes users who possess no prior knowledge of Kubeflow. The objective is to guide them in using Kubeflow for the deployment of machine learning models. This research provides comprehensive information and analytics on the performance of Kubeflow, including many aspects such as the setup on multiple cloud platforms and the deployment of our trained model for serving over the internet.**

Keywords*— *Data Science, Kubeflow, Kserve, Kubernetes, Docker, Machine Learning, Deep Learning, Artificial Intelligence

I. INTRODUCTION

The popularity of containers has risen significantly in response to the growing use of Cloud Computing and the creation of Distributed Systems, which are necessitated by the substantial volume of data and internet traffic. This may be attributed to the convenience and scalability attributes shown by containers. Numerous corporations have made substantial investments in the management and implementation of containerized applications. Kubernetes is an open-source technology that facilitates the automation of deployment, scaling, and administration of containerized applications.

Based on recent research published in CNCF as shown in Fig. 1, the number of Kubernetes users exceeds 5.6 million as of 2021.

The convergence of the significant surge in corporate investments in Machine Learning and the corresponding rise of Kubeflow provide an ideal alignment.

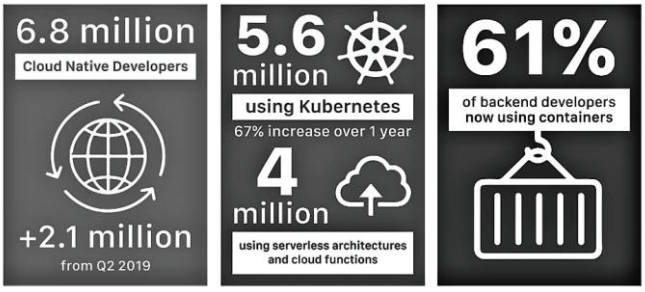

Fig. 1. The surging popularity of Kubernetes has been seeing a notable surge in recent times.

Kubeflow is situated in the convergence of Machine Learning, DevOps, and Data Engineering, encompassing the domain of MLOps in a cloud-based environment.

Kubeflow addresses many challenges encountered in the field of MLOps as shown in Fig. 2, including (i) the inefficiency of existing tools and infrastructure, (ii) the absence of iterative deployment capabilities, (iii) the significance of automated continuous integration and continuous deployment (CI/CD) pipelines, and (iv) the management of data and computing power scalability.

An additional crucial aspect to take into account pertains to the escalating intricacy involved in the deployment and upkeep of extensive machine learning systems on cloud platforms. The process of implementing machine learning models in production environments has unique challenges and requires distinct strategies, which vary significantly from the relatively basic and well-documented procedures used in developmental settings.

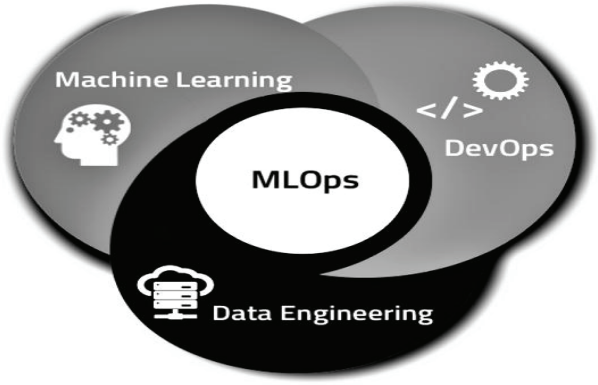

Fig. 2. Depiction of MLOps Structure

Kubeflow assists in mitigating this technical debt to a certain degree by using standardisation and containerization techniques for machine learning processes

