National College of Ireland

Serverless AI: Leveraging Cloud Functions For GPU-Optimized Machine Learning Deployment and Comparative Analysis with Traditional Methods

MSc Research Project
Ms Cloud Computing

Muhammad Qamar Chaudhry
Student ID: 23198028

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Muhammad Qamar Chaudhry
Student ID:	X23198028
Programme:	MSc Cloud Computing **Year:** 2024 - 2025
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	24/04/2025
Project Title:	Serverless AI: Leveraging Cloud Functions For GPU-Optimized Machine Learning Deployment and Comparative Analysis with Traditional Methods.
Word Count:	5885 **Page Count** 21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: QAMAR CHAUDHARY

Date: 24/04/25

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Serverless AI: Leveraging Cloud Functions For GPU-Optimized Machine Learning Deployment and Comparative Analysis with Traditional Methods

Muhammad Qamar Chaudhry

23198028

ABSTRACT

The fusion of serverless computing and GPU-accelerated machine learning (ML) marks a new approach in AI deployment at the cloud level. Proposed strategies incorporating virtual machines or containers seem to struggle with scaling, cost efficiency, management overhead, and infrastructure. This work analyzes the hybrid serverless architecture based on AWS Lambda and API Gateway for managing inference workloads using GPU-backed EC2 instances. The focus of the research is on two machine learning models - cyberbullying detection model with ML algorithms and an object detection model - YOLOv8, monitoring latency, resource consumption, processing time, and cost. The findings illustrate that lightweight models perform consistently under serverless configurations, but heavy workload models take advantage of dynamic offload to GPU-backed EC2 instances. In addition to facing latency spikes and inconsistent resource utilization, delay-sensitive computing GPU models exhibit tremendous resource needs. Nevertheless, the hybrid model achieves some balance of diminishing returns with performance and cost. The study illustrates the capability of such serverless architectures to configure with lower responsiveness for modern AI workloads, increasing the potential to enable these approaches with suitable requirements.

Keywords: Serverless Computing, Machine Learning Deployment, GPU Acceleration, AWS Lambda, Cloud Computing, Hybrid Architecture, Cost Optimization, Inference Performance, YOLOv8, Cyberbullying Detection.

1 Introduction

Over the past few years, technologies like artificial intelligence and machine learning have progressed greatly. They have caused changes in many industries like medicine, finance, self-driving vehicles, and language understanding. Usually, the implementation of a machine learning model meant having the infrastructure to support it, at the very least a VMs or containers, which involves intricate choreography, substantial spending, and

difficulty scaling. With the need for more agile, low-cost, and easily adaptable solutions, serverless computing has emerged as a pioneering paradigm in cloud computing(Liu et al., 2022).

Combining serverless computing models with AI/ML workflows is referred to as Serverless AI(Shafiei et al., 2019). It eliminates the need for supervision of infrastructure resources by enabling developers to put their code into cloud functions which scales automatically depending on the demand. Amazon Web Services Lambda, Google Cloud Functions, and Azure Functions are examples of vendors who offer serverless environments capable of supporting GPU acceleration. AI models that were once complicated to integrate into event-driven architecture are now within reach(Nagar et al., 2024).

This study analyzes the opportunity of deploying machine learning models utilizing GPU acceleration through serverless cloud functions in comparison to other deployment modalities. The research evaluates how the recent progress in serverless architecture, especially with GPU resources, alters the deployment model of modern AI applications in contexts that require rapid scaling and immediate response.

1.1 Background

Recent developments in deep neural networks (DNNs) have sparked significant interest in Machine Learning (ML). These models are extremely powerful, but at the same time, they are highly resource demanding during both training and inference phases. Typically, ML applications are hosted on virtual machines or containerized architectures that provide a high level of control and flexibility(Kandavel, 2024). However, these approaches often require a certain degree of infrastructure management, which alongside the persistent operational expense remains problematic.

In contrast, serverless computing offers a more effective option for hosting ML workloads. This new paradigm of cloud computing enables developers to run their code without provisioning or managing cloud servers. Functions are invoked and paid for according to actual consumption which leads to financial and scalability benefits. Serverless models are especially suited for low-cost, low-latency inference scenarios, real-time processing, as well as event-driven architectures where workloads are sporadic.

Regardless of its advantages, serverless computing has always been restricted by short execution timeouts, limited memory and CPU resources, and, most importantly, the absence of GPU acceleration(Golec et al., 2024). Such constraints are particularly detrimental to ML applications, especially those involving large models or demanding low-latency responses. This is the primary reason why serverless platforms are underused in high-performance computing environments.

There is an emerging trend toward enhancing serverless platforms to accommodate the growing demands of ML workloads, particularly with GPU-enabled serverless runtimes and relaxed execution time limits. These innovations, along with other advancements, are starting to resolve the issues of performance requirements and the serverless mode(Bhoyar et al., 2024)l. As these changes develop, it could make serverless computing a viable, and even preferable, option for deploying a wide variety of ML applications, owing to its scalability, cost-effectiveness, and user-friendliness.

1.2 Research Motivation

With the rise of artificial intelligence (AI) in sectors like healthcare, finance, and even autonomous systems, the need for efficient computing resources has risen tremendously. To perform modern AI workloads, GPUs have become the standard, drastically increasing the speed of deep learning training and inference. However, deploying these AI workloads on VMs or GPU servers is challenging due to infrastructure costs, resource overutilization, and complicated scaling systems.

