KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Science and Engineering

Second cycle, 30 credits

Comparative Study of the Inference of an Image Quality Assessment Algorithm

Inference Benchmarking of an Image Quality Assessment Algorithm hosted on Cloud Architectures

JESPER PETERSSON

Stockholm, Sweden, 2023

Comparative Study of the Inference of an Image Quality Assessment Algorithm

Inference Benchmarking of an Image Quality Assessment Algorithm hosted on Cloud Architectures

JESPER PETERSSON

Degree Programme in Computer Science and Engineering
Date: May 23, 2023

Supervisors: Wafaa Mushtaq, Christina Jenkins
Examiner: Aristides Gionis
School of Electrical Engineering and Computer Science
Host company: Devoteam Creative Tech
Swedish title: En Jämförande Studie av Inferensen av en Bildkvalitetsbedömningsalgoritm
Swedish subtitle: Inferens Benchmark av en Bildkvalitetsbedömingsalgoritm i olika Molnarkitekturer

Abstract

The utilization of Machine Learning has a wide range of applications, with one of its most popular areas being Image Recognition or Image Classification. To effectively classify an image, it is imperative that the image is of high quality. This requirement is not always met, particularly in situations where users capture images through their mobile devices or other equipment. In light of this, there is a need for an image quality assessment, which can be achieved through the implementation of an Image Quality Assessment Model such as BRISQUE.

When hosting BRISQUE in the cloud, there is a plethora of hardware options to choose from. This thesis aims to conduct a benchmark of these hardware options to evaluate the performance and sustainability of BRISQUE's image quality assessment on various cloud hardware. The metrics for evaluation include inference time, hourly cost, effective cost, energy consumption, and emissions. Additionally, this thesis seeks to investigate the feasibility of incorporating sustainability metrics, such as energy consumption and emissions, into machine learning benchmarks in cloud environments.

The results of the study reveal that the instance type from GCP was generally the best-performing among the 15 tested. The Image Quality Assessment Model appeared to benefit more from a higher number of cores than a high CPU clock speed. In terms of sustainability, it was observed that all instance types displayed a similar level of energy consumption, however, there were variations in emissions. Further analysis revealed that the selection of region played a significant role in determining the level of emissions produced by the cloud environment. However, the availability of such sustainability data is limited in a cloud environment due to restrictions imposed by cloud providers, rendering the inclusion of these metrics in Machine Learning benchmarks in cloud environments problematic.

Keywords

Machine learning, Cloud Computing, Benchmark, Image Quality Assessment

Sammanfattning

Maskininlärning kan användas till en mängd olika saker. Ett populärt verksamhetsområde inom maskininlärning är bildigenkänning eller bildklassificering. För att utföra bildklassificering på en bild krävs först en bild av god kvalitet. Detta är inte alltid fallet när användare tar bilder i en applikation med sina telefoner eller andra enheter. Därför är behovet av en bildkvalitetskontroll nödvändigt. BRISQUE är en modell för bildkvalitetsbedömning som gör bildkvalitetskontroller på bilder, men när man hyr plats för den i molnet finns det många olika hårdvarualternativ att välja mellan.

Denna uppsats avser att benchmarka denna hårdvara för att se hur BRISQUE utför inferens på dessa molnhårdvaror både när det gäller prestanda och hållbarhet där inferensens tid, timpris, effektivt pris, energiförbrukning och utsläpp är de insamlade mätvärdena. Avhandlingen söker också att undersöka möjligheten att inkludera hållbarhetsmetriker som energiförbrukning och utsläpp i en maskininlärningsbenchmark i molnmiljöer.

Resultaten av studien visar att en av GCPs instanstyper var generellt den bäst presterande bland de 15 som testades. Bildkvalitetsbedömningsmodellen verkar dra nytta av ett högre antal kärnor mer än en hög CPU-frekvens. Vad gäller hållbarhet observerades att alla instanstyper visade en liknande nivå av energianvändning, men det fanns variationer i utsläpp. Ytterligare analys visade att valet av region hade en betydande roll i bestämningen av nivån av utsläpp som producerades av molnmiljön. Tillgången till sådana hållbarhetsdata är begränsade i en molnmiljö på grund av restriktioner som ställs av molnleverantörer vilket skapar problem om dessa mätvärden ska inkluderas i maskininlärningsbenchmarks i molnmiljöer.

Nyckelord

Maskininlärning, Molntjänster, Jämförelse, Bildkvalitetsbedömning

Acknowledgments

Prior to presenting this thesis, I would like to extend my heartfelt gratitude to the individuals who played instrumental roles in the selection, planning, and execution of this research. First and foremost, I express my sincere appreciation to Devoteam for granting me the invaluable opportunity to undertake my master's thesis in collaboration with their esteemed organization. I am profoundly grateful for their warm reception and unwavering assistance throughout the course of this endeavor.

Furthermore, I would like to extend a special acknowledgement to my supervisors at Devoteam, namely CJ Jenkins and Klaus Müller, whose exceptional guidance and unwavering support proved invaluable to the successful completion of this thesis. Their profound expertise, valuable insights, and constant encouragement have significantly enriched the quality and depth of my research. In addition, I am deeply grateful to my supervisor at KTH, Wafaa Mushtaq, for guiding me through this thesis.

Stockholm, May 2023
Jesper Petersson

Contents

List of Figures

List of Tables

Listings

List of Acronyms and Abbreviations

AWS	Amazon Web Services
BRISQUE	Blind/Referenceless Image Spatial Quality Evaluator
CO_{2eq}	Carbon Dioxide or Equivalents
DMOS	Differential Mean Opinion Score
FaaS	Function as a Service
FR	Full-Reference
GCP	Google Cloud Platform
GGD	Generalized Gaussian Distribution
IaaS	Infrastructure as a Service
IQA	Image Quality Assessment
kWh	Kilowatt-Hour
ML	Machine Learning
MLaaS	Machine Learning as a Service
MSCN	Mean Subtracted Contrast Normalized
NR	No-Reference
OR	Outlier's Ratio
OS	Operating System
PaaS	Platform as a Service
PLCC	Pearson's Linear Correlation Coefficient
RMSE	Root Mean Square Error
RR	Reduced-Reference
SaaS	Software as a Service
SROCC	Spearman's Rank Order Correlation Coefficient

SVM	Support Vector Machine
SVR	Support Vector Regression
TDP	Thermal Design Power
TTA	Time-to-Accuracy
VM	Virtual Machine

Chapter 1

Introduction

The interest in machine learning has grown over the past decade. It is now widely used in various domains such as computer vision, natural language processing, time series prediction, and more. To make accurate computer vision models, images of good quality are needed. However, this might not always be the case in a production setting. Pictures often contain some kind of distortion that can prevent the model from accurately classifying the image. To deal with this distortion one can use Image Quality Assessment (IQA) algorithms to determine if an image contains too much distortion for the model to be able to correctly classify it. Newer IQA algorithms take advantage of the strengths of machine learning to assess the quality of an image.

Machine learning consists of two tasks, training and inference and as machine learning models are becoming more and more complex, the time and resources needed to perform these tasks have become exceedingly large. This has made it hard for companies to train and host the model on-premise since the cost of scaling, maintaining, and acquiring relevant hardware is high. Therefore, more professionals have turned to cloud computing[1]. Although, when choosing a cloud platform to host the model one might be overwhelmed by all the alternatives. First, there is the choice of provider, then it is the choice of hardware to run it on.

Right now there is a gap in research where the inference of IQA algorithms hosted on cloud platforms have been evaluated. Thus this thesis seeks to evaluate the differences in terms of performance and sustainability of IQA algorithms hosted in three different cloud platforms as well as benchmark the hardware while running inference of a IQA model.

1.1 Background

The project aims to explore the area of cloud computing combined with machine learning, more specifically comparing and benchmarking different cloud hardware and platforms to investigate which is more suitable for a Image Quality Assessment algorithm in terms of performance and sustainability.

Since cloud-based infrastructures have become more popular in recent years during development, and with machine learning algorithms becoming more expensive in terms of resources and more complex in terms of structure, hosting these algorithms may present an issue in today's mobile phones. Using cloud-based infrastructures to store and use these models might help solve that problem.

The interest in this project is from Laryngectomy patients and Atos Medical. Laryngectomy is a major surgery that removes either a person's Larynx or Voice-box. It is commonly done on patients that have throat cancer to remove it. After the surgery is done the patients have a permanent stoma on their necks where the patients breathe from.

Atos Medical is a company that creates breathing tools for former cancer patients who have had part of their throat removed (Laryngectomy). The patients have a hole in their throat that needs to be closed with a medical device that allows them to function normally. The device is placed in the hole, but since each patient has a unique throat, it sometimes does not fit well.

To address this issue Atos Medical is to develop an application that can be used by patients to gather relevant data on the stomas, based on images taken from their phones. However, when the patients take these images the quality of the images need to be evaluated before saving them into a database. While there are many components for such an application, this master thesis will investigate the optimal cloud hardware for an IQA algorithm for this use-case, this will be done by exploring the performance and sustainability aspect of hosting such an algorithm on 15 different hardware options within three different cloud platforms, Google Cloud Platform (GCP), Amazon Web Services (AWS), and Microsoft Azure, to determine the optimal inference and hosting platform for the algorithm. The thesis will be done together with Devoteam Creative Tech which is in collaboration with Atos Medical to develop this application.

1.2 Problem

As previously mentioned in the introduction, there is not much research on inference benchmarks that evaluate cloud hardware and software for IQA algorithms. This makes it hard for companies and professionals to decide which platform is optimal for their use case. The knowledge gained from this thesis could therefore act as a basis when evaluating cloud platforms and hardware when using similar algorithms in a cloud environment.

1.2.1 Original problem and definition

The problem is that there is not much research done on the performance and sustainability difference of the Infrastructure as a Service (IaaS) architectures. Hence, this thesis will compare three different IaaS infrastructures and five of their instance types in terms of performance and sustainability. In this case, performance refers to inference time whilst sustainability refers to cost, energy consumption, and carbon emissions.

1.2.2 Scientific and Engineering Issues

This thesis focuses on comparing both the performance and sustainability aspects of running an image quality assessment model in a cloud environment and aims at answering the question, ”What is the difference in terms of performance and sustainability between different hardware on IaaS architectures for machine learning inference?”. This can be divided into three separate research questions:

- **RQ1:** How does the inference time of running BRISQUE compare among different cloud providers and cloud hardware?
- **RQ2:** How do cost, energy consumption, and carbon emissions released by running BRISQUE compare among different cloud providers and cloud hardware?
- **RQ3:** Evaluate the inclusion of sustainability metrics, such as energy consumption and carbon emission, in a benchmark conducted in a cloud environment.

1.3 Purpose

There are four purposes with this thesis, firstly it is to support professionals and companies when choosing a cloud platform and hardware to use for inference of their image quality assessment model. Secondly, to contribute to the academy a base of the performance and sustainability of the inference of an image quality assessment machine learning model so that other researchers can expand on it later. And thirdly to help Atos Medical find a suitable cloud platform and cloud hardware for their machine learning model.

People that benefit from this are professionals and companies that are thinking about using cloud services together with machine learning but do not know which one to choose. The academy by gaining an entry into the performance and sustainability differences between the IaaS infrastructures.

Lastly, to evaluate the availability of sustainability metrics such as energy consumption and emissions when benchmarking cloud infrastructures.

1.4 Goals

The goal of this project is to do a comparison of IaaS hardware. This has been divided into the following four sub-goals:

1. To find pros and cons in terms of performance between the different IaaS hardware on different cloud platforms.

2. To find pros and cons in terms of sustainability between the different IaaS hardware on different cloud platforms.

3. Compare IaaS hardware on different cloud platforms.

4. Evaluate sustainability metrics for inference benchmarks in a cloud environment.

Consequently, the deliverable and results for the project are a well-motivated discussion on the pros and cons in terms of sustainability and performance of the IaaS infrastructures and their hardware supported by the metrics included in the comparison.

1.5 Research Methodology

The project aligns more with quantitative research than with qualitative research, since the thesis seeks to compare three different groups or three IaaS architectures[2]. There were two research methodologies that were considered for this thesis, experimental research, and comparative research. However, the latter was chosen since the project follows a similar approach to that of comparative research. The implementation was set up to gather relevant metrics from all three computer systems which were then evaluated against each other, this could be seen as a benchmarking methodology as well[3].

1.6 Delimitations

The thesis project will only examine three cloud platforms, AWS, GCP, and Microsoft Azure. These platforms have been chosen because they are the most used platforms in this area which will be explained in a later section. Only the inference of the model will be in consideration in the comparison since the cost of training can get expensive, especially for models with large inputs such as images. The thesis will neither include a detailed explanation of how the machine learning model works as the objective of the project is to analyze and compare the IaaS architectures, not the actual model.

Since there are a large number of different cloud hardware options available by the different cloud providers, the number of tested hardware options had to be limited due to time constraints, thus only 15 different cloud hardware were tested in this thesis, five for each cloud platform.

1.7 Structure of the Thesis

Chapter 2 presents relevant background information about cloud computing and image quality assessment. Chapter 3 presents how the different metrics were implemented and how the comparison was done. Chapter 4 goes through how the experiments were implemented in the project. Chapter 5 presents the results of the experiments whilst chapter 6 contains a discussion of the results. Lastly, chapter 7 presents the conclusions of the thesis and future work.

Chapter 2

Background

This chapter provides basic background information about cloud computing and virtual machines. Additionally, this chapter describes IQA terminology and basic concepts of one IQA algorithm. Lastly, the chapter describes related work such as several machine learning inference benchmarks.

2.1 Virtual Environment

2.1.1 Virtual Machine

A Virtual Machine (VM) is a layer of software that is run on physical computer hardware. Programs that run in this environment are given the picture that the software is run on dedicated physical hardware, without it actually being run on the underlying physical hardware. Instead, it is run on virtualized hardware. One or more VMs can run on the same physical machine, the VMs are often called the ”guests” and the physical machine the ”host”. Each of these guests is unaware of the other guests that run on the physical system and the guests can have different operating systems and function differently depending on the guest[4][5]. An example can be a guest VM with Windows as the Operating System (OS) running on a physical system with MacOS as the running OS.

2.2 Cloud Computing

Cloud computing lets its users borrow computing resources on demand while not having to worry about maintaining the hardware. The resources are often highly configurable, and scalable, and can be provisioned without human interaction. Common services within cloud computing are configuring networks, servers, storage, applications, and other services together with tools for analyzing them.