II. LITERATURE

There has been prior research conducted on the subject of MLOps and the use of pipelining and orchestration systems. One such instance is that Airflow is a workflow management tool designed for data engineering pipelines, which operates under an open-source framework. The development of the workflow management system was first undertaken by AirBnb with the purpose of facilitating the creation, scheduling, and monitoring of their workflows. In essence, the platform may be described as a generic task orchestration system. Although Airflow has similarities with Kubeflow, it was not specifically designed for Kubernetes and is better suited for general applications. Initially, Airflow was not originally designed for machine learning pipelines, but rather focused on orchestration and workflow management tasks. Secondly Argo is a container-native workflow engine for Kubernetes that is open source. The issue at hand pertains to the broader challenge of task orchestration, specifically within the context of native Kubernetes operations. A component of Kubeflow is constructed using the Argo framework. MLFlow [7] has many similarities with Kubeflow since it is a solution specifically designed for managing workflows and pipelines in the field of machine learning. MLFlow is backed by Databricks and offers flexibility in terms of deployment, since it is not limited to Kubernetes and may be executed in the environment preferred by the user. The tool primarily focuses on machine learning and addresses the challenges of experiment tracking and model versioning. Holmberg et al. provided the outcomes of the parameter tuning procedure and conducted an analysis of the performance of the deployed models in relation to inference throughput. They also offered valuable insights for future research in this area. Additionally, they exhibited enhancements in both the reliance on flavour and the accuracy of the energy response in comparison to the conventional baseline for jet energy adjustments [1]. In their study, Priyankasingh, R.J. et al. used the key performance indicator (KPI) as the dataset, which was created via preprocessing the counter value derived from network data. Additionally, they confirmed whether the framework supports both algorithms or not [2]. Patel, A.N. et al presented detailed implementations of AutoLFADS. The offered tooling showcases effective approaches for using large-scale computing, while also promoting the dissemination and repeatability of the research [3]. Kienzler et al. created the CLAIMED framework, which has shown efficacy in scientific research by effectively resolving the challenges of repeatability and reusability in contemporary data-driven science. The CLAIMED framework facilitates the construction of reusable operators and scalable scientific workflows. It empowers scientists by enabling them to use prior work via the re-composition of workflows from established libraries of coarse-grained scientific operators. Despite the presence of several implementations, CLAIMED is a programming language, scientific library, and execution environment that is not limited to any one programming language, scientific library, or execution environment [4]. Lovén et al. introduced the neural publish/subscribe paradigm as a unique method for coordinating artificial intelligence activities inside extensive distributed AI systems in the computing continuum [5]. Gafurov et al. introduced an innovative strategy for a future DevOps framework that is characterised by being "application-inspired" and capitalises on the progress made in the creation of cloud-based testbeds [6]. Buleje et al. introduced a data-centric and security-focused data fabric specifically tailored for digital health applications. The growing fascination in digital health research has resulted in a notable upswing in the quantity of data generated by Internet of Things (IoT) devices such as smartphones, wearables, and ambient sensors [7-8]. Cha, J.H. et al. conducted a study on the manual work patterns of operators, which are considered unstructured data. They formulated these patterns into power source consumption patterns and analysed them in conjunction with image information. The objective of their research was to develop a manufacturing management platform that could be applied to manual-based, multi-variety, small-volume production methods. Additionally, they aimed to utilise this platform for operator training by integrating it with three-dimensional visualisation technology [9-10]. Cerar et al. conducted a study on the automation of Machine Learning Operations, specifically emphasising feature management and feature store selection [11-12]. Wazir, S. et al. contributed to the development of user-friendly software, focusing their study on MLOps methodologies [13-14]. The study conducted by Lee, S. et al. primarily examined the tangible network infrastructure, with a specific emphasis on its implementation inside a distributed high-performance computing (HPC) environment. In the given context, it is possible for performance degradation to arise as a consequence of network latency while executing distributed deep learning tasks, provided that cluster nodes are allocated to distinct zones or regions, signifying a collection of suitably dispersed high-performance computing nodes [15-16]. Pham et al. proposed a methodological approach aimed at capturing misleading behaviour in order to get insights into the tactics used by hostile actors on the web. The analysis of detrimental solicitations obtained by cyber traps or honeypots is used to train machine learning (ML) models designed for the identification of online attacks [17]. Sorvisto et al. examined the potential implications of the increasing prevalence of generative artificial intelligence (AI) on the advancement of data science. They also proposed the construction of a continuous integration/continuous deployment (CI/CD) pipeline for our toolkit and explored the utilisation of pre-existing cloud services for model deployment [18-19]. Chatzieleftheriou, L.E. et al. used the notion of Network Intelligence Orchestrator (NIO), which has been introduced in recent literature, to establish the

precise criteria for Network Intelligence (NI) algorithms [20].

