Performance Evaluation of AWS and Azure Cloud Platforms Using TensorFlow Framework

MSc Research Project
Master of Science in Cloud Computing

Hima Shree Manyam Sunil
Student ID: X23189851

School of Computing
National College of Ireland

Supervisor: Shreyas Setlur Arun

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Hima Shree Manyam Sunil		
Student ID:	X23189851		
Programme:	MSc in Cloud computing	**Year:**	2024/2025
Module:	MSc Research Project		
Supervisor:	Shreyas Setlur Arun		
Submission Due Date:	12/12/2024		
Project Title:	Performance Evaluation of AWS and Azure Cloud Platforms Using TensorFlow Framework		
Word Count:	7295	**Page Count**	22

Signature:	Hima Shree Manyam Sunil
Date:	12-12-24

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.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Performance Evaluation of AWS and Azure Cloud Platforms
Using TensorFlow Framework

Hima Shree Manyam Sunil
x23189851@student.ncirl.ie
Dataset Link: https://rb.gy/rlcs74

Abstract

This paper presents a comparative analysis of two leading cloud computing platforms: Amazon Web Services (AWS) and Microsoft Azure. The focus of this analysis is on the implementation of TensorFlow, a widely used machine learning framework. The objective is to evaluate both platforms based on several criteria, including performance, cost-efficiency, scalability, and ease of implementation. Through benchmarking tests obtained through the cloud monitors and application performance assessments, this study aims to provide insights into which platform is more suitable for TensorFlow-based applications.

1. Introduction

Software and machine learning industries have benefitted from cloud systems, as it allows for a greater possibility of model deployment in a cost-effective and flexible manner. When it comes to the cloud service providers that support deep learning frameworks like TensorFlow, Microsoft Azure and Amazon Web Services rank highest. Both of these platforms provide fully-featured IaaS and PaaS solutions allowing easy deployment, scaling, and management of machine learning models for data scientists and research scientists.

However, insufficient attention has been given to the training phase of deep learning across the considered platforms which employs CPU resources. This is where the presented work fits in by focusing on a critical review of two Cloud Platforms AWS ec2 and Azure VM. The assessment will cover issues of CPU load, disk I/O operations, memory consumption, cost per task, time per task, and accuracy of the task performance employing GoogleNet based deep learning.

1.1 Motivation for the Study

In recent years machine learning (ML) and deep learning (DL) algorithms have invaded many fields like healthcare, finance, and computer vision among others. Nevertheless, the majority of deep learning modeling requires enough computational power which lends considerable interest to cloud services. Powerful computing capabilities are available on demand, such as those provided by AWS and Azure, but their effectiveness and cost depend on the type of workload used in the application.
With regard to ecommerce cloud applications, both AWS and Azure have a considerable number of options that facilitate training of machine learning models. It is worth underscoring that AWS EC2 instances are recognized for their versatility and efficient scaling which makes them a better match for training machine learning models with a high resource demand. On the

other side, Azure VM instances are normally cheaper and have easier integration which makes them suitable for small applications. Nevertheless, since TensorFlow can be utilized in both, it is fundamental to assess the training costs, times, and performance metrics using processor-based resources in both platforms.

1.2 Research Questions and Objectives

- **How does AWS EC2 compare with Azure VMs in terms of CPU Utilization, read write operations, memory used, costs and application performance when training deep learning models in TensorFlow?**

The specific areas of this research are:

1. To benchmark the CPU performance of AWS EC2 and Azure VM instances while training a GoogleNet deep learning model.
2. To evaluate read /write operations and their impact on training time and efficiency.
3. To measure memory utilization during training and assess how effectively each platform handles our selected datasets.
4. To compare the cost efficiency of training deep learning models on different platforms, considering various instance types and pricing models.
5. To analyze the training time and model accuracy to determine which platform delivers the best performance in terms of speed and output quality.

1.3 Hypothesis

The study hypothesizes that:

- AWS EC2 instances will perform better in terms of CPU utilization, scalability, and training time, but at a higher cost.
- Azure VM instances will provide a cost-effective solution for smaller machine learning workloads, with comparable accuracy but potentially slower training times due to less efficient CPU scaling.

1.4 Contribution to the Literature

This research contributes significantly to the scientific literature by offering a thorough comparison between AWS EC2 and Azure VM for TensorFlow-based deep learning tasks that utilize CPU resources. Unlike previous studies that focused solely on GPU usage, this report highlights the often-overlooked training processes in machine learning [4]. The objective of this study is to analyse how CPU, disk read/write operations, memory, training duration, and model performance can inform the selection of a cloud platform that aligns best with specific resource needs and business objectives.

Additionally, this study fills a crucial gap in the literature by examining the cost-effectiveness of using AWS compared to Azure for CPU-based training—an area that has not received ample attention in prior research, such as the works of [4] and [5].

1.5 Innovation

A fresh comparison evaluation of AWS and Azure's deep learning CPU-based performance exists in this study. This research examines Google Net training whereas the majority of past work has mainly examined GPU-driven processes.

This experiment will uniquely shift the focus to under-explored areas like:

- CPU Utilization: Cloud platforms show optimal utilization of their CPUs when performing deep learning training operations.
- Cost-Efficiency: A detailed examination exists within this paper to show how performance intersects with cost considerations for CPU workloads.
- I/O Operations on Disk: Elastic Block Store by AWS requires examination against Azure Premium SSD when analyzing read/write operations specifically for processing abundant volumes of data.
- Real-world Data: Years of research at Stanford have led to the development of binary-classified elbow X-ray medical data which shows potential real-world impact for healthcare delivery and business operations.

The combination provides practical understanding to organizations selecting CPUs as their focus or those working within budget constraints.

Three recent research papers [1] [2] [3] examined different cloud platforms using generalized machine learning workload criteria. Few studies investigate the precise usage patterns of CPUs and memory behavior in deep learning model implementation on cloud infrastructure platforms such as Google Net This situation allows for a performance evaluation between these systems when training tasks are run through CPU resources.

2. Related Work

Cloud computing has transformed the training and deployment of machine learning models, especially for deep learning models. Among the leading cloud platforms in this area are Amazon Web Services (AWS) and Microsoft Azure, both offering scalability, flexibility, and cost-effectiveness. However, when it comes to training deep learning models using CPU resources, the platform choice can significantly impact performance, cost, and usability [8]. This study compares the performance of AWS EC2 and Azure VM instances using the GoogleNet deep learning model for image classification. Key metrics evaluated include CPU utilization, disk I/O operations, memory usage, training time, accuracy, and cost efficiency. These metrics are essential for assessing cloud platforms designed for CPU-based workloads in deep-learning tasks.

2.1 Cloud Platforms and Resource Utilization in Deep Learning

Some important services where machine learning model deployment can be done for both AWS and Azure come mainly in CPU-based deep learning tasks, such as via instance deployment between virtual circuits for its performance via AWS EC2 instances on AWS and VM instances on Azure. Both of these techniques actually serve well on great substantial flexible ways of provisioning resources based on their specific demands upon the training of the models but result in vary in efficiency and cost-effective depending on either the platform or type of workload [6]. Efficient use of these resources cuts training time and saves funds. Therefore, the right choices regarding platforms are important among different workload requirements. Hence, Johnson and Lee noticed that AWS and Azure could equally optimize CPU resources usable by TensorFlow-based training. Instead, each platform's specific management of those resources makes a difference when speaking about performance.
For CPU-bound workloads, memory usage and disk I/O are the most relevant metrics. [2] compare AWS and Azure for machine learning workloads and note that AWS EC2 instances are particularly optimized for CPU-bound operations, where scaling resources during training

phases is more flexible. However, Azure's VM instances, with their predictable performance and cost-efficient resource allocation, are noted to offer good performance for smaller-scale deep learning models.