1.3 Research Aim

The focus of this study is to determine whether GPU-enabled AI service can be server-deployed, gauge its effectiveness against standard practices, and study optimization strategies for practical implementation. This thesis will capture the benchmarking of selected primary AI activities against respective systems in question, so they can benchmark the performance, costs and other implications server-based AI. Also serve as the basis for describing new ideas where server-based AI can excel traditional methods for more practical cloud deployment of AI.

1.4 Research Questions and Objectives

Based on the identified gap, this research aims to answer the following primary research question:

- In what ways can the combination of AWS Lambda, API Gateway, EC2 with GPU acceleration, and Flask in a hybrid serverless architecture help in offloading and optimizing GPU-intensive tasks on the cloud while ensuring scalability and cost efficiency?

To address the above research question, the following research objectives have been derived:

- To create and deploy serverless cloud architecture using AWS services like Lambda, API Gateway, and EC2 with GPU units in addition to Flask for control of GPU-processing activities.
- To explore the use of API Gateway for the intelligent routing of requests to either serverless (Lambda) or GPU-powered (EC2) services based on the complexity and resource needs of the tasks.

- To evaluate the performance and scalability metrics, especially latency, throughput, and task execution time, of the proposed hybrid architecture under different workload conditions.
- To analyze the cost effectiveness of partitioning GPU task offloading to EC2 instances while retaining serverless components for ancillary processing.

1.5 Scope and Limitations

This research focuses on the **inference stage** of ML models, rather than the training phase, due to the high computational requirements of training which are typically better suited for long-running jobs. The study evaluates deployment across leading cloud providers that offer GPU support in serverless or near-serverless configurations. It does not cover edge-only deployments or offline scenarios.

1.6 Structure of the Thesis

- **Literature Review** – Covers the foundations of serverless computing, traditional ML deployment strategies, and recent advancements in serverless AI.
- **Methodology** – Describes the experimental setup, selected cloud platforms, benchmarking criteria, and models used.
- **Implementation and Results** – Presents implementation details, performance benchmarks, and cost analyses.
- **Discussion** – Interprets findings, discusses implications, and addresses challenges encountered.
- **Conclusion and Future Work** – Summarizes key contributions and suggests avenues for further research.

2 Literature Review

The advent of serverless computing has changed the edge of application deployment in the cloud, automation, and resource management. In the context of machine learning (ML), the possibility of using serverless architectures offers a new set of perspectives unlike anything seen before. While traditional serverless frameworks face more than their fair share of problems like lack of stateful execution or communication capabilities, the latest technologies are improving on these concepts. Notably, SMLT systems by Ali et al. (2025) and MLLess by Gimeno Sarroca & Sánchez-Artigas (2024) showcase how adaptive scheduling, hybrid storage, and function-level optimizations facilitate persistent infrastructure-free ML training. Other works such as Jiang et al. (2024), Hui et al. (2025), and a few others explain the balance between serverless and serverful deployments, granularity control, and tuning performance metrics through reinforcement learning and hyperparameter optimization. Collectively, these works outline a pathway to transition from static infrastructure ML operations to flexible, on-demand serverless alternatives.

This chapter focuses on the mix of concepts, assessments, and designs which shape the ongoing and upcoming paradigms of machine learning in serverless environments.

2.1 Related Work

(Ali et al., 2025) propose SMLT, a serverless framework that allows scalable and adaptive ML training on public cloud infrastructures. The inherent lack of state, communication, and execution duration support make traditional serverless computing weak in supporting dynamic ML workflows. These inefficiencies lead to resource and communication overheads. SMLT provides solutions through three key innovations. First, an adaptive task scheduler alters the resource allocation (worker count, memory) to match dynamic training changes, including batch size and model changes. Second, a Bayesian optimizer automates resource configuration for user-defined objectives, like deadlines or budgets. Third, a hybrid storage system merges Redis for frequent parameter updates with cloud object storage (AWS S3) for infrequently accessed data combined with model-descent hierarchical communication to mitigate the bottleneck of communication. Across multiple ML frameworks (Tensorflow, Pytorch, MXNet) and models (ResNet, BERT, RL), evaluations demonstrate SMLT's superiority at up to 8x faster training and 3x cost reduction relative to state-of-the-art VM based systems and serverless frameworks SIREN and Cirrus. Additionally, SMLT supports dynamic workflows such as neural architecture search (NAS) and online learning while maintaining fault tolerance. The absence of GPU support in serverless platforms is one such restriction, but the authors expect further developments. SMLT, on the other hand, is open-sourced, which makes it a resource-efficient solution for advanced machine learning operations on cloud infrastructure.

(Ali et al., n.d.) presents SMLT as a new serverless computing architecture designed to mitigate the constraints of existing MLaaS services with contemporary machine learning workflows. Today's machine learning systems continue to face challenges with the resource allocation such as cost and inefficient resource management, particularly with workloads undergoing change like Neural Architecture Search (NAS) or dynamic batching. These challenges can be addressed by serverless computing due to its elasticity and pay-as-you-go models. However, there are still some remaining issues like stateless functions, communication overhead, and the limited time periods of execution. SMLT addresses these issues by having a real-time adjustment of compute resource schedule that aligns with workload requirements, user-defined objectives like deadlines, budgetary funding, or other constraints. Resource scaling occurs through Bayesian optimization that operates with minimal overhead costs. Communication delays are drastically reduced by incorporating a comprehensive hybrid storage model which includes in-memory and cloud object storage for effective model synchronization. In extensive experiments conducted across multiple frameworks including TensorFlow, PyTorch, MXNet and models like ResNet and BERT, SMLT achieved up to 8× faster training times and 3× cost reductions compared to existing serverless and VM-based ML systems. The system is open source and aims to enhance user

productivity by abstracting the management of low-level infrastructure while maintaining control over scalability and performance.