The key contributions of the study are summarised in the phrases as follows:

Throughout the research utilization of kubeflow for deploying machine learning models across several cloud providers has been discussed.

The importance of Kubernetes, Docker and Kubeflow has been discussed in this research work.

The future prospects have been discussed in the conclusion of this research work.

This work will be helpful to academics who are closely associated with the monitoring of the AI models that are expected to show anomalies.

Many factors are considered, including optimisation tactics and how they affect the results.

The following sections make up the study: Segments three and four present the Kubernetes Architecture and components, Segment five, six presents the setups. Segment seven shows the results and Segment eight concludes the research work followed by references.

III. Kubernetes Architecture and Components

The notable advantage of Kubeflow is its lack of dependency on any one Cloud Provider. Every service provider, such as Amazon SageMaker, IBM Watson, and others, offers comparable services. However, Kubeflow has the capability to abstract these services, enabling them to be executed on any cloud provider that supports Kubernetes. The examination of the fundamental principles and notions of the major backbone and Kubeflow are done as shown in Fig. 3.

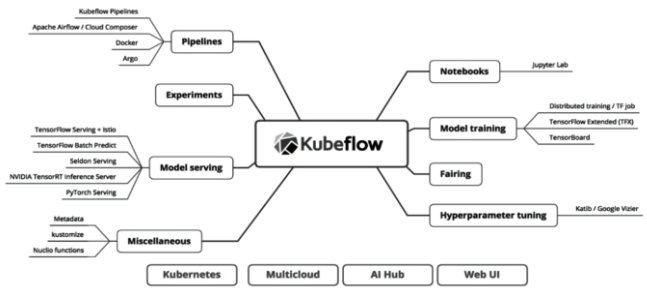

Fig. 3. The components and architecture pertaining to Kubeflow.

Kserve is made to handle the difficulties associated with installing and serving ML models at scale. By using Kubernetes' containerization features to deploy models as containers, it makes scalability, repeatability, and isolation simple. Additionally, it offers an adaptable and expandable architecture that works with a variety of ML frameworks and serving techniques.

Pipelines serve as the fundamental infrastructure of Kubeflow. End-to-end machine learning processes are constructed using them.

The primary objective of Kubeflow pipelines is to attain comprehensive orchestration, facilitate seamless experimentation, and promote the effortless reuse of components and pipelines to expedite the development of end-to-end solutions, eliminating the need for repetitive reconstruction.

Notebooks provide a means of executing web-based development environments inside a Kubernetes cluster by deploying them within Pods. Kubeflow offers inherent compatibility with JupyterLab, RStudio, and Visual Studio Code.

Hyperparameter Tuning, implemented via the use of Katib, is a Kubernetes-native initiative aimed at facilitating the automation of machine learning processes (AutoML). Katib provides support for many techniques in machine learning, including hyperparameter tuning, early halting, and neural architecture search (NAS). It offers the capability to optimise hyperparameters for multiple machine learning frameworks, such as TensorFlow, MXNet, PyTorch, XGBoost, and several more.

Model serving involves the hosting of machine-learning models, either on cloud-based platforms or on local infrastructure, with the purpose of making their functionalities accessible via an application programming interface (API). This enables applications to seamlessly integrate artificial intelligence capabilities into their systems.

In this context, KServe (formerly known as KFServing) is used since it offers native support.

IV. Fundamental Cloud Architectures

In order to provide a reference point for evaluating the ease-of-use and performance features of Kubeflow, the training tasks are performed on two distinct platforms.

1. The NYU Greene Cluster and Server configuration included the execution of MNIST training on the NYU Greene Cluster, a High-Performance Computing (HPC) cluster designed to facilitate research across several implementations and disciplines. The cluster provides support for various task kinds and sizes that need the utilisation of numerous CPU cores, GPU cards, large amounts of memory, or even single-core jobs. This process involves training the Machine Learning code directly on the hardware and saving the model weights for future use.

