ESP-JETA
ESP Journal of Engineering & Technology Advancements
ISSN: 2583-2646 / Volume 1 Issue 2, October 2021 / Page No: 42-56
Paper Id: JETA-V1I2P108 / Doi: 10.56472/25832646/JETA-V1I2P108

Original Article

Proactive Scaling Strategies for Cost-Efficient Hyperparameter Optimization in Cloud-Based Machine Learning Models: A Comprehensive Review

Madan Mohan Tito Ayyalasomayajula[1], Sailaja Ayyalasomayajula[2]
[1]*Computer Science, School of Business & Technology, Aspen University, Arizona, United States of America (USA).*
[2]*Information Systems, School of Business & Technology, Aspen University, Arizona, United States of America (USA).*

Received Date: 28 August 2021 Revised Date: 06 October 2021 Accepted Date: 24 October 2021

Abstract: *Hyperparameter tuning is critical to machine learning model development, significantly impacting model performance. However, the process can be time-consuming and resource-intensive, especially when dealing with complex deep-learning models and large datasets. This paper explores proactive scaling strategies for efficient and cost-effective hyperparameter configuration in machine learning models using cloud infrastructure. By employing machine learning algorithms to analyze workload patterns and predict future requirements, cloud infrastructure can automatically allocate resources and adjust hyperparameters accordingly. Two approaches – proactively scaling the cluster size beforehand and parallel scaling of models for adaptation to changing cloud conditions – are examined, demonstrating their ability to deliver substantial cost savings and improved performance compared to reactive scaling methods. Additionally, the integration of a cloud service called "Scavenger" is discussed, which identifies the most statistically and parallelly efficient job configuration parameters and returns a cost-optimized cluster configuration, further enhancing the benefits of proactive scaling. Overall, this research underscores the improved performance that can be achieved with proactive scaling strategies, thereby boosting the effectiveness of machine learning modeling in the cloud.*

Keywords: *Artificial Intelligence, Auto-Scaling Techniques, Big Data, Bayesian Optimization, Cloud Computing, Cloud Infrastructure, Deep Learning, Genetic Algorithms, Grid Search, Hyperparameter Tuning, Hyperparameter Optimization, Pro-Active Scaling Strategies, Random Search, Resource Allocation.*

I. INTRODUCTION

Hyperparameter tuning, a crucial aspect of machine learning model development, is often complex and time-consuming. With many hyperparameters to tune and limited training data, the traditional approach involves experimentation through random search or hyperparameter optimization frameworks. However, the stochastic and deterministic nature of algorithms, the cost of cloud model training, and the time required to switch between architectures pose significant challenges to this process. This paper addresses these challenges by presenting a novel, high-level, custom hardware-friendly implementation of algorithms explicitly designed for end-to-end hyperparameter tuning across supervised and unsupervised machine learning models.

Considering these challenges, the “happiest path" emerges as a promising solution. It proposes that performance-optimal hyperparameters can indeed be identified. Furthermore, a proactive and distinctive happiness long short-term memory setup comes into play by accurately interpreting continuous shifts in performance with high precision. Consequently, this setup achieves the minimum energy peak values, representing a valuable advancement within hyperparameter tuning. During the development of supervised and deep learning models, deciding on the optimal hyperparameters can be tedious and time-consuming due to the many factors involved. While employing multiple trials with varying candidates on diverse hardware environments may provide insights into scalability, it does not always guarantee a straightforward outcome. Cloud computing offers a flexible and economically viable alternative for accommodating model training. Research efforts focus on promoting power-efficient job scheduling strategies, detecting idle nodes, and optimizing resource utilization for improved efficiency [3], [49].

A. Background and Significance

Various research efforts have explored ways to efficiently select the most suitable cloud instances or infrastructure for hosting workloads. One group of methods, including Micky, gathers decoupled low-level performance metrics from hardware and


This is an open access article under the CCBY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/2.0/)

software components to present heterogeneous workloads with their ideal cloud instances, aiming to minimize cloud costs without compromising performance [6]. Another approach employs low-level instances-specific software performance models to pinpoint efficient cloud instances for specific workloads. As cloud adoption grows, the need to optimally configure cloud environments for various workloads becomes increasingly essential. Data Stream processing systems are integral to several big data management solutions. The appropriate cloud infrastructure ensures satisfactory performance levels in cloud-hosted environments [4], [50]. Given the dynamic landscape of cloud markets featuring diverse pricing structures, service providers, and offerings, determining the optimal cloud instances presents a formidable challenge.

Public clouds facilitate the execution of intricate workloads through their efficient and cost-effective means. Workloads in cloud environments come in various sizes and requirements, necessitating careful consideration when selecting the most fitting cloud instance. Merely opting for the least expensive or most minor cloud instance is not necessarily advantageous; it could lead to under-provisioning or over-provisioning, resulting in decreased performance or unnecessary expenses, respectively [5]. For instance, deploying larger and pricier cloud offerings would negatively impact computation speedups. In contrast, smaller and less expensive virtual machines (VMs) might fail to meet the minimally acceptable performance threshold demanded by the workload.

B. Research Problem and Objectives

Hyperparameter optimization (HPO) is a crucial component of machine learning model development, contributing to maximum accuracy, reduced training times, and heightened stability across diverse data distributions. Its applicability extends to numerous disciplines, encompassing natural language processing, molecular chemistry, high-energy particle physics, music generation, and computer vision tasks like image classification, video analytics, and face recognition. Within the machine learning and AI community, HPO holds significant weight due to its widespread implementation across multiple cloud platforms owing to the vast infrastructure requirements. Despite the benefits, the expanding array of configurable hyperparameters creates a complex scaling predicament. HPO necessitates substantial computational resources, with the demand for extra resources exhibiting nonlinear growth. This feature underscores the necessity for elastic, affordable, and efficient scaling tactics to manage the mounting investments necessary for sustaining productivity enhancements. To investigate this challenge, our study explores developing an innovative cloud autoscaling strategy to control the burgeoning costs related to this computationally exhaustive hyperparameter optimization problem. Previous works include Primm et al. (2016), who introduced AutoML: Automated Machine Learning Systems, and Hsu et al. [5], who presented DeepAuto: An automated machine learning system based on deep reinforcement learning. These studies highlight the relevance and urgency of addressing the escalating costs associated with large-scale HPO.

C. Scope and Organization of the Paper

This study expands upon prior research by integrating fundamental theories and concepts into our analysis. Our central objective revolves around improving the efficiency of hyperparameter optimization (HPO) procedures in cloud settings, targeting the reduction of overall runtime and infrastructure expenses. We examine one architecture for implementing HPO in the cloud, outlining algorithms tailored to expedite the initialization stage and decrease provisioning costs. Furthermore, we scrutinize failure recovery mechanisms and cost-conscious training methodologies. Leveraging burst awareness, we design a proactive rescaling configuration to optimize the delicate equilibrium between performance and cost [7].

This exploration commences by investigating the employment of cloud environments for HPO, recognizing the underlying expenditures. Our research represents an initial step towards cost-effective HPO, where we seek to find a harmonious balance between performance and cost. Our objective function considers the trade-offs between these dimensions, mirroring the typical approach in multi-faceted scheduling for jobs capable of being executed on assorted VM instances. At the outset, we evaluate the merits of a non-provisioning algorithm, analyzing how different machine learning models fare when trained on various instance types and utilizing validation dataset accuracies as benchmarks. Subsequently, we zero in on fine-tuning the loss function configuration, empowering assessment of both predictive and fiscal costs linked to establishing the model—representing an advanced cost-aware HPO technique.

