Journal of Internet Services and Applications, 2025, 16:1, doi: 10.5753/jisa.2025.4891

Training Neural Networks in Cloud Environments: A Methodology and a Comparative Analysis

Cláudio Márcio de Araújo Moura Filho [**Universidade Federal Rural de Pernambuco** | ***claudiomarciofilho@hotmail.com***]
Erica Teixeira Gomes de Sousa [**Universidade Federal Rural de Pernambuco** | ***erica.sousa@ufrpe.br***]

Universidade Federal Rural de Pernambuco, Rua Dom Manuel de Medeiros, Recife, PE, 52171-900, Brazil.

Received: 03 September 2024 • **Accepted:** 19 April 2025 • **Published:** 30 May 2025

Abstract

Deep neural networks are solutions to problems that involve pattern recognition, and various works seek to optimize the performance of these networks. This optimization requires suitable hardware, which can be expensive for small and medium organizations. This work proposes a methodology to evaluate the performance and cost of training deep neural networks by assessing how much impact factors such as environment setup, frameworks, and datasets can have on training time and, along with this task, evaluating the total financial cost of the environment for the training process. Experiments were performed to measure and compare the performance and cost of training deep neural networks on cloud platforms such as Azure, AWS, and Google Cloud. In this sense, factors such as the size of the input image and the network architecture significantly impact the training time metric and the total cost.

Keywords: Cloud Computing, Neural Networks, Performance Evaluation, Cost Evaluation

1 Introduction

Deep neural networks are potential solutions for many problems that involve pattern recognition, classification, and prediction. To become a reliable solution, these DNNs undergo a training process to understand data patterns and classify information correctly, which will be provided later LeCun *et al.* [2015].

This training process can be very costly because, to obtain satisfactory results, it is necessary to process a large volume of data. One of the most significant challenges in creating DNN models is the time required to train a model to solve a problem LeCun *et al.* [2015].

One of the alternatives to accelerate the training process of a DNN is to use specialized hardware for data processing, such as Graphic Processing Units (GPUs). However, these processing units have considerably higher acquisition costs than conventional processors, such as central processing units (CPUs) LeCun *et al.* [2015].

Cloud computing emerges as an alternative to reduce the high cost of a GPU. With cloud computing, building a computational environment for all tasks is possible without having the physical machine and paying only for what is used Kansal *et al.* [2014].

These custom built environments can be very promising for building solutions with neural networks, as we can have all the necessary resources for efficient training execution with a cost based on resource usage Juve *et al.* [2009] Jackson *et al.* [2010].

Due to the need to develop deep neural network solutions quickly and the need for a suitable environment to develop them, this work is dedicated to proposing a methodology to evaluate the training of neural networks in cloud environments, assess the performance and cost of training the networks in this environment.

The following topics will be covered in the following sections: The related work will be shown in Section 2, and the methodology will be presented in Section 3. Finally, Sections 4 and 5 discuss the results of the experiments and the conclusion of this work.

2 Basic Concepts

For a better understanding of this work, some basic concepts about cloud computing and neural networks are presented in this section. .

2.1 Cloud Computing

According to the National Institute of Standards and Technologies (NIST), cloud computing is a model that provides access to a shared set of resources (processors, memory, disk, etc.) that can be provisioned with minimal management effort or interaction with the service provider Correia [2011].

With cloud computing, building a computing environment for various services is possible by paying only for the computing resource used Kansal *et al.* [2014]. Furthermore, this provisioned environment can be adjusted according to the task's demand for computing resources.

2.2 Cloud Pricing

In general, the pricing models for cloud services are static or fixed payment models and dynamic payment models.

The static payment model is a model where the cloud provider predefines the price for the service and remains the same, regardless of whether the customer uses the full capacity of the contracted service. This model is often very beneficial for providers and somewhat unfair to customers, as each customer has specific needs and customers can usually pay for a service that may be poorly dimensioned Wu *et al.* [2019].

The charges of the dynamic payment model are based on the resources that are being used. In general, the dynamic model is usually more fair for both the provider and the customer, but it faces the dilemma of how to correctly price each service/resource Wu *et al.* [2019].

Preemptive machines are computing resources that adopt a pricing scheme in which cloud providers offer idle resources from the cloud cluster at a reduced price to stimulate their use. However, resources provisioned through this scheme can be deallocated abruptly and sometimes without warning. This modality is not recommended for running high-priority services where maintaining execution is essential. Still, this type of provisioning becomes very interesting for services where interrupting execution and resuming it after some time is possible.

2.3 Deep Neural Networks

Deep Neural Networks (DNNs) are a class of artificial intelligence algorithms capable of solving complex problems involving pattern recognition, classification, and prediction LeCun *et al.* [2015].

DNNs adopt the backpropagation algorithm to adapt to the data that pass through their multiple processing layers and, in this way, they can learn to interpret the data and consequently become capable of finding solutions to problems such as speech recognition and object recognition in images, among other diverse possibilities LeCun *et al.* [2015].

Convolutional neural networks (CNNs) are a special type of deep neural network that is capable of dealing with images that are used mainly to recognize objects within images and classify the image or highlight the object found in it LeCun *et al.* [2015].

An essential application of CNNs is in the medical field, where neural networks can be used to identify cancer cells in images. Neural networks can correctly predict more than 90% of the time when an image has or does not have a cancer cell pattern Jain *et al.* [2022].

In addition to image recognition, neural networks can also be used for speech recognition, as is the case with neural networks used for natural language processing. The most recent case is ChatGPT, a tool capable of recognizing user input and providing contextualized answers according to the questions asked Alahmed *et al.* [2023].

2.4 Graphic Processor Unit (GPU)

One point in common between deep neural networks is that they rely on a lot of training data to learn how to solve problems. This massive amount of data needs to be processed by the various layers of the network to recognize the pattern in the data and, consequently, learn to solve the problem Dally *et al.* [2021].

This processing can be highly costly in computational and time terms, as training takes place by passing data through the processing layers multiple times, each pass taking hours depending on the network's size and the data's size Dally *et al.* [2021].

One way to speed up this processing is to use hardware specializing in complex calculations, such as Graphic Processor Units (GPUs) Dally *et al.* [2021].

GPUs can process data several times faster than Central Processing Units (CPUs), which are the processors in conventional architectures Dally *et al.* [2021].

However, despite the excellent processing capacity of this type of hardware, the acquisition of this equipment is accompanied by high prices, which can make it unaffordable to many organizations Dally *et al.* [2021].

3 Related Work

The topic of deep neural networks is of great interest for research, mainly in the performance area; however, most research is related to the performance of DNNs concerning their precision in classifying and/or recognizing objects Zhu *et al.* [2018], that is, how well neural networks can solve a problem.