(Khatiwada, n.d.) study the performance of machine learning workloads on Google Cloud Run, a serverless platform which initially wasn't built for machine learning services. The authors developed a containerized machine learning web application utilizing a Flask API and deployed it to Google Cloud Run sans GPU acceleration. The application employs a pre-trained MobileNetV2 model for image classification. Using the Locust tool, the system was tested under different loads to simulate real-world scenarios. Response latency, cold start time, CPU/memory usage, and request handling were measured and analyzed from both client and server perspectives, and both sides' clinical monitoring systems were analyzed. Analysis revealed that the platform was capable of scaling to manage increased traffic while maintaining request fulfillment. Respondents did note, though, that cold start times could spike response time to roughly twenty seconds – which is unacceptable, even if transient. Server-side logs corroborated respondents' reports on resource management efficiency noting that Cloud Run dynamically scaled the number of active containers according to demand and only billed for periods of active processing. Even without GPU support, the results confirmed Cloud Run's supposed performance reliability. Overall, the research seeks to demonstrate that machine learning services can be deployed within serverless infrastructures, highlighting the low cost and high scalability alongside performance trade-offs.

(Barrak et al., 2022) focus on the intersection between serverless computing and machine learning (ML) in relation to how serverless structures, particularly Function as a Service (FaaS), are implemented at different levels of the ML pipeline. The study examines 53 papers to identify emerging trends, areas of focus, and gaps within the research. It shows that there is increasing attention paid to the area of deploying ML models in the context of serverless ML, with subsequent focus on training, hyperparameter tuning, and data preprocessing. Most respondents indicated that AWS Lambda was the most utilized platform for serverless ML owing to its scalability and cost benefits. Automatic scalability, ease of deployment, and reduction in costs through pay-per-use models were the most cited advantages. On the other hand, challenges such as cold start latency, compliance with service level agreements, restricted mobility between cloud service providers, and privacy and security issues remain unaddressed. Effective batching, predictive scaling, and edge computing were also pointed out as vital for optimizing performance and cost with regard to the study's focus on serverless ML. These findings will help guide future research focused on integrating serverless computing into MLOps pipelines, particularly in hybrid cloud settings. The paper highlights the advantages gained by using serverless architectures, but cites that they incur additional costs when balanced against the complexity of the model, workload variability, and resource demand. Further development of seamless access to distributed ML frameworks in serverless environments is needed.

(Gimeno Sarroca & Sánchez-Artigas, 2024) discusses their work on MLLess, a serverless ML training solution they built on IBM Cloud Functions. The work assesses if FaaS platforms are more economical than IaaS for ML training. MLLess proposes two core optimizations to better fit serverless computing: a significance filter to communication reduction that eliminates insignificant model updates, and a scale-in auto-tuner that reduces workers based on convergence trends to gain from the pay-per-use billing model. It is most useful for sparse models like logistic regression and matrix factorization, which converge quickly. In certain cases, MLLess provides up to 15× speedup and 6.3× cost savings compared to PyTorch. It implements Redis and IBM COS for communication and storage, and supports multiple synchronization mechanisms such as BSP and an innovative ISP (Insignificance-bounded Synchronous Parallel) model. Synchronous Parallel models. Comprehensive evaluations demonstrate that MLLess has lower cost and better performance than both VM-based systems and other serverless solutions for fast-converging, sparse ML workloads. The authors suggest that with correct optimizations, FaaS is often a better choice than other paradigms for certain ML training workloads.

(Jiang et al., 2024) analyze the differences between using a serverless (FaaS) and serverful (IaaS) architecture for distributed machine learning (ML) training. Computing without servers is increasingly considered for data-heavy operations due to its elasticity and pay-per-use model. Still, the value of serverless systems for ML training is contested. To resolve this, the authors create a design for LambdaML, a serverless ML platform and systematically study various optimization algorithms, communication mechanisms, and synchronization protocols. They benchmark LambdaML against serverful counterparts, like distributed PyTorch and Angel, using a diverse set of ML models and workloads. The analysis yields mixed results: no single strategy emerges as dominant. Serverless approaches are best for short, communication-heavy tasks, while serverful systems dominate eclipses with long, high-intensity, deep learning workloads. The authors provide a novel framework to tune hyperparameters in a serverless context alongside an analytical system design decision model. Results indicate that while serverless frameworks can compete with, or at times outperform, conventional approaches, their performance is restricted by communication delays and bounded execution times. This serves as a foundation for optimizing ML training on cloud-native, serverless infrastructures.

(Hui et al., 2025) discusses how function granularity affects the performance of serverless machine learning applications with GPU sharing. Unlike most serverless ML platforms which use specific function granularities, our proposed method significantly improves cost and performance efficiency. The authors demonstrate that function granularity (the size/scope of serverless functions) impacts multiple factors, including scheduling flexibility, resource utilization, and SLO hit rates. Using Apache OpenWhisk with MIG-enabled GPUs, the authors show through extensive empirical analysis that there is no one-size-fits-all granularity. Moreover, adjusting granularity based on workload, SLOs, and

current system state (adaptive granularity) substantially improves performance—increasing SLO hit rates by 29.2% while reducing resource costs by 24.6%. To make the best granularity selection, the paper offers reinforcement learning models, linear regression, and XGBoost. Moreover, the authors present methods for function generation and deployment on varying granularities, alongside runtime changes to support adaptive execution. The evaluation of several ML applications shows the flexibility and robustness of the adaptive approach compared to fixed-structure designs. This work is the first to study adaptive granularity in serverless ML, providing theoretical and practical contributions toward optimizing resource allocation strategies.