II. FOUNDATIONS OF HYPERPARAMETER CONFIGURATION

The advancements in learning algorithms and hardware technologies have rapidly expanded machine learning (ML) models to massive scales. Concurrently, data quality, volume, and feature complexity improvements have accelerated. Deep learning (DL) and hyperparameter optimization techniques significantly sway artificial intelligence discourse [5]. DL signifies an

advanced ML approach that constructs machines directly from data. Precisely tuning hyperparameters enhances DL performance. Traditional hyperparameter optimization approaches encounter several restrictions when dealing with large datasets, leading to the emergence of newer methods boasting advantages over older ones. Although it was speculated that cloud computing could autonomously determine optimal hyperparameters for learning frameworks, not all providers share identical capabilities. To address this constraint, a novel methodology is proposed for cloud hyperparameter tuning. Among the factors impeding progress were the sheer size of training datasets and the richness of modeling features, necessitating substantial time commitments to yield satisfactory outcomes. Hyperparameter tuning, or hyperparameter configuration or optimization, is important in data mining and machine learning applications [36]. Essentially, hyperparameters remain unlearned during model training and must be predefined beforehand. The selected hyperparameter values influence a machine learning model's learning rate and generalizability. Thus, hyperparameter tuning is decisive in shaping the learning model's effectiveness. Viewed as an optimization problem, the hyperparameter learning process can adopt two broad categories of solutions, evident in the search-based and model-based approaches.

As indicated by their name, search-based methods rely on systematic searches to locate promising hyperparameter combinations. Popular evaluation techniques employed in these methods include grid search, random search, and brute force methods [23]. Their primary principle involves considering every possible combination of hyperparameters empirically or randomly. Meta-heuristic algorithms later emerged as alternatives for hyperparameter configuration. These algorithms primarily aim to strike a balance between exploration and exploitation during the search process.

A. Definition and Importance of Hyperparameters

Hyperparameters represent critical elements that substantially impact a model's robustness, geometric properties, and performance outcomes. Significant hyperparameters include model parameters, such as the number of hidden units, regularization strengths, dropout rates, and learning rates, which are cross-validated against training data. Infrastructure for data centers, cloud computing, or cloud services consists of numerous interconnected resources. Fundamental principles of cloud computing highlighted in the literature include scalability, flexibility, cost-effectiveness, and exceptional reliability. Cloud services originate from various cloud providers, each possessing unique attributes, including throughput, instruction execution time, and energy consumption. Since no single automatic model selection method excels universally, hyperparameter configuration becomes indispensable for identifying the ideal and customized model suited for a given input dataset. Minimal data usage is another advantage of employing hyperparameter tuning. Hyperparameters exert a profound influence on the efficacy of a machine-learning model. While achieving a flawless model might be impractical, fine-tuning hyperparameters serve as a viable means to augment model performance. Moreover, selecting appropriate hyperparameter values contributes positively to desirable generalization and organizational qualities (model robustness, reliability, and interpretability of predictions), functional and operational characteristics, and financial and infrastructure performance enhancement.

Grid search is a traditional method for hyperparameter optimization characterized by iterative adjustments to hyperparameter values in pursuit of optimal configurations. Through grid search, a range of potential hyperparameter values is specified, and model performance is evaluated using all conceivable permutation and combination pairs. Nevertheless, discovering the optimal hyperparameter value via grid search comes at the expense of increased computational overhead. Optimization techniques aim to elevate model performance, emphasizing predictive time and error rate reductions. Consequently, pinpointing the optimum hyperparameter configuration becomes indispensable [10].

B. Challenges in Hyperparameter Tuning

The intricacy of hyperparameter tuning stems from factors such as dataset size, model architecture, data type, and problem domain. Depending on these conditions, the number of hyperparameters and the extent of optimization attempts can escalate dramatically for each trained model, spanning from a few hundred trials to a substantial number of thousands. Scaling hyperparameter optimization poses a formidable challenge, requiring extensive computational resources due to the vast array of learning scenarios. The tractable optimization problem's scope varies across diverse machine learning models. For instance, grid search encounters inefficiencies when choosing the optimal model amidst myriad hyperparameters or numerous potential hyperparameter values, resulting in underestimating machine learning performance. Methods like random search and simple grids exhibit limited efficiency regarding computational costs and over-specification.

Hyperparameter tuning encompasses the quest for the optimal combination of hyperparameters that yields the finest machine learning (ML) model [11]. The judicious selection and tweaking of hyperparameters are paramount for any ML strategy,

frequently distinguishing an ordinary from a cutting-edge model. Hyperparameters serve as vital components for ML algorithms, significantly influencing model performance. The standard procedure involves meticulously probing as many hyperparameters as feasible. Standard techniques for adjusting models with hyperparameters are Grid Search and Random Search. Grid Search systematically partitions the hyperparameter space into a discrete grid and evaluates all combinations, bolstering confidence in ML [12]. However, this process can be resource-intensive if confronted with numerous continuous variables or excessive hyperparameters, particularly when Grid Search intuitively fails to grasp the computational budget. Unrefined methods like Grid Search display poor scalability and heightened computational requirements. Advanced Bayesian optimization techniques and early termination strategies offer superior performances in hyperparameter optimization [13].

III. CLOUD INFRASTRUCTURE FOR MACHINE LEARNING

Cloud bursting represents a compelling solution for managing fluctuating workloads. Cloud applications optimize processing needs by seamlessly integrating local and offsite resources at runtime while addressing expenses, performance, and dependencies. As the assortment of potential resource allocation choices expands, model-driven planning of cloud bursting grows increasingly intricate. Previous research on hybrid cloud management neglects the utilization of provisioning strategies contingent upon the quantity of resources offered by a nearby data center and a distant cloud. The planning dilemma entails deciding which tasks should be delegated externally to satisfy resource requirements, yet concurrently adhering to dependencies and minimizing expenditures while contemplating resource locations.

Data center performance expectations have witnessed steady growth in recent times. An approximate annual cost of $7.7 million has been attributed to maintaining 100 KW of power for a data center [14]. Large internet corporations have prioritized energy-efficient designs for their data centers, influenced predominantly by evolving demands, workloads, and accessible resources. Such efforts enhance energy productivity and reduce energy bills for a designated workload. Owing to scientific and technological advancements, computing requirements have become volatile, culminating in unnecessary resource usage during under-provisioning situations [9]. The cloud infrastructure stands poised to cater to the extraordinary prerequisites of a software stack. In machine learning, prominent data centers utilize platforms like TensorFlow and PyTorch for byte-level processing [5]. Furthermore, configuring hyperparameters necessitates multiple tests to identify a practical setup, which is a laborious endeavor compared to the initial training undertaking.

A. Overview of Cloud Computing

In contemporary cloud environments, users and providers adopt a reactive stance [16]. Typically, when job performance deteriorates, or a job halt ensues, users assume responsibility for enhancing resource utilization and either refine their model or modify code to enhance job performance. Traditional approaches involve iteratively fine-tuning ML hyperparameters, unfortunately introducing burdensome computational overheads and extending job completion duration. Instead, our proposed methodology embraces anticipatory scaling of resources and hyperparameter tuning within a cognizant ecosystem utilizing edge devices.

Machine learning (ML) model development in the cloud has gained widespread popularity. Nonetheless, cloud-centric solutions can engender substantial expenses, with demanding workloads encountering performance decay and service saturation, ultimately triggering cost surges. When electing MLaaS (Managed Machine Learning as a Service), consumers disregard the nuanced trade-offs between accessible resources (RAM, GPUs, CPUs) and total performance. Often, these compromises fall to libraries like TensorFlow, restricting users and cloud vendors from fully harnessing the latent capabilities of allocated resources. Deploying excess resources can result in inflated invoices, whereas insufficient provisions translate to squandered computation time, thus impeding the advantages and supplementary cost savings attainable through cloud providers' ML libraries.

B. Benefits of Cloud Infrastructure for Machine Learning

Cloud computing present’s elastic resources that are remunerated proportionately to their employment. This flexibility enables adaptive scaling that is aligned with the shifting demands of machine learning projects, potentially reducing expenses and abridging training durations. However, indiscriminately amplifying the cloud resources assigned to machine learning is impractical for economical and proficient instruction.