Cloud environments, in turn, have already been addressed in high-performance computing research, as seen in Juve *et al.* [2009] and Elshawi *et al.* [2021], which use AWS virtual machines to evaluate their performance in executing high-performance tasks, and Carneiro *et al.* [2018], which uses the Google Colaboratory environment to accelerate neural network training. In these works, the authors focus on understanding how to use cloud computing to perform tasks that require high computational power, such as training neural networks.

The authors Zhu *et al.* [2018] and Elshawi *et al.* [2021] focus their efforts on understanding the mechanisms to evaluate the performance of neural networks considering various factors such as the dataset used, the task the neural network will perform, the training time, the framework in which the network is built, and finally, the resources of the environment. These two works propose benchmark models to evaluate the performance of neural networks, defining various parameters such as resource usage and analyzing this performance, considering the frameworks and datasets used.

The works Shi *et al.* [2016] Wu *et al.* [2018] Liu *et al.* [2018] Shams *et al.* [2017] and Xie *et al.* [2023] seek to perform experimentation by varying the possible factors that most impact the performance of a neural network to understand the behavior of the networks during training to improve the effectiveness of the neural network solution. These works observe the recurring use of frameworks such as TensorFlow and MXNet and datasets such as MNIST and CIFAR-10, in addition to performing various tests in different environments with and without GPUs.

In the works Adebayo *et al.* [2020] Edinat [2018] Kandpal *et al.* [2017] and Wu *et al.* [2019], the authors seek to synthesize information on how cloud services are priced and aim to answer how to price cloud services appropriately while maintaining fair prices for both users and providers. There is a tendency to consider the dynamic model as the fairest pricing model because it allows users to pay only for the amount of resources they allocate and use and enables the provider to offer resource sets that adequately cover their costs. Table 1 presents the related work to performance and cost evaluation of neural network training in cloud environments.

Given the above, the proposal of this work, and what differentiates it from other existing works, is the proposition of a methodology to evaluate the performance of neural network training in public cloud environments, considering the cost of training the neural network in this type of environment as a performance metric.

4 Methodology for Evaluating Neural Network Performance in Cloud Environments

The proposed methodology aims to present activities that allow the evaluation of neural network performance in cloud environments, as shown in Figure 1. This methodology consists of five activities: Understanding the Environment, Planning Experiments, Environment Configuration, Measurement, and Analyzing Metrics. In this section, the objective of each of these activities will be better detailed.

To understand the environment, it is first necessary to remember the objective we want to achieve. The aim in question is to analyze the training of neural networks in a cloud environment from a performance and financial cost perspective.

With the objectives defined, performance and cost metrics can be selected, these metrics will be very important to choose an adequate measurement tool in the Measurement activity. Many metrics can be selected, such as performance metrics such as resource utilization, execution time, and throughput. In contrast, the cost metric can be the total resource cost. The metrics selected must be in order with the objective we set; this way, we can evaluate the training process correctly.

The cloud service provider can be chosen to provide the environments where the experiments for data collection will be conducted. Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform are examples of providers that can be adopted.

The activity of Planning Experiments aims to define the factors and their levels for determining performance and cost experiments.

To conduct the experiments, it is important to define the types of virtual machines used. Virtual machines can be instantiated with only CPUs or both CPUs and GPUs.

Frameworks are code libraries that help speed up application development. In the case of neural networks, they provide the necessary structure to model networks and apply training and optimization functions. Many frameworks, such as Caffe, Chainer, CNTK, TensorFlow, and PyTorch, are available; therefore, selecting the most interesting ones for detailed analysis is necessary.

Another important factor for neural network training is the dataset used, as it defines the problem the neural network will

Table 1. Related Work

Related Work	Dataset	Framework	Metric	Pricing Model
Juve *et al.* [2009]	-	-	• EXECUTION TIME	-
Jackson *et al.* [2010]	-	-	• EXECUTION TIME • SUSTAINED SYSTEM PERFORMANCE	-
Shi *et al.* [2016]	• MNIST • CIFAR-10	• Caffe • Torch • TensorFlow • MXNet • CNTK	• TIME/BATCH	-
Shams *et al.* [2017]	• ILSVRC 2012 • CIFAR-10 • MNIST	• Caffe • Apache SINGA • TensorFlow	• EXECUTION TIME • IMAGES/MILLISECOND	-
Wu *et al.* [2018]	• MNIST • CIFAR-10 • ILSVRC 2012	• Caffe • Torch • TensorFlow • Theano	• TRAINING TIME • TEST TIME • ACCURACY • CPU UTILIZATION • MEMORY UTILIZATION	-
Liu *et al.* [2018]	• CIFAR-10 • ImageNet	• Caffe2 • Chainer • TensorFlow • MXNet • CNTK	• ACCURACY • TRAINING TIME • IMAGES/SECOND	-
Carneiro *et al.* [2018]	• MS-COCO	• TensorFlow	• EXECUTION TIME • EVALUATIONS/SECOND	-
Zhu *et al.* [2018]	• ImageNet1K • IWSLT15 • VOC 2007 • LibriSpeech • Downsampled • ImageNet • Atari 2006	• TensorFlow • MXNet • CNTK	• THROUGHPUT • FLOATING POINT OPERATIONS UTILIZATIONS • CPU UTILIZATIONS • MEMORY USAGE • GPU UTILIZATIONS	-
Elshawi *et al.* [2021]	• MNIST • CIFAR-10 • CIFAR-100 • SVHN • IMDB Reviews • Penn Treebank • Many things: English to Spanish • VOC 2012	• TensorFlow • Keras • PyTorch • MXNet • Theano • Chainer	• TRAINING TIME • ACCURACY • CPU UTILIZATION • MEMORY UTILIZATION • GPU UTILIZATION	-
Xie *et al.* [2023]	• MNIST • CIFAR-10 • Flower Recognition	• TensorFlow • Pytorch • PaddlePaddle	• TRAINING TIME • ACCURACY • CPU UTILIZATION • GPU UTILIZATION • MEMORY UTILIZATION	-
Kandpal *et al.* [2017]	-	-	-	• Static • Dynamic
Edinat [2018]	-	-	-	• Static • Dynamic
Adebayo *et al.* [2020]	-	-	-	• Static • Dynamic • Hybrid
Wu *et al.* [2019]	-	-	-	• Dynamic