To facilitate inference hosting, the linserv machine was used, which necessitates the establishment of an Apache Web Server. The user's text does not provide any information to rewrite in an academic manner. The current configuration being used lacks the essential components that Kubeflow offers, namely automated pipelines. Given that all aspects, including environment setup and resource seeking, must be performed manually, this serves as a suitable foundation for evaluating the advantages and disadvantages of using Kubeflow. Furthermore, it should be noted that there is a lack of support for Docker or Kubernetes, which sets it apart from our subsequent baseline.

2. Fundamental Kubernetes (on IBM Cloud): - This configuration included the creation of Container and Kubernetes components to facilitate the training and inference of Deep Learning models, which were hosted on an IBM Kubernetes cluster. In order to carry out the required operations, it is necessary to generate and publish a Docker image to Docker Hub [15]. Subsequently, this image may be used in the YAML files for the purpose of establishing jobs and containers. Therefore, the training and hosting of the MNIST model were conducted inside the IBM Kubernetes

cluster. In contrast to the previous configuration, the current arrangement delegates the management of the environment and resource components to Kubernetes. This setup is capable of supporting generic containers. However, it still requires the execution of the "kubectl apply -f" command in order to deploy each YAML file to the Kubernetes cluster. An area for improvement is in the development of a comprehensive automated system that effectively oversees all stages of the machine learning process, including data preparation, training, and inference. This is precisely the objective accomplished via the use of Kubeflow.

Based on the established baselines, implementation of two distinct configurations is done to evaluate the performance and functionality of Kubeflow.

- Running the MNIST dataset on Kubeflow inside the Google Cloud Platform (GCP) environment may be achieved via the use of an end-to-end (E2E) approach and a code-based methodology.
- Running the MNIST dataset on Kubeflow inside the IBM Cloud infrastructure is achieved via the use of an end-to-end (E2E) and code-based approach.

V. Setup on Google Cloud

During the investigation of the online documentation available on the official website of Kubeflow, various methods for setting up Kubeflow using Google Cloud were identified as shown in Fig. 4. One of these methods entailed the creation of a Kubernetes Cluster on the Google Cloud Platform (GCP), followed by the establishment of a Notebook instance on GCP's AI Platform. This Notebook instance facilitated the opening of a JupyterLab instance, subsequently allowing for the creation of a new instance. The objective is to establish a pipeline on a unified platform, enabling the connection between two components, ultimately facilitating the generation of new code inside the notebook. The pipeline is executed using the previously instantiated instance.

Fig. 4. Integration of Kubernetes and Google Cloud Platform.

VI. Setup on IBM Cloud

When configuring Kubeflow on IBM Cloud, there were primarily two sources of documentation available for the procedure. In reference to the Kubeflow Docs site, The development team of IBM has provided the information.

There are two primary setups of Kubernetes on which Kubeflow may be implemented on IBM Cloud. The subject of discussion is the Classic IBM Cloud Kubernetes cluster. The IBM Cloud Kubernetes cluster in the VPC-Gen2 environment.

Both of these platforms exhibit distinct setup procedures, particularly in terms of storage, authentication, network access, and other related aspects.

VII. RESULTS

In relation to the code/pipelines that are developed and executed on Kubeflow, the adhere to two primary methodologies, namely the Code Approach and the End-to-End (E2E) Approach.

A. Two Attributes Of Kubeflow

In this project, our primary emphasis was on examining the many attributes of Kubeflow and the comprehensive implementation of the Machine Learning Pipeline.

1. Executing a container image straight via a TensorFlow Job (End to End). In this study, a Docker image is used to execute a training code inside the Kubeflow framework. The procedure involves the execution of a comprehensive pipeline, including data loading, pre-processing, and model serving on KServe. The experiment was conducted using the MNIST dataset. The objective of this research is to demonstrate the whole process of creating and executing a pipeline using Kubeflow, including elements such as hyperparameter tweaking and automated machine learning (AutoML).

2. The process of developing a kubeflow pipeline involves the composition of TensorFlow code on top of a foundational picture. Additionally, we endeavour to execute a pipeline that incorporates a Neural Network model developed by our team. This is accomplished by turning our original TensorFlow code into a more streamlined Kubeflow component, which is then deployed inside the pipeline. Additionally, the task included the recognition of digits using the MNIST dataset. Users have the capability to effortlessly execute their pre-existing Machine Learning model on Kubeflow with few adjustments. Additionally, they may take use of Kubernetes's image pull functionality to train models, providing them with enhanced flexibility.