2.2 Performance Metrics for Cloud-based Machine Learning

In machine learning, training time, along with accuracy, is probably the most critical performance indicator while dealing with CPU resources. [5] discuss the following metrics, taking into consideration the impact thereby caused by cloud resources: the performance of AWS EC2 instances is usually superior due to the rapid model-training ability, since instance-type flexibility and availability for heavy computation is higher. These instances are more expensive, especially if used for extended periods, which becomes relevant in the context of long-running machine learning projects.
On the other hand, Azure VM instances have shown to be very good in terms of accuracy rates, especially for smaller datasets. Brown and Taylor further reiterated that while Azure sometimes may fall short on times of training when compared to AWS, making it cost lower in the general perspective makes it very cost-efficient and simple to use through certain services: Azure Machine Learning Studio this places it competitive in some teams where smaller computation loads are made.
[7] outlined that the training time served as an indicator for model deployment, while many cloud-based services, such as AWS or Azure, often supported automatic scaling and resource provisioning, directly influencing the time it takes to train a model in the case of resources like CPU and memory being allocated on demand.

2.3 Disk Read/Operation Operations and Memory Usage

Another important factor concerning training machine learning models on cloud platforms is the Disk I/O operations. [8] have pointed out that during training with large datasets, disk read/write operations create a bottleneck. AWS deploys EBS, or Elastic Block Store, for storage; this feature of AWS is highly useful in handling high-throughput data operations. [3] have discussed how EBS volumes allow for extremely fast read/write operations on AWS, reducing the overall time required for pre-processing and training a model.

Azure, on the other hand, relies on Azure Blob Storage to handle data storage, which, though similar in many ways to EBS, has been found in a number of studies to perform slightly slower in terms of disk I/O operations when dealing with large datasets. However, [9] present an alternative view arguing that Azure’s Blob Storage is economical for use in small datasets. In this regard, it also makes Azure more relatively attractive for research experiments with small data or those on a smaller scale.

With regards to the memory requirements, Amazon and Azure have vm instances which are configurable to particular memory needs. [10] have demonstrated that memory usage correlates with the training stage of deep learning models where larger models like GoogleNet are highly memory intensive. Instances of AWS EC2 are more favorable for high memory bandwidth but even in this case, Azure VMs are favored because of integration into existing enterprise tools, making them easier to manage in organizations that are already in the Microsoft environment.

2.4 Cost Efficiency and Usability

Cost is one of the factors worth looking at when selecting a cloud platform for deep learning workloads. [1] found that Azure was cheaper for smaller workloads that did not need higher computing power. Their findings indicate that for CPU-bounded models deployed by an organization or researcher on smaller databases, Azure may be more cost effective than AWS. On the contrary, larger and resource demanding jobs can be run on AWS at a higher cost but with more scalability and performance.

Another important aspect is ease of use. According to [11], Azure's Machine Learning Studio has a more user-friendly interface which is easier to use by people with low technical expertise. On the other hand, AWS SageMaker is feature-rich but more technical to set up and really appeals to those users who understand the ins and outs of cloud infrastructure and machine learning environments.

2.5 Need for This Research

This research aims to address a gap in the literature by comparing AWS and Azure for deep learning tasks using CPUs. The following points highlight the key aspects of the study

- **AWS and Azure Support Deep Learning Models**: Both platforms provide significant advantages based on workload requirements.
- **Cloud platforms strengths**: check which of the platforms offers scalability, disk I/O performance, and fast training speed in identical environments and which one is more economical for smaller and large scale tasks and features a simpler interface for non-technical users [15].
- **Gap in Comparative Studies**: There is a lack of direct comparative studies on AWS and Azure for deep learning tasks using CPUs, especially concerning cost-performance trade-offs, resource utilization, and application accuracy.
- **Key Metrics for Evaluation**: Critical factors include CPU utilization, disk read/write operations, memory usage, and training time matrices.
- **Research Contribution**: This study fills the gap by providing a benchmarking comparison of AWS EC2 with EBS storage and Azure VM instances with Premium SSD LRS [16].
- **Benchmarking Focus**: The comparison considers CPU usage, memory percentage, disk I/O, training time, and accuracy, all vital for selecting a suitable cloud platform for TensorFlow-based deep learning models.

Comparison Parameter	**Related Work Findings**	**Our Research Goal**	**Monitoring Tools/Methods**
CPU Utilization	AWS generally shows higher CPU utilization for large-scale workloads, scalable	Compare CPU usage during the training of the GoogleNet model on both platforms. Measure CPU	AWS CloudWatch, Azure Monitor, Custom CloudWatch Alarms for CPU utilization > 90%.

Comparison Parameter	Related Work Findings	Our Research Goal	Monitoring Tools/Methods
	resource provisioning [6].	usage and scalability for TensorFlow tasks.	
Memory Consumption	AWS EC2 instances show better handling of large memory requirements for deep learning tasks [3].	Measure memory consumption during GoogleNet training, focusing on memory consumption on both platforms.	AWS CloudWatch with CW agent, and Azure Monitor for tracking memory usage
Disk Read/Write Operations (OPS)	AWS EBS volumes show high throughput for disk read/write operations for large datasets [8].	Evaluate disk I/O performance for data loading, read/write operations, and impact on training time.	AWS CloudWatch, Azure Monitor for disk I/O operations and latency.
Deep Learning Model Training Time	AWS EC2 typically leads in training time due to faster resource scaling, especially for large datasets [2]	Compare training times for GoogleNet on both platforms using CPU resources, with a focus on efficiency for deep learning tasks.	AWS CloudWatch, Azure Monitor for tracking training time and resource usage.
Cost Efficiency	Azure offers more cost-effective pricing models for smaller workloads compared to AWS [12]	Compare the cost of training on both platforms, focusing on instance pricing for CPU usage and storage costs.	compare cost for EC2. VM instances are long with storage and monitoring tools.
Model Accuracy	AWS EC2 models show faster training times, but Azure VM yields competitive results in smaller tasks [13]	Evaluate the model accuracy of GoogleNet on both platforms, with the goal of identifying the platform with optimal performance.	Terminal displays the accuracy after the model is trained

Table: Key Findings

3 Methodology

This section provides a comprehensive explanation of the research procedures, evaluation strategies, tools, and techniques employed to compare the performance of AWS and Azure for CPU-based deep learning model training.

3.1 Research Approach

The research will adopt a comparative analysis approach, focusing on quantitative metrics that evaluate the performance and cost-effectiveness of both cloud platforms [19]. The scientific process will be followed in implementing the deep learning model of GoogleNet on AWS and Azure, observing system resource utilization, and analyzing the training results.