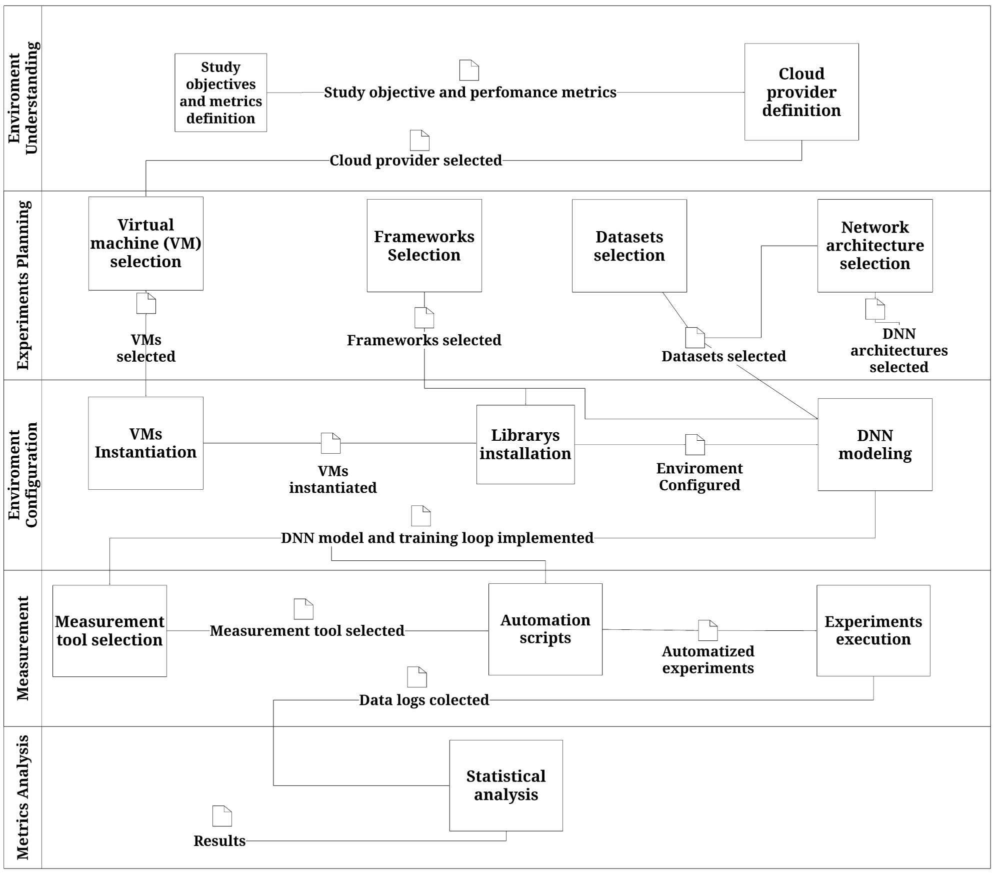

Figure 1. Methodology for Evaluating Neural Network Performance in Cloud Environments.

solve. High-quality datasets ideally have many images well-distributed among classes, but this data balance may not be possible depending on the situation.

With the datasets and frameworks defined, it is now necessary to define a neural network model to train. The model depends on the problem being solved; it can be a completely novel model or an existing architecture that can be used for other purposes.

Selecting a variety of frameworks, datasets, and network architectures will lead us to a fascinating analysis since we can mix those different levels of factors and create a good number of scenarios to experiment.

Other factors like image sizes can also be used to create even more scenarios to experimentation.

To execute the neural network training, it is important to configure the environment correctly with the necessary libraries. The Environment Configuration activity involves creating virtual machines, installing the required libraries to execute the neural network training, and creating the neural network models. A misconfigured environment can lead to problems in the training process, such as GPUs not being enabled to be used.

The Measurement activity focuses on selecting an appropriate tool to monitor and execute the experiments. To select the tool, it is necessary to understand which metrics will be used. Many measurement tools can be selected. The ideal one is the one that can collect our defined metrics. Furthermore, more than one tool can be used if necessary.

Last, the Analysis of Metrics activity aims to analyze the data collected in the experiments statistically. This analysis is done through the statistical summary of the data collected and the presentation of the performance and cost metrics analysis.

5 Results and Discussion

This section aims to evaluate whether the proposed methodology can be applied to assess the performance and cost of neural networks in cloud environments. Thus, a series of case studies are presented.

The first study aims to validate the proposed methodology. The second seeks to enable a comparison between the Amazon AWS and Microsoft Azure cloud providers and will follow the same methodology applied in Case Study 1. The last case study will add Google Cloud Platform to the comparison, while the other network architectures will also be evaluated.

5.1 Case Study 1 - AZURE

This case study aims to validate the methodology for which each methodology activity must be carried out.

5.1.1 Understanding the Environment

Following the activities proposed in the methodology, the metrics for the study were first selected. The metrics to be collected were: training time, CPU utilization, memory utilization, GPU utilization, and GPU memory utilization, with the first metric measured in seconds (s) and the subsequent four all measured in percentage (%). Those metrics are intended to make possible to understand with resources commonly used by the training process and the total time of training will be needed to calculate the training cost.

After selecting the metrics, a cloud provider needs to be chosen. Microsoft Azure, one of the most used cloud platforms, was chosen.

5.1.2 Planning of Experiments

The factors in the experiments are the framework, the dataset, the image dimension, the network architecture, and the environment. Each of these factors will have different variations, referred to as levels. These factors and their levels can be seen in Table 2 and will be discussed further below.

Table 2. Factors and Levels of the Experiments.

Factor	Level 1	Level 2	Level 3
Framework	MXNet	PyTorch	Tensor Flow
Dataset	MNIST	CIFAR-10	-
Image Size	32x32	64x64	-
Network Architecture	Network 1	Network 2	-
Environment	CPU	GPU	-

The selected virtual machines are used as the environment for training neural networks. Table 3 shows the configurations of the adopted virtual machines.

Table 3. Virtual Machine Configuration.

VM Name	vCPUs	RAM	GPU	SO	Cost/hour
F4s_v2	4	8GB	-	Ubuntu 22.04	U$0,169
NC4as_T4_v3	4	28GB	NVIDIA T4 16GB	Ubuntu 22.04	U$0,526

Due to their usability, PyTorch, MXNet, and TensorFlow were chosen. These frameworks are widely used in academic projects and the industry, allowing for quick and simplified model construction. In addition to network modeling, they facilitate the creation of training loops for the networks in an equally simple manner.

The selected datasets were MNIST LeCun and Cortes [2005], a dataset of black-and-white handwritten digit images, and CIFAR-10 Krizhevsky [2009], a dataset of colored images with 10 different classes, mixing animals and vehicle types. These datasets are widely used in other works in the field, as shown in the related works section, and were therefore chosen.

The MNIST dataset consists of 70.000 images divided into 60000 images for training and 10.000 for testing. These images are 28x28, black-and-white divided in 10 classes that are the handwritten digits 0 to 9 LeCun and Cortes [2005].

The CIFAR-10 dataset is a collection of 60000 images split into 50.000 as the training set and 10.000 as the testing set. The images are equally distributed between 10 classes, they are RGB images with a 32x32 image size Krizhevsky [2009].

Although the datasets selected are quite simple, the focus of this work is not on solving a specific problem, so there is no need to use large datasets.