2.2.1 Definition

There are a lot of different definitions of cloud computing. Qian *et al.,* defines it as ”a kind of computing technique where IT services are provided by massive low-cost computing units connected by IP networks”. In addition, five technical characteristics define cloud computing, general-purpose, large-scale computing resources, shared resource pool, dynamic scheduling, and high scalability and elasticity[6].

Another definition comes from Peter Mell and Timothy Grance’s white paper ”The NIST Definition of Cloud Computing” where they define cloud computing more as a model that has on-demand network access to different configurable computing resources that are maintained and released with minimal service provider interaction. The model is composed of five technical characteristics, three service models, and four deployment models[7]. This definition is a more fitting description for the use case in this thesis and the definition will be further explained in this section.

2.2.2 Characteristics

- *On-demand self-service:* Computing capabilities can be provisioned on-demand without human interaction.

- *Broad network access:* The computing resources are easily accessible over the network.

- *Resource pooling:* To serve several consumers simultaneously computing resources are pooled with virtual and physical resources being dynamically distributed and re-distributed depending on the demand from the consumer. The consumer has no control over where the computing resources are allocated, giving a sense of location independence. However, the consumer may control the location of the provided resource on a higher level of abstraction such as country or data center.

- *Rapid elasticity:* Computing resources are highly scalable and can be flexibly distributed to the consumer, meaning that the consumer can scale up and down the number of resources they need at any time.

- *Measured service:* The resources provided are controlled and optimized by the provider. The usage of the resources is monitored and reported to provide a base when it comes to the cost of the used service.

2.2.3 Cloud Service Models

There are three standard service models, each coming with different levels of abstraction and freedom[7][8][9]. There exist more types than just these three, such as Function as a Service (FaaS) and Machine Learning as a Service (MLaaS)[10][11]. However, only the types mentioned in Peter Mell and Timothy Grance's white paper, "The NIST Definition of Cloud Computing" will be explained in this section.

The service models can be seen as a layered structure where the higher layers depend on the lower layers which are shown in figure 2.1. It shows which layer the consumer and provider manage of the system with each of the types. Among the layers are the application, data, runtime, middleware, O/S, virtualization, servers, storage, and network. In addition, the figure shows a comparison with on-premise devices where the consumer has to manage every aspect of the computer system.

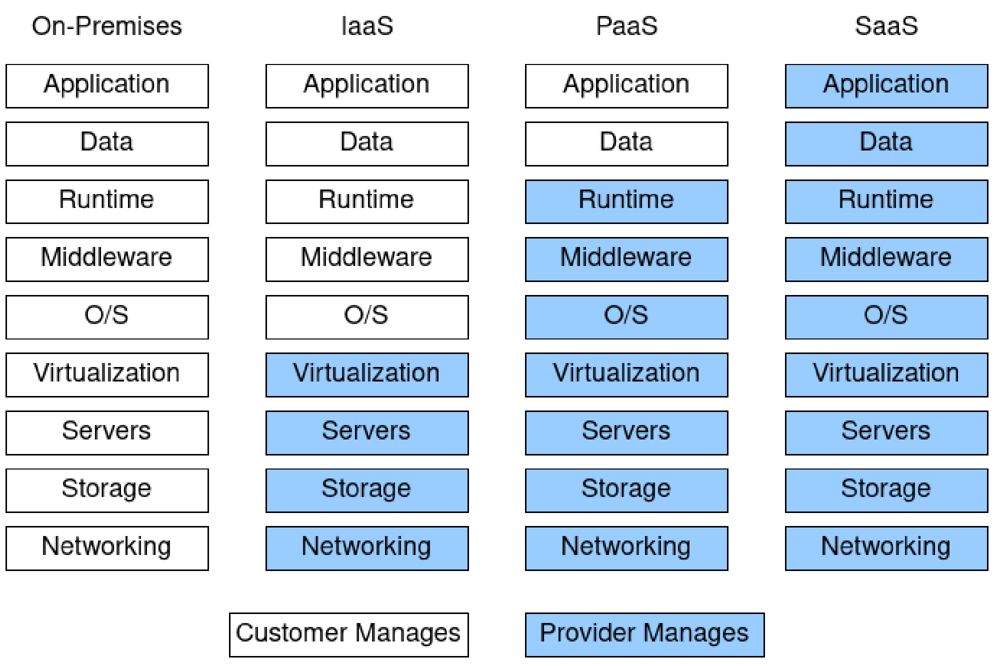


Figure 2.1: The different model service types and corresponding management of hardware

2.2.3.1 Infrastructure as a Service

As seen in figure 2.1 when using IaaS the consumer manages everything themselves except virtualization, servers, storage, and networking which is provisioned by the provider. An example of users for this service model is network architects.

2.2.3.2 Platform as a Service

Platform as a Service (PaaS) is between IaaS and Software as a Service (SaaS) at the abstraction level. It is more configurable than SaaS but less configurable than IaaS. The consumer only has to manage the application and the data of the system while the provider manages the rest, as can be seen in figure 2.1. Common use cases for this service model are server hosting for application developers.

2.2.3.3 Software as a Service

SaaS is the highest abstraction level out of the three mentioned in this section. It is a complete end-user application that is running on cloud infrastructure where the provider manages every layer shown in figure 2.1. Gmail, Google Docs, and Office 365 are examples of applications served by the SaaS model.

2.2.4 Deployment Models

Clouds are divided into four cloud types, private, community, public, and hybrid clouds. These are classified based on who controls the infrastructure for the deployment and the location of the infrastructure. The decision of deployment model is important since they satisfy different customer needs and the cost may differ among them[12].

2.2.4.1 Private Cloud

Private clouds are exclusively used within a single organization. The infrastructure can be provisioned on-premise or at another suitable location, whilst the infrastructure might be owned and managed by a third-party provider, by the organization itself, or a combination of the two. There is another alternative to private cloud, namely ”virtual private cloud”, where the cloud is located within the infrastructure of a public cloud[13].

2.2.4.2 Community Cloud

For organizations that have similar interests and concerns, a community cloud might be a better solution. A community cloud is a shared cloud between organizations in a community that shares similar concerns or interests. The cloud may be owned, managed, and operated by one or several of the companies in the community, by a third party, or by a combination of the two. The infrastructure can be hosted on or off-premises.

2.2.4.3 Public Cloud

In a public cloud, the infrastructure is operated and managed by an academic, business, company, or government and it is open for use by the general public. The infrastructure itself is located on-site of the organization that provides the cloud.

2.2.4.4 Hybrid Cloud

A hybrid cloud is a cloud where the infrastructure is comprised of several distinct cloud infrastructures and managed by a single entity. The different cloud infrastructures inter-operate through technologies that can handle application and data portability.

2.2.5 Machine Learning as a Service

The term Machine Learning as a Service was first mentioned in scientific literature by Ribeiro *et al.,* in 2016[11]. Machine Learning as a Service is a cloud service model as well. Although this service is more directed toward data engineers, data scientists, or organizations that are looking for machine learning solutions. MLaaS is a web service where consumers can train, build and deploy their Machine Learning (ML) models[14].

The benefits of using MLaaS are that the cloud already has other systems that suit the needs for Machine Learning. For example data management since clouds offer storage that is managed by the provider.

2.3 Machine Learning

Machine learning is a subfield of artificial intelligence that revolves around developing algorithms that can imitate human behavior. The four common approaches within machine learning are supervised, unsupervised, semi-supervised, and reinforcement learning. The first approach, supervised learning, is the most relevant for this thesis. Supervised learning consists of two phases, a training phase, and an evaluation phase. During the training phase, the model is supposed to map the input to the correct output, this is done with the help of labels which are earlier known mappings of certain input data. This way a model can learn features related to a group of earlier known inputs and then use this knowledge to classify new input data, this type of approach is also known as classification[15]. Support Vector Machine (SVM) is one type of classification algorithm which was utilized in this thesis.

2.3.1 Support Vector Machine

An SVM is a classification algorithm that uses a hyperplane to separate two groups of input data. The data is separated based on the underlying features of the input. In other words, the hyperplane can be used to identify new unseen observations. There may exist more than one separation of the two groups of observations, however, an SVM tries to find the hyperplane that best separates the groups. It does so by maximizing the distance from the hyperplane to a point in each of the observation groups, that distance is often referred to as the margin and the observations on the margin are called the supporting vectors. This is shown in figure 2.2.

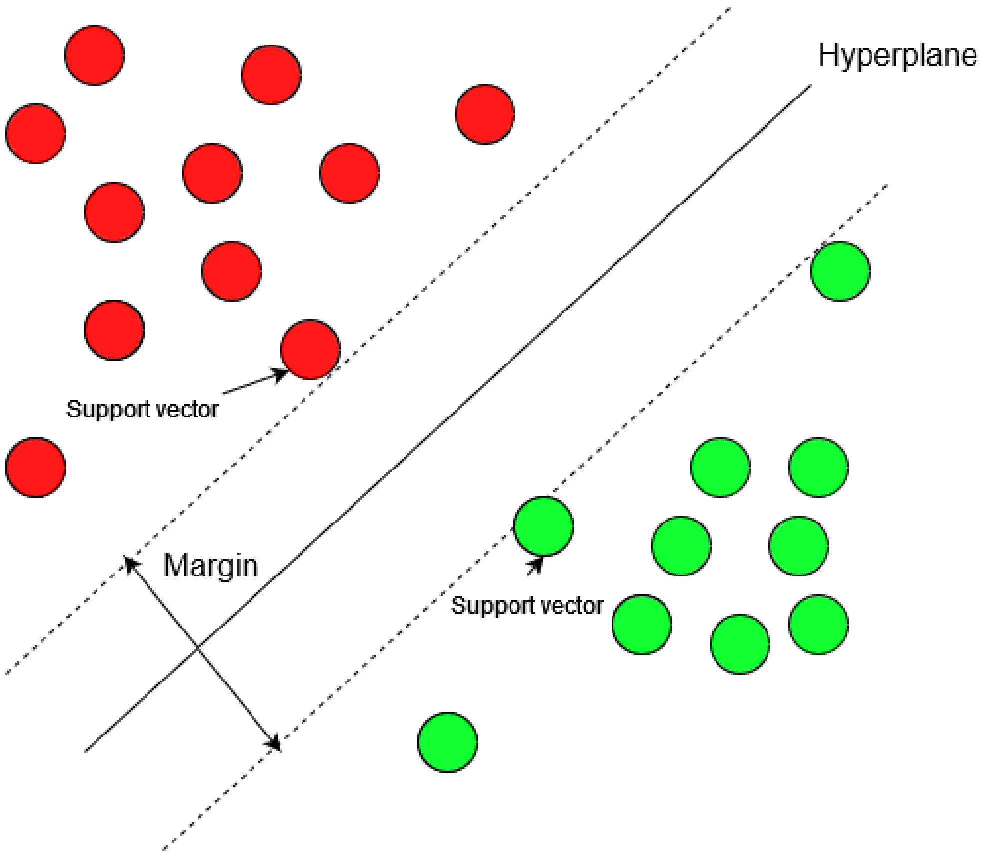


Figure 2.2: A hyperplane separating two groups of observations.

An SVM can be both linear and nonlinear, an example of a linear SVM is shown in figure 2.2 above. The complexity of an SVM can vary depending on the number of features in the observation set. For example in figure 2.2 only two features are available, thus the hyperplane is a line. The dimension of the hyperplane is one less than the number of features available in the feature space. If there would have been three features it would have been a three-dimensional space and the hyperplane would correspond to a two-dimensional plane. For an SVM to be linear the observation set needs to be linearly separable, this means that the two groups of observations can be separated by a linear hyperplane[16].

There are other ways of deciding the hyperplane than maximizing the distance between the closest points in each group of the observation set. Another approach is to have a soft margin where some points are allowed to be misclassified. This is because not all datasets are linearly separable which allows some data points to be misclassified and might still give a good performance for the SVM. The misclassified points are still within the margin but still on the wrong side of the separating hyperplane, this is implemented as a parameter that the user has to set(number of misclassified data points)[17].

There are other ways of dealing with non-separable data, commonly with the use of the kernel function, which is another user-specific parameter withing the domain of SVMs. The function of the kernel is to decide the shape and type of the hyperplane used for classification. It allows the SVM to handle non-linearly separable data by employing a mathematical trick that projects the data points into a higher-dimensional space, where a hyperplane can be found to effectively separate the groups of data points. Some commonly used kernel functions include the linear kernel, polynomial kernel, Gaussian (RBF) kernel, sigmoid kernel, and more. The choice of kernel function depends on the specific problem, data characteristics, and the desired complexity of the decision boundaries[17].

SVM does not only do classification, it can be used to do regression as well. When doing regression with an SVM it is often called a Support Vector Regression (SVR). It works similarly to an SVM doing classification but instead they are used for regression, all the concepts explained earlier are still the same. The point with regression is to fit a function onto as many data points as possible. In SVR it is the hyperplane that has the maximum number of points on it that is the best fit[18].

2.4 Image Quality Assessment

According to Bovik *et al.*, IQA is the subjective human perception of an image, i.e if a human can recognize the contents of an image[19]. There are three different categories depending on the availability of a reference image. The first category is Full-Reference (FR), which assesses the quality of an image in comparison to a pristine reference image which together with the distorted image is sent into the assessment algorithm. One of the other categories is Reduced-Reference (RR) where the algorithm has the distorted image and some information about the reference image but not the actual reference image. Lastly, there is No-Reference (NR) which is going to be used in this project. NR refers to when the algorithm only has the distorted image without any information about a reference image[20].

2.4.1 No-Reference Image Quality Assessment

As stated in the previous section, NR-IQA refers to the methods of calculating the perceptual quality of an image when there is no information about a pristine reference image available. Within the area of IQA there exists four types of distortions that can affect an image. They are the following: synthetic distortions, with blur, JPEG, or noise, authentic distortions with poor lighting conditions or sensor limitations and GAN-based distortion[21].

Traditionally algorithms focused on synthetically distorted images. The problem with these images is that they contain limited distortion diversity, and content, and do not encapsulate the more complex distortions caught in real-world images. However, as more datasets with distorted real-world images have emerged, more focus have shifted to understanding the complex distortions of authentic images[22][23][24][25].

There exist several metrics that are used for evaluating NR IQA algorithms. Among the most usual are Outlier's Ratio (OR), Pearson's Linear Correlation Coefficient (PLCC), Root Mean Square Error (RMSE), Spearman's Rank Order Correlation Coefficient (SROCC). However, in recent benchmarks, SROCC and PLCC are the ones used the most. SROCC is a metric that calculates the relation between two variables that are ranked. The range of the values for SROCC is between -1 and 1. Values close to 1 mean that the variables have a positive correlation meanwhile values close to -1 indicate that the variables have a negative correlation. If it is a value closer to 0 then it means that the two variables are not correlated. PLCC is a metric to compare the relation between two variables as well, similar to SROCC. The range and the interpretation of the values are the same as for SROCC, but the two are calculated with different formulas[26].