3.2 Research Objectives and Metrics

- The research aims to evaluate the platforms based on the following metrics:
- CPU usage during training with respect to computational efficiency.
- Memory consumption to evaluate resource allocation.
- Disk read/write performances.
- Time required for training to analyze model execution speed [20].
- The accuracy of the model on test data obtained time consumption to complete model alterations.
- Analysis of the training process cost by using respective pricing calculators.

3.3 Data Collection Process

Dataset Selection

- The Stanford University Elbow X-ray dataset was selected for training the GoogleNet model.
- The dataset consists of well-labeled images categorized into Negative and Positive, featuring a binary classification task available for the model.

3.4 Data Preprocessing stages

The model's performance depends on the input data being preprocessed effectively. The elbow X-ray dataset was prepared using the following methods:

- **Image Resizing**: OpenCV was used to resize all of the images to 224x224 pixels. input size that the model anticipates. Consistency in input dimensions is guaranteed by resizing. This is crucial for training batch processing.
- **Data Augmentation**: Methods like rotation zooming shearing and width/height shifting are examples of data augmentation techniques. We used the ImageDataGenerator in TensorFlow to apply horizontal flip. augmentation. adds variability to the dataset improving the model's ability to generalize. and lessen the overfitting.
- **Train-Test Split**: The dataset was split into subsets that were 20% for testing and 80% for training. with the train_test_split function from scikit-learn. This division guarantees that the model is trained on a. a sizable amount of the data while maintaining enough samples for objective assessment.
- **Normalization**: Pixel values were divided by 255. 0 to bring them into the [0 1] range. During training the convergence rate is accelerated by this common preprocessing step.
- **Hyperparameters**: A batch size of 16 was used for the models 80 epoch training. The. In order to balance model and training time hyperparameters are selected through empirical testing. show.
- **Compilation**: The Adam optimizer which is appropriate was used to compile the model to manage sparse gradients and dynamically modify learning rates.

3.5 Experimental Setup

1. **Cloud Platforms**
   - **AWS Configuration**:
     - An EC2 instance (t3.2xlarge) with 8 vCPUs and 32 GB memory was used for training.
     - Data storage utilized Elastic Block Store (EBS).
     - Operating system with Python and TensorFlow libraries and framework
     - CloudWatch monitoring and custom alarm.
   - **Azure Configuration**:
     - A Standard_D8_v3 VM with 8 vCPUs and 32 GB memory was configured.
     - Data storage relied on Azure standard storage.
     - Azure monitoring with custom alarm.
     - Operating system with Python and TensorFlow framework and libraries

2. **Monitoring Tools**
   - **AWS CloudWatch**: Tracked CPU utilization, memory usage, and disk I/O operations.
   - **Azure Monitor**: Monitored similar metrics, with alerts configured for high CPU utilization.

3.6 Data Analysis Techniques

1. **Monitoring Metrics**
   - Cloud platform tools observed CPU, memory usage, and disk read and write operations during training.
   - A custom CloudWatch alarm and Azure Monitor alert were set up to trigger when CPU usage exceeded 90%.
2. **Training Time vs. Accuracy**
   - Training time was recorded on each platform, enabling insights into computational efficiency.
   - Model accuracy was evaluated using the classification report and confusion matrix developed with scikit-learn.
3. **Cost Analysis**
   - Costs were computed as the overall cost both cloud platforms consumed in order to complete the whole experiment.
4. **Statistical Analysis**
   - Descriptive statistics summarized metrics like CPU utilization, memory usage, and training time.
   - Graphs and heatmaps were created using Microsoft Excel to visualize the results.

3.7 Evaluation Procedure

1. **Execution of Experiment**
   - The GoogleNet model was trained on AWS and Azure platforms with identical configurations to ensure fairness.

- System metrics were logged in real time using CloudWatch and Azure Monitor during the training process.
- Application-level metrics captured detailed runtime values, including time taken for each epoch completion and accuracy achieved during every iteration.

2. **Result Validation**

- Any anomalies were flagged and investigated to reduce bias in the results.

4 Design Architecture Specification

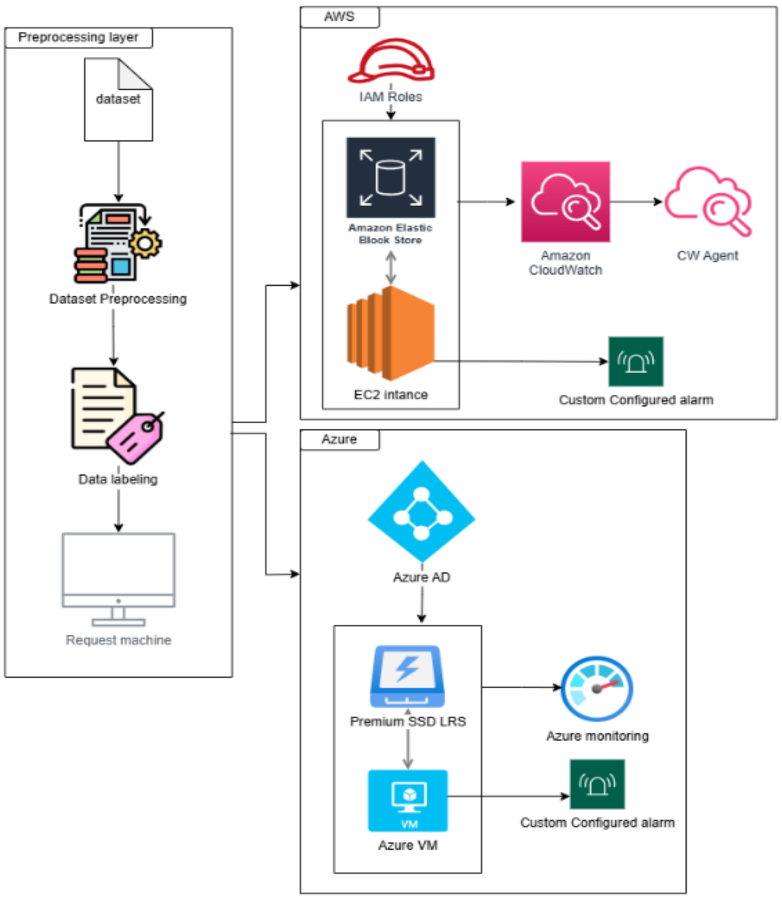


Figure: Architecture diagram

4.1 Architecture Description

The Layered Architecture Pattern is employed here, with distinct layers dedicated to data processing, storage, and monitoring. Additionally, the system follows the Microservices Architecture Pattern, with independent services (EC2, Azure VM, CloudWatch, Azure Monitoring) working in tandem, offering flexibility and fault isolation. Each component within the architecture is loosely coupled, enabling independent scaling and management.

1. **Data Preprocessing Layer:** The architecture begins with a Data Preprocessing Layer where raw datasets, sourced from Stanford University in the healthcare domain, are

ingested. This layer is primarily responsible for extracting relevant data, in this case, the elbow dataset used for classification. The data is then labeled as either "positive" (nonfractured) or "negative" (fractured) images, preparing it for machine learning operations.