With these factors in mind, the chosen network architecture was similar to what is done in VGG networks Simonyan and Zisserman [2014], which combines convolutional layers, activation functions, pooling layers, and fully connected layers. The architecture of the networks can be seen in Figures 2 and 3.

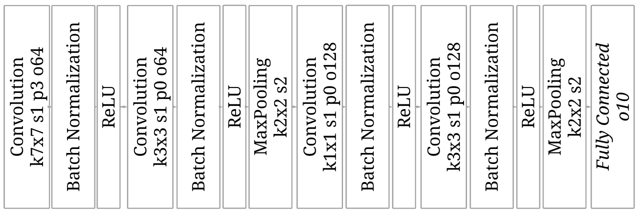

Figure 2. Network Architecture 1.

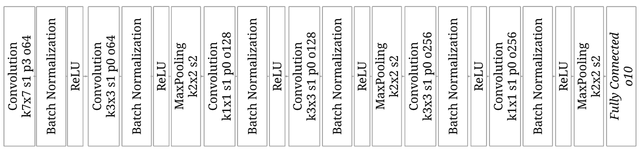

Figure 3. Network Architecture 2.

In the figures, we can see some letters in the layers:

- k represents kernel and the window size that the layer will observe on the image at each step.
- s represents the stride, or how many pixels the kernel window will move across the image with each interaction.
- p refers to padding, or how many pixels will be added to the edges of the image.
- Finally, o denotes the number of output channels of the layer or the number of outputs of the layer.

These two architectures will generate workloads on our environment to be evaluated. The combination of these different factors will lead us to a variety of scenarios, the main focus of which is to determine how well the resources are being used and the impact of the variation of those factors on the training time and, therefore, the final cost.

Table 4 shows the combination of factor levels forming numbered scenarios to improve their identification.

5.1.3 Environment Setup

For the environment setup, libraries must be installed to run the neural network training code. Below are the versions used for the main libraries. TensorFlow Version 2.14.0, PyTorch Version 2.2.0, and MXNet Version 1.9.1 was used. These versions were the most stable version of the frameworks when the experiments were conducted.

In addition to the framework versions, another required installation is for GPU usage libraries. Since the graphics card

Table 4. Scenarios

S	Env.	Framework	Dataset	Img Size	Net. Architect.
C1	CPU	MXNet	MNIST	32x32	Net. 1
C2	CPU	MXNet	MNIST	32x32	Net. 2
C3	CPU	MXNet	MNIST	64x64	Net. 1
C4	CPU	MXNet	MNIST	64x64	Net. 2
C5	CPU	MXNet	CIFAR-10	32x32	Net. 1
C6	CPU	MXNet	CIFAR-10	32x32	Net. 2
C7	CPU	MXNet	CIFAR-10	64x64	Net. 1
C8	CPU	MXNet	CIFAR-10	64x64	Net. 2
C9	CPU	PyTorch	MNIST	32x32	Net. 1
C10	CPU	PyTorch	MNIST	32x32	Net. 2
C11	CPU	PyTorch	MNIST	64x64	Net. 1
C12	CPU	PyTorch	MNIST	64x64	Net. 2
C13	CPU	PyTorch	CIFAR-10	32x32	Net. 1
C14	CPU	PyTorch	CIFAR-10	32x32	Net. 2
C15	CPU	PyTorch	CIFAR-10	64x64	Net. 1
C16	CPU	PyTorch	CIFAR-10	64x64	Net. 2
C17	CPU	TF	MNIST	32x32	Net. 1
C18	CPU	TF	MNIST	32x32	Net. 2
C19	CPU	TF	MNIST	64x64	Net. 1
C20	CPU	TF	MNIST	64x64	Net. 2
C21	CPU	TF	CIFAR-10	32x32	Net. 1
C22	CPU	TF	CIFAR-10	32x32	Net. 2
C23	CPU	TF	CIFAR-10	64x64	Net. 1
C24	CPU	TF	CIFAR-10	64x64	Net. 2
C25	GPU	MXNet	MNIST	32x32	Net. 1
C26	GPU	MXNet	MNIST	32x32	Net. 2
C27	GPU	MXNet	MNIST	64x64	Net. 1
C28	GPU	MXNet	MNIST	64x64	Net. 2
C29	GPU	MXNet	CIFAR-10	32x32	Net. 1
C30	GPU	MXNet	CIFAR-10	32x32	Net. 2
C31	GPU	MXNet	CIFAR-10	64x64	Net. 1
C32	GPU	MXNet	CIFAR-10	64x64	Net. 2
C33	GPU	PyTorch	MNIST	32x32	Net. 1
C34	GPU	PyTorch	MNIST	32x32	Net. 2
C35	GPU	PyTorch	MNIST	64x64	Net. 1
C36	GPU	PyTorch	MNIST	64x64	Net. 2
C37	GPU	PyTorch	CIFAR-10	32x32	Net. 1
C38	GPU	PyTorch	CIFAR-10	32x32	Net. 2
C39	GPU	PyTorch	CIFAR-10	64x64	Net. 1
C40	GPU	PyTorch	CIFAR-10	64x64	Net. 2
C41	GPU	TF	MNIST	32x32	Net. 1
C42	GPU	TF	MNIST	32x32	Net. 2
C43	GPU	TF	MNIST	64x64	Net. 1
C44	GPU	TF	MNIST	64x64	Net. 2
C45	GPU	TF	CIFAR-10	32x32	Net. 1
C46	GPU	TF	CIFAR-10	32x32	Net. 2
C47	GPU	TF	CIFAR-10	64x64	Net. 1
C48	GPU	TF	CIFAR-10	64x64	Net. 2

is an NVIDIA card, the necessary libraries are the CUDA Toolkit and cuDNN. The CUDA Version used is v11.7, and cuDNN is Version 8.6.0. These versions were selected for compatibility reasons. The MXNet with CUDA is only available until 11.7, so this version should be used.

5.1.4 Measurement and Analysis

To collect metric data, a script runs in parallel with the neural network training and captures and logs the metrics every 10 seconds in a comma-separated values file. In addition to this script, some commands in the training loop track the training time for each epoch.

This script was selected because it is easy to configure and very light, not having a chance to compromise the training process with a more complex and maybe resource-consuming tool.

To ensure the quality of the experimental data, each training was executed 5 times, so the initial 48 scenarios became 240 runs, varying the different levels of the defined factors.

Each experiment involves training a neural network for 10 epochs, meaning all images from the datasets were passed through the networks 10 times. Since we are not adding more data trough the epochs the time needed to execute one epoch will not have a high variation between epochs, moreover this training is not intended to reach the end of convergence so there is no need to a high number of epochs but the small number of epochs used can be very helpful to stablish expectations towards the training cost for a higher number of epochs.

Besides the number of epochs, the batch size was fixed at 128 examples per iteration, meaning the network is fed 128 images at a time until the network has processed all images.