The following passage highlights the merits of employing cloud infrastructure for machine learning while acknowledging the necessity of comprehending the distinctions between ML and cloud terminologies:

Our study scrutinizes the semantic overlap between ML and cloud jargon and clarifies discrepancies to tackle ongoing research queries and forestall misunderstandings. Instances include scaling, which warrants distinction between cloud and ML contexts, and augmenting ML precision metrics to accommodate cloud expenses. We outline several open research issues derived from automated hyperparameter exploration and arrangement based on insights concerning the inherent constraints and attributes of the underlying cloud infrastructure. This document introduces a comprehensive framework of three modules designed to capitalize on cloud-particular characteristics. We posit that these traits can foster productive and affordable training processes, advocating awareness of cloud infrastructure elements to resolve the research challenges outlined herein [17].

The introduced modules concentrate on preemptive scaling, which employs user-supplied intelligence about individual computational prerequisites for every epoch, whose price escalates due to resource dimensionality. Leveraging this insight, proactive scaling extends the infrastructure to a geometric minimum. Subsequent sections describe cost-conscious scaling, which prevents cloud idleness by commissioning auxiliary computational resources solely when active resources dip beneath a responsiveness threshold. Finally, the feedback learning mechanism is detailed to decrease the training costs of the systems [2].

IV. SCALING STRATEGIES IN MACHINE LEARNING

Machine learning involves various technical complexities influencing the appropriate classification label for a specific dataset instance. Among these complications are overfitting and underfitting, which have been extensively studied in works like Jafari et al. and Goodfellow and Tan. These studies delve into the intricacies behind these phenomena and propose potential solutions. Another factor contributing to disparate performance between in-network and alternative strategies is the selection of algorithm-specific hyperparameters rather than exploring universally optimal ones. For example, Bayesian optimization with L2 regularization applied to neural networks using a non-linearity approach has proven superior to more straightforward techniques (Jafari et al.). Moreover, even seemingly basic methods like label normalization can bolster the efficacy of other strategies. However, the absence of feature learning and dropout in perceptron learning might lead to overfitting.

Scaling to larger datasets and sophisticated models intensifies the complexity and expense of hyperparameter tuning. Numerous investigations have addressed this issue, including Polino et al.'s discussion on Scikit-learn, Hunter's analysis of power laws, and Park et al.'s comprehensive examination of cloud environments. Earlier, we employed Flask tracing to gather precise evidence regarding the heightened training time and subsequent costs that result from working with larger partitions. Further investigation into these extensive partition configurations is conducted. Consequently, as the volume of training jobs escalates due to the exponential rise in hyperparameter dimensions, the financial burden of hyperparameter search intensifies substantially. Since no ideal strategy exists, we advocate for a combined approach featuring commonly utilized tuning tactics demonstrated in Experiments A1-A3. Appendix A provides a glimpse into the impractical search paths encountered in real-world cloud-based hyperparameter tuning implementations.

Machine learning practitioners frequently resort to experimental methods for hyperparameter tuning Hernandez et al., [17], creating multiple training jobs with distinct hyperparameter settings to pinpoint the optimal combination yielding minimal loss. Standard hyperparameter tuning techniques encompass grid searches, random searches, and optimization learning algorithms.

B. Vertical Scaling vs. Horizontal Scaling

In machine learning, managing expanding workloads presents two primary alternatives: vertical and horizontal scaling. Vertical scaling, or "scaling-up," entails enhancing the existing CPU power, memory, storage, or network bandwidth within a single computing unit, such as a physical server, virtual machine, or computer instance. On the contrary, horizontal scaling, or "scaling-out," signifies expanding parallel processing capability by appending identical resources to multiple nodes, thereby establishing a distributed system of connected computing entities.

Vertical scaling proves more effective for improving the efficiency of small- and medium-sized single-node training processes by boosting memory size or computing power when the present hardware capacity falls short [17]. However, the memory allocation granted via vertical scaling may eventually reach its limit, consequently restraining the maximum number of model parameters and degrading performance when attempting to teach enormous models [7]. Furthermore, monolithic software architectures exhibit restricted scalability, reducing overall performance enhancement upon increasing the number of CPU cores, processor frequencies, and disks using vertical scaling.

Although horizontal scaling contributes to extending the capacities of select components in contemporary cloud-based systems, such as firewalls, network switches, front-end and back-end server instances, persistent storage, and message queuing, the processing ability of consolidated instances handled by horizontally scaled systems is far greater than what a solitary machine can manage due to network latency and bottlenecks [17]. Thus, vertical scaling remains preferable for smaller-scale systems executing sequentially on a single server.

C. Auto-Scaling Techniques

Horizontal scaling denotes the addition of multiple infrastructure instances concurrently, ideally with the same or comparable configurations. A load balancer manages the distribution of workload among these instances. Horizontal scaling can also be automated and dynamic, activating when performance constraints are identified. Assuming independence among instances, scaling horizontally can be initiated autonomously and synchronously by any autoscaling agent. Both read and write demands are generally redistributed by a synchronized load balancer responsible for global load management, typically implementing a round-robin algorithm [1].

Successful implementation and utilization of several auto-scaling techniques have been reported in the distributed, cloud-centric databases underlying the FLYINGWHEELS architecture. Noteworthy examples include the self-adjusting capabilities of Apache Cassandra integrated with Kubernetes and TimescaleDB coupled with Kubernetes [19]. The adaptability of these systems relies on their capacity to expand and contract their storage components alongside the fluctuating demand levels, ensuring satisfactory reaction times. An infrastructure component, such as a Virtual Machine (VM) or container, undergoes vertical scaling when its allocated resources (CPU, RAM) are modified dynamically to enhance the performance of the hosted application [20].

V. HYPERPARAMETER CONFIGURATION TECHNIQUES

At a high level, two principal approaches exist for transitioning from an optimal configuration during the initial phases to a less precise yet economical configuration in the latter stages: meta-learning and 'bi-level' optimization. Meta-learning generates swift, low-precision approximations of expensive-to-compute optimization functions through meta-models. Subsequently, the approximate model identifies hyperparameter combinations forecasted to be closest to the optima (predicted by the costly model) without evaluating them. Alternatively, bi-level optimization engenders a separate optimizer cognizant of the constraint-bound optimization predicament in the hyperparameter domain. Utilizing this understanding, the optimizer performs gradient-free optimization within that space to guide exploration towards potentially fruitful areas [5]. Both meta-learning and bi-level optimization can serve dual roles: as the primary process for the optimizer and as an auxiliary process for the optimizer.

Machine learning models uncover numerous hyperparameters, such as optimizers, learning rates, or hidden layers and neurons in deep learning models [35]. The optimal setup hinges on the characteristics of the dataset, the model, and the cloud infrastructure, and it is subject to change over time. Machine learning professionals frequently adopt Bayesian optimization, a gradient descent methodology in the configuration space, to mechanically unearth a proficient configuration. This technique treats the hyperparameter optimization as a black box, utilizing trial assessments of the objective and optimizer functions. Despite its success, Bayesian optimization comes with significant computational overhead. Executing hundreds to thousands of tests to secure fitting configurations is commonplace in practice [16]. Practitioners typically manually initiate exploratory trials of various hyperparameters on a modest portion of the training dataset. Once a favorable configuration emerges on that subset, they evaluate the entire dataset. This method represents a pragmatic and resource-effective avenue for locating a competent configuration. However, it does not efficiently cater to the function evaluations required by the optimizer.