B. Three Forms Of Pipeline Modelling

- One of the tasks performed by Katib is hyperparameter tweaking. In order to thoroughly examine the many elements of Kubeflow, our research initiative involves an in-depth exploration of Katib. Katib is a project that is native to Kubernetes and serves as a valuable tool for tasks such as hyperparameter tweaking and early halting. In this analysis, we examined the challenge of hyperparameter adjustment.
- The objective of the experiment was to minimise the loss throughout the training process of the MNIST model, with the specific target of achieving a loss value of 0.001. The code uses a Docker image, namely "docker.io/liuhougangxa/tf-estimatormnist," which employs LeNet, a machine learning model designed for image classification, to optimise the hyperparameters. The chosen approach involves using a random search technique to explore the hyperparameters. This method will systematically search for optimised hyperparameters within a certain range, specifically focusing on the learning rate (ranging from 0.01 to 0.05) and batch size (ranging from 80 to 100). The algorithm will choose

values randomly and without replacement, ultimately identifying the combination that results in the lowest loss.

- The TFJob Training Task is a specialised resource in the Kubenetes framework designed to streamline the execution of TensorFlow training tasks on a Kubernetes instance generated by MiniKF. This stage will use the optimal hyperparameters identified in Katib's experiment to train the same model on which the hyperparameters were fine-tuned.
- The KServe Inference resource is a widely-used platform for Model Inference that is capable of serving Machine Learning models on many frameworks such as TensorFlow, PyTorch, and others. The aforementioned process generates a serving component Uniform Resource Locator (URL) that will be used in the inference of the model. One of the primary benefits of using this asset is its ability to automatically adjust its capacity and intelligently distribute incoming traffic for the purpose of load balancing. This stage also includes the utilisation of volume, which will be established in the future step, as persistent data storage for our machine learning model.

To optimise the model's performance, we used AutoML techniques by integrating Katib with Kubeflow. In our study, we used random, Bayesian, and step algorithms.

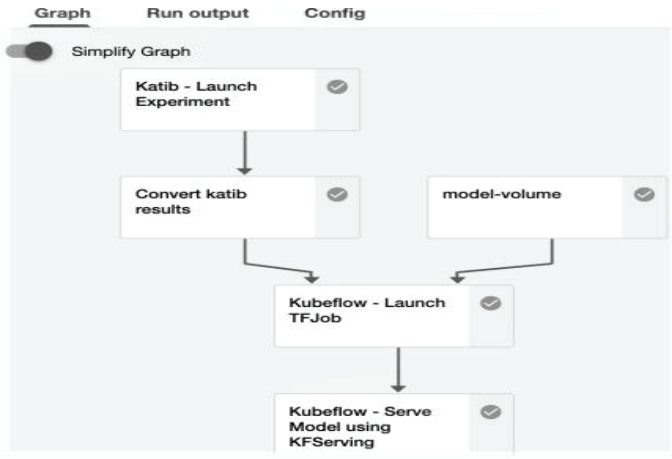

Fig. 5. Pipeline Completion Depiction after running the notebook.

C. Model Serving Using Kserve

In the context of model serving, KServe is being used to establish the endpoints. Kubeflow simplifies the process of creating an endpoint, allowing users to conveniently feed test data to the API endpoint through a CuRL request and get the corresponding response. This task may also be accomplished programmatically by using any request module. The experiment conducted on the linserv platform included the development of a rudimentary Flask application. This application was designed to load a pre-trained PyTorch model and provide predictions. Subsequently, the application was deployed on the NYU Server. In order to demonstrate the functionality of an end-to-end machine learning (ML) based application, we have also developed a basic user interface (UI). The deployment process for the UI Component closely resembles the deployment of any application on the Kubernetes platform.

A basic static HTML website is developed to facilitate user picture uploads and perform API queries to the inferencing API. The resulting output is then shown to the user.

Next, we proceed to containerize the HTML file using Docker and afterwards deploy it using the Nginx web server in a very efficient manner. After obtaining the docker image, the application may be deployed to the Kubernetes platform. A basic Pod is used for this purpose.