The following are figures (Figures 4 to 7) with graphs of the averages of the collected metrics, obtained through the execution of neural network training across the 48 scenarios previously mentioned.

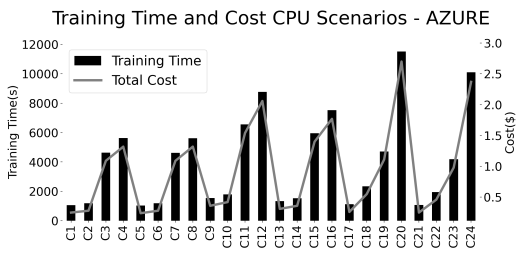

Figure 4. Average Training Time and Cost Graph in Azure CPU Environment.

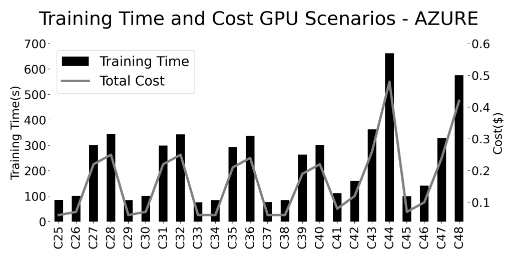

Figure 5. Average Training Time and Cost Graph in Azure GPU Environment.

The results presented in Figures 4 and 5 show that as the image size increases, the training time of the neural network also increases. A similarity can be observed in the graphs when comparing the CPU against GPU environments; scenarios with 64x64 images increase the training time by about

200% compared to 32x32 images.

The complexity of the chosen network also impacts the training time. This can be seen in the graphs where scenarios use Network 2; these scenarios require more time to train the network.

The GPU environment is at least 10 times faster than the CPU environment, even though the cost per hour for the GPU environment is three times that of the CPU scenario. The total cost of the CPU environment was approximately 4 to 6 times lower than that of the GPU environment.

Therefore, careful consideration must be given to the dimensions of the input image and the network architecture used, as they significantly impact the total training time of the neural network and consequently on the total cost.

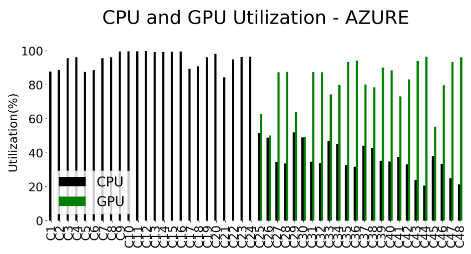

Figure 6. CPU and GPU Utilization graph in Azure Environments.

Considering Figure 6, it can be observed that all frameworks make good use of processing resources, with PyTorch being the framework that utilizes CPU resources the most, with values consistently close to 100%. As for the GPU scenarios, it is noted that scenarios with 64x64 images tend to consume twice as much GPU resources compared to CPU.

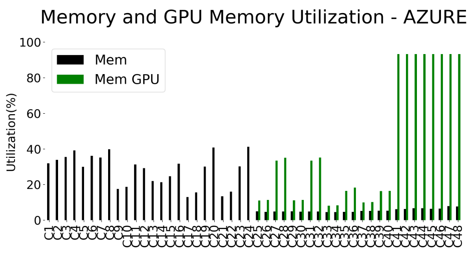

Figure 7. Memory and GPU memory utilization graph in Azure Environments.

Figure 7 shows that TensorFlow is the framework that uses GPU memory the most across all scenarios, tending to utilize 100% of GPU memory. However, this framework also exhibits training times that are 2 to 3 times longer compared to the others it is compared with. For PyTorch, it is noted that its performance places it between MXNet and TensorFlow in terms of training time. Additionally, PyTorch fully utilizes the CPU, using about 40% of its memory, and uses minimal GPU memory, consuming about 20% despite utilizing more than 80% of the GPU.

The dataset does not generally affect the metrics, but it is observed that scenarios with CIFAR-10 had shorter training times compared to those using MNIST.

Finally, the total amount spent as indicated by the Microsoft Azure cost management panel was approximately \$31, summing up the costs. This cost is divided as follows, \$27 to CPU scenarios and \$4 to GPU scenarios.

5.2 Case Study 2 - AWS

With the first case study, it was possible to observe the capability of GPUs to accelerate neural network training and a potential reduction in costs. However, as mentioned in Section 4, there are various cloud providers, and to make a comparison between different providers, this case study was conducted, using most of the definitions of the first case study with slight changes that will be discussed in this section.

5.2.1 Understanding the environment

For this second case study, the main objective is to compare the performance and cost of training between the scenarios executed on a Microsoft Azure VM and a VM of another cloud platform. The Amazon AWS cloud provider was selected, primarily due to its popularity.

5.2.2 Planning of Experiment

A similar virtual machine was chosen to run the experiments to keep the comparisons fair. The VM in question is named g4dn.xlarge, and like the Microsoft Azure F4s_v2 machine, it has four vCPUs, 1 NVIDIA T4 16GB, and the installed operating system was Ubuntu 22.04. The only difference, therefore, was the amount of available RAM, with the Amazon AWS instance having 16GB of RAM compared to 28GB on the Microsoft Azure virtual machine, as shown in Table 5.

Table 5. Virtual Machine Configuration.

VM Name	vCPUs	RAM	GPU	SO	Cost/hour
g4dn.xlarge	4	16GB	NVIDIA T4 16GB	Ubuntu 22.04	U$0,526

Only the GPU scenarios were considered in this second study, but all other selections and configurations were maintained. To facilitate understanding, the terminology defined in Table 4 will also be preserved, so the scenarios analyzed on the Amazon AWS VM will range from C25 to C48.

5.2.3 Measurement and Analysis

Figures 8, 9, and 12 show the metrics obtained in the experiments with the Amazon AWS instance, and Figures 10, 11,13, 14 shows the percentage difference between the scenarios in Microsoft Azure and Amazon AWS using the Microsoft Azure values as a base.

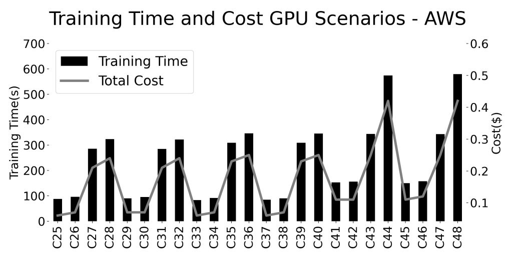

Figure 8. Average Training Time and Cost Graph in AWS GPU Environment

As shown in Figure 8, the training time bars and cost line are similar to the scenarios depicted in Figure 5, indicating comparable performance between the instances from the two providers.

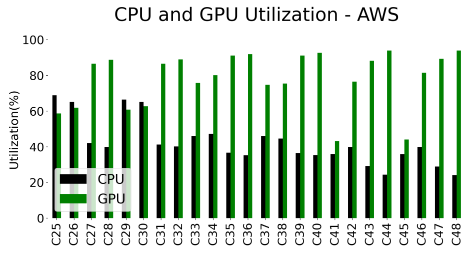

Figure 9. CPU and GPU utilization graph in AWS Environments.