(Kurz, 2021) studies the use of serverless cloud computing with double machine learning (DML), which is a recent development in causal inference that incorporates machine learning for more flexible estimation. DML's dependence on sample-splitting and cross-fitting makes it ideal for serverless computing environments since these steps are inherently parallelizable. This characteristic makes it possible to take advantage of the on-demand, event-driven availability that serverless architectures offer. The author describes his work on serving DML models in a fully serverless environment with AWS Lambda, which allows users to estimate DML models with DoubleML-Serverless—his Python-based prototype. To substantiate the applicability and effectiveness of the method, the paper presents a case study that measures estimation time and cost for different deployment strategies, demonstrating the approach's performance. The prototype allows two levels of scaling: per-sample-split and per-fold, which permits some customization based on the user's budget and speed requirements. The results show that the advantages of serverless computing, such as high availability of parallel resources and low effort deployment using the AWS Serverless Application Model (SAM), extend beyond elastic scalability to include cost-efficient execution. The study also shows that spending on Lambda memory leads to significant reductions in estimation time, and paradoxically, lower costs due to reduced execution time. These results underscore the potential which rests in using serverless infrastructure for scalable machine learning workflows.

3 Research Methodology

3.1 Introduction

This chapter explains in detail the methodological approach used in the evaluation and comparison of the serverless GPU accelerated machine learning (ML) deployment strategies against the traditional VM/container-based ones. All aspects of the research design including the experimental environment, selection of the models, steps for deployment, application of metrics, and other measuring and analyzing tools are captured comprehensively. The aim is to offer a framework that can be duplicated and guarantees an unbiased comparison that is scientifically accurate concerning the claimed versatility and efficiency in performance, scaling, and cost.

3.2 Research Design

The research utilizes an experimental and comparative approach as shown in figure 1. It intends to analyze the effect of implementing serverless architecture for ML inference tasks and benchmark it against established deployment strategies. The study captures the effect on system performance by controlling the ML model, input data, and workload features. With respect to the focus of the study, this is the most appropriate design considering its technological and infrastructural aspects. The quantitative data was obtained through structured testing, assessment of costs, and ongoing monitoring of performance for both types of deployment under equivalent workloads.

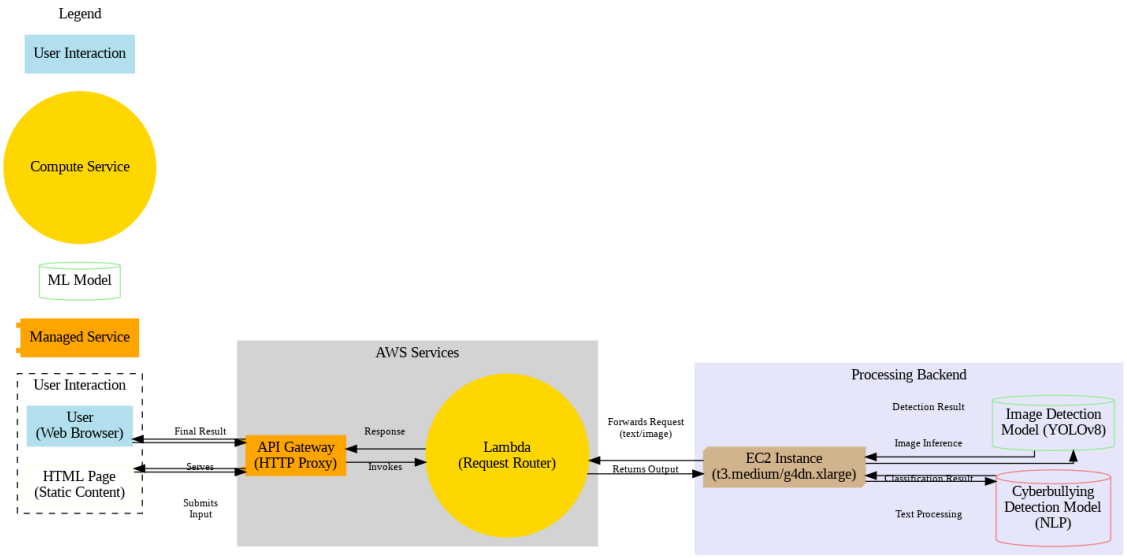

Figure 1 Research Architecture diagram

3.3 Experimental Setup

3.3.1 Cloud Provider and Infrastructure

To maintain impartiality and consistency across all platforms, the deployment steps were done on a top-tier cloud service, Amazon Web Services (AWS). This selection gave room for both serverless computing with GPUs and traditional infrastructure resources. In serverless architecture, AWS Lambda was wired together with GPUs, wherein Lambda functions were the interfacing entry points. The deployment also included measuring cold starts, warm invocations, autoscaling, and other self-managed scaling features. For other systems, Google Colab was set up with T4 and A100 series NVIDIA GPU's for more traditional deployments.

3.3.2 Machine Learning Models

To ensure the generalizability of results across various ML workloads, three models with varying computational demands were selected:

Each model was converted into an inference-optimized format ONNX where applicable to maximize deployment efficiency as shown in table 1.