The last phase is exposing the system to the internet, which is accomplished by using a load balancer.

D. Additional Tools Integration With Kubeflow Deployment

The Kubeflow deployment incorporates many supplementary tools, which include the following:

- Istio is a service mesh platform that facilitates the management of microservices in a distributed system. Istio is an open source framework that is used by Kubeflow in order to provide comprehensive authentication and access control mechanisms. The aforementioned is an open-source implementation of a service mesh that exhibits great performance. It is used to depict the intricate network of microservices and their corresponding interactions. Kubeflow encompasses a collection of tools, frameworks, and services that collaborate harmoniously to establish comprehensive machine learning workflows. The workflows are generated by the integration of several services and components. Kubeflow facilitates the integration of these components by providing the necessary infrastructure. Kubeflow leverages Istio to enhance the security of service-to-service communication inside a Kubeflow deployment, using robust identity-based authentication and authorisation mechanisms. The additional criteria may include aspects such as failure recovery, metrics, monitoring, and traces for traffic inside the deployment, including cluster entrance and egress.
- KServe, formerly referred to as KFServing, is a platform that offers an inferencing service on Kubernetes. It is designed to provide efficient and abstract interfaces for popular machine learning frameworks like as TensorFlow, scikit-learn, XGBoost, and PyTorch. This platform aims to address the requirements of serving production models. The platform provides support for sophisticated deployment strategies such as canary rollout, experiments, ensembles, and transformers. KServe offers comprehensive support for current serverless inference workloads, including essential functionalities like as autoscaling, networking, health checking, and server setup. Additionally, it provides advanced serving capabilities, including GPU autoscaling, scale to zero, and canary rollouts, which are particularly beneficial for machine learning deployments.

E. Hyperparameter Tuning On Katib Kubernetes Native Project

Experiments were conducted on the subject of Katib in order to determine the optimal values of hyperparameters.

This was achieved by the use of three distinct methodologies, namely Grid Search, Random Search, and Bayesian optimisation search.

Grid Search is a method that systematically explores the whole hyperparameter space by training a model for each conceivable combination of hyperparameter values in a sequential way. The use of grid search in practical applications is very infrequent due to the exponential growth in the number of models that need to be trained as the number of hyperparameters is raised. This process exhibits a significant lack of efficiency in terms of time utilisation. It is evident from Fig. 6 that as the number of runs increases, the grid search method requires more time in comparison to the Random and Bayesian optimisation search methods.

The primary distinction between random search and grid search is in the selection of values for testing. Unlike grid search, random search does not evaluate all possible values, but rather randomly selects values to be examined. One benefit of using randomised search is the ability to expand the search boundaries without the need for further iterations, as seen in Fig. 6. This approach avoids the time-consuming nature of raising the number of iterations. The primary focus is in the use of this method for identifying precise boundaries, hence facilitating an extensive investigation inside a more confined area.

Bayesian optimisation is a method of sequential model-based optimisation that leverages the outcomes of past iterations to inform the selection of subsequent hyperparameter values for the model. Bayesian optimisation search demonstrates efficiency due to its ability to intelligently choose hyperparameters. When the complexity of the model is lower, Bayesian Optimisation requires fewer iterations to reach the global optimum, but Random and Grid search may need a higher number of iterations to get the same result. In the present scenario, the training of our model is of a somewhat intricate nature, which explains the comparatively shorter duration required for random search as opposed to Bayesian and Grid Search methods.

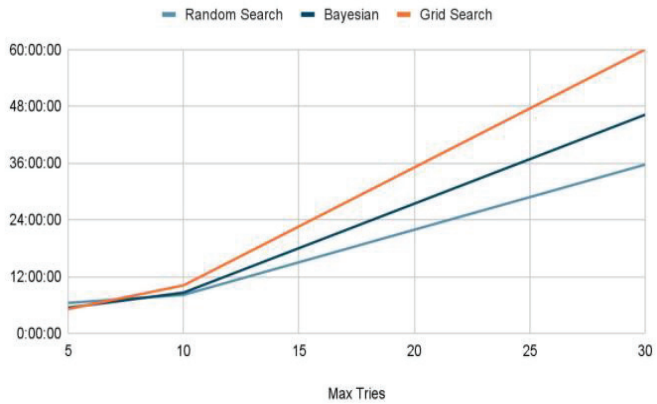

Fig. 6. Hyperparameter tuning algorithms depiction

F. Comparison Of Inference Time Across All Platforms

In this study, a total of four experiments were conducted. Every experimental trial included of a training instance for the model, followed by the deployment of an inference service on the cloud platform.