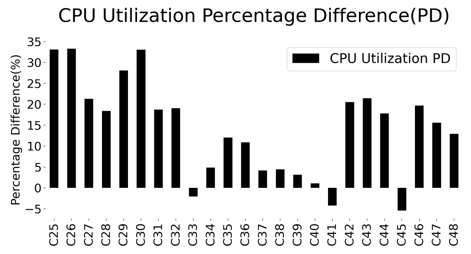

Figure 10. CPU Utilization Percentage Difference.

Comparing Figure 9 and Figure 6, a strong similarity in resource utilization is observed again. However, there is a noticeably higher CPU utilization on the Amazon AWS VM compared to the Microsoft Azure VM, as shown in Figure 10, especially in scenarios C25 to C32, these scenarios are using MXNet as a framework, we can see an increase of almost 20% at least in these scenarios and more than 30% at most.

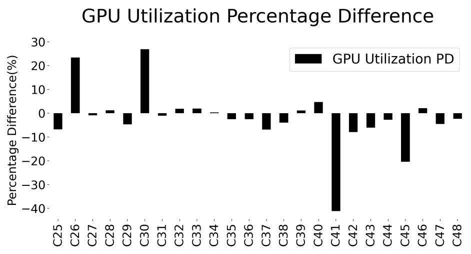

Figure 11. GPU Utilization Percentage Difference.

Regarding GPU utilization, a slight overall decrease is seen in Figure 11, but this decrease is more pronounced in Scenarios C41 and C45, where both CPU and GPU utilization fall below 50%. These graphs indicate that the AWS VM managed resources differently but achieved similar training time and cost results.

Finally, Figure 12 shows memory and GPU memory utilization. It is important to remember that the Azure virtual machine has 75% more memory RAM than the Amazon AWS VM. Considering this difference for a more fair analysis, the percentage difference was calculated using the amount of memory utilized in GB. It is noticeable in Figure 13 that scenarios using the MXNet framework (C25-32) utilize about 100% more main memory and the Tensorflow scenarios (C41-48) utilize approximately 40% more main memory in the Amazon AWS instance, the PyTorch scenarios (C33-40), on the other hand, can utilize 20% less main memory.

Regarding GPU memory, as shown in Figure 14, scenarios using images 32x32 evidence a 10% increase in utilization. Moreover, PyTorch scenarios(C33-40) had a volatile utilization, increasing 40% at most and reaching a 10% decrease. TensorFlow scenarios(C41-48) showed a 1% increase in general demonstrating a very stable utilization of this resource.

The total cost of this second case study is around $4 as much as the first study with the Microsoft Azure Cloud, and this shows that the performance does not depend on the cloud provider.

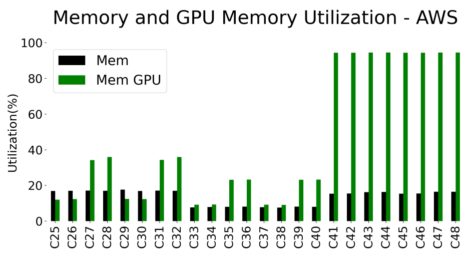

Figure 12. Memory and GPU memory utilization graph in AWS GPU Environments.

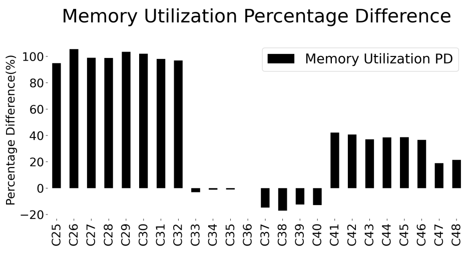

Figure 13. Memory Utilization Percentage Difference.

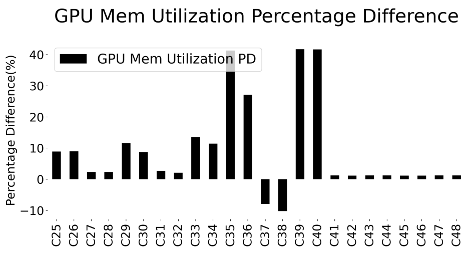

Figure 14. GPU Memory Utilization Percentage Difference.

5.3 Case Study 3 - Google Cloud Platform

This third case study aims to add a third cloud platform to the comparison. This time, the Google Cloud Platform(GCP) was also selected because of great popularity.

5.3.1 Replicating Previous Experiments

This part of the case study will follow the same pattern as Study 1 with Microsoft Azure and Study 2 with AWS, with a slight difference in the available RAM in the utilized VM, as seen in Table 6. This difference was already shown to have little impact on training time.

The Figures 15, 16 and 17 show the results of the metrics

Table 6. Virtual Machine Configuration.

VM Name	vCPUs	RAM	GPU	SO	Cost/hour
n1-standard-4	4	15GB	NVIDIA T4 16GB	Ubuntu 22.04	U$0,54

collected in the experiments on the GCP VM environment that was set.

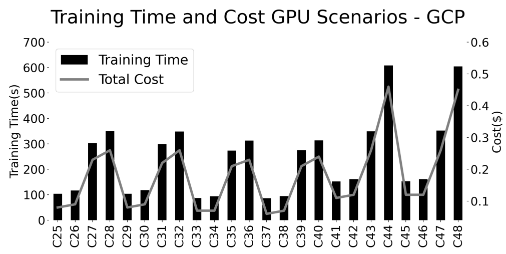

Figure 15. Average Training Time and Cost Graph in GCP Environment

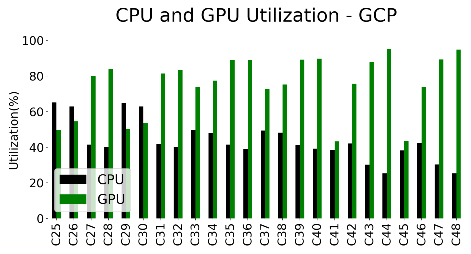

Figure 16. CPU and GPU utilization graph in GCP Environments.

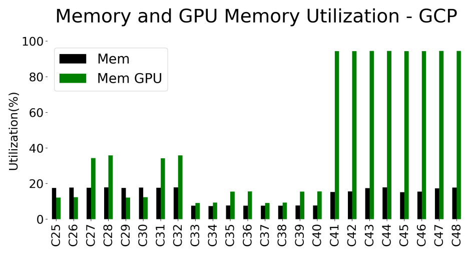

Figure 17. Memory and GPU memory utilization graph in GPC Environments.