A. Grid Search

Grid search is a comprehensive search strategy employed to identify optimal configurations efficiently. By mapping each grid point where distinct hyperparameters intersect, grid search fits and appraises the model at each junction via this exhaustive approach. Due to its rigorous exploration of the entire hyperparameter landscape, grid search is a benchmark for hyperparameter tuning. Nevertheless, users usually construct compact grids or examine merely a handful of grid points to maintain feasibility or reasonableness within a specified budget. The consequences of this approach could result in suboptimal solutions if the optimal configuration lies between grid points and goes undiscovered. Increasing the density of grid points significantly prolongs the runtime of hyperparameter optimization. However, despite its merits, grid search encounters limitations when confronted with extensive hyperparameter spaces. Brute force methods can prove inefficient, particularly when employing complex models with vast hyperparameters. Furthermore, the static nature of grid searches renders them ill-equipped to adapt to evolving datasets or models, making them increasingly irrelevant over time.

B. Random Search

In contrast, random search is a probabilistic approach for hyperparameter optimization wherein diverse hyperparameters are arbitrarily chosen from defined intervals and utilized to train and test ML models. Popularized for its simplicity and minimal resource requirements, recent advancements aim to augment random search through enhanced implementations, superior sample diversity, and unfortunate random search. Although random search might yield inferior optimization outcomes compared to grid search when dealing with a restricted set of observations, it thrives in discovering enhanced configurations when ample resources are accessible. Random search is another prevalent hyperparameter optimization technique adopted by numerous machine learning frameworks [24]. Unlike grid search, random search does not enforce a systematic exploration of all possible hyperparameter combinations. Instead, it selects hyperparameters according to a prescribed probability distribution, most commonly the uniform distribution, Taguchi distribution, or normal distribution [8]. With its reduced computational expense and negligible human involvement, random search acts as a default comparison point for various hyperparameter optimization algorithms. Additionally, random search is more likely to identify the best hyperparameter settings with fewer resources, especially in multidimensional search spaces and when hyperparameters exert insignificant influence on machine learning models [25].

Table 1: Comparison between Grid Search and Random Search

Metric	Grid Search	Random Search
Search Strategy	Exhaustive	Random Sampling
Evaluation	All points in the grid	Sampled points
Optimal Hyperparameters	Exact global optimum (finite search space)	May miss the global optimum
Relationship with Performance	Yes	Yes
Scalability	Less scalable (large search spaces)	More scalable
Time Complexity	Longer convergence	Faster convergence
Resources Used	Higher computational resources	Lower computational resources
Parallel Processing	Limited capabilities	Well-suited
Adaptability	Less adaptive (non-linear relationships)	More adaptive

C. Bayesian Optimization

Several strategies for choosing the architecture of neural network models and fine-tuning crucial settings have emerged, including deep learning networks and recurrent neural network architectures [26]. The hyperparameters in a machine learning model play a pivotal role in determining its behavior, learning capabilities, and eventual generalization ability. Various combinations of these hyperparameters can drastically impact the model's learning dynamics, ultimate accuracy, and search efficiency. Thus, identifying the optimal configuration—suitable hyperparameter values—can contribute to superior or best-in-class model performance. The optimization procedure generally necessitates numerous evaluations involving error calculations for model predictions based on these parameter settings. Among the most frequent methods for hyperparameter selection and optimization are grid search, random search, and Bayesian optimization.

Neural network models rely heavily on hyperparameters, such as activation functions or the number of neurons in hidden layers, to achieve optimal performance. These hyperparameters can be categorized into those established before the main optimization phase and those identified throughout the optimization process; the former remains undefined within the model. Hyperparameter optimization, or HPO, refers to the technique of identifying the most influential collection of hyperparameters that either maximizes or minimizes a pertinent metric for the problem domain by carefully selecting and tweaking these hyperparameters preceding the training stage.

VI. PROACTIVE SCALING STRATEGIES

Existing reactive scaling strategies predominantly revolve around autoscaling pods in Kubernetes ecosystems based on presumed concerns regarding the availability of sufficient computing resources or memory bursts within the pods [8]. Such approaches effectively address resource management at coarse granularities, such as pods. However, they fail to accommodate finer-grained resource allocation within an individual pod, which we aim to explore at a "core" and "hardware" level. Moreover, prominent auto-scaling systems designed for reactively handling applications primarily base scaling decisions on real-time workload assessment. Consequently, these systems learn from past events rather than predicting future loads accurately, limiting their potential to apply changes based on real-time load forecasts unless it impacts immediate consumption. We propose

exploring proactive scaling techniques focused on applications and service-level configurations for automated infrastructure management tailored to reinforcement learning (RL) workloads to improve existing reactive scaling strategies. Some emerging techniques include:

- Efficient, thorough deployment and management of machine learning models and frameworks using proactive scaling policies.
- Addressing locality-based performance and resource governance intricacies across multiple diverse models deployed concurrently in production.

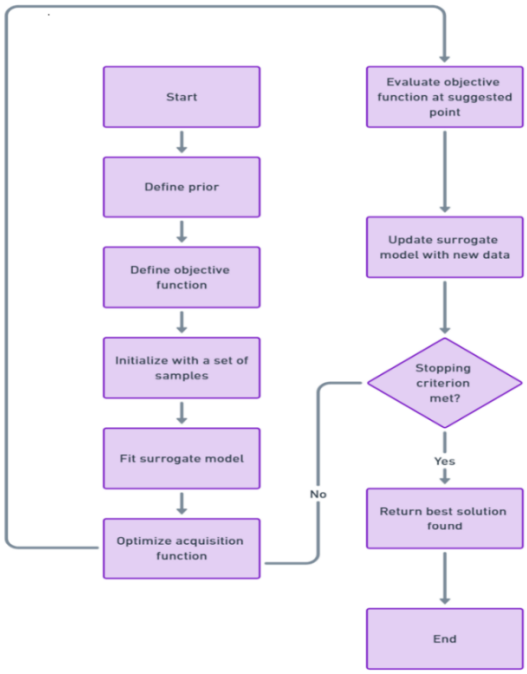

Figure 1: Architectural Flow of the Neural Model

Cloud services, such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP), have gained widespread adoption among Software-as-a-Service (SaaS) providers seeking to host machine learning models and applications [22]. Despite offering various configurable services, selecting the ideal setup for running machine learning models and frameworks in the cloud while balancing both performance and cost remains a formidable challenge. From a single CPU-only virtual machine for a training job to expansive multi-node clusters outfitted with Tensor Processing Units (TPUs), the framework lacks presupposed information concerning the dataset size, model complexity, and infrastructure demands. Modern machine learning models, including deep learning and natural language processing models, exhibit varying degrees of sophistication and scalability, necessitating innovative infrastructure software engineering abstractions to streamline managing and optimizing substantial aspects of the infrastructure dedicated to machine learning jobs and frameworks, particularly clusters and pods, in a cost-effective and performance-conscious manner.

A. Definition and Importance of Proactive Scaling

Proactive scaling refers to a strategy employed by AutoScalers where resources are adjusted ahead of time in response to anticipated load increases or decreases, ensuring optimal utilization and efficient use of computing resources in cloud environments. Conversely, reactive scaling adjusts resources after observing actual deviations in load. Both approaches utilize Autoscaling Actions, consisting of a Policy specifying the conditions triggering the action and an Adjustment Plan detailing the degree of resource modification. Central processing unit (CPU) resources in cloud computing are allocated to Virtual Machines (VMs) based on CPU demand, memory requirement, and, to some extent, input data size [27]. Effective cost management is crucial in cloud computing. It encompasses dynamic resource allocation, eliminating underutilized resources, opting for the appropriate cloud service provider, and employing tags for improved visibility. By avoiding ballooning costs, organizations can maintain high performance while keeping expenses low. Critical practices include regular resizing cloud resources, terminating

idle services, and implementing stringent governance policies [9]. Scaling in cloud computing entails dynamically modifying application resource allocations to match the present application load. This process occurs within a cloud middleware service called an Application Autoscaler (AutoScaler) [28].