2.4.2 Machine Learning Approach

Recently, as more datasets with distorted images have emerged researchers have been trying to utilize Machine Learning to get more accurate results. The approach used in this thesis used an SVM to capture the distortions of images.

2.4.3 BRISQUE

This model was proposed by Mittal *et al.,* in 2012[20]. The purpose of the model is that it should predict the Differential Mean Opinion Score (DMOS) of an image with the help of a machine learning model, DMOS is the subjective quality of an image. The authors did a spatial approach, meaning it analyzes the pixel-level characteristics of an image to estimate its perceived quality. This attribute makes it particularly useful in scenarios where a reference image is unavailable or difficult to obtain. The algorithm can be explained in a two-step process.

First, it extracts a set of natural scene statistics features from the image, which capture information related to luminance, contrast, and local texture. 36 features are extracted from two scales of the same image. 18 features are extracted from the original image scale and the same 18 features are extracted from the same image but at a reduced resolution. The 18 features that are extracted from both versions of the image are, shape and variance of the luminance values of the image, which are defined as the Mean Subtracted Contrast Normalized (MSCN) coefficients fitted on a Generalized Gaussian Distribution (GGD). The other features shape, mean, left variance, and right variance for the statistical relationship between neighboring pixels across four different orientations, horizontal, vertical, main diagonal, and secondary diagonal. It is done by pairwise products of neighboring MSCN coefficients across the four orientations. In the second step, when all the features have been extracted from the image they are sent as input to a SVR which then outputs the predicted DMOS of the image.

2.5 Related work area

2.5.1 Benchmarks

2.5.1.1 DAWNBench

In 2017, Coleman *et al.,* created DAWNBench, an end-to-end deep learning benchmark. The benchmark measures both the training and inference of a machine learning model and is most known for introducing the metric Time-to-Accuracy (TTA) when benchmarking the training of a machine learning model. When DAWNBench was released in 2017 it supported benchmarking in the following domains: image classification and question answering. It was later extended to include segmentation, machine translation, and video classification as well. For the inference part of the benchmark, latency and cost was first used as metrics, additionally, they added energy consumption later. In this article, cost refers to money(in USD) and latency refers to the time it takes for the model to process a single item. Both metrics are calculated by taking the average latency of processing a single item and the average cost of processing one item for 10000 items[27]. The paper is similar to how the experiments are going to be conducted in this thesis and it includes similar metrics. However, the focus is on benchmarking the training of deep learning models.

2.5.1.2 AIBench

Gao *et al.,* published AIBench in 2019[28] and it is another end-to-end benchmark that focuses on three different internet service domains, search engines, social networks, and e-commerce. Among these domains, they identified 16 representative machine-learning tasks that the benchmark can evaluate, such as classification, image generation, image-to-text, and object detection to name a few. The AI benchmark is both configurable and flexible to include data input, AI problem, domain, online inference, offline training, and deployment tools. The online inference is the most interesting part since it is relevant to this thesis. The online inference in this paper consists of loading a trained model onto a serving system such as Tensorflow serving and then performing the experiments. From a performance aspect, the online inference includes metrics such as query response latency, tail latency, and throughput. In addition, it contains inference energy consumption and inference accuracy. This paper is more in line with the work of this thesis since it includes a benchmark for online inference. The paper does not include an explanation of the details of the metrics or how they are gathered. Inference energy consumption and inference time are two metrics that are included as a metric in this thesis. The inference accuracy is a bit vague of what they are referring to and accuracy is not a metric suitable for evaluating the performance of an infrastructure which is the focus of this thesis, instead it includes inference time.

2.5.1.3 MLPerf Inference Benchmark

MLPerf inference benchmark was published in 2020 by Reddi *et al.,* [29]. It is a benchmark for evaluating inference on different systems and is motivated by over 30 organizations and more than 200 machine learning practitioners from industry and academia. MLPerf supports image classification, object detection, and machine translation and is divided into four different scenarios: single-stream, multi-stream, server, and offline. Related to this thesis is the offline scenario which includes one metric, inference time. This thesis differs from the paper by including several metrics for comparison. MLPerf should be a general benchmark for several different domains and areas whilst the comparison in this thesis only seeks to investigate one specific model on IaaS systems. Therefore the other metrics included in this thesis are crucial to making a fair comparison and as this thesis is more specific it lets it include more metrics that suit this comparison. For example, cost and emissions are rather specific to cloud platforms and inference latency is a valuable metric for the system, this is supported by the other benchmarks that have included these metrics as well. They are sustainability metrics as well which are not included in MLPerf Inference Benchmark which is only a benchmark to evaluate performance.

2.5.1.4 A Study of Machine Learning Inference Benchmarks

The paper ”A Study of Machine Learning Inference Benchmarks” by Alvarado Rodriguez *et al.,* published in 2020[30], is a study comparing two benchmarks, MLPerf and MLMark and the results gathered from these benchmarks. The paper does have an emphasis on edge devices. As mentioned earlier, MLPerf supports different scenarios, single-stream, multi-stream, server, and offline with 90th-percentile latency, number of streams, queries per second, and throughput as metrics respectively. Whilst MLMark includes two metrics, 95th-percentile latency, and throughput. In the comparison four metrics were used, one of the most used performance metrics, latency which is another word for inference time, in addition to cost and energy consumption was used as well. From the paper, it can be derived that the performance of machine learning models differs from various systems, hence why it is necessary to do such comparisons.

2.5.1.5 InferBench

The last benchmark mentioned is Inferbench by Zhang *et al.,* which was published in 2021[31]. This is a benchmark for comparing deep learning serving systems. It includes three tiers of measurement: hardware, software, and pipeline. For benchmarking hardware it includes metrics such as latency, throughput, and cost. When it comes to benchmarking different cloud providers they support energy consumption and CO2 emissions as well. In their software tier, they include tail latency as a metric. Throughput is measured as the number of requests per second and latency is the time in milliseconds it takes for an instance to be processed. Throughput and latency are measured as an average over several experiments while cost is measured as cloud cost, energy consumption, and CO2 emissions, with cloud cost refereeing to hourly rates or cost per request. This paper is one of more similar to this thesis since similar metrics are included and gathered in the same way. Additionally, the task of inference is chosen by the practitioner and not preset like in the other benchmarks. InferBench takes the whole system into consideration and not only the performance of the hardware. This thesis also follows a similar approach for measuring the metrics as in this paper, therefore this is the most relevant work to this thesis.

2.5.1.6 Comparison of Benchmarks for Machine Learning Cloud Infrastructures

In 2021 Madan and Reich published ”Comparison of Benchmarks for Machine Learning Cloud Infrastructures” where they compare different machine learning benchmarks and how suitable they are for comparing cloud infrastructures[32]. The comparison includes which datasets and metrics the benchmarks contain. The benchmarks included in the comparison are MLPerf, DAWNBench, DeepBench, DLBS, TBD, AIBench, and ADABench. The paper primarily focuses on training benchmark and their metrics and not so much on inference benchmarks but their discussions about suitable metrics for benchmarking cloud platforms and that today’s benchmark is not suitable to compare the infrastructures of cloud providers are highly relevant to this thesis since there does not exist a benchmark that considers this. The arguments are that there needs to be more suitable metrics for benchmarks for cloud infrastructures such as cost, and compute and storage performance since they are missing at the moment in benchmarks. This is the case for some of the benchmarks mentioned in this thesis as well, however, newer benchmarks have started to include such metrics. Additionally, there are no guidelines on how to implement these benchmarks with some metrics being hard to impossible to evaluate without tools provided by the cloud providers. They also talk about other sustainability metrics such as energy consumption which this thesis seeks to evaluate.

2.5.2 CO2 Emissions Within Machine Learning

In the paper ”Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning” by Henderson *et al.,* published in 2020[33] the authors introduce a framework for reporting energy and carbon usage. They state that the compute and energy demands of machine learning methods are growing exponentially, thus working towards more energy and carbon-efficient methods is crucial for a sustainable future as machine learning systems have the potential to contribute to carbon emissions. The metrics energy and carbon usage are measured in Kilowatt-Hour (kWh) and Carbon

Dioxide or Equivalents (CO_{2eq}) respectively. The sustainability aspect presented is what this thesis has tried to incorporate as well. Energy usage metrics used in many of the benchmarks above do not consider the sustainability aspect of the machine learning algorithms, instead, they mainly include the metric for benchmarking edge devices since their energy is limited. However as training and inference of a model become more complex and resource-heavy, the energy consumption metric can play a bigger part in benchmarks. In addition, some of the newer benchmarks support these types of metrics for evaluating the sustainability of a machine learning model. Thus the reason for including emissions as a metric in this thesis.

2.6 Summary

Here a summary of the background together with the related work is presented as well as a table showing an overview of the inference benchmarks mentioned in the related work section. The chapter started with an introduction to the cloud where different cloud service and deployment models were mentioned. After came an explanation of IQA and specifically NR-IQA and its' relation to machine learning. The related work section compared four inference benchmarks to the metrics of this thesis and discussed what this thesis does similar and how it differentiates from the other works. Table 2.1 gives an overview of the benchmarks with their tasks, domains, and metrics. MLPerf do have a training benchmark as well but they are seen as separate benchmarks unlike DAWNBench, that is why an overview of MLPerfs training benchmark was not included, because it is a separate benchmark and it is not relevant for the thesis.

Benchmark	Tasks	Domains	Metrics	Relevance to thesis
DAWNBench	Training, Inference	Image classification, Question answering, Segmentation, Machine translation, Video Classification	Training time, Training cost, Time-to-accuracy, Inference latency, Inference cost, Energy, Sample complexity	Includes latency, cost, and energy. Emphasis on training rather than inference. No guidelines for the implementation of metrics. Does not cover all aspects of offline inference thus some metrics are not representative of this thesis.
AIBench	Training, Inference	Component benchmarks such as Image classification, Image generation, Text-to-text, etc, and micro-benchmark including convolution, fully connected, etc	Query response latency, Tail latency, Throughput, Accuracy, Energy Consumption	Inclusion of the energy consumption metric. Vague or no explanation of the implementation and usage of the metrics.
MLPerf	Inference	Image classification, Object detection, Machine translation	90th-percentile latency Number of streams subject to latency bound Queries per second Throughput Inference Time	Includes inference time as a metric
InferBench	Inference	Custom	Latency Throughput Cost Energy consumption CO2 emissions Memory and computation Tail latency Resource usage Advanced features	Relevant and useful metrics. Allows for custom domains. Most in line with this thesis, especially the hardware part. More general than this thesis. No information about how the metrics are implemented is provided.

Table 2.1: Summary of all inference benchmarks mentioned in this thesis. The summary consists of the name of the benchmark, its' tasks, domains, metrics, and lastly what it did well and what it could have done better.

Chapter 3

Method

The purpose of this chapter is to provide an overview of the research method used in this thesis. Section 3.1 describes the research process. Section 3.2 details the research paradigm. Section 3.3 focuses on the data collection techniques used for this research. Section 3.4 describes the experimental design and the software and instance types used in this thesis.

3.1 Research Process

To solve the research question asked in this thesis benchmarking is going to be used to collect data about the performance and sustainability factors when running the chosen model in a cloud environment. The research process for this thesis can be split up into three different stages. The first stage contains the data collection, choice of the model, training of the chosen model, or finding an already trained model. In the second stage, the experiments are implemented and in the last stage, they are conducted. An overview of the different steps in the process is given below.

Overview of the process:

Step 1 Gather information about the cloud

Step 2 Finding data and a model for the problem

Step 3 Training the model on the chosen dataset

Step 4 Implement experiments

Step 5 Implementing the trained model together with the experiments on the different IaaS architectures

Step 6 Conducting the experiments on the IaaS architectures

Step 7 Analyze and discuss the results

In addition, figure 3.1 shows the different stages of the research process for this thesis. As shown the first stage is setting up the experimental design by either training the model or finding an already trained model. Stage 2 is conducting the experiments of the thesis, there we need to create an VM, and upload the code for the experiments together with the data to conduct them. After that, the requirements are installed and the environment is ready to run the experiments.

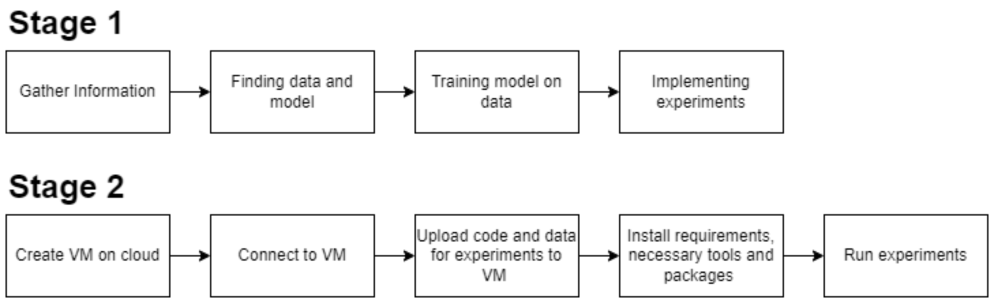


Figure 3.1: Research Process

3.2 Research Paradigm

In terms of research philosophy, this research is more similar to positivism rather than constructivism or pragmatism, in the sense that only quantitative methods are used to acquire data. Furthermore, the research is investigating the relationship between two or more variables instead of the reason behind it where the variables are the different cloud providers. As stated before the project is more similar to quantitative research than qualitative research since the thesis seeks to compare three different groups or three cloud platforms and their hardware[2]. There were two research methodologies that were considered for this thesis, experimental research, and comparative research. The latter was chosen since the project follows a similar approach to that of comparative research. The implementation was set up to gather relevant metrics from all three cloud providers which were then evaluated against each other, this could be seen as a benchmarking methodology as well[3].