As could be seen in the graphics, we reach similar results as the previous case studies showing some consistency about what was already analyzed and pointing to the fact that the cloud provider does not have much impact on training performance, becoming a matter of preference or even availability of a specific configuration of a VM, over a meaningful impact choice in the training process.

5.3.2 Others DNN Models

The previous experiments show us that the cloud provider chosen is more a matter of preference than a impactful factor in reducing costs. In addition to these experiments already performed on other cloud platforms, we conducted some new tests with different types of DNN, one LSTM model, and a very relevant DNN nowadays, the YoLov7 Wang *et al.* [2023]

These experiments will use the same metrics as the first ones, so we can establish some connection between the training processes of the different types of models and the models we used to conduct the first experiments.

LSTM

For or LSTM model we used a model presented on Jackson *et al.* [2010] in this work the authors use a specific model for each of the datasets they are trying to experiment, in our case we will use the model utilized on the Penn Treebank(PTB) dataset with 2 LSTM layers, since its easy to reproduce and simple to understand. This model was be configured in the three frameworks used along the case studies until here, MXNet, PyTorch, and Tensorflow.

For the frameworks that we be utilized as well as them versions can be seen on the Table 7 below. The PyTorch suffered a downgrade in its version because of compatibilities with another library that was need in the experiments with LSTMs, the torchtext v0.10.0.

The CUDA and CuDNN versions are the same as the previous experiments.

Table 7. Frameworks versions for LSTMs and YoLo experiments

Framework	Version
MXNet	1.9.1
PyTorch	1.9.0
Tensorflow	2.14.0

The Penn Treebank is a collection of text data from the papers of the Wall Street Journal that aggregates more than 1 million examples for training.

In the figures below, we can see the results of the training experiments for the LSTM model.

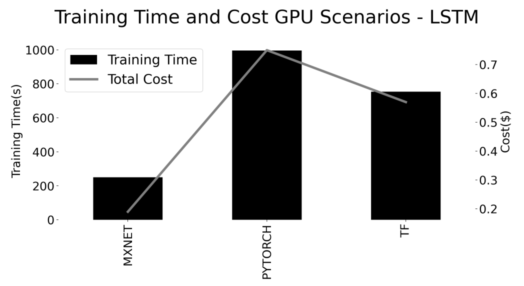

Figure 18. Average Training Time and Cost Graph for LSTM model.

Figure 18 shows the average training time for the LSTM model. Here, we can see that the MXNet performs much better than the other two frameworks in terms of training time.

In Figure 19 it is explicit the well use of the CPU by the MXNet framework part, almost reaching the 80% in comparison with the past 20% of the PyTorch and TensorFlow frameworks. This explains why the MXNet outperformed the other frameworks in the training time metric, it used 4x more resources available to train the model.

Figure 20 shows the memory utilization rates. We can see a slight increase in memory utilization compared with the first experiments with CNNs models. Reaching over 20% in MXNet and PyTorch scenarios shows us that a signifi-

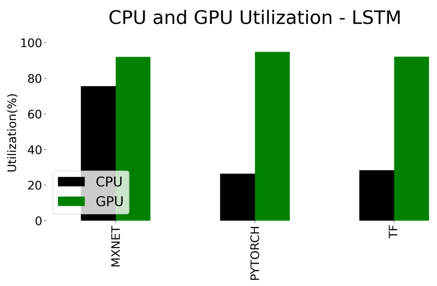

Figure 19. CPU and GPU utilization graph for LSTM model.

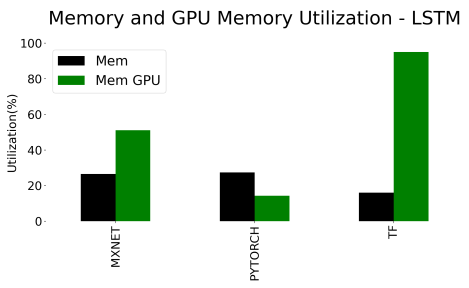

Figure 20. Memory and GPU memory utilization graph for LSTM model.

cant amount of memory is not strictly necessary for training DNNs.

YoLov7

YoLov7 is a DNN used for multiple purposes, such as image segmentation and object detection. It is implemented using the PyTorch framework, and since this is a quite complex model, we will not convert it to other frameworks in this work.

The dataset utilized for training the YoLo model was the MS COCO 2017 Lin *et al.* [2015] dataset, a collection of over 100.000 RGB images that have large image sizes and comes along with annotations for the training dataset that need to be used to train the model for the object detection task.

Table 8. Virtual Machine Configuration.

Time(h)	CPU(%)	Mem(%)	GPU(%)	GPU Mem(%)	Cost($)
13.99	78.13	61.87	76.79	90.18	10.91

The Table 8 above shows the metrics collected from the YoLov7 training process. The training, again, does not intend to reach the end of convergence; it only requires some workload to collect the metrics so we can see how well the environment is being used.

The training was conducted over a total of 10 epochs. Unlike the original work, which used an image size of 640x640, this work set the image size to 480x480 since the original training loop caches the images in the GPU memory before the start of the training, and the instantiated VM does not have enough GPU Memory for that.

We can see that a very large DNN consumes a very long time to train, 10 epochs makes the training process we conduct almost 14 hours long with an average of over 75% CPU and GPU utilization, over 60% memory utilization and 90% GPU memory utilization but because the images utilized have 5600x more pixels compared to the 64x64 images used in the first experiments these numbers reveal to us that the amount of memory the environment has is not too impactul in the training time becoming a resource that can be dimensioned if strictly needed.

6 Conclusion

This work presents a methodology for evaluating the performance of neural networks in cloud environments based on experimental design techniques. Additionally, it presents three case studies that collected data from virtual machines provided by Microsoft Azure, Amazon AWS, and Google Cloud Platform while training neural networks.

The first study shows the validity of the proposed methodology for evaluating this training using cost as a metric.

The second study intended to compare the capacity of two cloud providers to train neural networks. This comparison resulted in the perception that the cloud provider does not have a meaningful impact on training time.

The third study adds another cloud provider to the comparison and shows that choosing a cloud provider does not impact the training process and can be selected as preference and availability of resources. Moreover, all cloud providers have VM options in the same price range.

A series of experiments were conducted. Other meaningful conclusions are that the amount of memory available is not a significant factor in training and can be a better dimensioned resource to avoid some expenses.

Factors such as the DNN architecture complexity and image size are more impactul on training time, they can exponentially increase the time costs without affecting at all the resources utilization metrics.

Also, this work shows that complex models such as YoLov7 can be trained on cloud environments since the VMs chosen for the experiments are some of the most basic ones available on the cloud platforms used, existing other VMs qualities that are more powerful and more adequate for the task.

The studies show that the cloud has the potential to offer a low-cost environment with accelerators (GPUs), allowing neural network development to be ten times faster than in environments without these accelerators.

The collected metrics demonstrate that, in addition to the reduced cost, the various existing frameworks can effectively utilize this environment.

