

LB/TH/41/2025
TH5998

SCALABLE CLOUD-DRIVEN PIPELINE FOR MACHINE LEARNING ALGORITHMS DEPLOYMENT

Ranasinghe Weerakkodi Pubudu Mahesh Meththananda

219368R

Degree of Master of Science in Computer Science

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

March 2025

SCALABLE CLOUD-DRIVEN PIPELINE FOR MACHINE LEARNING ALGORITHMS DEPLOYMENT

Ranasinghe Weerakkodi Pubudu Mahesh Meththananda

219368R

Thesis/Dissertation submitted in partial fulfillment of the requirements
for Master of Science in Computer Science specializing in Cloud
Computing

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

March 2025

DECLARATION

I declare that this is my own work, and this thesis/dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to the University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part, in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature:

Date: 10/07/2025

Name: Pubudu Mahesh Meththananda

The above candidate has carried out research for the Master of Science thesis under my supervision.

Signature of the supervisor: Date: ...21/07/2025.....

Name: Dr. Sunimal Rathnayake

ABSTRACT

As machine learning (ML) continues to permeate industry and research, the need for robust, scalable, and maintainable deployment pipelines has become critical. Despite advances in model development, operationalizing ML models remains a non-trivial challenge due to the complexity of managing infrastructure, ensuring reproducibility, and integrating with heterogeneous data sources and consumers. This thesis proposes a generalized, cloud-oriented deployment framework for ML algorithms, addressing these challenges through modular and extensible architecture.

The framework is built upon containerized microservices, enabling environment-agnostic deployment and seamless transition from local development to production. A core component of the system is its pluggable inference interface, which abstracts model-specific logic, allowing arbitrary ML algorithms to be integrated with minimal engineering overhead. The pipeline supports dynamic parameterization of models, automatic resource provisioning, and concurrent execution, promoting scalability and adaptability to varying workloads.

A lightweight, web-based interface facilitates user interaction with the system, supporting dataset upload, model parameter configuration, execution of inference tasks, and retrieval of outputs in multiple formats. Workflow orchestration, monitoring, and logging are integrated into the pipeline to ensure operational transparency, fault tolerance, and streamlined debugging. All system components are loosely coupled, promoting maintainability and compatibility with multiple cloud service providers and on-premises environments.

The generality of the framework is validated through the integration of the Growing Self-Organizing Map (GSOM) algorithm and deployment on Amazon Web Service (AWS), demonstrating its ability to support computationally intensive models while preserving system responsiveness and reliability. However, architecture is agnostic to specific ML methods or cloud platforms, making it applicable across a wide range of deployment scenarios. This work contributes to a reusable and scalable Machine Learning Operations (MLOps) aligned infrastructure that bridges the gap between ML development and production deployment, supporting continuous delivery, reproducibility, and operational efficiency in modern ML workflows.

Keywords: Machine Learning Deployment, Cloud Computing, MLOps, CI/CD Pipeline, RESTful APIs, Containerization, Scalable Architecture, Model Operationalization

ACKNOWLEDGMENT

I would like to express my gratitude to **Prof. Damminda Alahakoon (Professor- La Trobe University, Australia)** and Mr. Prasad Maduranga (PhD. Candidate- La Trobe University, Australia), for initiating this project and providing guidance and motivation throughout the development of this cloud-driven ML pipeline. Their vision to enhance the accessibility and scalability of the GSOM algorithm as a cloud service has shaped the objectives and implementation of this research.

I extend my appreciation to my supervisor, **Dr. Sunimal Rathnayake (Senior Lecturer- Department of Computer Science and Engineering, University of Moratuwa)**, for his continuous support, insightful feedback, and technical guidance. His expertise in cloud computing and ML deployment has been instrumental in refining the methodologies and ensuring the success of this project.

Furthermore, I acknowledge the University of Moratuwa for providing the necessary resources and academic support to undertake this research. Special thanks to my colleagues and others who contributed through discussions, testing, and valuable suggestions, which greatly improved the quality of this work.

Finally, I would like to express my gratitude to my family for their encouragement and unwavering support throughout this journey. Their belief in my abilities has been a constant source of motivation in successfully completing this project.