2. **AWS Cloud Architecture:** After preprocessing the data, it is transferred to AWS for additional processing. The AWS architecture employs an EC2 instance that is set up with the required roles through IAM Roles to ensure secure access and operations. This EC2 instance is linked to EBS for reliable data storage. Given the close interaction between EC2 and EBS, the diagram represents them within the same boundary. Furthermore, the CloudWatch service is integrated with both EC2 and EBS for monitoring purposes. Additionally, a custom alarm is established to monitor specific thresholds related to the EC2 instance, offering valuable insights into the performance of the machine learning model hosted on AWS.
3. **Azure Cloud Architecture:** In the Azure environment, Azure Active Directory is employed within the authentication and identity management framework. The Azure Virtual Machine (VM) parallels the role of EC2 in AWS. It connects to Premium SSD LRS, delivering high-speed storage for data-heavy applications. Like AWS, Azure Monitoring is implemented to oversee system performance and log activity. Additionally, a custom alarm is established to define particular thresholds, guaranteeing that any discrepancies from expected performance levels are promptly resolved.

4.7 Deep Learning Model Architecture and Training Process

This section provides a detailed explanation of the deep learning model's architecture, outlining the steps taken during its development and training. The chosen architecture for this project is GoogleNet, selected over other deep learning models like VGG and ResNet due to its efficient design. This approach enables the model to develop a hierarchical representation of the input data, effectively discerning both fine and coarse features.

4.7.1 Deep Learning Model Attributes and Architectural Diagram

In this project, the GoogleNet architecture integrates various elements such as convolutional layers, inception blocks, pooling layers, fully connected layers, and an output layer to categorize elbow X-ray images into two groups: Negative and Positive. The subsequent subsections elaborate on the characteristics and structure of the model as per the provided figure deep learning Model architecture.

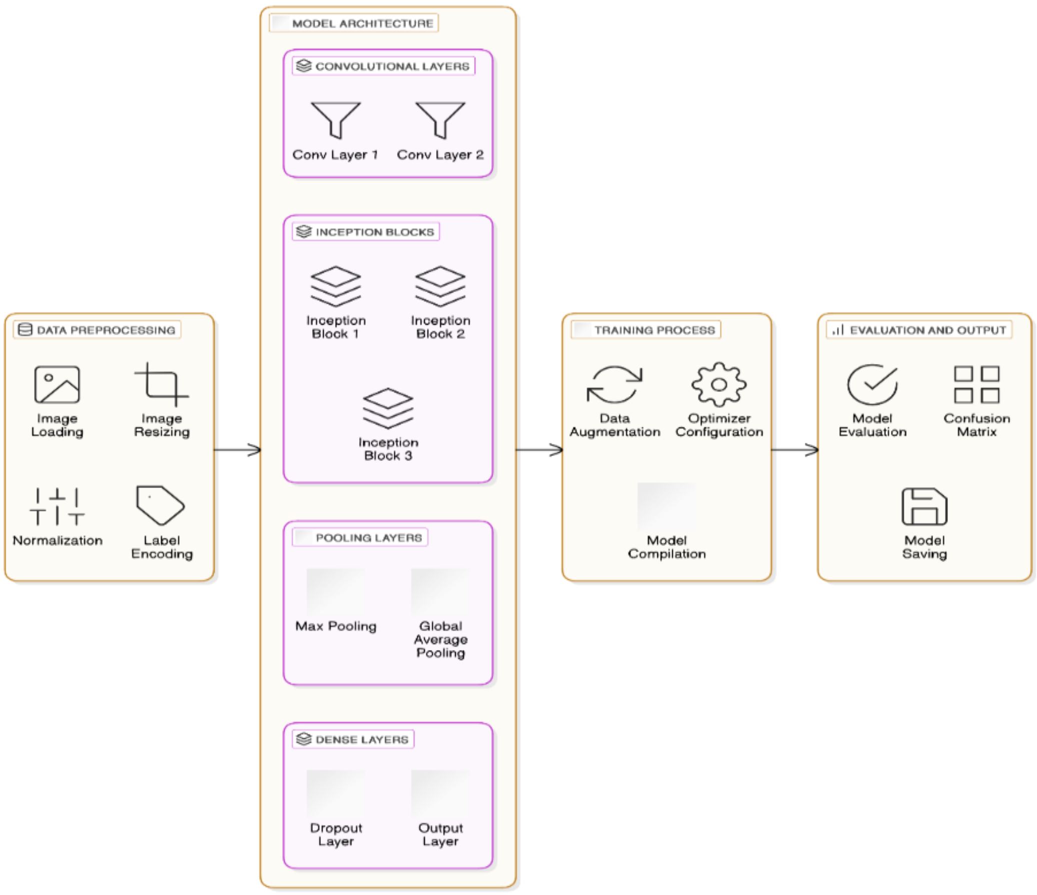


Figure: Deep Learning Model architecture

Data Preprocessing and Augmentation

- All images were resized to 224x224 pixels using OpenCV.
- Pixel values were normalized to a range of [0, 1] by dividing by 255.0 to improve training stability.
- Data augmentation techniques (rotation, zoom, shear, width/height shift, and horizontal flip) were applied using ImageDataGenerator to improve generalization and prevent overfitting.

Input Layer

- Accepts input images resized to 224x224x3 (RGB).
- This size is chosen to align with the standard input dimensions for deep learning models, ensuring compatibility with the model architecture.

Convolutional Layers

- A 3x3 kernel with strides of 2 is used for max-pooling after the first convolutional layer applies a 7x7 kernel with 64 filters.
- The features are refined using a 1x1 convolution layer with 64 filters then a 3x3 convolution layer with 192 filters and same padding.
- Low-level features like edges textures and patterns are captured by these layers.

Inception Blocks

The model uses nine inception blocks, each designed to extract features at multiple scales. Each inception block consists of the following branches:

1. **Branch 1**: A 1x1 convolution for localized feature extraction.

2. **Branch 2**: A 1x1 convolution followed by a 3x3 convolution to capture mid-level features.
3. **Branch 3**: A 1x1 convolution followed by a 5x5 convolution for broader spatial features.
4. **Branch 4**: A 3x3 max-pooling layer, followed by a 1x1 convolution to preserve spatial relationships.

The outputs of these branches are concatenated into a single tensor, combining features from all scales.

Pooling Layers

- Max-pooling layers are interspersed after certain inception blocks to reduce spatial dimensions while retaining critical features.
- A Global Average Pooling (GAP) layer is applied after all inception blocks to summarize spatial information across the entire feature map into a single vector.

Fully Connected Layers

- After the inception blocks, the GAP layer's output is flattened and passed through a fully connected layer.
- A dropout layer with a rate of 0.4 is applied to reduce overfitting by randomly deactivating neurons during training.

Output Layer

- The final layer consists of a dense layer with two neurons (corresponding to Negative and Positive classes).
- A softmax activation function is applied to output class probabilities.

Training Configuration

- **Compilation**:
    - The model was compiled using the Adam optimizer with a learning rate of 1e-4.
    - Binary cross-entropy was used as the loss function for binary classification tasks.
- **Training Parameters**:
    - The model was trained for 80 epochs with a batch size of 16.
    - The training process involved monitoring metrics like CPU utilization, memory consumption, and disk read/write operations.