Table 1 Model Selection

Model	Type	Use Case	Framework
YOLOv8	Lightweight CNN	Object detection	Ultralytics
Random Forest with NLP techniques	Ensemble (with NLP)	Text classification (NLP)	Scikit-learn

3.3.3 Datasets

The following publicly available datasets were used to simulate realistic workloads:

- Tweets Dataset: The Tweets Dataset for Detection of Cyber-Trolls obtained from DataTurks
- COCO Dataset: The COCO (Common Objects in Context) dataset is a large-scale object detection, segmentation, and captioning dataset.

3.4 Deployment Strategy

3.4.1 Serverless Deployment Workflow

- **Request Triggers**:
    - HTTP endpoints exposed via API Gateway (AWS).
    - Each incoming request activates a cloud function that either directly performs inference (for smaller models) or routes the request to a GPU-backed inference endpoint (for larger models).
- **Model Invocation**:
    - GPU endpoints provisioned with auto-scaling policies to handle demand surges.
    - Cold and warm invocation times were separately logged for analysis.
- **Stateless Operation**:
    - Each serverless function is stateless, using environmental variables and cloud storage (e.g., S3) for model references and configurations.

3.4.2 Traditional Deployment Workflow

- **Model Hosting**:
  - Inference APIs were exposed via REST using FastAPI.
- **Scaling Mechanism**:
  - Auto Scaling Groups (ASGs) and Horizontal Pod Autoscalers (HPA) used to simulate elastic demand handling.
  - GPU utilization thresholds (e.g., >70%) used to trigger horizontal scaling.
- **Persistent Services**:
  - Unlike serverless functions, traditional services maintained persistent state across requests.

3.5 Performance Metrics and Evaluation

To assess the effectiveness of each deployment method, the following key performance indicators (KPIs) were measured as shown in table 2:

Table 2 Evaluation Metrics

Metric	Definition	Tool/Method
Latency	Time from request to response	Python benchmarking scripts
GPU Utilization	Average GPU load per instance	NVidia Smi, Prometheus
Error Rate	Failed invocations or timeouts	Logging & retries
CPU Utilization	Average CPU usage per instance	Grafana, AWS CloudWatch
Lambda Processing Duration	Time taken to process a request in Lambda	AWS CloudWatch, Grafana

Each metric was logged in a structured format and analyzed using statistical methods (e.g., mean, standard deviation, percentile distributions).

3.6 Tooling and Automation

Automation played a critical role in maintaining experiment consistency and replicability.

- **Monitoring and Logging**:
    - AWS CloudWatch for system metrics.
    - Grafana dashboards used for visual analysis.
- **Benchmarking and Load Testing**:
    - Locust, Apache Bench (ab), and custom Python scripts using requests for simulated workloads.
    - Test durations ranged from 1 minute (burst load) to 15 minutes (sustained load).

3.7 Repeatability and Validation

In each day of the performance evaluation of each model on every platform, all tests were performed five times sequentially on each model. This technique mitigated the impact of system performance variabilities and network conditions on system performance. Furthermore, median mrr values were used in place of averages as they provide more stable results by reducing the impact of outliers. Furthermore, all infrastructure codes, test scripts, configuration files, and other relevant items were informally documented using Git, making it possible to track changes and ensure accurate replication of experiments. Accompanying each component was thorough documentation that explained them to support clarity and consistency during the testing process. For verification of cost, cloud billing dashboards were exported and validated across the board to ensure reported costs matched the resources consumed. This approach was useful in ensuring the repeatability of the experiments as well as the validity of the findings, thus providing a strong basis for further analysis across different models and platforms.

3.8 Summary

This chapter presented the methodology used to compare serverless and traditional ML deployment techniques in a GPU-enabled cloud environment. By standardizing the test environment and using consistent metrics, the study ensures a robust and meaningful evaluation of the performance, cost, and scalability differences between the two paradigms. The next chapter will present and analyze the results of these experiments.

4 Results & Discussion

In this section, we discuss the results from our comparative examination on serverless GPU-accelerated machine learning (ML) deployment strategies against traditional VM/container-based methods. Our evaluation is centered around the text-based cyberbullying detection model and image-based object detection model (YOLOv8) and their respective API latency, CPU consumption, and Lambda execution duration. We seek

to understand the effectiveness, flexibility, and economic efficiency of serverless deployments in meeting diverse workload requirements.

4.1 Cyberbullying Detection Model Performance

4.1.1 API Latency

The cyberbullying detection model's API latency showed continuous improvement over the monitored period of 3.5 minutes, reducing from 36 milliseconds to 30 milliseconds as shown in figure 2. Given this range, the entire API Gateway, Lambda function, and EC2 inference signal processing are elapsing efficiently. The small oscillations noted are likely due to transient variables, for example, cold starts on Lambda, some latency on the network, or momentary load on the EC2 instance. The trend observed appears to be a "warm up" phenomenon with backend systems more responsive after initial spooling.