TABLE OF CONTENTS

DECLARATION	i
ABSTRACT.....	ii
ACKNOWLEDGMENT	iii
LIST OF FIGURES	viii
LIST OF TABLES	x
LIST OF ABBREVIATIONS	xi
1. CHAPTER 01: INTRODUCTION	1
1.1 Overview	1
1.2 Problem statement.....	2
1.2.1 Leveraging Cloud Technologies for Scalability	2
1.2.2 Automating the ML workflow	3
1.3 Aim and objectives.....	4
1.4 Methodology	5
1.4.1 Case study	5
1.5 Organization of the Thesis	6
1.5.1 Chapter 2	6
1.5.2 Chapter 3	6
1.5.3 Chapter 4	6
1.5.4 Chapter 5	6
2. CHAPTER 02: LITERATURE REVIEW	7
2.1 Overview	7
2.2 ML deployment challenges	7
2.3 Cloud-based solutions for ML	8
2.3.1 Role of cloud platform.	8
2.3.2 Data management in ML pipelines.	8
2.3.3 Containerization	10
2.3.4 Serverless computing	10
2.3.5 On-Premises deployment	11
2.3.6 Cloud-based deployment.....	13
2.4 CI/CD Pipelines	14
2.5 Monitoring and Logging	14

2.5.1	Log management challenges	15
2.6	Docker and Containerization	16
2.7	Amazon Sagemaker for ML model deployment.....	17
2.7.1	Disadvantages of Amazon Sagemaker.....	19
2.7.2	Registry in ECR	20
2.8	AWS CodeBuild	20
2.9	Comparative analysis between existing solutions.....	23
2.9.1	A TensorFlow-Based Production-Scale ML Platform [28]	23
2.9.2	Developments in ML flow: A System to accelerate the ML lifecycle [10]	24
2.9.3	A ML model for improving healthcare services on cloud computing environment [29].....	25
2.9.4	Hidden Technical Debt in ML Systems [30]	27
2.9.5	Efficient and Robust Automated ML [16]	28
2.9.6	Speeding up End-to-End Auto ML via Scalable Search Space Decomposition [32].....	29
2.9.7	Elastic ML Algorithms in Amazon SageMaker [33]	30
2.9.8	AI and ML Integration with AWS SageMaker: Current Trends and Future Prospects [34]	31
2.9.9	Revolutionizing ML Model Serving with Containerization [19].....	32
2.9.10	Serverless ML: Building and Deploying AI Models in the Cloud [15]	33
3.	CHAPTER 03: METHODOLOGY	35
3.1	System Architecture Overview	35
3.2	Process Flow	36
3.3	Approach.....	37
3.3.1	Designing cloud architecture.....	37
3.3.2	Implement ML algorithm.....	38
3.3.3	Containerization	38
3.3.4	Integration with versioning controlling system	39
3.3.5	CI/CD pipeline	39
3.3.6	Cloud deployment.....	40
3.3.7	Integrate cloud storage	41
3.3.8	Deployment status check.....	42
3.3.9	Web interface	42

3.3.10	Execution status check	44
3.4	Case study	44
3.4.1	Self-Organizing Map (SOM)	44
3.4.2	Growing Self-Organizing Map (GSOM)	45
3.4.3	Technology stack	46
3.5	Pipeline architecture design	47
3.6	Containerization	48
3.7	Dataset handling	49
3.8	Monitoring and metrics	50
3.9	AWS Sagemaker	50
3.10	Summary	52
4.	CHAPTER 04: EVALUATION	53
4.1	Overview	53
4.2	Scalability and dynamic workloads	53
4.3	Cost-efficient pipeline for deployment	53
4.4	Accurate and reliable results from the pipeline.....	54
4.5	Accessible is the pipeline for non-technical users	54
4.6	Evaluation	54
4.6.1	Data set generation.....	54
4.6.2	Execution in local environment	55
4.6.3	Cloud-based environment	58
4.7	Evaluation matrix	60
4.7.1	Scalability.....	60
4.7.2	Model complexity	64
4.7.3	Size of dataset	66
4.7.4	Computational resources	66
4.7.5	User experience	71
4.7.6	End to end user experience evaluation.....	74
4.7.7	Error handling and troubleshooting	78
4.7.8	Performance and usability observations	79
4.7.9	Output validation.....	81
4.7.10	Challenges	81
4.7.11	Conclusion	82

5.	CHAPTER 05: CONCLUSION AND FUTURE WORK	83
5.1	Conclusion	83
5.2	Future work	83
6.	REFERENCES.....	86
7.	APPENDICES	91
7.1	Appendix A: Code	91
7.2	Appendix B: Survey	96
7.3	Appendix C: Output result	99