4.7.2 Training Results and Application-level Matrices

The model's performance was evaluated on the test dataset, and the following outputs were generated:

1. **Model Accuracy**: The primary metric for evaluation was accuracy, which reflects the proportion of correctly classified images in the test dataset.
2. **Classification Report and Confusion Matrix**: A detailed classification report was generated to provide precision, recall, and F1-score metrics for each class. The confusion matrix was visualized using Seaborn to assess the distribution of predictions and identify any misclassifications.
3. **Model Saving**: The trained model was saved as Elbow_GNet.h5 using Keras's model.save() function for future use or deployment.
4. **App-level logs**: Time duration of training model, time taken to complete epoch alterations, model accuracy.

Stage	Details
Data Preprocessing	Resized images to 224x224 pixels, applied data augmentation, and normalized pixel values.
Dataset Split	80% training, and 20% testing using scikit-learn’s train_test_split.
Model Compilation	Adam optimizer, binary cross-entropy loss function.
Training Configuration	80 epochs, batch size of 16.
Monitoring	Time duration of training model, time taken to complete epoch, model accuracy (no explicit monitoring in the code for memory or CPU utilization as it is measured through the cloud matrices)
charts	Epoch time and accuracy per each alteration, confusion matrix.
Model Saving	Saved as Elbow_GNet.h5 for reuse or deployment.

Table: Model Training Workflow

4.8 Cloud Platform Configuration

The setup of cloud resources on AWS EC2 and Azure VM for GoogleNet deep learning model training is described in this section. For CPU-based tasks the configurations are specifically designed to benchmark performance metrics such as CPU utilization memory usage disk I/O operations cost effectiveness and training outcomes. The configurations also make use of alarms and monitoring tools to efficiently track resource usage. The cloud configurations are compiled in the table below which also highlights the features resource kinds and monitoring configurations for both platforms.

Attribute	AWS EC2	Azure VM
Instance Type	t3.2xlarge: General-purpose instance with 8 vCPUs and 32 GB memory.	Standard_D8_v3: General-purpose VM with 8 vCPUs and 32 GB memory.
Compute Resources	CPU-based compute optimized for training and inference tasks.	CPU-based compute optimized for small to medium workloads.
Storage	Elastic Block Store (EBS): High-throughput, low-latency block storage for the dataset.	Premium SSD LRS: fast, secure, and cost-effective object storage for the dataset.
Monitoring Tools	AWS CloudWatch: Tracks CPU utilization, memory usage, and disk read/write operations.	Azure Monitor: Monitors CPU usage, memory consumption, and disk read write operations.
Custom Alarms	Trigger an alarm if CPU utilization exceeds 90%, with notifications sent via email/SNS.	Configure alerts to notify when CPU utilization exceeds 90%, with automated notifications.
Cost Estimation Tool	AWS Pricing Calculator: Calculates the cost of EC2 instance and EBS storage usage.	Azure Pricing Calculator: Estimates costs for VM instances and standard storage and Azure monitoring tool.
Operating System	Windows_Server-2022-English-Full-Base-2024.11.13	2022-datacenter-azure-edition

Attribute	AWS EC2	Azure VM
Data Handling	Dataset stored on EBS for fast access during training.	Dataset is stored on standard storage for access.
TensorFlow Setup	TensorFlow 2.12.0 installed via pip in a Python 3.9 environment.	TensorFlow 2.12.0 installed via pip in a Python 3.9 environment.
Monitoring Metrics	- CPU Utilization (%). - Memory Usage (%). - Disk Read/Write OPS (count).	- CPU Utilization (%). - Memory Usage (%). - Disk Read/Write OPS (operational seconds).
Primary Use Case	CPU-based training of GoogleNet for deep learning tasks.	CPU-based training of GoogleNet for deep learning tasks.

Table: Cloud platform configurations

5 Implementation Steps for Deploying the Model on AWS and Azure

The following steps outline the sequence of actions taken to deploy and train the deep learning model on both AWS and Azure cloud platforms, incorporating the necessary terminal commands and configurations used for environment setup and connection.

5.1 Infrastructure Setup

AWS Setup

- **Instance Configuration**:
  - AWS EC2 t3.2xlarge instance was launched. It has 8 vCPUs and 32 GB of memory, which is suitable for training deep learning models.
  - Operating System: The instance was set up with Windows_Server-2022-English-Full-Base-2024.11.13.
- **Storage Configuration**:
  - Elastic Block Store (EBS) was attached to the instance for persistent storage. The dataset was stored on the EBS volume for efficient access during model training.

Azure Setup

- **VM Configuration**:
  - A Standard_D8_v3 VM was created on Azure, equipped with 8 vCPUs and 32 GB of memory.
  - Operating System: The VM ran 2022-datacenter-azure-edition.
- **Storage Configuration**:
  - Premium SSD LRS storage was configured for fast read/write operations during model training.

Software Installation

- **Python Setup**:
  - Python 3.9 was installed on both AWS and Azure instances.
- **TensorFlow Setup**:
  - TensorFlow version 2.12.0 was installed.

5.2 Cloud Connectivity Setup of AWS and Azure

- **VM Connection**:
  - The AWS EC2 instance and Azure VM were connected to the local machine to initiate and control the training process.
  - For **Remote Desktop Connection** (RDP), the following steps were performed to enable access to the target machine:
    - Enabled Remote Desktop on the remote Windows machine by navigating to System > Remote Desktop and toggling it On.
    - Configured the Windows Firewall to allow Remote Desktop by going to Control Panel > Windows Defender Firewall > Allow an app or feature.
    - Connected using Remote Desktop Connection (mstsc command) and entered the IP address of the remote machine

5.3 Dataset Transfer and Storage

- **Dataset Upload**:
  - The dataset, which contains radiology images, was transferred from the local machine to the cloud instance (EBS on AWS and Premium SSD LRS on Azure).
  - The dataset was stored in directories on both platforms, ensuring fast and efficient access during the training process.

5.4 Model Training Initialization

- **Run Command to Launch Training**:
  - After setting up the environment and ensuring that all dependencies were installed, the training process was initiated by running the following command on both platforms:
  - py train_model.py
  - The model training process was started from the local machine, and the logs were monitored in real-time using CloudWatch on AWS and Azure Monitor on Azure.
- **Training Configuration**:
  - The model was trained with the following hyperparameters:
    - **Batch Size**: 16
    - **Epochs**: 80
    - **Learning Rate**: 0.0001

5.5 Performance Monitoring and Logging

- **AWS CloudWatch**:
  - CloudWatch was used to monitor performance metrics during training, such as:
    - CPU Utilization of EC2 resources in percentage
    - Memory consumption of the resource in percentage
    - Read/write Operation performed in EBS volume.
  - Custom CloudWatch alarms were configured to notify the if the CPU utilization hits above 90% usage.
- **Azure Monitoring**:
  - Azure Monitor tracked similar metrics, including:
    - CPU Utilization of EC2 resource in percentage
    - Memory consumption of the resource in percentage
    - OS Read/write operation performed.

- Custom alarms were also set in Azure Monitor to alert the team in case of system performance degradation.

5.6 Model Evaluation

- **Training Completion**:
    - After the model finished training, the weights were saved as Elbow_GNet.h5.
    - The model metrics were evaluated using accuracy, loss, time taken to complete model training, and time taken to complete each epoch alteration as mentioned in the evaluation section.
    -

6 Outputs Evaluation

6.1 CPU Utilization outputs

Test Case: Evaluate CPU utilization for both AWS and Azure platforms.