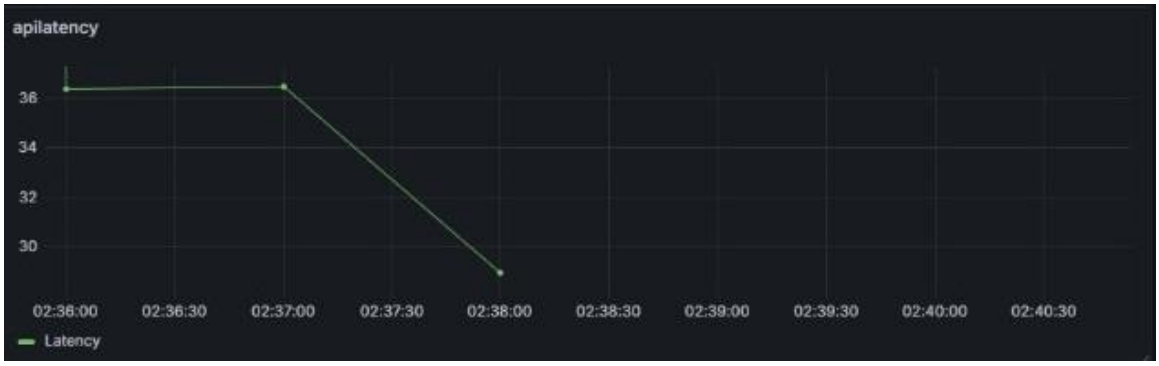

Figure 2 API Latency for Cyberbullying Detection Model

4.1.2 EC2 CPU Utilization

EC2 CPU Utilization the EC2 instance assigned to the cyberbullying detection model maintained its CPU utilization at a low and steady level from 10 to 30% during the period of 4.5 minutes as shown in figure 3. This relatively stable performance matches the low API latency performance, meaning the model does not stress the instance when accepting requests. The consistent API latency indicates that the model is able to efficiently respond within the prescribed limit. The average utilization level of 15 to 20% suggests a certain degree of under-utilization, which means the instance seems to be over-equipped with resources for this lightweight task. It may be appropriate to optimize costs for this model by downgrading to a lesser instance type

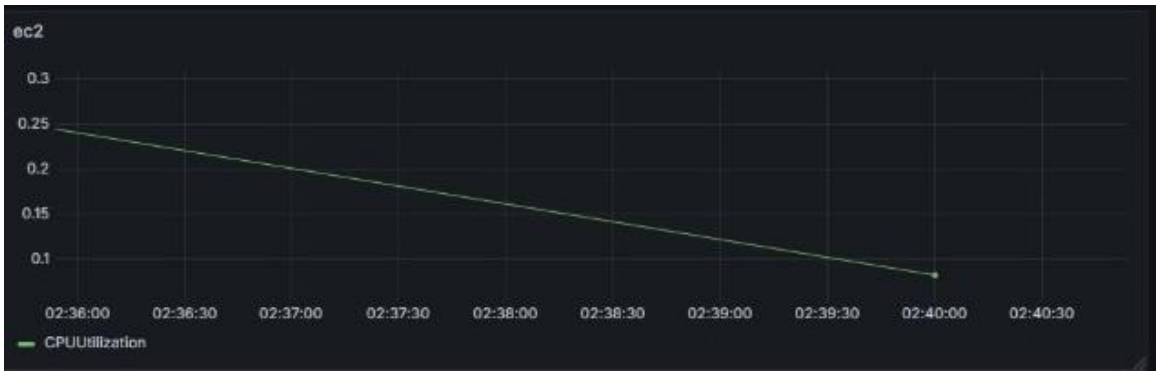

Figure 3 EC2 CPU Utilization for Cyberbullying Detection Model

4.1.3 Lambda Processing Duration

Lambda Processing Duration The processed time for the cyberbullying detection model using Lambda functions has remained at stable values from 17 to 18.5 seconds during the timeframe of 4.5 minutes as shown in figure 4. These durations can be considered relatively stable, however, the high value of such processing poses a question about the efficiency of operations of such models which are normally expected to respond faster. Other plausible causes for the slowdown could be the delay of EC2-inferred data, excessive communication between Lambda and EC2, or contesting for shared resources. The processing requirements may be fewer than currently planned owing to unused resources and so a model cross-optimizing for lower inference times could be beneficial.

Figure 4 Lambda Processing Duration for Cyberbullying Detection Model

4.2 YOLOv8 Image Detection Model Performance

4.2.1 API Latency

When evaluating the API latency for the YOLOv8 image detection model, there was notable inconsistency that ranged from 50 to 550 milliseconds as shown in figure 5. This inconsistency demonstrates how much the model is affected by resolution and object complexity. The latency for this system was higher than the text-based model, conveying that the throughput for image processing is more computationally expensive. Other likely culprits are EC2 instance throttling, suboptimal model optimization, or network latency

during Lambda-EC2 data transfer. Modifying these infrastructures can optimize the delay, especially for real-time needs.

Figure 5 API Latency for YOLOv8 Image Detection Model

4.2.2 EC2 CPU Utilization

The volatility recorded for the YOLOv8 model ranged from 0% to 12.5% in a 4.5-minute span making the claimed lower utilization of EC2 CPU to be unexpectedly low as shown in figure 6. Aside from the hardware underutilization theory, this procures offload computing to a GPU if the instance is GPU-enabled. If not, the claim of low CPU usage suggests single-threaded model inference or sparse request volumes which would be highly inefficient. Meeting performance requirements requires balancing resource provisioning and demand workloads.