LIST OF FIGURES

Figure 2.1: ML model deployment [13].....	10
Figure 2.2: ML workflow in serverless deployment.....	11
Figure 2.3: On-premises ML model deployment.....	12
Figure 2.4: Cloud-based deployment [16]	13
Figure 2.5: Continuous Integration (CI) and Continuous Deployment (CD)	14
Figure 2.6: Logging and monitoring relationship between problem solving strategy	15
Figure 2.7: Impact of containerization on ML model deployment and collaboration [19].....	17
Figure 2.8: Amazon Sagemaker model deployment [20]	18
Figure 2.9: Amazon Sagemaker functional workflow [21]	19
Figure 2.10: AWS CodeBuild [27]	21
Figure 2.11: Sample CodeBuild YAML file.....	22
Figure 2.12: AWS CodeBuild release process [27]	23
Figure 3.1: Scalable cloud driven pipeline for ML algorithms deployment (conceptual diagram).....	35
Figure 3.2: Scalable cloud driven pipeline for ML algorithms deployment.....	47
Figure 3.3: Containerization	48
Figure 3.4: AWS Sagemaker configuration	51
Figure 4.1: Test data generation script.....	55
Figure 4.2: Test automation and collecting data script	56
Figure 4.3: Performance analysis with increasing data file size in local environment	57
Figure 4.4: Test automation script for cloud-based deployment.....	58
Figure 4.5: Response time with increasing data file size increasing in cloud	60
Figure 4.6: Response time. Cloud vs. Local	61
Figure 4.7: Response time in cloud environment with and without scalability	63
Figure 4.8: With and without scalability - cloud watch AWS	63
Figure 4.9: CPU utilization for AWS Sagemaker execution	64
Figure 4.10: Disk utilization for AWS Sagemaker	65
Figure 4.11: Memory utilization	65
Figure 4.12: Choose a registered ML algorithm	75
Figure 4.13: Pipeline execution.....	76
Figure 4.14: Pipeline execution result.....	77
Figure 4.15: GSOM plot	77
Figure 4.16: Output result in csv format	78
Figure 4.17: Error logging.....	78
Figure 4.18: AWS Cloud watch service.....	79
Figure 7.1: Zoo_gsom python code part 1	91
Figure 7.2: Zoo_gsom python code part 2	92
Figure 7.3: New plot python method to support cloud deployment	93
Figure 7.4: Build spec YAML file	93

Figure 7.5: Requirement txt file to setup python package	94
Figure 7.6: Docker file	94
Figure 7.7: Sagemaker endpoint access	95
Figure 7.8: Prediction result download API.....	95

LIST OF TABLES

Table 2.1: TFX solution vs. scalable cloud-based pipeline solution.....	24
Table 2.2: ML flow vs. scalable cloud base pipeline.....	25
Table 2.3: ML for healthcare with cloud computing vs scalable cloud-based pipeline	26
Table 2.4 Hidden technical debt in ML system vs scalable cloud computing pipeline.	27
Table 2.5: Efficient and robust automated ML vs scalable cloud computing pipeline.	28
Table 2.6: Speeding up end to end auto ML vs scalable cloud driven pipeline.....	29
Table 2.7: Elastic ML using Sagemaker vs Scalable cloud-based pipeline.....	30
Table 2.8: ML integration with Sagemaker vs scalable cloud based generic pipeline	31
Table 2.9: Revolutionizing ML models serving with containerization vs scalable cloud base generic pipeline	33
Table 2.10: Deploying AI models in the cloud using serverless architecture vs scalable cloud-driven pipeline	34
Table 4.1: Response time for each data set in local environment deployment.....	57
Table 4.2: Response time for cloud-based deployment with respect to data size.....	59
Table 4.3: Testing approaches for local deployment and cloud deployment.....	60
Table 4.4: Response time comparison between local and cloud deployment.....	61
Table 4.5: Response time comparison scalable cloud deployment vs. non-scalable cloud deployment.....	62
Table 4.6: Infrastructure setup cost.....	66
Table 4.7: Compute cost	67
Table 4.8: Storage cost.....	68
Table 4.9: Network and bandwidth cost	69
Table 4.10: Software and licensing cost	69
Table 4.11: Maintenance cost	70
Table 4.12: User experience evaluation metric.....	71
Table 4.13: Survey summary	74

LIST OF ABBREVIATIONS

Abbreviation	Description
AWS	Amazon Web Service
API	Application Programming Interface
BMU	Best Matching Unit
CD	Continuous Development
CI	Continuous Integration
DevOps	Developer Operations
ECR	Elastic Container Registry
GCP	Google Cloud Platform
GSOM	Growing Self-Organizing Map
HTTP	Hyper Text Transfer Protocol
IaaS	Infrastructure as a Service
IT	Information Technology
KMS	Key Management Service
ML	Machine Learning
MLOps	Machine Learning Operations
OCI	Open Container Initiative
OCI	Open Container Initiative
RBAC	Role-Based Access Control
REST API	Representational State Transfer Application Programming Interface
S3	Simple Storage Service
SOM	Self-Organizing Map
UI	User Interface