3.3 Data Collection

To collect the data needed to do the comparison, five different metrics have been defined, inference time, hourly cost, effective cost, emissions, and energy consumption. Data will be collected based on these five metrics. Inference time is measured as the time it takes for the model to process one image. Hourly cost represents the monetary expenditure in USD associated with the utilization of computing resources per hour. Effective cost is the expenditure in USD to process one item. Emissions are measured as the amount of carbon emissions produced by the model per image, the unit of this metric is $\text{CO}_{2\text{eq}}$. To calculate the emissions carbon intensity is used. Schmidt *et al.,* define carbon intensity as a weighted average of the emissions produced from generating electricity from different energy sources, a so-called energy mix[34]. Carbon intensity refers to how many grams of carbon dioxide(CO_2) it takes to produce one unit of kilowatt per hour. The last metric, energy consumption, is measured as the amount of energy it takes for one image to be processed, the unit is kWh.

3.4 Experimental Design

In this section, the setup and the experiment conducted are explained in detail. Furthermore, all the tools used during the experiment and setup are given.

3.4.1 Test Environment

The experimental environment of this thesis is that the experiment is conducted on a VM that is hosted on the cloud that is under test. These environments can be run on different hardware which is chosen upon creation of the VM. Therefore, to conduct this experiment one would need to have an account on the three different cloud platforms mentioned in section 3.4.2 below. Once an account has been created the process of conducting the experiment can be started which is explained below:

Step 1 Create a VM on the cloud platforms IaaS solution

Step 2 Upload code and data needed to conduct the experiment

Step 3 Connect to VM that is under test

Step 4 Install requirements needed to conduct the experiment

Step 5 Run experiment

This process is then done for every different setup of hardware investigated in this thesis. The hardware that was tested can be seen in table 3.1 in section 3.4.6.

3.4.2 Cloud Platforms

The thesis is a comparative study of three cloud infrastructures. More could have been included to make a more extensive comparison but then the cost of the comparison would have been higher. For that reason only the three most used cloud platforms were included, AWS, Azure, and GCP, according to Synergy Research Group[35].

3.4.3 Machine Learning Model

For the image quality assessment model, Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE)[20] was chosen because it is a popular IQA model and there is already a lot of support for the implementation of the model with libraries that support the implementation and trained parameters. Other models that were in consideration were MANIQA, MetaIQA, and RM-IQA[36][37][22]. However, due to the limited time of the thesis they were not chosen because no implementation was available that only used CPUs, and implementing them from the beginning would have been too time-consuming. Therefore a model with an already existing implementation was chosen.

3.4.4 Dataset

There were different options for the choice of a dataset. Among these were CIDIQ[38], KonIQ-10k[24], KADID-10K[39], LIVE[23], PIPAL[40], and TID2013[41]. Most of the previous datasets only focused on synthetically distorted images which presents a challenge when used together with NR IQA method since the images do not capture complex distortions from the real world. Among the datasets considered in this thesis, LIVE and KonIQ-10k include images that are more similar to real-world distortions. LIVE was then chosen because there existed a library, Image Quality, that included an already trained version of the BRISQUE model on that dataset.

3.4.5 Metrics

The metrics chosen for the comparison are inference time, hourly cost, effective cost, emissions, and energy consumption. Inference time is the most common metric in machine learning inference benchmarks[30] hence why it were chosen in this comparison. Hourly Cost is a common metric used in machine learning inference benchmarks when comparing on cloud platforms or hardware [31][29] hence why it was included. However, different cloud providers offer varying pricing models and resource options. Including effective cost in a machine learning benchmark allows for direct comparison of the financial impact of running workloads on different cloud platforms. It enables users to assess which provider offers the most cost-effective solution for their specific instance type. In many practical scenarios, machine learning models are deployed in resource-constrained environments where cost optimization is crucial. By including effective cost as a benchmark metric, the evaluation becomes more representative of real-world considerations, allowing users to assess the trade-offs between model performance, computational resources, and financial constraints.

Newer benchmarks have started to include energy consumption as a metric for inference and training, to consider the sustainability aspect of the models and infrastructures. Further, both energy consumption and emissions are crucial to understanding the potential impacts on climate, from machine learning models in research and production [33]. However, using emissions and energy consumption as a metric comes with a challenge. The ways of measuring those metrics are a complex process because only the cloud providers have access to the exact measurements. All the cloud providers compared in this thesis include a service that calculates the carbon footprint produced from the usage of their services. However, this calculator is still in early development and only calculates the quarterly or monthly produced carbon footprint for all usage on the cloud platform. Instead, there are tools that estimate these values in a cloud environment. The estimations are conducted using the CodeCarbon emissions tracker*, a joint effort from authors of [42] and [43].

3.4.6 Hardware

The hardware chosen for the comparison was based on the different categorization made by the providers. All three providers include four similar categories, general purpose, compute-optimized, memory-optimized, and accelerated computing. This thesis is going to test and evaluate three different pieces of hardware from their general purpose category and two different pieces of hardware from their compute-optimized category, for each of the

* https://codecarbon.io/

cloud platforms. Consequently, in order to provide a more comprehensive and focused comparison, a deliberate decision was made to select and assess only two categories. The general purpose category was selected as it represents a versatile and commonly used hardware option suitable for a wide range of applications and workloads. Additionally, the compute-optimized category was chosen for further analysis due to its relevance in scenarios that demand high computational power, such as machine learning.

While acknowledging that other categories may hold relevance and value as well, the decision to focus on only two categories is a pragmatic one, considering the time constraints associated with conducting a comprehensive evaluation. This selection enables the thesis to allocate sufficient attention and resources to thoroughly test and evaluate the chosen hardware options, resulting in a more reliable and informative comparison.

Each setup of the virtual machine on which the experiments were conducted ran one of the hardware listed in table 3.1 together with information about the environment. Since the hardware is on different platforms and the platforms do not implement their VMs the same way the configurations might differ slightly depending on the cloud platform. The configurations of each VM are the same on their respective cloud platform, meaning all VMs on AWS are identical except for the hardware, the same goes for the other cloud platforms. Configurations of the VMs for each cloud platform are listed in table 3.2. Separate regions were used for the gathering of data for the performance and sustainability metrics because the carbon intensity of some regions was not available and only noticed after gathering the performance metrics.

Instance Type	Cloud Platform	CPU	CPU Family	Frequency	CPU Count	Memory	Cost (hourly)
t2.medium	AWS	Intel Xeon CPU E5-2676 v3	Intel Haswell or Broadwell	2.4 GHz	2	4 GiB	$0.0464
t3.medium	AWS	Intel Xeon Platinum 8259CL	Intel Skylake	2.5 GHz	2	4 GiB	$0.0472
m5.large	AWS	Intel Xeon Platinum 8259CL	Intel Skylake or Cascade Lake	2.5 GHz	2	8 GiB	$0.1120
c5.large	AWS	Intel Xeon Platinum 8124M	Intel Xeon 8000 series	3.0 GHz	2	4 GiB	$0.1010
c5a.large	AWS	AMD EPYC 7R32	AMD EPYC 7002 series	N/A	2	4 GiB	$0.0910
D2s_v3	Azure	Intel Xeon CPU E5-2673 v4	Intel Haswell	2.4 GHz	2	8 GiB	$0.1120
D2as_v4	Azure	AMD EPYC 7763	AMD EPYC Rome and Milan	N/A	2	8 GiB	$0.1120
B2s	Azure	Intel Xeon Platinum 8370C	Intel IceLake	2.8 GHz	2	4 GiB	$0.0472
F2s_v2	Azure	Intel Xeon Platinum 8370C	Intel IceLake	2.8 GHz	2	4 GiB	$0.1010
F4s_v2	Azure	Intel Xeon Platinum 8372CL	Intel Skylake	2.6 GHz	4	8 GiB	$0.1940
n2-standard-2	GCP	Intel Xeon	Intel Ice Lake and Cascade Lake	2.8 GHz	2	8 GB	$0.09712
n2d-standard-2	GCP	AMD EPYC 7B12	AMD EPYC Rome and Milan	2.25 GHz	2	8 GB	$0.0980
n1-standard-2	GCP	Intel Xeon	Intel Haswell, Broadwell, or Skylake	2.3 GHz	2	8 GB	$0.1050
c2-standard	GCP	Intel Xeon	Intel Cascade Lake	3.1 GHz	4	16 GB	$0.2297
c2d-standard	GCP	AMD 7B13	AMD EPYC Milan	N/A	2	8 GB	$0.0908

Table 3.1: Hardware investigated in the experiment

Provider	**Regions**	**OS**	**Disk**
AWS	eu-west-2 & eu-west-3	Ubuntu 20.04 on x86 architecture	8 GiB general purpose SSD (gp3)
Azure	West Europe & UK South	Ubuntu 20.04 on x86 architecture	Standard SSD
GCP	eu-west2 & eu-west9	Ubuntu 20.04 on x86 architecture	New balanced persistent disk 10 GB

Table 3.2: Configuration of the VMs on each cloud platform

3.4.7 Software

To conduct the experiment a variety of different software is going to be used. These are listed in table 3.3 below:

Name	Type	Area of Use
Azure Virtual Machine	Cloud Service	One of the services in the comparison and benchmark.
Compute Engine (GCP)	Cloud Service	One of the services in the comparison and benchmark.
EC2 (AWS)	Cloud Service	One of the services in the comparison and benchmark.
Secure Copy Protocol (SCP)	Command	Used to upload files to the virtual machine.
Secure Shell (SSH)	Command	Used to connect to the virtual machine.
Bash	Command Prompt	Used to run scripts.
CodeCarbon	Package	Used to measure emissions and energy consumption.
Image Quality	Package	Has a trained BRISQUE model available.
Pillow	Package	Used to load image into python.
Time	Package	Used to measure the inference time and throughput.
Python	Programming Language	Used to carry out the experiments.

Table 3.3: Software used in this project

In table 3.3 all software that was used to conduct this experiment is listed. It is defined as the name of the software together with what type of software it is and what it is used for. Microsoft Azure, Compute Engine, and EC2 are the IaaS solutions for the different cloud providers and are the environment where the experiments were conducted. The Packages listed were used for setting up and conducting the experiments with the listed metrics. For the experiments, the ”Image Quality” library provided the machine learning model since it had a pre-trained implementation of BRISQUE set for testing.

Chapter 4

Implementation

This part is split up into three sections where the implementation of the experiments is described in more detail. In the first section the creation of the VMs is going to be described, one section for each cloud provider since there are some differences in setup between them. Then a description of the start-up script that was used is given. Lastly, the implementation of the experiments is going to be detailed as to how the metrics were acquired.

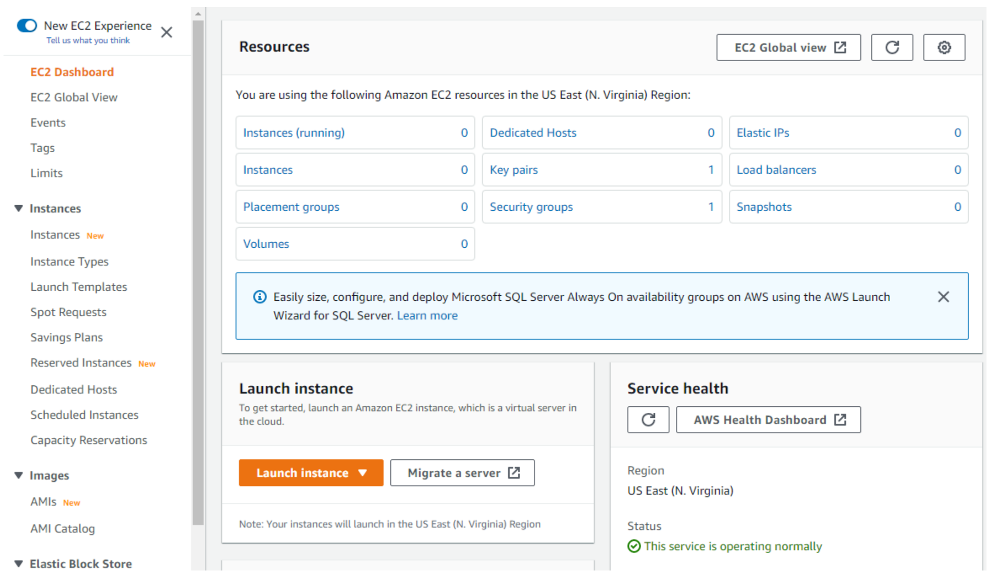
4.1 Creation of VMs on the different cloud platforms

The creation of the VMs all start similarly with first logging into the account of the respective cloud provider and navigating to the service mentioned in table 3.3. These are the services that host the VMs and it is where they are launched.

4.1.1 Amazon Web Services

Once finished navigating to the EC2 console, the user starts with the creation of a security group and a key pair in the region where the experiments are being conducted. These are used to securely connect to the VM, they are optional to create since one could connect to the VM without them, however at reduced security. The thesis will not go through how these are created, AWS has good tutorials on how to both create the security group[44] and the key pair[45].

Further, upon entering the console of the EC2 the user is prompted with a dashboard with information about the resources the user has allocated. This page is shown in figure 4.1. There are two ways of launching a VM from here, first is to click the ”Launch Instance” button which will forward the user to the page where the VMs are created.



New EC2 Experience
Tell us what you think

EC2 Dashboard
EC2 Global View
Events
Tags
Limits
▼ Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Scheduled Instances
Capacity Reservations
▼ Images
AMIs New
AMI Catalog
▼ Elastic Block Store

Resources
EC2 Global view

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	0	Dedicated Hosts	0	Elastic IPs	0
Instances	0	Key pairs	1	Load balancers	0
Placement groups	0	Security groups	1	Snapshots	0
Volumes	0				

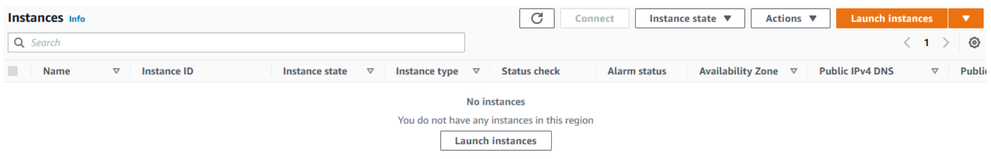
Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more

Launch instance
To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.
Launch instance ▼ Migrate a server
Note: Your instances will launch in the US East (N. Virginia) Region

Service health
AWS Health Dashboard
Region
US East (N. Virginia)
Status
This service is operating normally

Figure 4.1: EC2 console page

The other way is to click on the button ”Instances” on the left toolbar under the category ”Instances”. When the button has been clicked the user will be forwarded to a page where all the instances are listed that have been created on that account, this overview page can be seen in figure 4.2. In the figure, an orange ”Launch instance” button can be seen in the top right corner, when pressed the user is taken to the VM creation page.