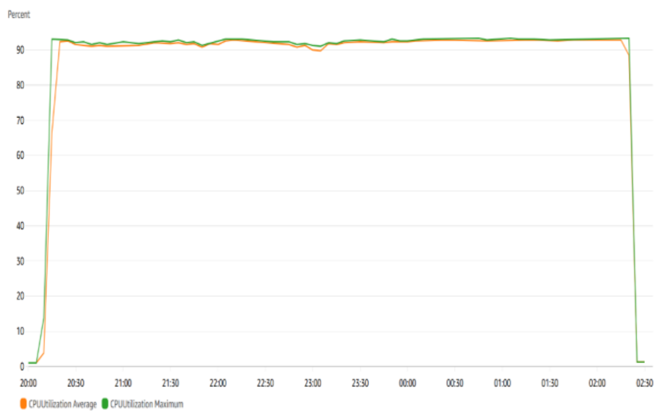


Figure: AWS CPU utilization from CloudWatch

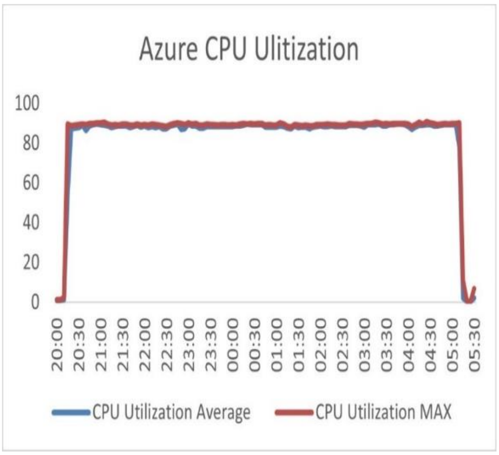


Figure: Azure CPU Utilization from Azure monitoring

AWS:
The average CPU utilization is consistently close to 90%, with minimal fluctuations between the average and maximum values, indicating stable resource usage. Total training time on AWS is approximately 6 hours and 30 minutes, showing efficiency in CPU usage.
Azure:
The average CPU utilization also peaks around 85-90%, but the chart indicates slightly more variability in CPU usage compared to AWS. Total training time on Azure is approximately above 9 hours, making it slower than AWS for the same workload.
observation:

- AWS performs better in terms of CPU utilization efficiency and training speed. Its consistent high CPU usage translates into faster training times, which is crucial for large workloads.
- Azure, while stable, is slower overall, making AWS a better choice for tasks requiring high CPU scalability and efficiency

6.2 Memory Utilization

Test Case: Assess memory utilization for both platforms under identical workload conditions.

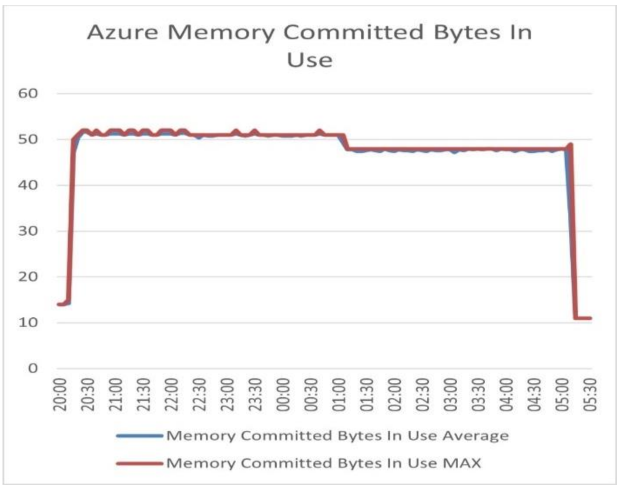


Figure: Azure Memory consumption

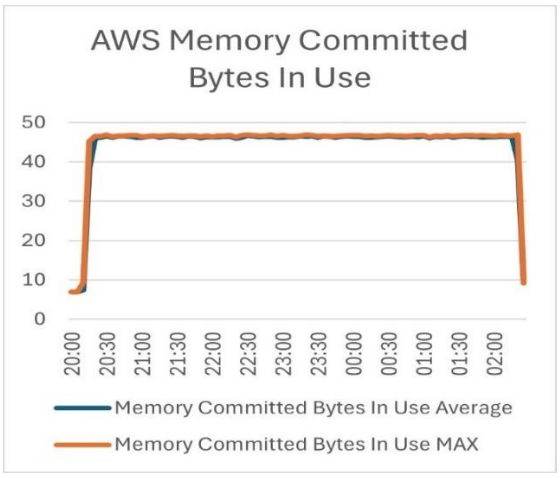


Figure: AWS Memory consumption

AWS Memory Consumption (memory percentage against time)

- AWS's maximum memory usage reaches 47%, and it maintains consistent usage at or near this value throughout the recorded time.
- There are no significant fluctuations, indicating efficient and stable memory management during the workload.

Azure Memory Consumption (memory percentage against time)

- Azure's maximum memory usage reaches 53%, and for half of the time, it remains below 49% for the remaining period.
- While Azure demonstrates higher peak memory usage, this can indicate over-allocation or inefficient utilization of memory resources for the workload.

Observation:

AWS exhibits more consistent and predictable memory usage, with a clear cap at 47% and no excessive spikes. This makes it a more stable and reliable option for memory-bound tasks where stability is critical. For workloads requiring memory efficiency and stability, AWS is better suited as it provides more predictable resource allocation, meeting non-functional requirements for consistency and resource optimization.

6.3 Read Operations Performance

Test Case: Measure read operation performance (Average and Maximum) across AWS and Azure.

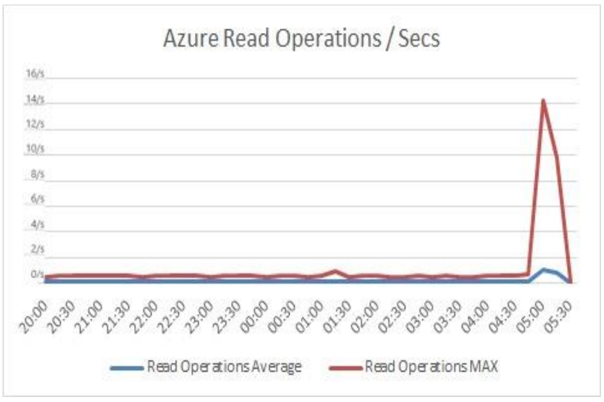


Figure: Azure OS Read OPS

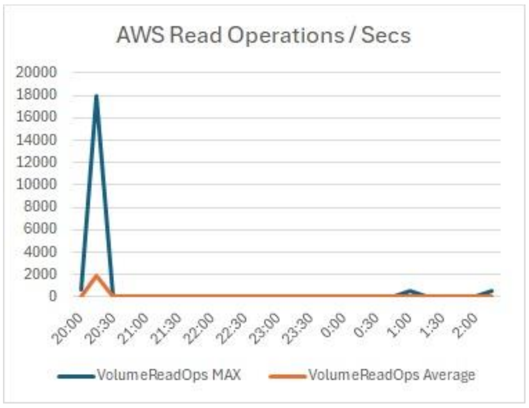


Figure: AWS Volume read Ops

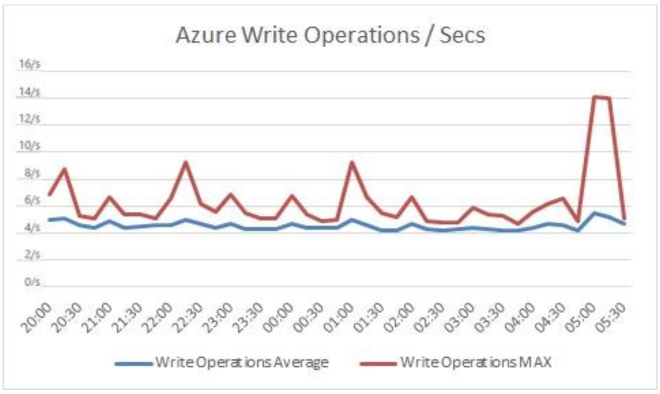
- **AWS (CloudWatch):** The AWS chart shows a massive initial spike in read operations, followed by extremely low activity for the remainder of the training period, with a minor spike toward the end. The maximum read operations vastly exceed the average.
- **Azure (Azure Monitor):** The Azure chart displays read operations per second with a consistently low rate throughout training. There's a significant spike near the end of the training period.