- The MNIST code was executed on the NYU Greene Cluster and the resulting model was placed to Linserv for inference.
- The objective of this task is to execute a fundamental MNIST image on the Kubernetes platform, specifically on IBM Cloud. Subsequently, the aim is to establish a basic application programming interface (API) and use a LoadBalancing service for hosting purposes.
- The execution of the MNIST dataset on Kubeflow inside the IBM Cloud environment, with subsequent serving facilitated by KServe, is explored.
- The execution of the MNIST dataset on Kubeflow inside the Google Cloud Platform (GCP), with the subsequent deployment facilitated by KServe.

G. Comparison Of Performance Of Kubeflow Across Clouds

The running time of our two techniques has been assessed.

(i) The customised model is being used to do digit recognition.
(ii) The end-to-end pipeline is implemented using Katib and Model Serving.

The comparative analysis indicates that, on average, Kubeflow on Google Cloud exhibits somewhat superior performance compared to IBM Cloud.

VIII. CONCLUSION

This research provides a comprehensive analysis of the integration of Kubeflow on both IBM Cloud and GCP, while conducting a comparative evaluation with identical models deployed on a K8s cluster. Additionally, models trained on NYU HPC without the use of Kubeflow are also included in the comparison. Our findings indicate that the length of the end-to-end run for Kubeflow on Google Cloud Platform (GCP) was the shortest. However, it is noteworthy that IBM Cloud outperformed all other models in terms of providing the quickest inference time. The observed outcomes may be attributed to the subtle variations in the configurations and infrastructures used by the two cloud service providers. Kubeflow serves as a very effective tool and platform for consumers and developers alike, facilitating the testing and productionization of comprehensive Machine Learning solutions. The convenience of establishing a pipeline, including the stages of Data Ingestion/Preprocessing, Parameter tweaking, and Model serving, makes it a useful asset for any organisation seeking to implement a machine learning solution. Furthermore, the use of Kubernetes as a foundation leverages the many benefits it offers in terms of scalability and orchestration. However, the implementation of Kubeflow requires significant effort in terms of setup and reliance on many services like Istio, KServe, etc., which might pose integration challenges within the framework. It is important to acknowledge that while Kubeflow is very effective for executing machine learning tasks on Kubernetes, it is essential to recognise that high-performance computing (HPC) and big data systems include unique variations of MLOps. It is firmly believed that the provision of enhanced documentation and community support will provide Kubeflow with the necessary tools to achieve success as a framework.

REFERENCES

[1] Holmberg, D., Golubovic, D. and Kirschenmann, H., 2023. Jet energy calibration with deep learning as a Kubeflow pipeline.

[2] Priyankasingh, R.J. and Vani, H.Y., 2023, March. Deployment and Serving ML Models Using Kubeflow and KfServing. In Proceedings of Third International Conference on Advances in Computer Engineering and Communication Systems: ICACECS 2022 (pp. 283-293). Singapore: Springer Nature Singapore.

[3] Patel, A.N., Sedler, A.R., Huang, J., Pandarinath, C. and Gilja, V., 2023. High-performance neural population dynamics modeling enabled by scalable computational infrastructure. Journal of Open Source Software, 8(83), p.5023.

[4] Kienzler, R., Khan, R., Nilmeier, J., Nesic, I. and Haddad, I., 2023. CLAIMED--the open source framework for building coarse-grained operators for accelerated discovery in science. arXiv preprint arXiv:2307.06824.

[5] Lovén, L., Morabito, R., Kumar, A., Pirttikangas, S., Riekki, J. and Tarkoma, S., 2023. How Can AI be Distributed in the Computing Continuum? Introducing the Neural Pub/Sub Paradigm. arXiv preprint arXiv:2309.02058.