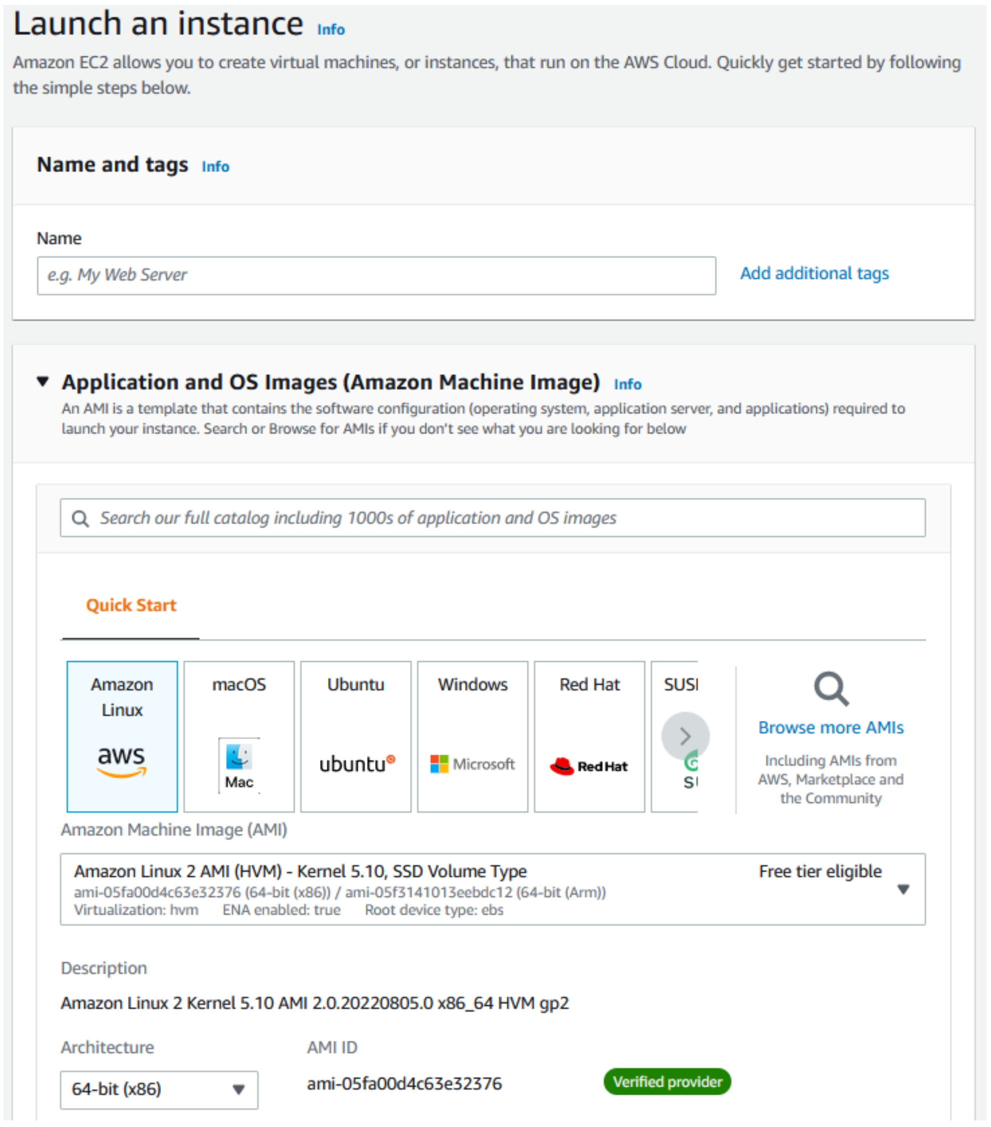
Instances Info
Connect | Instance state ▼ | Actions ▼ | Launch instances ▼
Search
1

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Publi

No instances
You do not have any instances in this region
Launch instances

Figure 4.2: EC2 Overview page

On the creation page, there are several things to consider before creating the VM. The first part of the page is shown in figure 4.3, where the first two steps of the creation process is displayed, naming of the VM and choosing the machine image of the instance. Here as stated in the configuration table in section 3.4.6 the Ubuntu 20.04 LTS version was chosen. The next step is to choose the instance type, the types used in this thesis are listed in table 3.1. Lastly, the user can choose to select their own key pair and security group for securely connecting to the instance. When these options are selected and the ”Launch Instance” button is pressed a VM is created.



Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

e.g. My Web Server

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type — Free tier eligible
ami-05fa00d4c63e32376 (64-bit (x86)) / ami-05f3141013eebdc12 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

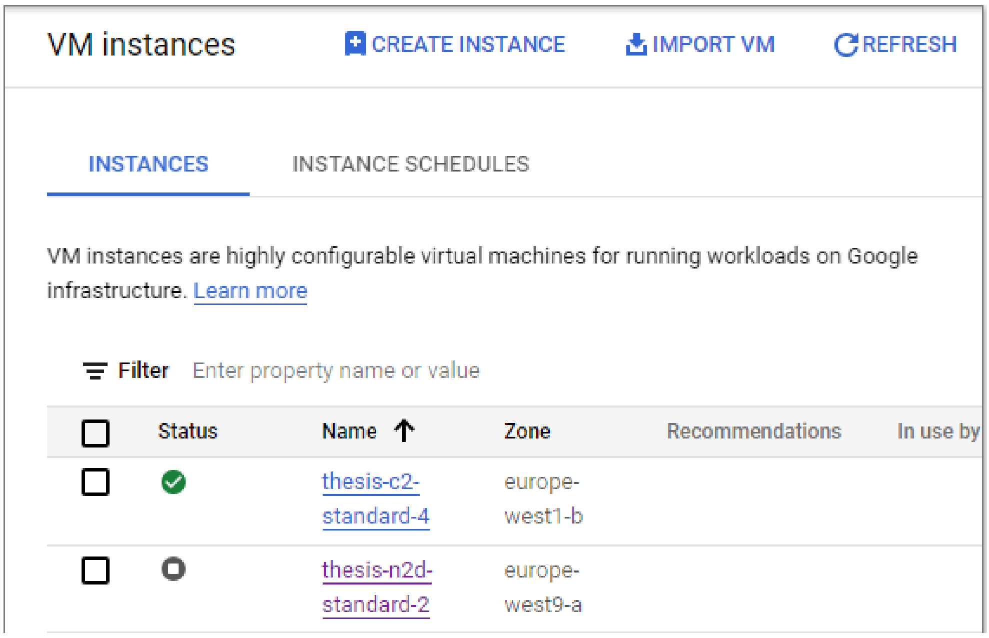
Amazon Linux 2 Kernel 5.10 AMI 2.0.20220805.0 x86_64 HVM gp2

Architecture	AMI ID
64-bit (x86)	ami-05fa00d4c63e32376 Verified provider

Figure 4.3: EC2 creation page

4.1.2 Google Cloud Platform

In GCP to access the VMs the user first needs to enable compute engine, which is done by navigating the compute engine page in their console. Once that is enabled the user has access to create and manage their VMs. To create a VM the user either presses the ”Create a VM” button or once again navigates to the compute engine console page. This page displays an overview of all active instances the user has at that moment. To create a new VM the user can press the ”CREATE INSTANCE” button on the toolbar at the top as shown in figure 4.4.



VM instances are highly configurable virtual machines for running workloads on Google infrastructure. Learn more

Filter Enter property name or value

	Status	Name ↑	Zone	Recommendations	In use by
☐	✓	thesis-c2-standard-4	europe-west1-b		
☐	●	thesis-n2d-standard-2	europe-west9-a		

Figure 4.4: Compute Engine Dashboard

The creation of the VM consists of several steps, not all the categories shown on the creation page were changed. Below is a list with all the categories that were altered during the creation, if it is not listed there the default values were used:

Step 1: Name First, the user is prompted with naming the instance, the name is what is displayed on the overview page.

Step 2: Region Next step is to choose which region the VM instance is going to be hosted on. The

	regions used in this thesis are listed in table 3.2.
Step 3: Machine Configuration	This step refers to choosing the instance configuration, first from which category of instances to choose from, then which machine type. The machine types used for GCP in this thesis are listed in table 3.1.
Step 4: Boot Disk	In this step the user chooses the type of storage the instance should contain. In this thesis, Ubuntu 20.04 was used as the OS and the configuration is listed in table 3.2.
Step 5: Create Instance	The last step is to create the instance when the above steps are done. The user can press a blue ”CREATE” button on the bottom of the creation page to create the VM instance.

4.1.3 Microsoft Azure

To create a VM instance on Microsoft Azure the user must first navigate to the ”Virtual Machines” page which can be done by either pressing the ”Virtual machines” button at the start page or searching for it in the toolbar at the top. Once at the dashboard for the virtual machines, the user has the option to press the button ”Create” which forwards to the creation page. On the creation page, the user will be prompted to select several options for the virtual machine, like basic information about the VM but also information about networking and disks. Below is a description of which options were changed in this thesis as well as references to which values were chosen, everything else was left as the default values. The process is similar to both AWS and GCP.

Step 1: Name	First, the user is prompted with naming the instance, the name is what is displayed on the overview page.
Step 2: Region	Next step is to choose which region the VM instance is going to be hosted on. The regions used in this thesis are listed in table 3.2.

Step 3: Image This step refers to choosing OS of the instance type. The OS used for Azure in this thesis was Ubuntu 20.04 which is shown in table3.2.

Step 4: VM Architecture In this step the user chooses which kind of architecture the VM should run on. The architectures used in this thesis were always x64 and can be seen in table 3.2.

Step 5: Instance type This step refers to choosing the instance type. The types used for Azure in this thesis are listed in table 3.1.

Step 6: Boot Disk In this step the user chooses the type of storage the instance should include. The storage configuration for Azure is shown in table 3.2.

Step 7: Create Instance The last step is to create the instance when the above steps are done. The user can press a blue ”Review + create” button at the bottom of the creation page and then the blue ”Create” button at the bottom corner again.

4.2 Startup Process

In this section, the process of setting up the experimental environment on the created VMs will be described. The first step in the process was to upload all the necessary files needed to conduct the experiment. Among these were the python program with the experiments, a dummy image used to conduct the experiments, the machine learning model, a text file stating all the required python packages, and a bash script that automates the startup process of the VM. These files were uploaded with a command named Secure Copy Protocol(SCP). The command was used together with the ”-i” option to specify the private SSH key that was used to connect and upload files to the VMs. Since there were multiple files that needed to be uploaded, a shell script was used to automate the process. Listing 4.1 shows the shell script used to upload all files, where ”X” is the IP address to the VM, ”Y” is the location where the private SSH key is stored, and ”U” is the name of the user decided at the cloud provider.

Listing 4.1: Uploading files to virtual machine

```
#!/bin/bash
echo "UPLOADING FILES"
VM_IP= X
KEY_LOCATION= Y
USER= U
scp -i $KEY_LOCATION experiments.py $USER@$VM_IP:
scp -i $KEY_LOCATION brisque_svm.txt $USER@$VM_IP:
scp -i $KEY_LOCATION normalize.pickle $USER@$VM_IP:
scp -i $KEY_LOCATION dummy_image.jpg $USER@$VM_IP:
scp -i $KEY_LOCATION requirements.txt $USER@$VM_IP:
scp -i $KEY_LOCATION startUp.sh $USER@$VM_IP:
echo "SUCCESS"
```

The next step is to install the required packages and tools used to conduct the experiments. Since there are multiple packages needed this step has been automated with a Bash shell script as well. This script is shown in Listing 4.2. First, a reinstall of the build-essential packages was done to ensure that all meta packages were up to date. Then the latest version of all packages was installed with ”sudo apt-get update” and ”sudo apt -y upgrade”. After all the packages are updated ”pip” is installed, and then the python packages are used for the experiment. The python packages needed are listed in table 3.3. Lastly, the program that conducts the experiment is started.

Listing 4.2: Installation of required packages and tools

```
#!/bin/bash
echo "Startup"
sudo apt-get install --reinstall build-essential -y
sudo apt-get update
sudo apt -y upgrade
sudo apt install -y python3-pip
pip install -r requirements.txt
pip install -e git+https://github.com/codecarbon.git@master#egg=c
python3 experiments.py
```

4.3 Experiments

To conduct the experiment a python program was written that collected all the metrics used in this thesis. For the inference time, a for loop was constructed that ran for a predefined number of times. In the loop, a timer was started with the inbuilt python module ”time”. When the timer had started a call to the model was made where the dummy image was passed as the argument. Once the model had been executed the timer was stopped and an elapsed time for the inference was calculated and stored in a list. This loop ran for 500 iterations. To calculate the emission and energy consumption the module CodeCarbon was used. An object called an emission tracker was created from the CodeCarbon package which had a function to start tracking emissions and a function to stop tracking emissions. Before starting the for loop that calculated the inference time the tracker was started and it was then stopped when the for loop had finished. An example of such a program is shown in listing 4.3.

Listing 4.3: The python program that executed the experiment described in this thesis

```
ITERATIONS = 500
tracker = EmissionTracker()
list = []

tracker.start()
for i to ITERATIONS
    startTime = time.start()
    model(dummyImage)
    stopTime = time.stop()
    elapsedTime = stopTime - startTime
    list.append(elapsedTime)
tracker.stop()
```

There are several ways of calculating the emissions and energy consumption of a program. CodeCarbon does it with the help of various tools depending on the OS. For systems using Windows or MacOS, CodeCarbon tracks the power consumption of the CPU with the help of Intel Power Gadget. When it comes to systems running on a Linux OS it tracks the power consumption from the RAPL files. When none of these tracking modes are available to track the power consumption on the system, CodeCarbon firsts finds the Thermal Design Power (TDP) of the CPU that the system is running on. It assumes that the power consumption of a CPU is on average 50% of the TDP, they themselves have empirically tested that 50% of the TDP is a good approximation of the power consumption of a CPU. To find the TDP of a CPU CodeCarbon have listed over 2000 different CPUs as well as their TDP[34].

To calculate the emissions of the power consumed by the computational system, CodeCarbon uses the following formula:

$$E = C * P \tag{4.1}$$

Where E is the carbon dioxide emissions (CO_2eq), C, is the carbon intensity, and, P, is the power consumed by the system. P, the power consumed is received from the description above, and, C, the carbon intensity is acquired from either the global carbon intensity of electricity for each cloud provider or for the country where the computational resource is located. If none of those are available for that resource but the energy mix for that country is available they use a specific table that contains the carbon intensity from different energy sources. Among these energy sources are coal, petroleum, natural gas, geothermal, hydroelectricity, nuclear, solar, and wind. The effective cost was calculated with the help of the hourly cost and how many items the machine learning model could process per hour. The formula is shown below:

$$\text{Effective Cost} = \text{Cost hourly}/\text{Items per hour} \tag{4.2}$$

Chapter 5

Results

In this chapter, the results of the experiments are presented. In the first section, the results of the experiments regarding performance and cost are presented in three tables, one for each cloud provider. In the second section, the results of the experiments regarding sustainability are presented in one table and two graphs.