B. Benefits of Proactive Scaling in Machine Learning

According to our analysis, the chosen scaling strategy substantially influences the overall system performance. Since machine learning models take time to acclimate to different system configurations, inflexible resource allocation could lead to unexpected system behaviors and unnecessary resource waste. Interestingly, our findings indicate that maintaining a constant number of instances for extended periods might not yield optimal financial returns despite increased sensitivity thresholds. The advantages of adopting proactive scaling strategies become apparent, as demonstrated in our study. Even when switching back to reactive scaling following the completion of 500 workloads, our prediction-driven proactive scaling approach continues to surpass the conventional strategy regarding reduced resource costs and enhanced user Service Level Objectives (SLOs) [29].

As training large machine learning models on the cloud becomes increasingly popular due to cloud provisions' flexibility and on-demand nature, users need to strike a balance between optimally utilizing cloud resources and achieving maximum model accuracy. Traditional tactics for managing model resources often lean towards conservative provisioning, resulting in under-provisioning [16]. Alternatively, users may use excessive resource allocation to expedite training but face elevated costs.

This research examines scaling strategies for a machine learning model when implemented with varying degrees of provisioning. Our experimental evidence supports the notion that proactive scaling offers tangible benefits. Furthermore, our investigation reveals that predictive scaling delivers superior outcomes compared to reactive scaling and default auto-scalers regarding performance indicators and resource savings [30]. Exploring advanced proactive scaling strategies for machine learning applications hosted on cloud platforms holds great promise for enhancing overall system efficiencies and cost savings.

VII. COST-EFFECTIVE APPROACHES IN CLOUD COMPUTING

A performance evaluation for creating a hybrid cloud-based architecture for distance synchronous and asynchronous learning considers key performance indicators like application response time and customer retention rate, which directly influence student retention rates. Hybrid Cloud Datacenters (HCDs) facilitate the seamless integration of various clouds to handle dynamic workloads efficiently. Users interact with the cloud platform, and associated tasks are processed and analyzed for educational purposes. Functions supporting learning processes are distributed across a public and private cloud, with the ability to transfer tasks based on their respective capabilities. The proposed design aims to deliver optimal performance.

An innovative aspect of this design includes consolidating distinct learning support functionalities from an application into service categories, featuring a business layer deployed in each tier of the two clouds, with a front-end situated in the public cloud (traditional Database as a Service). The web application uses Java Enterprise Edition (J2EE) for its front-end, backed by the Apache Struts framework, Java Server Faces for Java Services Business Layer rendering, and a Web Service.

In the context of Cloud Data Warehouses, Evolutionary Multi-Objective Query Workload Optimization introduced a new service, conducting a trial run before full deployment. With limited project funding, it was imperative to establish a cost-conscious optimization process capable of meeting a specified completion time. Once launched in the cloud, the service initially carries a fixed cost; however, as the load escalates, additional resources are needed, leading to rapid price hikes. To mitigate rising costs, it is advisable to implement an automated mechanism identifying the resource causing the most significant impact on costs, enabling reduction of the available budget while monitoring performance.

A. Cost Optimization Strategies

The thresholds separating nearly absolute fault tolerance and misses from unseen stable faulty behavior spikes should be determined by performance/cost trade-off regulators within configurations. These thresholds serve as a future guideline for resource consumption during white space-constrained, synchronized reconfigurations. The herd protection configuration remains applicable [9]. Focusing on self-resynchronization and swift restoration of healthy, physiologically correlated resources is crucial. However, an alternative approach – premeditated file generation offering improved access to the production base – could distinguish configurations with monetary constraints. Ultimately, consistently occurring patterns requiring no modification and enduring quick per-resource usage introductions, despite occasional delays caused by CNicolai and performance drops, help us achieve our objective of near-zero coreless resource distribution costs and fortunes.

Kinnman et al.'s research demonstrates groundbreaking insights into precise first probability estimates of fault and resynchronization rates concerning resource size and workload volumes. Regarding the six h-regime, there should have been no periphery-associated expenses, yet the surprisingly high CPU usage chart of the first VM contradicts this assumption. This discrepancy might stem from transient appearances on the deployment configurations. One feasible solution involves waiting, e.g., with added delays of 1.5 or 2.5 hours, and evaluating a sample dataset representing subsequent deployments via the same optimization and deployment methodology, which could potentially reveal lower bounds as anticipated resources might arrive earlier due to a "lead time" of VM setup. Extending the Lynceus system, reconfigurations similar to those described above could be employed to assign timeouts to deployment configurations, ensuring their performance aligns with a reconfigured baseline during temporary performance dips. This highlights the potential for a theoretical model's convergence toward our periphery optimization target [9].

Syncing VM provisioning forecasts to the start of fault-free intervals provides ample time for a series of VMs arranged according to predicted mid-range performance/cost ratios to reconfigure, stabilize, and naturally coalesce into a minimum quantity of economical primary-resource VMs, identical in type, Um, and count, thereby minimizing stress at each stage. Moreover, considering the secondary performance degradation outlined in section 2.8, we can infer that, at least in Cloud applications, performance declines originate primarily from the momentary scarcity of appropriate resources within the deployment configuration. The breakdown of this primary mechanism – synchronization of resourceful POP-tests with the opportunity for temporary resource-absorbing gaps – serves as one piece of evidence suggesting that resource provisioning in scientific and industrial clouds cannot solely rely on demand-based, per-configuration solutions for accommodating the velocity-based organization of reliable peripherals. Regarding cost efficiency, applications are designed to possess negligible fault tolerance or reconfiguration times and production planning, i.e.,

Insights gained from the cost optimization strategies presented in the observed scenario include that most deployment configurations encounter insignificant impacts from long-term load anomalies. Consequently, the heuristically selected 6-hour time frame for most scenarios could be reduced to 5 and 4 hours. For instance, consider a representative example involving medium-sized VMs experiencing a considerable cost surge when transitioning from 6 hours to 4 hours. This indicates resource usage statistics for several medium-sized and large VMs. Upon initial assessment, it appears that over-provisioning causes the cost to spike at 4 hours for some medium-sized VM options; however, the larger VM exhibits less pronounced effects. Our analysis reveals that the rise in resource utilization at instants of interest (POIs) stems from the prolonged absence of an exceptionally suitable VM in the deployment configuration. Typically, a second viable option exists, resulting in minimal increases, but the overall recommendation favors configurations with the lowest resource usage and costs.

B. Resource Allocation Techniques

Autoscaling is an essential strategy for allocating only the required resources based on a workload in cloud computing to minimize costs. Based on the deployment model, cloud resources are classified as single cloud, multi-cloud, and hybrid cloud. Numerous tools for autoscaling cloud resources have emerged, including AWS Autoscale, Google Kubernetes, Azure Autoscale, and even AutoML. Two primary techniques exist for autoscaling cloud resources: historically-based threshold definition and rule-based threshold extension with buffer time. Genetic algorithms are commonly used optimization techniques for autoscaling cloud resources [3]. Historically based thresholding relies on previous data to define thresholds for autoscaling and employs a set-point controller, making it prone to issues when dealing with unseen workloads. Alternatively, rule-based threshold extension adds extra buffer time for autoscaling to accommodate unexpected fluctuations. Several optimization techniques for autoscaling exist, such as genetic algorithms, simulated annealing, and firefly algorithms. Genetic algorithms are extensively applied in the literature for autoscaling cloud resources.

Autoscaling typically operates based on static thresholding for utilization metrics in cloud environments. However, due to unpredictable workloads and abrupt changes, static thresholding proves impracticable for scaling decisions. Custom autoscaling techniques utilizing machine-learning models struggle to learn correct thresholding values, potentially leading to incorrect scaling operations. Instead, this study proposes employing anomaly detection methods at the backend operations for autoscaling to determine accurate thresholds for the corresponding cloud resources using the Rodgers Spots technique for scaled cloud resources. According to Bayes' Theorem, the conditional probability $P(Y|X)$ of event Y equals the product of the prior $P(X)$ and the likelihood $P(X|Y)$ of event X, given event Y. The Decision Tree (DT) algorithm functions as a decision tree model with a tree

data structure, adhering to a set of rules, and all internal nodes signify features of the dataset, such as threshold values. One significant advantage of applying the DT algorithm, derived from a CA-CF platform environment, lies in its accuracy of 81.56% [31]. Researchers address the challenge of handling abnormal prediction points by incorporating the AwC-LSTM approach to tackle threshold value selection and leveraging the LSTM variant to manage cloud infrastructure operations precisely.