However, the increased utilization of CPU, GPU, and memory resources does not consistently affect training time, suggesting that an investigation varying the amount of available resources is necessary.

Other aspects, such as studying the training of networks with parallelized machines, were not addressed and may be considered for future research.

Authors' Contributions

Cláudio Márcio de Araújo Moura Filho contributed to the conception of this study and He performed the experiments. He is the main contributor and writer of this manuscript. Erica Teixeira Gomes de

Sousa supervised the work and wrote and revised the text. All authors read and approved the final manuscript. This work is a extension of Moura Filho and Sousa [2024].

Competing interests

The authors declare no conflicts of interest.

Availability of data and materials

The MNIST and CIFAR-10 datasets can be downloaded through the frameworks used. The Penn Treebank can be found on: https://www.kaggle.com/datasets/aliakay8/penn-treebank-dataset/data. The MS COCO 2017 can be downloaded from https://cocodataset.org/download. The scripts for the experiments and the data obtained are available upon request to the authors.

References

Adebayo, I., Manganyela, N., and Adigun, M. O. (2020). Cost-benefit analysis of pricing models in cloudlets. *2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, pages 1–5. DOI: 10.1109/IMITEC50163.2020.9334114.

Alahmed, Y., Abadla, R., Badri, A. A., and Ameen, N. (2023). How does chatgpt work. examining functionality to the creative ai chatgpt on x's (twitter) platform. *2023 Tenth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 1–7. DOI: 10.1109/SNAMS60348.2023.10375450.

Carneiro, T., Da Nóbrega, R. V. M., Nepomuceno, T., Bian, G.-B., De Albuquerque, V. H. C., and Rebouças Filho, P. P. (2018). Performance analysis of google colaboratory as a tool for accelerating deep learning applications. *2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, 6:61677–61685. DOI: 10.1109/ACCESS.2018.2874767.

Correia, F. (2011). Definição de computação em nuvem segundo o nist. Avalable at: `https://plataformanuvem.wordpress.com/2011/11/21/definicao-de-computacao-em-nuvem-segundo-o-nist/` Accessed: 2024-08-31.

Dally, W. J., Keckler, S. W., and Kirk, D. B. (2021). Evolution of the graphics processing unit (gpu). *IEEE Micro*, 41(6):42–51. DOI: 10.1109/MM.2021.3113475.

Edinat, A. (2018). Cloud computing pricing models: A survey. *International Journal of Scientific Engineering and Research (IJSER)*. DOI: 10.14257/ijgdc.2013.6.5.09.

Elshawi, R., Wahab, A., Barnawi, A., and Sakr, S. (2021). Performance analysis of high performance computing applications on the amazon web services cloud. *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, page 2017–2038. DOI: 10.1109/CloudCom.2010.69.

Jackson, K. R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, H. J., and Wright, N. J. (2010). Dlbench: a comprehensive experimental evaluation of deep learning frameworks. *Cluster Computing*, 24:159–168. DOI: 10.1007/s10586-021-03240-4.

Jain, D., Singh, P., Pandey, Kumar, A., Singh, M., Singh, H., and Singh, A. (2022). Lung cancer detection using convolutional neural network. *2022 3rd International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, pages 1–4. DOI: 10.1109/ICICT55121.2022.10064513.

Juve, G., Deelman, E., Vahi, K., Mehta, G., Berriman, B., Berman, B. P., and Maechling, P. (2009). Scientific workflow applications on amazon ec2. *2009 5th IEEE International Conference on E-Science Workshops*. DOI: 10.1109/ESCIW.2009.5408002.

Kandpal, M., Gahlawat, M., and Patel, K. R. (2017). Role of predictive modeling in cloud services pricing: A survey. *2017 7th International Conference on Cloud Computing, Data Science Engineering - Confluence*. DOI: 10.1109/CONFLUENCE.2017.7943158.

Kansal, S., Singh, G., Kumar, H., and Kausha, S. (2014). Pricing models in cloud computing. *Proceedings of the 2014 International Conference on Information and Communication Technology for Competitive Strategies*. DOI: https://doi.org/10.1145/2677855.267788.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Available at: `https://api.semanticscholar.org/CorpusID:18268744`.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444. DOI: https://doi.org/10.1038/nature14539.

LeCun, Y. and Cortes, C. (2005). The mnist database of handwritten digits. Available at: `https://api.semanticscholar.org/CorpusID:60282629`.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2015). Microsoft coco: Common objects in context. DOI: 10.48550/arXiv.1405.0312.

Liu, J., Dutta, J., Li, N., Kurup, U., and Shah, M. (2018). Usability study of distributed deep learning frameworks for convolutional neural networks. *SIGKDD Conference on Knowledge Discovery and Data Mining*. Available at: `https://www.kdd.org/kdd2018/files/deep-learning-day/DLDay18_paper_29.pdf`.

Moura Filho, C. and Sousa, E. (2024). Uma metodologia para a avaliação de desempenho e custos do treinamento de redes neurais em ambientes de nuvem. In *Anais do XXIII Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, pages 1–12, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/wperformance.2024.1986.

Shams, S., Platania, R., Lee, K., and Park, S.-J. (2017). Evaluation of deep learning frameworks over different hpc architectures. *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. DOI: 10.1109/ICDCS.2017.259.

Shi, S., Wang, Q., Xu, P., and Chu, X. (2016). Benchmarking state-of-the-art deep learning software tools. *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, pages 99–104. Available at: `https://api.semanticscholar.org/CorpusID:3040291`.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556. Available at: `https://`

api.semanticscholar.org/CorpusID:14124313.

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.48550/arXiv.2207.02696.

Wu, C., Buyya, R., and Ramamohanarao, K. (2019). Cloud pricing models: Taxonomy, survey, and interdisciplinary challenges. *ACM Comput. Surv.*, 52(6). DOI: 10.1145/3342103.

Wu, Y., Liu, L., Pu, C., Cao, W., Sahin, S., Wei, W., and Zhang, Q. (2018). A comparative measurement study of deep learning as a service framework. *IEEE Transactions on Services Computing*, 15:551–566. Available at: https://api.semanticscholar.org/CorpusID:53106015.

Xie, X., He, W., Zhu, Y., and Xu, H. (2023). Performance evaluation and analysis of deep learning frameworks. In *Proceedings of the 2022 5th International Conference on Artificial Intelligence and Pattern Recognition*, AIPR '22, page 38–44, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3573942.3573948.

Zhu, H., Akrout, M., Zheng, B., Pelegris, A., Jayarajan, A., Phanishayee, A., Schroeder, B., and Pekhimenko, G. (2018). Benchmarking and analyzing deep neural network training. *2018 IEEE International Symposium on Workload Characterization (IISWC)*, pages 88–100. DOI: 10.1109/IISWC.2018.8573476.