5.1 Performance

There are three tables in this section, one for each of the different cloud providers. They all show the performance results of the instance types tested on that provider. Highlighted values indicate the best-performing instance type for that metric. The result consists of inference time and cost, both hourly and effective.

In table 5.1 the performance results of AWS instances are shown. The instance type ”c5.large” had the best inference time, whilst ”t2.medium” was the cheapest in terms of hourly cost and ”t3.medium” was the most cost-effective instance type on AWS.

Instance Type	**Inference Time**	**Cost** Hourly	Effective
t2.medium	3.96s	$**0.0464**	0.000050 $USD/item
t3.medium	3.87s	$0.0472	**0.000049** $USD/item
m5.large	3.45s	$0.1120	0.000110 $USD/item
c5.large	**3.33**s	$0.1010	0.000094 $USD/item
c5a.large	3.38s	$0.0910	0.000086 $USD/item

Table 5.1: Result for AWS, shows inference time and cost for the different instance types when running BRISQUE.

For Azure, the results are shown in table 5.2. There it can be seen that the best-performing instance type was the ”D2as_v4” in terms of inference time and effective cost. However, ”B2s” had the cheapest hourly cost.

Instance Type	**Inference Time**	**Cost** Hourly	Effective
B2s	3.61s	$**0.0472**	0.0000756 $USD/item
D2s_v3	4.05s	$0.1120	0.000126 $USD/item
D2as_v4	**2.41**s	$0.1120	**0.000075** $USD/item
F2s_v2	3.70s	$0.1010	0.000104 $USD/item
F4s_v2	3.44s	$0.1940	0.000185 $USD/item

Table 5.2: Result for Azure, shows inference time and cost for the different instance types when running BRISQUE.

On GCP, ”n2d-standard-2” performed the best on inference time and effective cost. Although, c2d-standard was the cheapest on hourly cost. This can be concluded from table 5.3.

Instance Type	**Inference Time**	**Cost** Hourly	Effective
n1-standard-2	4.38s	$0.1050	0.000126 $USD/item
n2-standard-2	3.48s	$0.09712	0.000093 $USD/item
n2d-standard-2	**2.23**s	$0.0980	**0.000060** $USD/item
c2-standard	2.98s	$0.2297	0.000190 $USD/item
c2d-standard	5.01s	$**0.0908**	0.000126 $USD/item

Table 5.3: Result for GCP, shows inference time and cost for the different instance types when running BRISQUE.

5.2 Sustainability

The results regarding sustainability are presented in this section with one table that involves the metrics gathered regarding sustainability. Table 5.4 shows the metrics gathered for the four instance types that could measure energy consumption. These instance types were, ”t3.medium” and ”m5.large” on AWS, ”D2as_v4” on Azure, and ”n2d-standard-2” on GCP. The sustainability metrics gathered were the amount of carbon dioxide equivalent in kg, emissions, and energy consumption represented by kWh. The power in watts was included because it was the basis to measure these metrics and therefore thought to be relevant to include in the same table. As in previous tables, highlighted values indicate the best value for that metric.

Instance Type	Provider	Power (W)		Emissions (kgCO2eq)	Energy Consumed (kWh)		
		CPU(TDP)	RAM		CPU	RAM	TOTAL
t3.medium	AWS	120	1.41574	0.00601	0.05647	0.00076	0.05723
m5.large	AWS	120	2.83445	0.00557	0.05167	0.00139	0.05306
D2as_v4	Azure	280	2.91186	0.03041	0.04782	0.00099	0.04881
n2d-standard-2	GCP	240	2.91019	0.00632	0.05224	0.00127	0.05350

Table 5.4: Sustainability results for the instance types that were eligible for measuring the energy consumption

Two graphs were generated for the two best-performing instance types. It shows a comparison between the amount of CO_{2eq} in kg the model would have released if run in a different region on the same instance type for that cloud provider. Figure 5.1 shows the graph described above for GCP’s instance type ”n2d-standard-2”. Figure 5.2 shows the graph described above for Azure’s instance type ”D2as_v4”.

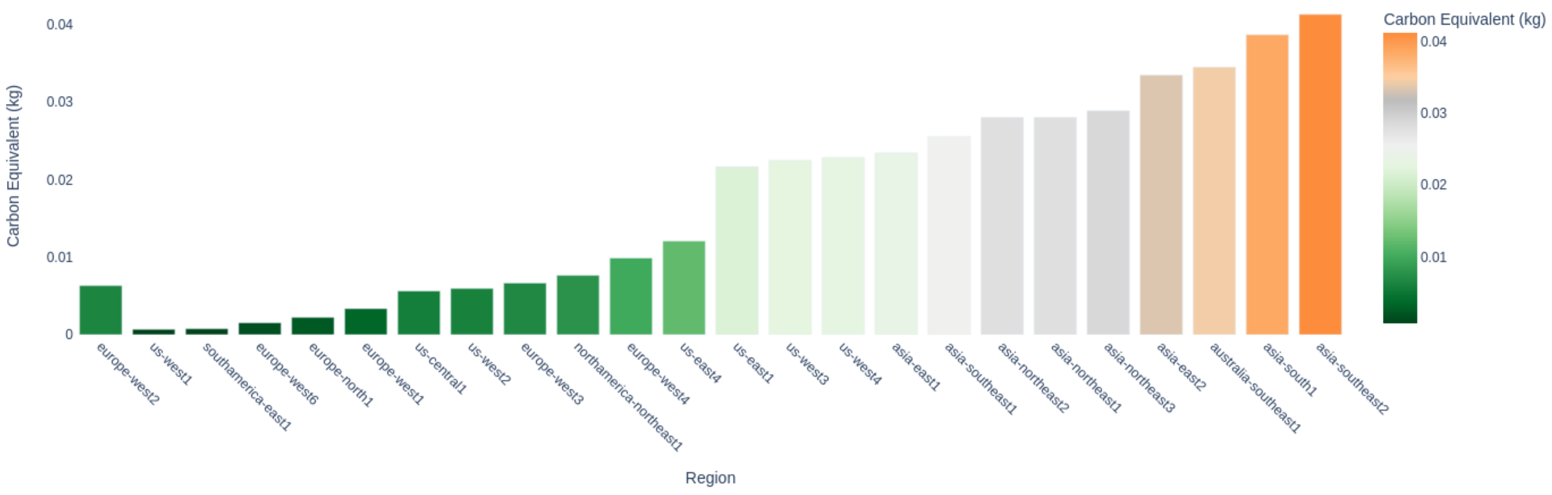


Figure 5.1: Carbon equivalent released per region when running BRISQUE on GCP's n2d-standard-2

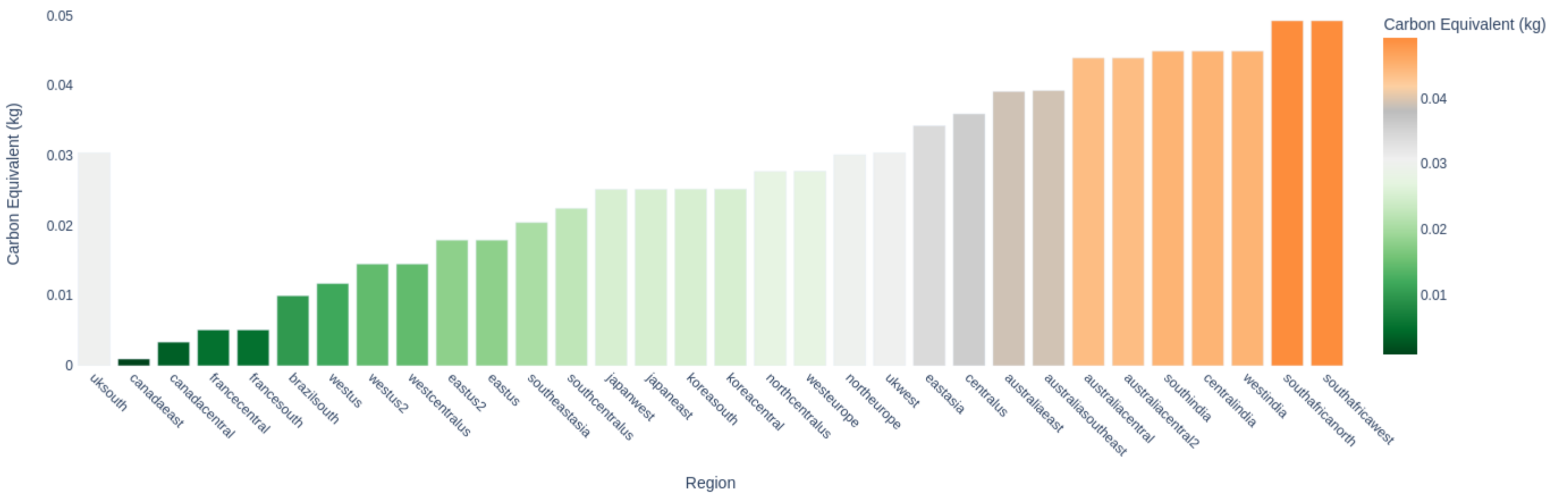


Figure 5.2: Carbon equivalent released per region when running BRISQUE on Azure's D2as_v4

5.3 Reliability Analysis

Both the sustainability and performance metrics gathered in this thesis were done multiple times and then averaged. The processing of one item through the model was done 500 times to collect the metrics. Inference time, emissions, and energy consumption are all an average of 500 measured runs. The effective cost, however, is only an average of five runs since to collect the metric the model was run for an hour and due to time constraints only five runs were feasible.

Chapter 6

Discussion

In this chapter, a thorough examination of the data collected from the conducted experiments is presented. This analysis includes a comprehensive comparison of the performance and sustainability of the different cloud providers and their respective instance types. Additionally, a discourse on the suitability of sustainability metrics as a benchmark in cloud environments is presented. Furthermore, the chapter also includes recommendations for improvements to the experimental design, which were not previously considered prior to the implementation of the experiments.

This section discusses the results in terms of the research questions that the thesis sought to answer,

- **RQ1:** How does the inference time of running BRISQUE compare among different cloud providers and cloud hardware?
- **RQ2:** How do cost, energy consumption, and carbon emissions released by running BRISQUE compare among different cloud providers and cloud hardware?
- **RQ3:** Evaluate the inclusion of sustainability metrics, such as energy consumption and carbon emission, in a benchmark conducted in a cloud environment.

The initial portion, as identified by section 6.1.1 and 6.1.2, conducts an examination of the findings pertaining to research questions one and two. Additionally, section 6.1.3 presents an analysis of the results in relation to research question three.

6.1 Analysis

6.1.1 Comparison of each cloud provider

AWS

As can be seen in table 5.1 ”c5.large” had the fastest inference time closely followed by “c5a.large”. The expectation was that the compute-optimized instances would be faster than the instances in the general purpose category, which seemed to be the case for AWS since both the mentioned instance types are from the compute-optimized category. However, all instance types performed similarly with only some differences in inference time.

Regarding the sustainability results for AWS, “t2.medium” had the lowest hourly cost which is also the instance type with the least powerful CPU. On effective cost, “t2.medium” and “t3.medium” performed similarly with the latter performing slightly better. This is probably because they were more than half the hourly cost of the other instance types but performed similarly in terms of inference time, only 15% slower than the rest.

Another surprising revelation from the result was that both ”t3.medium” and ”m5.large” are using equivalent CPU but their inference time differs by around 10%. An explanation for this would be that the other hardware components of ”m5.large” specifically the RAM is faster. In table 3.1 in section 3.4.6 we can see that the RAM is bigger and in table 5.4 we can see that it consumes more energy, which could indicate that the RAM in ”m5.large” is more powerful than the one in ”t3.medium”. Why the RAM affects the inference time in this way could be because of the calculations inside the model, many reads from the RAM might be needed since we are working with images and doing calculations with the help of the image’s pixels, both during the first and second phases of the model.

Regarding the energy consumption, only two of the five instance types used in this experiment had information about their thermal design power available and could therefore measure the energy consumption. The two instance types were ”t3.medium” and ”m5.large”, the results can be seen in table 5.4. Strangely, the table also shows that ”t3.medium” had more total energy consumed than ”m5.large”, however, they have very similar hardware except for the RAM which was more powerful in the ”m5.large”, thus the thought was that the total energy consumed was going to be higher for ”m5.large” than for ”t3.medium”. One explanation is that the ”m5.large” runs faster as it has a lower average inference time and therefore takes less time to execute the experiment resulting in lower energy consumed.

Azure

For Azure, ”D2as_v4” performed the best in terms of inference time, with 2.4093s. The other instance types were over one second slower than the ”D2as_v4”. Further, the hourly cost of the instance was in line with the others except for the ”B2s” which was less than half the price of the other instances. In addition, it was the most cost-effective as well with only ”B2” being close. This did not come as a big surprise though, since it was the fastest and had a relatively low hourly cost.

However, why it performed much better than the others may have been because of the underlying architecture. “D2as_v4” was the only instance type running on AMD hardware, and could be the reason for it performing better than the other instance types. It is hard to know why their AMD processors performs better since they do not leave out information about the processors and most of them are custom-made for Azure’s cloud computing.

Both the ”B2s” and ”F2s_v2” use the Intel Xeon Platinum 8370C, nonetheless, the former cost half as much. The performance of the two instances is of the same quality. The inference time does not differ more than 3%, and both use two CPUs and the same amount of RAM. The reason for this is that the B-series of Azure introduces a cost-effective way to manage and execute code on those instances. They use a feature that runs the CPU with low performance and then spikes when a workload is executed on the instance, this lets the instance maintain a lower cost. For our experiments that feature proved beneficial to reduce the overall cost.

The only instance type that was eligible to measure the sustainability metrics, energy consumption and emissions, was ”D2as_v4”. Figure 5.2 shows that the amount of emissions produced could have been reduced if another region were chosen. An example is the Canada east region which would have reduced the emissions by a little less than 0.03 kg of carbon equivalent. However, the choice of the region includes other factors as well, such as latency or security, but the graph gives a good understanding of how important the choice of the region is when configuring an instance.