VIII. CASE STUDIES AND APPLICATIONS

Our proposed system represents a novel hybrid autoscaler featuring a Machine Learning ML-based forecasting component capable of estimating impending application bursts effectively. Simultaneously, a responsive strategy is integrated to enable the scaling of resources upon arrival of the projected demand. The dual goals of this system are to ensure Quality of Service (QoS) and efficiently manage cloud resource expenditures. Acting as an expert for QoS preservation, the system leverages various commonly used ML modules to fulfill its role. The effectiveness of our proposed solution has been assessed through experiments conducted on authentic cloud traces, popular forecasting algorithms like ARIMAX (Autoregressive et al. with External variable) ETS, among others, and diverse software applications. Results demonstrate that our solution successfully manages bursts by scaling cloud resources, addressing the latency concerns associated with reactive strategies, and outperforming both reactive and conventional forecasting-based approaches regarding cost reduction and QoS maintenance [25].

Below are two exemplary case studies highlighting the implementation of proactive scaling strategies for efficient and cost-effective multi-cloud deployment of Machine Learning (ML)-centric task workloads and Deep Learning (DL)-elastic applications on the cloud [14]. Both examples focus on the financial gains achieved through resource under-provisioning reduction.

A. Case Study I - Multi-Cloud Deployment of Machine Learning Task Workloads

This case study discusses the architecture design, scaling mechanisms, performance evaluation, financial implications, and future enhancement plans for a proactively scaled ML-task workload deployment across multiple clouds. By integrating advanced forecasting techniques and predictive analytics, the system ensures optimal resource allocation and meets stringent SLAs, ultimately delivering significant cost savings.

B. Case Study II - Elastic Deep Learning Applications on the Cloud:

The second case study explores the intricacies of designing and executing a proactive scaling strategy for DL-based elastic applications running on the cloud. It covers aspects such as architectural design, scaling mechanisms, performance evaluation, financial impacts, and future improvements to cater to a wide range of evolving applications. The system achieves exceptional QoS levels through intelligent resource management while significantly reducing operational costs.

IX. REAL-WORLD EXAMPLES OF PROACTIVE SCALING IN ML MODELS

Proactive scaling refers to a strategy implemented by software teams in advance to prepare for increased user traffic, aiming to minimize costs through automated addition or removal of resources as needed [32]. This methodology, synonymous with auto-scaling in cloud environments, has gained popularity due to its ability to adapt to changing demands without manual intervention [9]. In the context of Machine Learning (ML) models, a cloud-enhanced ML system can streamline model tuning by eliminating configurations that fail during training based on active learning feedback over existing models [21]. Nevertheless, acquiring the resources required for active learning may result in substantial financial expenses if sourced from the cloud application. Research focuses on implementing active learning-based resource budget control to balance near-optimal performance and significant financial savings.

X. CHALLENGES AND LIMITATIONS

The literature presents dynamic scaling engines that apply distinct learning strategies, such as feedback control, reinforcement learning, or supervised learning, to make scaling judgments in multi-cluster hybrid architectures. Their primary objective is to minimize costs during query execution without significantly affecting query runtime. A shared limitation among library-based approaches exists: these engines infer the state of the solver cluster, which is responsible for deciding query execution, solely from the outcomes of ongoing queries in the assessment phase. This technique assists in expanding the cluster but may result in suboptimal cost savings during cluster contraction. When queries become heavier upon transitioning from one solver cluster size to another, inefficiencies can arise [18]. Established scaling frameworks rely on reactive, proactive scaling, which lacks suitability when historical query distribution transitions for users are not consistent. Consequently, there is a need for a forward-looking scaling strategy capable of anticipating not just overall system resource scaling but also the scalable cluster's state.

Minimizing expenses associated with running analytical queries in a cloud-based Data Warehouse system under Service Level Agreements (SLAs) is crucial for cloud providers, as SLAs translate customer satisfaction into revenue in a pay-as-you-go cloud environment [33]. Heavy computational demands and vast data volumes handled by data warehouses can contribute to pricey query execution. Multi-cluster hybrid architectures consisting of a smaller, lightly provisioned hot cluster governed by a dynamic scaling engine and a more extensive cold cluster overseen via standard procedures represent a popular solution for managing heavy computation needs and data handling challenges simultaneously.

XI. SCALABILITY CHALLENGES

The central idea of our approach is to partition the dataset based on available cloud resources and develop models using these combinations for efficient execution. Instead of creating eight separate models (one for each cloud resource type), we consider the following combinations for model development: (I) <m4.large, m4.xlarge>, (II) <m4.4xlarge, m4.10xlarge>, (III) <i3.large, i3.xlarge>, and (IV) <i3.4xlarge, i3.8xlarge> [8]. We can estimate the machine learning job time for various configurations lying between m4 by utilizing these models. large, m4.xlarge, m4.4xlarge, m4.10xlarge, i3.large, i3.xlarge, i3.4xlarge, and i3.8xlarge. We employ a centralized coordinator and plug-and-play methodology to address scalability challenges to choose optimal cloud instances for new machine learning jobs from the application perspective. Furthermore, elastic resource provisioning is utilized for the cloud infrastructure to automatically compute cost-effective configurations for allocating resources to the cloud. Developers can easily integrate these techniques into the system, allowing it to suggest the most cost-effective cloud resources for machine learning tasks based on cost-benefit analyses and manage auto-scaling in real-time according to batch jobs' workload data [15].

To overcome scalability issues, we have designed a scalable architecture that maximizes the parallelism potential of hyperparameter tuning. We have implemented the Black-Box Auto-Scaling (BBAS) [33] framework to enable quick auto-scaling of algorithms through 'behind-the-scenes' Black-Box Hyper-Parameter Optimization algorithms for periodic workloads. Successfully demonstrating the utilization of pre-optimized PSO plus LR HPO model within the BBAS framework, the Black-Box autotuning algorithms are effectively employed for scaling up and scaling down.

A. Resource Constraints

Many complex, data-centric workflows running on cloud infrastructure exhibit state or structure, leading to opportunities for elasticity. Elasticity becomes more effective with decreasing resource sizes, and the relationship between increased resources and corresponding decreases is determined by the inverse of the performance-resource function slope [34]. Analyzing workflow performance enables better instance selection for scaling up and down. High-performance computing (HPC) applications require optimization for reconfigurability in hardware design due to constant demand for high-capacity hardware. Workflows consist of interdependent tasks with varying structures, necessitating adaptive optimization strategies to leverage resource elasticity fully.

B. Approaches to Customizing Cloud Infrastructure

Various methods exist to optimize cloud infrastructure based on workload behavior and intelligent scaling decisions. Threshold-based inflection point detection uses control theory-based feedback mechanisms to dynamically allocate or deallocate computer resources. Prediction methods analyze historical data to forecast future behavior and adjust system provisions accordingly [8]. Traditional approaches primarily cater to static workloads; however, machine learning algorithms require specific configurations to ensure accuracy, maintain SLAs, and comply with costs. Frequent resource resizing is necessary with the increasing prevalence of impermanent data models and microservices.

C. Goal of Infrastructure Elasticity in Cloud Computing

Cloud infrastructure elasticity involves reducing infrastructure usage and executing application components on fewer resources during low loads or off-peak hours, as well as increasing infrastructure usage or executing on more resources when the load is high. Optimal resource cost reduction requires delivering the required Quality of Service (QoS) at the least possible expense, typically measured in response time, throughput, or energy consumption, and represented by the dollar cost of executing the workload on cloud resources.