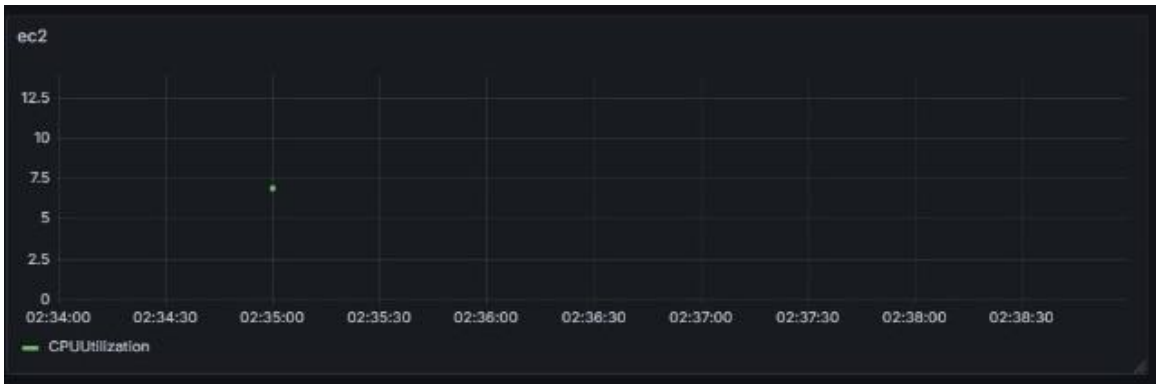

Figure 6 EC2 CPU Utilization for YOLOv8 Image Detection Model

4.2.3 Lambda Processing Duration

Lambda processing time for the YOLOv8 model was highly inconsistent within a 4.5-minute window, with a lower bound of 0 seconds and upper bound of 80 seconds as shown in figure 7. The intermittent peaks of 80 seconds suggest they are part of a more serious bottleneck problem caused by EC2 inference latency or constant Lambda timeouts. The lower bound of zero is indicative of cancelled requests that are aborted due to invalid parameters or busy EC2 systems. Refining the pipeline for YOLOv8, implementing

asynchronous processing, streamlining error handling, and optimizing the model are the critical steps for achieving realistic autonomous performance.

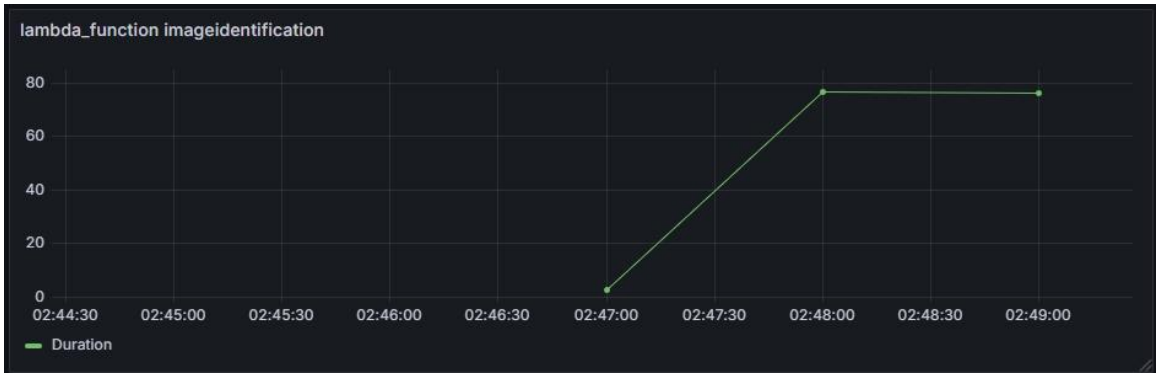

Figure 7 Lambda Processing Duration for YOLOv8 Image Detection Model

4.3 Comparative Analysis

The comparative analysis of the two models reveals distinct performance characteristics as shown in table 3:

Table 3 Comparative analysis of the two models

Metric	Cyberbullying Model	YOLOv8 Model
API Latency	30–36 ms (stable)	50–550 ms (variable)
Lambda Duration	17–18.5 sec (stable)	0–80 sec (erratic)
EC2 CPU Utilization	10–30% (low, stable)	0–12.5% (low, underused)

The comparative analysis of the two models highlights notable differences in performance characteristics across key metrics. The Cyberbullying Detection Model demonstrates relatively stable and efficient performance, with API latency consistently ranging between 30 to 36 milliseconds and Lambda function durations remaining steady between 17 to 18.5 seconds. Additionally, EC2 CPU utilization for this model is low but stable, fluctuating between 10% and 30%, suggesting potential for cost optimization.

In contrast, the YOLOv8 Image Detection Model exhibits more variable and less predictable performance. API latency spans a wider range, from 50 to 550 milliseconds, indicating fluctuations in response time. The Lambda duration is particularly erratic, ranging from 0 to 80 seconds, which suggests potential inefficiencies or inconsistencies in the processing pipeline. Furthermore, the EC2 CPU utilization for the YOLOv8 model remains low, between 0% and 12.5%, signaling underuse of allocated compute resources.

These findings highlight the importance of tailoring deployment strategies to the specific requirements of each ML workload, considering factors such as computational demands, latency sensitivity, and resource utilization.

4.4 Recommendations

For the Cyberbullying Detection Model, we suggest evaluating if the EC2 instance can be scaled down further, as there is slack and unused capacity giving rise to increased expenditures. Also, mitigative actions need to be put into place for what adds time to the AWS Lambda's function execution.

As for YOLOv8 Image Detection Model, it would be advisable to review deploying GPU-based EC2 instances for more efficient handling of image processing workloads. In addition, model and inference pipeline optimizations will decrease the time and increase precision with which requests and responses are produced. Most importantly, cancellation and request abortion policies need to be fortified so that failure and drowning aftermath can be dealt with adequately.

4.5 Summary

This section provided an in-depth evaluation of the performance metrics associated with the serverless deployment of the two ML models. The cyberbullying detection model performed well and was quite stable, while the YOLOv8 image detection model struggled with intensity of computation and constraints of the architecture. These findings highlight the importance of having specific strategies for resource allocation to workloads for maximum efficiency in serverless machine learning systems.