GCP

As presented in Table 5.3, Google Cloud Platform (GCP) exhibited a range of performance in terms of inference time among its various instance types. Specifically, the ”n2d-standard-2” instance type had the fastest average inference time at 2.23 seconds per item, while the ”c2-standard” instance type had the slowest average inference time at 2.98 seconds per item, despite having a significantly higher hourly cost. Interestingly, the ”c2d-standard” instance type had the lowest hourly cost of the three but also had the highest inference time. This result is surprising, as the ”c2d-standard” is one of GCP’s compute-optimized instance types and has similar hardware to the ”n2d-standard-2” instance type. One possible explanation for this discrepancy could be that the CPU of the ”c2d-standard” instance type is not well-suited for the calculations required by the machine learning model, leading to the slow inference time. Alternatively, there may have been an issue with the experimental set-up during the run of this instance type, which could have resulted in the high inference time. Further experimentation would be necessary to distinguish between these two possibilities. Additionally, GCP provides less information about their Intel processors than other cloud providers, making it difficult to conduct in-depth comparisons. The only information provided is the family type of the hardware, but not the specific CPU.

From a sustainability perspective, the energy consumption of only the ”n2d-standard-2” instance type was eligible to be measured. Table 5.4 illustrates that the ”n2d-standard-2” performed similarly to the other instance types in terms of emissions and total energy consumed. This outcome is somewhat surprising, as the ”n2d-standard-2” had superior performance compared to the other instance types, leading to the expectation that it would have a greater impact on sustainability factors. However, this was not the case. Furthermore, Figure 5.1 illustrates that there are more sustainable options available when considering the choice of region. Specifically, the us-west1 region had the lowest emissions in terms of kg carbon equivalent, while the closest region in Europe, europe-west6, had only slightly higher emissions. However, over time, these marginal differences in emissions can accumulate if the model is used for an extended period. Thus, it is evident that the choice of the region plays an important role in determining the sustainability of an instance.

6.1.2 Comparison Between the Different Cloud Providers

6.1.2.1 Performance

Overall, it was found that the instance type ”n2d-standard2” from GCP exhibited the most efficient inference time, with a time of 2.23 seconds. This was followed by Azure’s ”D2as_v4” with an inference time of 2.41 seconds. The only other instance type that demonstrated an inference time under three seconds was GCP's ”c2-standard” with a time of 2.98 seconds. Notably, the instance types that utilized an AMD processor performed better in terms of inference time. While a comprehensive explanation for this phenomenon is difficult to provide without further information regarding the underlying architecture, similar results have been observed in other benchmarks, such as an Online Transaction Processing (OLTP) database benchmark[46]. Furthermore, the SPEC CPU benchmark does include the processor used in Azure's ”D2s_v4” instance type, and it has been observed to perform better than most Intel counterparts. However, none of the Intel processors utilized by the instance types in this study are included in the benchmark[47]. One potential explanation for the superior performance of AMD processors in the inference time metric could be their larger number of cores compared to their Intel counterparts. For example, the processors in the third generation AMD EPYC 7003 series scale from 8 to 64 cores, whereas their Intel counterparts, such as the Ice Lake, Sky Lake, and Cascade Lake processors, top out at 40 cores. However, it should be noted that Intel processors often have a higher CPU clock speed to compensate for their lower number of cores. Given that a significant portion of the inference time is spent on pre-processing, where multiple independent features must be calculated, it is possible that the parallel processing capabilities of processors with more cores may lead to a reduction in inference time, as opposed to processing each feature separately but at a faster rate.

6.1.2.2 Sustainability

From an economic standpoint, it was determined that the instance types offered by Azure and Amazon Web Services (AWS) had the lowest hourly cost. Specifically, the ”t2.medium” instance type had the lowest cost at $0.0464 per hour, followed closely by the ”t3.medium” and ”B2s” instances at $0.0472 per hour. These instance types were also deemed to be the most cost-effective. However, it is important to note that when considering other factors such as inference time, the ”n2d-standard-2” instance type may be a more suitable choice due to its combination of low cost and high performance. On the other hand, the ”c2-standard” instance type had the highest hourly cost at $0.2297 and was deemed to be the least cost-effective due to its relatively slow performance for its cost.

Obtaining sustainability metrics proved to be challenging, as information about TDP was only provided for four instance types. The TDP values varied between the different instance types, as can be seen in table 5.4. The two

AWS instances had a TDP of 120W, while the ”D2as_v4” and ”n2d-standard-2” instances from Azure had TDP values of 280W and 240W, respectively. Initially, it was assumed that the instance types with higher TDP values would have higher total energy consumption. However, the data revealed that all instance types consumed approximately the same amount of energy during the tests, with the instance type with the highest TDP having the lowest energy consumption. This discrepancy may be explained by the fact that the faster-performing instances consumed less energy over the course of the experiments.

The results from the sustainability experiments, as presented in graph 5.2 and 5.1, indicate that the region in which the experiments were conducted can have a significant impact on the emissions of instance types. Specifically, the ”D2as_v4” instance type had the highest emissions despite consuming less energy than other instance types. This suggests that the region used for the experiments, in this case, the UK south region for Azure, was not optimal. Figure 5.2 illustrates this, showing that if a different region, such as France central, had been used instead, emissions could have been reduced by approximately 0.0025 kg of carbon equivalent.

6.1.3 Sustainability Metrics in Cloud Benchmarks

As previously discussed in related literature, the emphasis on measuring energy consumption and emissions of machine learning models is becoming increasingly important. However, the methods tested in this thesis for measuring energy consumption on cloud instances have proven to be unsuccessful. The main hurdle in measuring energy consumption on cloud instances is the lack of information about the system provided by providers on the tools necessary for doing so.

For Linux systems, a common method is to examine the data provided in RAPL files, however, this information is not available on cloud instances running on a Linux system. Another method is to use the TDP to approximate energy consumption, however, only a small number of the tested instance types provided this information, likely due to the use of custom-made CPUs for the provider.

Other methods for measuring energy consumption, such as using Intel Power Gadget on Windows or MacOS systems, were not explored in this thesis as the instances used in the study were running on a Linux operating system. Additionally, using carbon calculators provided by cloud providers, such as AWS, Azure, and GCP, was considered, but ultimately deemed impractical due to limitations in data collection and the need for several months to gather the necessary information.

In conclusion, unless cloud providers begin to provide more information on tools and methods for measuring energy consumption, it will be difficult to include such metrics in machine learning benchmarks for cloud computing.

6.2 Suggestions for Improvement

There are three main improvements that would benefit this thesis. The first improvement would be to run the data used for the cost-effective metrics more times. It was only run five times due to time and cost limitations in the cloud. This would make the comparison and the data gathered more robust. Additionally include more instance types in the comparison. There are several more instance types that could be investigated and included in the benchmark, only five instance types for each cloud provider were included in this thesis because of time and cost limitations. For all providers, there are multiple instance types available for testing, both in the general purpose category and in other categories not explored in this thesis.

Other improvements that would make the comparison more robust are to either add or change some of the instance types. An example is that the experiment for Azure and GCP uses two instance types that should use equally powerful hardware but from different CPU manufacturers, one from Intel and one from AMD. They are also the newest general-purpose instance types for Azure and GCP. AWS does not include its latest instance type, it does include its second latest, the ”m5.large” but not its counterpart ”m5a.large”. Having that said GCP does not include a low-cost instance unlike AWS and Azure which include their low-cost instance types, t2 and t3 series for AWS and ”B2s” for Azure. Adding these changes would make a more robust and complete comparison. With time, all instance types would be needed to compare how the BRISQUE model performs on different clouds, but this gives an initial understanding and basis for academia and professionals.

The last improvement mentioned here is to include median values for the inference time. Tests made after the experiments were already finished showed that there was a slight difference between the median and mean values of the inference time. These tests were made on a system not in the cloud, thus it is not certain that the same observations would apply in that case. Adding these values would not impose a challenge, however, as most values during the calculations of the mean were saved.

Chapter 7

Conclusions and Future Work

The final chapter of this thesis presents a comprehensive conclusion which integrates the results obtained throughout the study with the original problem statement and research questions. Additionally, an in-depth examination of potential future research endeavors and a critical examination of the relevant economic, social, environmental, and ethical implications of the findings are presented.

7.1 Conclusions

In this study, a benchmark was created to compare how an image quality assessment model performed in regard to both performance and sustainability on cloud infrastructure. The goal of the study was to analyze if there was a performance difference between the different cloud infrastructures and their hardware and evaluate the inclusion of sustainability metrics, specifically energy consumption and carbon emission, in a benchmark. To answer the first research question, how does the inference time compares among the different instance types and cloud providers, all providers had similar performance results where GCP had the best-performing instance type followed by Azure. From the results, it could be derived that the instance types with an AMD processor performed better than those that ran on Intel hardware. Surprisingly, the hardware with higher CPU clock speed did not perform as well as initial assumptions, instead, instance types with a higher number of cores seemed to have an advantage. Regarding the categorizations made by the cloud providers only on AWS using an instance type of the compute-optimized category yielded an increase in performance for the other providers they yielded a decrease in performance instead. However, due to time limitations, all instance types were not tested and there might have been other instance types that performed differently.

Further analysis revealed that, in terms of energy consumption, all instance types consumed approximately equal amounts. The instance type ”D2as_v4” demonstrated the lowest total energy consumption among the analyzed instances. However, it did not exhibit the lowest amount of emissions released, as that distinction was held by the instance type ”m5.large”, with ”n2d-standard-2” closely following. The results of this investigation underline the importance of considering one key factor when selecting an instance type with respect to sustainability: the region in which the instance is hosted. This factor was found to have the largest impact on emissions released. Contrary to expectations, the TDP of the hardware was not directly correlated with higher energy consumption. One potential explanation for this phenomenon is that the instance types with higher TDPs may run faster, compensating for the energy consumed by completing execution in a shorter amount of time. However, it must be noted that these conclusions were drawn only from a limited subset of instances, as the cloud services lacked the information and features needed for a more comprehensive analysis. Therefore, additional research is required to arrive at a more robust and conclusive understanding of the relationship between instance type and energy consumption.

This leads to the final research question of this thesis, which aims to assess the feasibility of incorporating sustainability metrics, such as energy consumption and emissions, into cloud benchmarks. The process of benchmarking clouds can be considered a ”grey box” as some data may be inaccessible due to a shortage of information provided by cloud providers. This presents challenges for professionals and academia in including sustainability metrics in evaluations and benchmarking frameworks for cloud services. Given these limitations, it is currently difficult to include these metrics in cloud benchmarks due to the unavailability of the necessary features and information. Despite this, there are emerging alternatives that offer promise, but further development is required before they can be integrated into benchmarking frameworks.

7.2 Limitations

Due to cost and time limitations, not all instance types could be included in the experiments leaving more work to be done to make a more comprehensive comparison. There are roughly over 100 different instance types if including all three cloud providers making it unfeasible to test all of them during the time span of this thesis. Some of these instance types were neither feasible to test due to their hourly cost. Another limitation was that training the image quality assessment model from scratch would have been too time-consuming for this thesis which would have reduced the number of instances types evaluated and tested in this thesis which was why an already pre-trained model was chosen. There was also a desire to make an end-to-end benchmark and evaluate and compare the training of the machine learning model as well, however, due to time limitations this was not feasible since the training of a machine learning model takes a lot of time and consequently, the thesis focused on the inference instead.

7.3 Future Work

The most obvious work to be done is to continue testing instance types so that future professionals have a larger set of instance types to compare with. This would make for a larger and better benchmark that would be more comprehensive. The thesis only focused on instance types with CPUs as hardware but one could include instance types with GPUs as well. These are often used with machine learning models in general and would be a good addition to this thesis.

Another work is to include an end-to-end benchmark and train the model on the instance as well. This would open up gathering more metrics and make a more extensive evaluation of the whole flow of the machine learning model. However, this would take up much time and cost to include several different instance types. Professionals could also extend the benchmark to include more image quality assessment models to test if they would perform differently on the instance types or to see how their model performs in the cloud.

This comparison focused on the IaaS services on their respective cloud provider, but there are other services that could be investigated and evaluated such as SaaS and MLaaS. On such services, other metrics could be explored as well, since the nature of the experiment might be more online rather than offline which could include metrics related to online benchmarks, for example, tail latency and the time it takes for the service to read from memory.

During the course of this thesis, different methods of estimating the energy consumption of an instance on the cloud were explored, however, there might be better and more reliable ways of doing this which were missed in this thesis. One method that might yield more accurate results once finished is the cloud provider's own emission calculators, this method would be more accurate than the one used in this thesis. However, during the thesis, the calculators were in an early stage of development and would not include the information necessary to conduct the experiments in this thesis.

7.4 Reflections

Relevant economic and social aspects to this thesis work are to act as a base of evaluation when other professionals are considering using an image quality assessment model in a cloud environment. There is a huge amount of options when choosing an instance in the cloud and the consequences of choosing one that is not suitable for the task meant to run on it might be unnecessarily costly. This can also be viewed by both academia and professionals when trying to measure energy consumption on cloud hardware, where the work carried out in this thesis helps the reader understand what worked and what did not in this thesis. Another more concrete social aspect is a contribution to open source frameworks. To carry out this thesis the open source framework, CodeCarbon, was used. This framework contained information that made it possible to measure energy consumption and carbon emissions. However, the carbon intensity values for the GCP cloud provider were outdated. These values were updated in this thesis and all experiments were run with the new updated values for the carbon intensity. Other relevant environmental aspects of this thesis are creating awareness of the environmental impact of running machine learning models in cloud environments. This could help reduce the carbon footprint of machine learning in general and the metric has started to be included in newer benchmarks in general. Making more people recognize it and consider it when choosing an instance type for hosting their machine learning model could both help further research in this area and reduce the environmental footprint of cloud computing in general.

References

[1] "Cloud Computing, Data Science and ML Trends in 2020–2022: The battle of giants," section: 2021 Jan Tutorials, Overviews. [Online]. Available: https://www.kdnuggets.com/cloud-computing-data-science-and-ml-trends-in-2020-2022-the-battle-of-giants.html/ [Page 1.]

[2] C. Wohlin, M. Höst, and K. Henningsson, "Empirical Research Methods in Software Engineering," in *Empirical Methods and Studies in Software Engineering: Experiences from ESERNET*, ser. Lecture Notes in Computer Science, R. Conradi and A. I. Wang, Eds. Berlin, Heidelberg: Springer, 2003, pp. 7–23. ISBN 978-3-540-45143-3. [Online]. Available: https://doi.org/10.1007/978-3-540-45143-3_2 [Pages 5 and 24.]