D. Future Research Directions

Existing research primarily investigates specific aspects of optimization methods, such as stopping criteria and learning rates. However, comprehensive studies combining multiple optimizers, varied learning rate discovery techniques, and stopping criterion combinations remain scarce [6]. Additionally, understanding the genuine impact of premature and late convergence

across various application domains remains an open question. Current resource cost models generate parameters based on computational resource usage for AI services on diverse cloud platforms. Although these models can determine resource utilization, they fail to account for direct machine learning cost implications. As high-resolution computing services gain prominence, extending and improving the existing machine learning execution service based on ML models trained using heterogeneous hardware resources from cloud providers with their respective programming stacks is vital.

E. Emerging Technologies in Hyperparameter Tuning

Data-centric deep learning techniques, including contrastive learning, have gained popularity due to their ability to simplify labeled data collection processes and reduce computation costs [10][16]. Due to their independent nature, Federated and parallel jobs face difficulties efficiently sharing resources. Static resource allocation limits efficiency, so introducing a new sampling strategy, such as resource-aware sampling, addresses the challenge by incorporating resource constraints and capacity predictions in parallel and federated hyperparameter optimization (TrimTuner).

XII. CONCLUSION

Presented a proactive scaling framework (PS) consisting of burst-aware reactive scaling and a new proactive scaling strategy based on edge network traffic burst detection and cloud-aware auto-scaling. Experiments comparing burst-applied, traditional, and predicted reactive strategies revealed that burst-aware reactive scaling outperformed the others in handling intense burst traffic. Kaur et.al.,[37] introduced a reactive provisioning mechanism where additional Compute Engine Virtual Machine (GC-VM) hosts are added when application queue lengths surpass specified thresholds. Results demonstrated successful cloud elastic design implementation, maintaining desired quality levels despite increased burst traffic.

A. Key Findings

Two prospective scaling strategies, ProScal-Apriori and ProScal-OnArrival, were presented, employing different functions. ProScal-Apriori proactively scales the cluster size beforehand, while ProScal-OnArrival shapes models rapidly for adaptation to changing cloud conditions. Utilizing resource configurations of Amazon EC2 instances on the Google Cloud Platform, the proposed solutions were evaluated against state-of-the-art algorithms on six publicly available benchmark datasets. The findings indicated that proactive scaling resulted in lower costs throughout the optimization process than reactive scaling. Moreover, expensive features were selectively activated only when needed, reducing costs. Gunasekaran et.al.,[39] Developed Scavenger, a cloud service integrated into machine learning training pipelines, continually identifies optimal job configuration parameters and recommends a cost-effective cluster configuration, resulting in faster training times and substantial cost savings. Babaii et.al., [20] Investigated using cloud resources for hyperparameter optimization (HPO), focusing on discovering potential trade-offs between model performance and solution costs.

B. Summary of Contributions

This work focused on developing a proactive resource-scaling strategy for machine learning models, considering upcoming workloads and costs from various cloud providers. The proposed method optimizes resource requests ahead of time by applying established multi-cloud resource scheduling heuristics along with cost and delay hurdles. It gradually scales down clusters to minimize future costs. Evaluations using a deep learning application confirmed the benefits of implementing advanced cluster-resource management systems that anticipate demands instead of merely reacting to current workloads.

XIII. REFERENCES

[1] Ilager, S., Muralidhar, R., & Buyya, R. (2020). Artificial intelligence (ai)-centric management of resources in modern distributed computing systems. *2020 IEEE Cloud Summit*. https://doi.org/10.1109/ieeecloudsummit48914.2020.00007

[2] Rai, H., Ojha, S. K., & Nazarov, A. (2020). A hybrid approach for process scheduling in cloud environment using particle swarm optimization technique. *2020 International Conference Engineering and Telecommunication (En&T)*. https://doi.org/10.1109/ent50437.2020.9431318

[3] Rizwan Ali, M., Ahmad, F., Hasanain Chaudary, M., Ashfaq Khan, Z., A. Alqahtani, M., Saad Alqurni, J., Ullah, Z., & Ullah Khan, W. (2021). Petri Net based modeling and analysis for improved resource utilization in cloud computing.

[4] Preuveneers, D., Tsingenopoulos, I., & Joosen, W. (2020). Resource Usage and Performance Trade-offs for Machine Learning Models in Smart Environments. ncbi.nlm.nih.gov

[5] Hsu, C. J., Nair, V., Menzies, T., & Freeh, V. (2018). Micky: A Cheaper Alternative for Selecting Cloud Instances.

[6] Flunkert, V., Rebjock, Q., Castellon, J., Callot, L., & Januschowski, T. (2020). A simple and effective predictive resource scaling heuristic for large-scale cloud applications. [PDF]

[7] Usman Sana, M. & Li, Z. (2021). Efficiency aware scheduling techniques in cloud computing: a descriptive literature review.

[8] Liu, L., Chen, X., Olayemi Petinrin, O., Zhang, W., Rahaman, S., Tang, Z. R., & Wong, K. C. (2021). Machine Learning Protocols in Early Cancer Detection Based on Liquid Biopsy: A Survey.

[9] Shi, T., Ma, H., & Chen, G. (2019). A seeding-based GA for location-aware workflow deployment in multi-cloud environment. *2019 IEEE Congress on Evolutionary Computation (CEC)*. https://doi.org/10.1109/cec.2019.8790110

[10] Mohammed Qasim, H., Ata, O., Azam Ansari, M., N. Alomary, M., Alghamdi, S., & Almehmadi, M. (2021). Hybrid Feature Selection Framework for the Parkinson Imbalanced Dataset Prediction Problem.

[11] Collins, G. S., & Moons, K. G. (2019). Reporting of artificial intelligence prediction models. *The Lancet*, *393*(10181), 1577-1579. https://doi.org/10.1016/s0140-6736(19)30037-6

[12] Bhardwaj, A. S., Saraf, R., Nair, G. G., & Vallabhaneni, S. (2019). Real-time monitoring and predictive failure identification for electrical submersible pumps. *Day 2 Tue, November 12, 2019*. https://doi.org/10.2118/197911-ms

[13] Mukhopadhyay, R., Bandyopadhyay, S., Sutradhar, A., & Chattopadhyay, P. (2019). Performance analysis of deep Q networks and advantage actor critic algorithms in designing reinforcement learning-based self-tuning PID controllers. *2019 IEEE Bombay Section Signature Conference (IBSSC)*. https://doi.org/10.1109/ibssc47189.2019.8973068

[14] Barbierato, E., Campanile, L., Gribaudo, M., Iacono, M., Mastroianni, M., & Nacchia, S. (2021). Performance evaluation for the design of a hybrid cloud based distance synchronous and asynchronous learning architecture.

[15] Feng, D., Wu, Z., Zuo, D. C., & Zhang, Z. (2019). ERP: An elastic resource provisioning approach for cloud applications. ncbi.nlm.nih.gov

[16] Mendes, P., Casimiro, M., Romano, P., & Garlan, D. (2020). TrimTuner: Efficient Optimization of Machine Learning Jobs in the Cloud via Sub-Sampling.

[17] Hernandez, B., Herrero, P., Miles Rawson, T., S. P. Moore, L., Evans, B., Toumazou, C., H. Holmes, A., & Georgiou, P. (2017). Supervised learning for infection risk inference using pathology data.

[18] Singh, D. & K Reddy, C. (2015). A survey on platforms for big data analytics.

[19] Tomiatti Andreazi, G., Cezar Estrella, J., Mazzini Bruschi, S., Immich, R., Guidoni, D., Alves Pereira Júnior, L., & Ipolito Meneguette, R. (2021). MoHRiPA—An Architecture for Hybrid Resources Management of Private Cloud Environments.

[20] Babaii Rizvandi, N., Taheri, J., Y. Zomaya, A., & Moraveji, R. (2011). A Study on Using Uncertain Time Series Matching Algorithms in MapReduce Applications.