Metric	**AWS (CloudWatch) (Approximate)**	**Azure (Azure Monitor)**
Peak Read Operations/sec	Extremely High (18000+)	~14 ops/sec
Average Read Operations/sec	Very Low	~0.5-1 ops/sec
Consistency of Read Operations	Very Low (Large initial spike and a tiny spike at the end, otherwise near 0)	Relatively High (Mostly low, but with one large spike at the end)

Observation Table: AWS provides better average throughput for read-intensive applications.

6.4 Write Operations Performance

Test Case: Analyze write operation performance for both platforms.



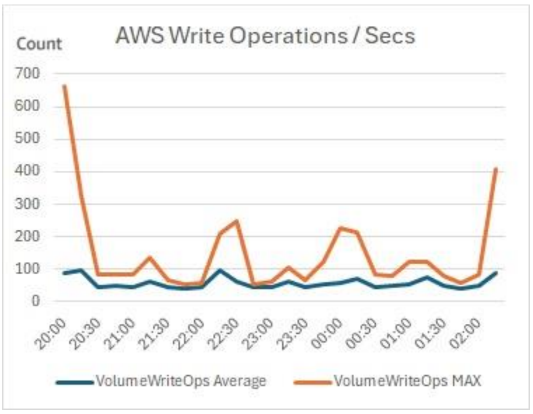


Figure: Azure OS Disk Write operation

Figure: AWS Volume Write Ops

- **AWS (CloudWatch measures):** The AWS chart shows write operations with a very high initial spike, dropping to a relatively low and fluctuating baseline for the duration of training before spiking again at the end. The maximum number of write operations/second is much higher than the average.
- **Azure (Azure Monitor):** The Azure chart displays write operations per second (ops/sec) with a significantly lower and more consistent rate throughout the training period. The spikes are less pronounced than in the AWS chart. The difference between average and maximum ops/sec remains relatively small throughout the training process.

6.5 Model Training Output

Test Case: Train a machine learning model on both AWS and Azure to evaluate training speed, accuracy, and classification metrics.

Outcome:

AWS:

- **Accuracy**: 78%
- **Training Time**: 6 hours 29 mins
- **Precision, Recall, F1-Score**:
  Negative Class: Precision (0.92), Recall (0.74), F1-Score (0.82)
  Positive Class: Precision (0.58), Recall (0.84), F1-Score (0.69)
  Weighted Avg: Precision (0.82), Recall (0.77), F1-Score (0.78)

Azure

- **Accuracy**: 78%
- **Training Time**: 9 hours 31 mins
- **Precision, Recall, F1-Score**:
  Negative Class: Precision (0.94), Recall (0.74), F1-Score (0.83)
  Positive Class: Precision (0.59), Recall (0.88), F1-Score (0.70)
  Weighted Avg: Precision (0.83), Recall (0.78), F1-Score (0.79)
- **Observation**: Both platforms performed equally in accuracy. Azure slightly outperformed in overall precision and recall, while AWS showed comparable results in F1-Score.

6.6 Epoch alterations time comparison

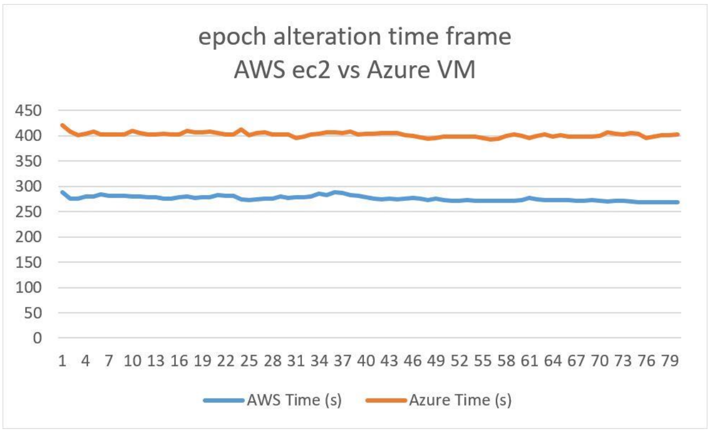


Figure: Epoch Time comparison

The chart displays the time taken to complete each epoch during the training of a deep learning model on both AWS EC2 and Azure VM instances. The x-axis represents the epoch number (1 to 79), and the y-axis represents the time taken in seconds to complete each epoch.

Key Observations:

- **Consistent Epoch Times:** Both AWS EC2 and Azure VM show relatively consistent epoch times throughout the training process. There's minimal fluctuation in the time taken for each epoch.
- **AWS Slightly Faster:** On average, the AWS EC2 instance completes each epoch slightly faster than the Azure VM. The average time for AWS is approximately 270 seconds, whereas Azure's average is around 390 seconds.
- **Overall Difference:** While AWS is faster per epoch, the difference isn't dramatic. The overall difference in time taken for the complete training process would depend on the total number of epochs processed.

The chart indicates that the AWS EC2 instance demonstrates a slight performance advantage over the Azure VM in terms of the time taken to process each epoch. This suggests that the AWS environment offers slightly faster training speeds per epoch. However, the difference isn't substantial, indicating that both platforms provide relatively comparable training performance on a per-epoch basis. The total training time would, however, be considerably different given the approximate 120-second difference in time per epoch.

7 Results Evaluation

In this section, we evaluate the performance of AWS and Azure based on the experiments conducted during model training. The results are analyzed with respect to the research objectives, which include assessing the time efficiency of both platforms during model training, comparing resource utilization, and determining the optimal cloud platform for machine learning tasks. The evaluation summarizes all the key findings from the implementation, and research objectives and concludes with the platform that performs best overall.