[3] L. M. Weber, W. Saelens, R. Cannoodt, C. Soneson, A. Hapfelmeier, P. P. Gardner, A.-L. Boulesteix, Y. Saeys, and M. D. Robinson, "Essential guidelines for computational method benchmarking," *Genome Biology*, vol. 20, p. 125, Jun. 2019. doi: 10.1186/s13059-019-1738-8. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6584985/ [Pages 5 and 24.]

[4] M. Rosenblum, "The Reincarnation of Virtual Machines: Virtualization makes a comeback." *Queue*, vol. 2, no. 5, pp. 34–40, Jul. 2004. doi: 10.1145/1016998.1017000. [Online]. Available: https://doi.org/10.1145/1016998.1017000 [Page 7.]

[5] M. Byström, *Design and realisation of an automated software testing system utilizing virtual machines*, 2014. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-154480 [Page 7.]

[6] L. Qian, Z. Luo, Y. Du, and L. Guo, "Cloud Computing: An Overview," in *Cloud Computing*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz,

C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, M. G. Jaatun, G. Zhao, and C. Rong, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, vol. 5931, pp. 626–631. ISBN 978-3-642-10664-4 978-3-642-10665-1 Series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-642-10665-1_63 [Page 8.]

[7] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," p. 7, Sep. 2011. [Pages 8 and 9.]

[8] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The Characteristics of Cloud Computing," in *2010 39th International Conference on Parallel Processing Workshops*, Sep. 2010. doi: 10.1109/ICPPW.2010.45 pp. 275–279, iSSN: 2332-5690. [Page 9.]

[9] I. Bojanova and A. Samba, "Analysis of Cloud Computing Delivery Architecture Models," in *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications*, Mar. 2011. doi: 10.1109/WAINA.2011.74 pp. 453–458. [Page 9.]

[10] T. Lynn, P. Rosati, A. Lejeune, and V. Emeakaroha, "A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms," in *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec. 2017. doi: 10.1109/CloudCom.2017.15 pp. 162–169, iSSN: 2330-2186. [Page 9.]

[11] M. Ribeiro, K. Grolinger, and M. A. Capretz, "MLaaS: Machine Learning as a Service," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2015. doi: 10.1109/ICMLA.2015.152 pp. 896–902. [Pages 9 and 11.]

[12] D. Rountree and I. Castrillo, "Chapter 1 - Introduction to the Cloud," in *The Basics of Cloud Computing*, D. Rountree and I. Castrillo, Eds. Boston: Syngress, Jan. 2014, pp. 1–17. ISBN 978-0-12-405932-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780124059320000013 [Page 10.]

[13] V. Delgado, "Exploring the limits of cloud computing," p. 93, Apr. 2010. [Page 10.]

[14] Y. Yao, Z. Xiao, B. Wang, B. Viswanath, H. Zheng, and B. Y. Zhao, "Complexity vs. performance: empirical analysis of machine

learning as a service," in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC '17. New York, NY, USA: Association for Computing Machinery, Nov. 2017. doi: 10.1145/3131365.3131372. ISBN 978-1-4503-5118-8 pp. 384–397. [Online]. Available: https://doi.org/10.1145/3131365.3131372 [Page 11.]

[15] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995. doi: 10.1007/BF00994018. [Online]. Available: https://doi.org/10.1007/BF00994018 [Page 12.]

[16] D. A. Pisner and D. M. Schnyer, "Chapter 6 - Support vector machine," in *Machine Learning*, A. Mechelli and S. Vieira, Eds. Academic Press, Jan. 2020, pp. 101–121. ISBN 978-0-12-815739-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128157398000067 [Page 13.]

[17] W. S. Noble, "What is a support vector machine?" *Nature Biotechnology*, vol. 24, no. 12, pp. 1565–1567, Dec. 2006. doi: 10.1038/nbt1206-1565 Number: 12 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nbt1206-1565 [Page 14.]

[18] M. Martin, "On-Line Support Vector Machine Regression," in *Machine Learning: ECML 2002*, ser. Lecture Notes in Computer Science, T. Elomaa, H. Mannila, and H. Toivonen, Eds. Berlin, Heidelberg: Springer, 2002. doi: 10.1007/3-540-36755-$1_24.ISBN978-3-540-36755-0pp.282--294.[Page$ 14.]

[19] A. C. Bovik, *Handbook of Image and Video Processing*. Academic Press, Jul. 2010. ISBN 978-0-08-053361-2 Google-Books-ID: UM_GCfJe88sC. [Page 14.]

[20] A. Mittal, A. K. Moorthy, and A. C. Bovik, "No-Reference Image Quality Assessment in the Spatial Domain," *IEEE Transactions on Image Processing*, vol. 21, no. 12, pp. 4695–4708, Dec. 2012. doi: 10.1109/TIP.2012.2214050 Conference Name: IEEE Transactions on Image Processing. [Pages 14, 15, and 26.]

[21] S. Yang, T. Wu, S. Shi, S. Lao, Y. Gong, M. Cao, J. Wang, and Y. Yang, "MANIQA: Multi-dimension Attention Network for No-Reference Image Quality Assessment," arXiv, Tech. Rep. arXiv:2204.08958, Apr. 2022, arXiv:2204.08958 [cs, eess] type: article. [Online]. Available: http://arxiv.org/abs/2204.08958 [Page 15.]

[22] S. A. Golestaneh, S. Dadsetan, and K. M. Kitani, "No-Reference Image Quality Assessment via Transformers, Relative Ranking, and Self-Consistency," *arXiv:2108.06858 [cs, eess]*, Jan. 2022, arXiv: 2108.06858. [Online]. Available: http://arxiv.org/abs/2108.06858 [Pages 15 and 26.]

[23] D. Ghadiyaram and A. C. Bovik, "Massive Online Crowdsourced Study of Subjective and Objective Picture Quality," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 372–387, Jan. 2016. doi: 10.1109/TIP.2015.2500021 Conference Name: IEEE Transactions on Image Processing. [Pages 15 and 26.]

[24] V. Hosu, H. Lin, T. Sziranyi, and D. Saupe, "KonIQ-10k: An Ecologically Valid Database for Deep Learning of Blind Image Quality Assessment," *IEEE Transactions on Image Processing*, vol. 29, pp. 4041–4056, 2020. doi: 10.1109/TIP.2020.2967829 Conference Name: IEEE Transactions on Image Processing. [Pages 15 and 26.]

[25] Z. Ying, H. Niu, P. Gupta, D. Mahajan, D. Ghadiyaram, and A. Bovik, "From Patches to Pictures (PaQ-2-PiQ): Mapping the Perceptual Space of Picture Quality," 2020, pp. 3575–3585. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Ying_From_Patches_to_Pictures_PaQ-2-PiQ_Mapping_the_Perceptual_Space_of_CVPR_2020_paper.html [Page 15.]

[26] V. Kamble and K. M. Bhurchandi, "No-reference image quality assessment algorithms: A survey," *Optik*, vol. 126, no. 11, pp. 1090–1097, Jun. 2015. doi: 10.1016/j.ijleo.2015.02.093. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003040261500145X [Page 15.]

[27] C. Coleman, D. Narayanan, D. Kang, T. Zhao, J. Zhang, L. Nardi, P. Bailis, K. Olukotun, C. Ré, and M. Zaharia, "DAWNBench: An End-to-End Deep Learning Benchmark and Competition," *Training*, vol. 100, no. 101, p. 102, 2017. [Page 16.]

[28] W. Gao, F. Tang, L. Wang, J. Zhan, C. Lan, C. Luo, Y. Huang, C. Zheng, J. Dai, Z. Cao, D. Zheng, H. Tang, K. Zhan, B. Wang, D. Kong, T. Wu, M. Yu, C. Tan, H. Li, X. Tian, Y. Li, J. Shao, Z. Wang, X. Wang, and H. Ye, "AIBench: An Industry Standard Internet Service AI Benchmark Suite," *arXiv:1908.08998 [cs]*, Oct. 2019, arXiv: 1908.08998. [Online]. Available: http://arxiv.org/abs/1908.08998 [Page 17.]

[29] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, R. Chukka,

C. Coleman, S. Davis, P. Deng, G. Diamos, J. Duke, D. Fick, J. S. Gardner, I. Hubara, S. Idgunji, T. B. Jablin, J. Jiao, T. S. John, P. Kanwar, D. Lee, J. Liao, A. Lokhmotov, F. Massa, P. Meng, P. Micikevicius, C. Osborne, G. Pekhimenko, A. T. R. Rajan, D. Sequeira, A. Sirasao, F. Sun, H. Tang, M. Thomson, F. Wei, E. Wu, L. Xu, K. Yamada, B. Yu, G. Yuan, A. Zhong, P. Zhang, and Y. Zhou, "MLPerf Inference Benchmark," *arXiv:1911.02549 [cs, stat]*, May 2020, arXiv: 1911.02549. [Online]. Available: http://arxiv.org/abs/1911.02549 [Pages 17 and 27.]

[30] O. Alvarado Rodriguez, D. Dave, W. Liu, and B. Su, "A Study of Machine Learning Inference Benchmarks," in *2020 4th International Conference on Advances in Image Processing*, ser. ICAIP 2020. New York, NY, USA: Association for Computing Machinery, Nov. 2020. doi: 10.1145/3441250.3441277. ISBN 978-1-4503-8836-8 pp. 167–171. [Online]. Available: https://doi.org/10.1145/3441250.3441277 [Pages 18 and 27.]

[31] H. Zhang, Y. Huang, Y. Wen, J. Yin, and K. Guan, "InferBench: Understanding Deep Learning Inference Serving with an Automatic Benchmarking System," *arXiv:2011.02327 [cs]*, Jan. 2021, arXiv: 2011.02327. [Online]. Available: http://arxiv.org/abs/2011.02327 [Pages 18 and 27.]

[32] M. Madan and C. Reich, "Comparison of Benchmarks for Machine Learning Cloud Infrastructures," *CLOUD COMPUTING*, p. 7, 2021. [Page 19.]

[33] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning," vol. 21, no. 248, p. 43, Jan. 2020. doi: DOI: 10.48550/arXiv.2002.05651 arXiv:2002.05651 [cs] type: article. [Online]. Available: http://arxiv.org/abs/2002.05651 [Pages 19 and 27.]

[34] V. Schmidt, K. Goyal, A. Joshi, B. Feld, L. Conell, N. Laskaris, D. Blank, J. Wilson, S. Friedler, and S. Luccioni, "CodeCarbon: Estimate and Track Carbon Emissions from Machine Learning Computing," 2021, publisher:Zenodo. [Online]. Available: https://github.com/mlco2/codecarbon [Pages 25 and 39.]

[35] "As Quarterly Cloud Spending Jumps to Over $50B, Microsoft Looms Larger in Amazon's Rear Mirror | Synergy Research Group," Feb. 2022. [Online]. Available: https://www.srgresearch.com/articles/as-quarterly-cloud-spending-jumps-to-over-50b-microsoft-looms-larger-in-amazons-rear-mirror [Page 26.]

[36] H. Zhu, L. Li, J. Wu, W. Dong, and G. Shi, "MetaIQA: Deep Meta-learning for No-Reference Image Quality Assessment," *arXiv:2004.05508 [cs, eess]*, Apr. 2020, arXiv: 2004.05508. [Online]. Available: http://arxiv.org/abs/2004.05508 [Page 26.]

[37] T. Yuan, C. Li, L. Tian, and G. Li, "RM-IQA: A new no-reference image quality assessment framework based on range mapping method," *Computers & Electrical Engineering*, vol. 96, p. 107508, Dec. 2021. doi: 10.1016/j.compeleceng.2021.107508. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790621004559 [Page 26.]

[38] X. Liu, M. Pedersen, and J. Y. Hardeberg, "CID:IQ – A New Image Quality Database," in *Image and Signal Processing*, ser. Lecture Notes in Computer Science, A. Elmoataz, O. Lezoray, F. Nouboud, and D. Mammass, Eds. Cham: Springer International Publishing, 2014. doi: 10.1007/978-3-319-07998-$1_22.ISBN978-3-319-07998-1pp.193--202.[Page$ 26.]

[39] H. Lin, V. Hosu, and D. Saupe, "KADID-10k: A Large-scale Artificially Distorted IQA Database," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, Jun. 2019. doi: 10.1109/QoMEX.2019.8743252 pp. 1–3, iSSN: 2472-7814. [Page 26.]

[40] G. Jinjin, C. Haoming, C. Haoyu, Y. Xiaoxing, J. S. Ren, and D. Chao, "PIPAL: A Large-Scale Image Quality Assessment Dataset for Perceptual Image Restoration," in *Computer Vision – ECCV 2020*, ser. Lecture Notes in Computer Science, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-58621-$8_37.ISBN978-3-030-58621-8pp.633--651.[Page$ 26.]

[41] N. Ponomarenko, L. Jin, O. Ieremeiev, V. Lukin, K. Egiazarian, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, and C. C. Jay Kuo, "Image database TID2013: Peculiarities, results and perspectives," *Signal Processing: Image Communication*, vol. 30, pp. 57–77, Jan. 2015. doi: 10.1016/j.image.2014.10.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0923596514001490 [Page 26.]

[42] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the Carbon Emissions of Machine Learning," Nov. 2019, arXiv:1910.09700 [cs]. [Online]. Available: http://arxiv.org/abs/1910.09700 [Page 27.]

[43] K. Lottick, S. Susai, S. A. Friedler, and J. P. Wilson, "Energy Usage Reports: Environmental awareness as part of algorithmic accountability," Dec. 2019,

arXiv:1911.08354 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1911.08354 [Page 27.]

[44] "Set up to use Amazon EC2 - Amazon Elastic Compute Cloud." [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/get-set-up-for-amazon-ec2.html#create-a-base-security-group [Page 31.]

[45] "Set up to use Amazon EC2 - Amazon Elastic Compute Cloud." [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/get-set-up-for-amazon-ec2.html#create-a-key-pair [Page 31.]

[46] K. McClellan, L. Carmel, C. Custer, Y. Miretskiy, S. Rosenberg, and J. Xing, "2022 Cloud Report," 2022. [Page 51.]

[47] J. Bucek, K.-D. Lange, and J. v. Kistowski, "SPEC CPU2017: Next-Generation Compute Benchmark," in *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '18. New York, NY, USA: Association for Computing Machinery, Apr. 2018. doi: 10.1145/3185768.3185771. ISBN 978-1-4503-5629-9 pp. 41–42. [Online]. Available: https://doi.org/10.1145/3185768.3185771 [Page 51.]

TRITA-EECS-EX-2023:245