[6] Gafurov, D., 2023. Next-generation DevOps for network and compute-intensive applications (Doctoral dissertation, University of Missouri--Columbia).

[7] Buleje, I., Siu, V.S., Hsieh, K.Y., Hinds, N., Dang, B., Bilal, E., Nguyen, T., Lee, E.E., Depp, C.A. and Rogers, J.L., 2023, July. A Versatile Data Fabric for Advanced IoT-Based Remote Health Monitoring. In 2023 IEEE International Conference on Digital Health (ICDH) (pp. 88-90). IEEE.

[8] Gill, K.S., Anand, V. and Gupta, R., 2023, May. Garbage Classification Utilizing Effective Convolutional Neural Network. In 2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN) (pp. 1-4). IEEE.

[9] Cha, J.H., Jeong, H.G., Han, S.W., Kim, D.C., Oh, J.H., Hwang, S.H. and Park, B.J., 2023, July. Development of MLOps Platform Based on Power Source Analysis for Considering Manufacturing Environment Changes in Real-Time Processes. In International Conference on Human-Computer Interaction (pp. 224-236). Cham: Springer Nature Switzerland.

[10] Gill, K.S., Sharma, A., Anand, V. and Gupta, R., 2023, March. Flower Classification Utilisizing Tensor Processing Unit Mechanism. In 2023 2nd International Conference for Innovation in Technology (INOCON) (pp. 1-5). IEEE.

[11] Anand, V., Gupta, S., Altameem, A., Nayak, S.R., Poonia, R.C. and Saudagar, A.K.J., 2022. An Enhanced Transfer Learning Based Classification for Diagnosis of Skin Cancer. Diagnostics, 12(7), p.1628.

[12] Cerar, G., Bertalanič, B., Mohorčič, M., Grobelnik, M. and Fortuna, C., 2023, June. Feature Management for Machine Learning Operation Pipelines in AI Native Networks. In 2023 International Balkan Conference on Communications and Networking (BalkanCom) (pp. 1-5). IEEE.

[13] Prema, C.E., Suresh, S., Krishnan, M.N. and Leema, N., 2022. A Novel Efficient Video Smoke Detection Algorithm Using Co-occurrence of Local Binary Pattern Variants. Fire Technology, 58(5), pp.3139-3165.

[14] Wazir, S., Kashyap, G.S. and Saxena, P., 2023. MLOps: A Review. arXiv preprint arXiv:2308.10908.

[15] Lee, S., Raza Shah, S.A., Seok, W., Moon, J., Kim, K. and Raza Shah, S.H., 2023. An Optimal Network-Aware Scheduling Technique for Distributed Deep Learning in Distributed HPC Platforms. Electronics, 12(14), p.3021.

[16] Banerjee, D., Kukreja, V., Hariharan, S. and Sharma, V., 2023, March. Fast and Accurate Multi-Classification of Kiwi Fruit Disease in Leaves using deep learning Approach. In 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA) (pp. 131-137). IEEE.

[17] Pham, V.H., Nghi, H.K. and Nguyen, H.Q., 2023. Deception and Continuous Training Approach for Web Attack Detection using Cyber Traps and MLOps. VNUHCM Journal of Science and Technology Development, 26(2), pp.2729-2740.

[18] Sorvisto, D., 2023. Deploying Stochastic Systems. In MLOps Lifecycle Toolkit: A Software Engineering Roadmap for Designing, Deploying, and Scaling Stochastic Systems (pp. 189-216). Berkeley, CA: Apress.

[19] Gundaboina, L., Badotra, S., Bhatia, T.K., Sharma, K., Mehmood, G., Fayaz, M. and Khan, I.U., 2022. Mining cryptocurrency-based security using renewable energy as source. Security and Communication Networks, 2022. [20] Chatzieleftheriou, L.E., Gramaglia, M., Camelo, M., Garcia-Saavedra, A., Kosmatos, E., Gucciardo, M., Soto, P., Iosifidis, G., Fuentes, L., Garcia-Aviles, G. and Lutu, A., 2023, June. Orchestration Procedures for the Network Intelligence Stratum in 6G Networks. In 2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit) (pp. 347-352). IEEE.