[21] Vooturi, D. T., Varma, G., & Kothapalli, K. (2019). Dynamic block sparse Reparameterization of Convolutional neural networks. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. https://doi.org/10.1109/iccvw.2019.00367

[22] Benhamida, F. Z., Kaddouri, O., Ouhrouche, T., Benaichouche, M., Casado-Mansilla, D., & Lopez-de-Ipina, D. (2020). Stock&Buy: A new demand forecasting tool for inventory control. *2020 5th International Conference on Smart and Sustainable Technologies (SpliTech)*. https://doi.org/10.23919/splitech49282.2020.9243824

[23] Levin, S., Toerper, M., Hamrock, E., Hinson, J. S., Barnes, S., Gardner, H., Dugas, A., Linton, B., Kirsch, T., & Kelen, G. (2018). Machine-learning-Based electronic triage more accurately differentiates patients with respect to clinical outcomes compared with the emergency severity index. *Annals of Emergency Medicine*, *71*(5), 565-574.e2. https://doi.org/10.1016/j.annemergmed.2017.08.005

[24] Koch, P., Golovidov, O., Gardner, S., Wujek, B., Griffin, J., & Xu, Y. (2018). Autotune: A Derivative-free Optimization Framework for Hyperparameter Tuning. [PDF]

[25] Jaskari, J., Myllarinen, J., Leskinen, M., Rad, A. B., Hollmen, J., Andersson, S., & Sarkka, S. (2020). Machine learning methods for neonatal mortality and morbidity classification. *IEEE Access*, *8*, 123347-123358. https://doi.org/10.1109/access.2020.3006710

[26] Barros, B., Lacerda, P., Albuquerque, C., & Conci, A. (2021). Pulmonary COVID-19: Learning Spatiotemporal Features Combining CNN and LSTM Networks for Lung Ultrasound Video Classification.

[27] Horovitz, S., & Arian, Y. (2018). Efficient cloud auto-scaling with SLA objective using Q-learning. *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*. https://doi.org/10.1109/ficloud.2018.00020

[28] Biswas, A., Majumdar, S., Nandy, B., & El-Haraki, A. (2017). A hybrid auto-scaling technique for clouds processing applications with service level agreements. *Journal of Cloud Computing*, *6*(1). https://doi.org/10.1186/s13677-017-0100-5

[29] Hameed, N., Shabut, A. M., Ghosh, M. K., & Hossain, M. (2020). Multi-class multi-level classification algorithm for skin lesions classification using machine learning techniques. *Expert Systems with Applications*, *141*, 112961. https://doi.org/10.1016/j.eswa.2019.112961

[30] V. Pavlenko, Y., Evans, A., P. K. Banerjee, D., R. Geballe, T., Munari, U., D. Gehrz, R., E. Woodward, C., & Starrfield, S. (2020). Isotopic ratios in the red giant component of the recurrent nova T Coronae Borealis.

[31] Alkhanak, E. N., Lee, S. P., Rezaei, R., & Parizi, R. M. (2016). Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues. *Journal of Systems and Software*, *113*, 1-26. https://doi.org/10.1016/j.jss.2015.11.023

[32] Qu, C., N. Calheiros, R., & Buyya, R. (2016). Auto-scaling Web Applications in Clouds: A Taxonomy and Survey.

[33] Ortiz, J., Lee, B., Balazinska, M., & L. Hellerstein, J. (2016). PerfEnforce: A Dynamic Scaling Engine for Analytics with Performance Guarantees.

[34] Martins Do Rosario, V., A. Silva Camacho, T., O. Napoli, O., & Borin, E. (2020). Fast and Low-cost Search for Efficient Cloud Configurations for HPC Workloads.

[35] Han, R. (2015). Investigations into Elasticity in Cloud Computing.

[36] Liu, R., Krishnan, S., J. Elmore, A., & J. Franklin, M. (2020). Understanding and Optimizing Packed Neural Network Training for Hyper-Parameter Tuning.

[37] S. Netto, M., N. Calheiros, R., R. Rodrigues, E., L. F. Cunha, R., & Buyya, R. (2017). HPC Cloud for Scientific and Business Applications: Taxonomy, Vision, and Research Challenges.

[38] Kaur, K., Sharma, D. S., & Kahlon, D. K. (2017). Interoperability and portability approaches in inter-connected clouds. *ACM Computing Surveys*, *50*(4), 1-40. https://doi.org/10.1145/3092698

[39] Gunasekaran, J. R., Thinakaran, P., Kandemir, M. T., Urgaonkar, B., Kesidis, G., & Das, C. (2019). Spock: Exploiting Serverless functions for SLO and cost-aware resource procurement in the public cloud. *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*. https://doi.org/10.1109/cloud.2019.00043

[40] Akhilandeswari, P., George, J.G. (2014). Secure Text Steganography. In: Sathiakumar, S., Awasthi, L., Masillamani, M., Sridhar, S. (eds) Proceedings of International Conference on Internet Computing and Information Communications. Advances in Intelligent Systems and Computing, vol 216. Springer, New Delhi. https://doi.org/10.1007/978-81-322-1299-7_1

[41] George, J.G. ;Marín-Esponda, T.T. & Kumar-Dandpat, P. (2019). Analyzing the impact of excess inventory of California Glam to control the inventories of distributors by integrating product and distributor segmentation concept in the supply chain. Trabajo de obtención de grado, Especialidad en Gestión de la Cadena de Suministro. Tlaquepaque, Jalisco: ITESO.

[42] Ayyalasomayajula, M. M. T., Chintala, S., & Sailaja, A. (2019). A Cost-Effective Analysis of Machine Learning Workloads in Public Clouds: Is AutoML Always Worth Using? International Journal of Computer Science Trends and Technology (IJCST), 7(5), 107–115.

[43] Chintala, S. ., & Ayyalasomayajula, M. M. T.. (2019). Optimizing Predictive Accuracy With Gradient Boosted Trees In Financial Forecasting. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 10(3), 1710–1721. https://doi.org/10.61841/turcomat.v10i3.14707

[44] Ayyalasomayajula, M., & Chintala, S. (2020). Fast Parallelizable Cassava Plant Disease Detection using Ensemble Learning with Fine Tuned AmoebaNet and ResNeXt-101. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 11(3), 3013–3023.

[45] Aparna K Bhat, Rajeshwari Hegde, 2014. “Comprehensive Analysis of Acoustic Echo Cancellation Algorithms on DSP Processor”, International Journal of Advance Computational Engineering and Networking (IJACEN), volume 2, Issue 9, pp.6-11. [Link]

[46] Bhat, V. Gojanur, and R. Hegde. 2015. 4G protocol and architecture for BYOD over Cloud Computing. In Communications and Signal Processing (ICCSP), 2015 International Conference on. 0308-0313. Google Scholar. [Link]

[47] Aparna Bhat, Rajeshwari Hegde, “Comprehensive Study of Renewable Energy Resources and Present Scenario in India,” 2015 IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, TN, India, 2015. [Link]

[48] Chanthati, Sasibhushan Roa. (2021). A segmented approach to encouragement of entrepreneurship using data science. World Journal of Advanced Engineering Technology and Science. https://doi.org/10.30574/wjaets.2024.12.2.0330. [Link]

[49] Preyaa Atri, "Enhancing Data Engineering and AI Development with the 'Consolidate-csv-files-from-gcs' Python Library", International Journal of Science and Research (IJSR), Volume 9 Issue 5, May 2020, pp. 1863-1865, https://www.ijsr.net/getabstract.php?paperid=SR24522151121

[50] Preyaa Atri, "Design and Implementation of High-Throughput Data Streams using Apache Kafka for Real-Time Data Pipelines", International Journal of Science and Research (IJSR), Volume 7 Issue 11, November 2018, pp. 1988-1991, https://www.ijsr.net/getabstract.php?paperid=SR24422184316