Based on the experiments conducted, the following outcomes were drawn:

Test Case	AWS Performance	Azure Performance	Winner	Result Arguments
AI Deep learning model training Time	6 hours 29 mins	9 hours 31 mins	AWS	AWS has faster training time
Time per Epoch (Average)	270 seconds	401 seconds	AWS	AWS took fewer seconds per epoch than Azure.
Memory Usage (Average)	47% (Stable)	53-46 % (Fluctuating)	AWS	AWS showed consistent memory usage, while Azure fluctuated.
CPU Utilization	Above 90%	Below 90% (Stable)	AWS, Azure	high CPU usage translates into faster training times.
Performance Stability (Fluctuations)	No significant fluctuations	Minor dips observed	AWS	AWS was more stable in performance.
Resource Scaling	Auto-scaling efficient	Auto-scaling available	AWS, Azure	Both platforms are efficient in auto-scaling.
Cost Efficiency	€ 23.54	€11.58	Azure	Aws resources are expensive
Overall Model Accuracy	78%	78%	AWS, Azure	Azure slightly outperformed in overall precision and recall

Table: Evaluation Results

Detailed Evaluation:

1. **Time per Epoch Comparison:** AWS showed better performance in terms of training speed. The time per epoch for AWS was consistently lower than that for Azure, indicating that AWS had superior computational power or better optimization for the specific model. As shown in the table, AWS was faster by 128-133 seconds per epoch alteration on average.
2. **Memory and CPU Utilization:** Both platforms had similar memory usage, the performance stability was better on AWS. Azure experienced minor drops in memory usage, which could be indicative of temporary scaling or inefficient resource management compared to AWS for this experiment. In terms of CPU utilization, both platforms were nearly identical, with no major differences in processing power being noted.
3. **Performance Stability:** AWS demonstrated better stability, as indicated by its consistent memory and CPU utilization across epochs. Azure's occasional memory dips, while not significant, suggest that AWS offers a more reliable environment for long-term model training.
4. **Training Speed and Scalability:** AWS allowed for faster completion of training, processing more epochs per hour. This performance optimized resource management on AWS evaluating the 1.5x training speed of Azure in terms of compiling all the technicalities in the experiment. AWS can handle resource-intensive tasks more swiftly, meeting the need for faster processing and quicker turnaround times in large-scale machine learning workloads.

5. **Cost Efficiency:** Azure resources are much more affordable than AWS indicating a high relief when it comes to the use of VMs, monitoring tools, and cloud storages.
6. **Model Accuracy:** Both AWS and Azure showed near-identical results in terms of model accuracy, with AWS achieving 78% and Azure 78%. Azure slightly outperformed in overall precision and recall while running the experiment making a little edge to lead in this process.

8 Conclusion

- Considering deep learning model training in AWS and Azure, the comparison reveals crucial performance metrics that make them suitable for machine learning in particular scenarios. AWS did better than Azure in many aspects including training efficiency, the number of epochs that were processed in an hour and efficient use of resources with respect to the number of CPUs and memory available.
- AWS took a training time of six hours while Azure took nine and a half hours which indicates that it is more appropriate for large computations. In addition, AWS had a more stable utilization rate of the CPU, while Azure had small variations which suggested possible weakness in resource management.
- On the other hand, Windows Azure was cheaper with cost per task being much lower than that of AWS and this was preferable for smaller or budget constrained projects.
- However, this and other positives of Azure all had a drawback in that it was designated cheaper and slower by the fact that when speed was a factor, AWS's superior capabilities shone through.
- The accuracy of the model for both platforms was similar at 78% with variable levels of precision and recall for positive class on Azure and AWS respectively. In this case however, the slight advantage in accuracy for AWS didn't matter as training was faster and more consistent in terms of performance.
- To sum up, for massive deep learning tasks that require more training time and effective scaling, the cloud computing platform of choice is AWS, whereas for smaller projects with comparable accuracy, Azure is still more cost-effective.

9 References

[1] H. Johnson and S. Lee, "Cost Efficiency and Scalability of Cloud Platforms for Machine Learning Workloads," *Cloud Computing and Big Data,* vol. 15, no. 1, pp. 45-59, 2021.

[2] Ramirez, "Scalability of TensorFlow on AWS and Azure: A Comparative Study.," *Journal of Machine Learning Infrastructure,* vol. 21, no. 2, pp. 67-81, 2023.

[3] L. e. a. Foster, "Comparative Analysis of Disk I/O Performance in Cloud Platforms for Machine Learning," *Cloud Performance Studies,* vol. 14, no. 3, pp. 235-249, 2020.

[4] S. e. a. Khan, "The Usability of Cloud Platforms for Machine Learning Development: A Comparison of AWS and Azure.," *Journal of Cloud Technology,* vol. 10, no. 2, pp. 85-94., 2020.

[5] E. e. a. Roberts, "Benchmarking TensorFlow Performance on AWS and Azure," *Cloud Performance Studies,* vol. 14, no. 3, pp. 237-246, 2021.

[6] P. e. a. Zhang, "TensorFlow on Cloud Platforms: AWS vs Azure.," *International Journal of Machine Learning,* vol. 8, no. 2, pp. 85-98, 2019.

[7] Q. e. a. Wang, "Automating TensorFlow Model Deployment on AWS.," *AI Deployment Quarterly,* vol. 7, no. 2, pp. 92-102, 2021.

[8] A. Miller and R. Singh, "AutoML and Model Deployment on Azure: A Comparative Review," *International Journal of Cloud Applications,* vol. 14, no. 1, pp. 18-33, 2023.

[9] J. e. a. Smith, "Cloud Computing and Machine Learning: A Comparative Review," *Journal of Cloud Engineering,* vol. 12, no. 4, pp. 112-130, 2020.

[10] E. e. a. Roberts, "Benchmarking TensorFlow Performance on AWS and Azure," *Cloud Performance Studies,* vol. 14, no. 3, pp. 237-24, 2021.

[11] J. Lee and S. Gupta, "Cost Analysis of Machine Learning on AWS and Azure," *Computing Resources Quarterly,* vol. 6, no. 2, pp. 203-215, 2020.

[12] G. e. a. Hernandez, "Predictable Pricing Models for Machine Learning in the Cloud," *Journal of Cloud Economics,* vol. 17, no. 1, pp. 43-52, 2023.

[13] D. e. a. Kim, "Advanced Customization in Cloud-Based Machine Learning," *Cloud Engineering Journal,* vol. 8, no. 3, pp. 165-178, 2020.

[14] P. Williams and J. Taylor, "Ease of Use and Usability of Cloud Platforms for Machine Learning," *Cloud Platforms Review,* vol. 8, no. 4, pp. 77-85, 2019.

[15] L. Garcia and H. Lopez, "Optimizing TensorFlow for Cloud Training," *Journal of Cloud Development,* vol. 11, no. 4, pp. 334-349, 2022.

[16] R. e. a. Jones, "Cloud-Based TensorFlow Optimization," *Journal of Applied AI,* vol. 15, no. 1, pp. 104-121, 2023.

[17] M. Patel and V. Gupta, "Cost Comparison of Cloud Platforms: AWS vs. Azure for Machine Learning," *Journal of Cloud Economics,* vol. 17, no. 3, pp. 45-56, 2019.

[18] K. Brown and S. Taylor, "The User Experience of Machine Learning Platforms: A Comparison of AWS and Azure," *UX Research Journal,* vol. 19, no. 2, pp. 55-68, 2021.

[19] J. Ravanello, L. E. B. Desharnais, A. A. Villalpando and A. Gherbi, "Performance Measurement for Cloud Computing Applications Using ISO 25010 Standard Characteristics," [Online].

[20] L. Davis and M. Liu, "Scaling Deep Learning Models with AWS SageMaker," *Proceedings of the Cloud Computing Symposium,* vol. 45, no. 1, pp. 23-29, 2021.