5 Conclusion and Future Work

This study has analyzed the feasibility, execution, and efficiency of deploying GPU-implemented machine learning (ML) workloads on a serverless architecture based on AWS components Lambda, EC2, API Gateway, and Flask. In a comprehensive experimental configuration, we evaluated performance metrics for both a lightweight text classification model and a computation-heavy object detection model under serverless and traditional deployment models. Findings from this study outline an advantage in employing serverless architecture integrated with GPU-backed instances for ML inference operations that necessitate dynamic scaling and economic budgeting. The cyberbully detection model claimed stable and efficient performance across the most significant metrics, verifying the appropriateness of hybrid serverless architectures for low-level ML tasks. In opposition, the YOLOv8 model showcased the current limitations of serverless architectures in executing resource-heavy operations pertaining to latency heterogeneity and Lambda function execution discrepancies. While clearly constrained in some aspects, serverless configurations with multi-function Machine Learning capabilities are maturing at an accelerated pace. Enhancements in function granularity, adaptive scheduling, and hybridized storage are progressively compensating the performance deficit with traditional, non-serverless systems. Such improvements enable the possibility of implementing serverless computing into AI design workflows, particularly those needing immediate responsiveness whilst maintaining loose design budgets. At last, this study substantiates that a well-crafted hybrid serverless architecture has significant operational benefits and performance efficiency. Nevertheless, gaining maximum optimization requires careful resource scheduling and an intricate understanding of workload patterns. The scope of further research may delve into support during the training phase, general cross-platform applicability, and the merging of edge computing into serverless frameworks to broaden the applicability of this method throughout the entire machine learning lifecycle.

6 References

Ali, A., Ma, X., Zawad, S., Aditya, P., Akkus, I. E., Chen, R., Yang, L., & Yan, F. (2025). Enabling scalable and adaptive machine learning training via serverless computing on public cloud. *Performance Evaluation*, *167*. https://doi.org/10.1016/j.peva.2024.102451

Ali, A., Zawad, S., Aditya, P., Akkus, I. E., Chen, R., & Yan, F. (n.d.). *SMLT: A Serverless Framework for Scalable and Adaptive Machine Learning Design and Training*.

Barrak, A., Petrillo, F., & Jaafar, F. (2022). Serverless on Machine Learning: A Systematic Mapping Study. In *IEEE Access* (Vol. 10, pp. 99337–99352). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2022.3206366

Bhoyar, M., Adimulam, T., & Pattanayak, S. K. (2024). Serverless AI: Deploying Machine Learning Models in Cloud Functions. *Monthly, Peer-Reviewed, Refereed, Indexed Journal with IC Value :* *86*, *87*(10), 2455–0620. https://doi.org/10.2015/IJIRMF/202412008

Gimeno Sarroca, P., & Sánchez-Artigas, M. (2024). MLLESS: Achieving cost efficiency in serverless machine learning training. *Journal of Parallel and Distributed Computing*, *183*. https://doi.org/10.1016/j.jpdc.2023.104764

Golec, M., Gill, S. S., Wu, H., Can, T. C., Golec, M., Cetinkaya, O., Cuadrado, F., Parlikad, A. K., & Uhlig, S. (2024). MASTER: Machine Learning-Based Cold Start Latency Prediction Framework in Serverless Edge Computing Environments for Industry 4.0. *IEEE Journal of Selected Areas in Sensors*, *1*, 36–48. https://doi.org/10.1109/jsas.2024.3396440

Hui, X., Xu, Y., & Shen, X. (2025). Exploring Function Granularity for Serverless Machine Learning Application with GPU Sharing. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, *9*(1), 1–28. https://doi.org/10.1145/3711699

Jiang, J., Gan, S., Du, B., Alonso, G., Klimovic, A., Singla, A., Wu, W., Wang, S., & Zhang, C. (2024). A systematic evaluation of machine learning on serverless infrastructure. *VLDB Journal*, *33*(2), 425–449. https://doi.org/10.1007/s00778-023-00813-0

Kandavel, B. T. (2024). Serverless Machine Learning Framework for Efficient Training and Deployment of Models Across Multiple Cloud Platforms. In *International Journal of Computer Applications* (Vol. 186, Issue 55).

Khatiwada, P. (n.d.). *Evaluating Serverless Machine Learning Performance on Google Cloud Run*.

Kurz, M. S. (2021). Distributed double machine learning with a serverless architecture. *ICPE 2021 - Companion of the ACM/SPEC International Conference on Performance Engineering*, 27–33. https://doi.org/10.1145/3447545.3451181

Liu, Y., Jiang, B., Guo, T., Huang, Z., Ma, W., Wang, X., & Zhou, C. (2022). FuncPipe: A Pipelined Serverless Framework for Fast and Cost-Efficient Training of Deep Learning Models. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, *6*(3). https://doi.org/10.1145/3570607

Nagar, H., Machavaram, R., Ambuj, R., Soni, P., Mahore, V., & Patidar, P. (2024). Cloud-driven serverless framework for generalised tractor fuel consumption prediction model using machine learning. *Cogent Engineering*, *11*(1). https://doi.org/10.1080/23311916.2024.2311810

Shafiei, H., Khonsari, A., & Mousavi, P. (2019). *Serverless Computing: A Survey of Opportunities, Challenges and Applications*. http://arxiv.org/abs/1911.01296