SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Seamless Synergy: Unifying Local Development and Cloud Execution in Machine Learning DevOps

Baraa Alnassan

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Seamless Synergy: Unifying Local Development and Cloud Execution in Machine Learning DevOps

Nahtlose Synergie: Vereinheitlichung von lokaler Entwicklung und Cloud-Ausführung in DevOps für maschinelles Lernen

Author:	Baraa Alnassan
Supervisor:	Jens Großklags
Advisor:	Derui Zhu (TUM), Kaio Giurizatto Utsch (IT4IPM)
Submission Date:	September 1st 2024

Acknowledgments

Abstract

Artificial Intelligence (AI) is a branch of computer science that aims to create systems capable of performing tasks that would typically require human intelligence. These tasks include learning and adapting to new information, understanding human language, recognizing patterns, solving problems, and making decisions. AI can be categorized into two main types: narrow or weak AI, which is designed to perform a specific task, and general or strong AI, which can perform any intellectual task that a human being can do. AI has various applications, such as self-driving cars, virtual personal assistants, and image recognition systems.
Machine Learning (ML) is a subcategory of AI. It is based on algorithms trained for decisions making that automatically learn and recognize patterns from data.
Machine Learning Operations (MLOps), is a practice that combines machine learning, data engineering, and DevOps practices to streamline the deployment and management of machine learning models. MLOps aims to bridge the gap between data science and DevOps, ensuring seamless collaboration and efficient deployment of ML models. It involves various components, such as model versioning, data pipeline management, model monitoring and observability, testing strategies for models, deployment orchestration, and choosing the right tools. Effective collaboration between data scientists and DevOps engineers is essential for successful MLOps practices.

This Bachelor thesis aims to highlight the difficulties of transitioning from local development environment to cloud based code execution in production environment in machine leaarning operations, and tries to find solutions to closeing the gap between local development and successful code execution on the cloud.

Contents

1 Introduction

ML is about the development of computer systems that can learn and adapt by following explicit instructions or any human interference. And it uses special algoriths and statistical models to analyze and predict the outcome from a given pattern of data. It was born to allow computers to learn and control their environment.
In today's rapidly evolving landscape of machine learning and software engineering, the integration of ML operations (MLOps) plays a crucial role. MLOps serves as the critical bridge between data science and DevOps practices, ensuring seamless collaboration and efficient deployment of ML models. As organizations increasingly rely on ML for decision-making, it becomes imperative to address the challenges associated with transitioning from local development to production environments.
To address these challenges the goal of this Thesis is to try to bridge the gap between local development and successful code execution on the cloud, in other words, how to make ML processes automated and operationalized so that more ML proof of concept can be brought into production.
To overcome those challanges, I conduct a mixed-method research endeavor to identify important principles of MLOps, carve out functional core components, highlight the roles necessary to successfully implement MLOps, and derive a general architecture for ML systems design. In combination, these insights result in a definition of MLOps, which contributes to a common understanding of the term and related concepts.
The remainder of this thesis is constructed as follows. I will first elaborate on the necessary foundations and related work in the field. Next, I will give an overview of the utilized methodology, consisting of a literature review, a tool review, and an interview study. I then present the insights derived from the application of the methodology and conceptualize these by providing a unifying definition. I conclude the paper with a short summary, limitations, and outlook.

2 Background

Before diving into the details of the main objective of the thesis, it is important to understand some background knowledge. In this chapter i will introduce some necessary detailed background of the rise of ML, and how the concenpt of MLOps came to be.

2.1 The Rise of Machine Learning

2.1.1 The Birth of Machine Learning (1950s)

The foundations of machine learning were laid in the 1950s by computer scientists and mathematicians who sought to develop algorithms that could enable computers to learn from data [Jac22; RVK24; Naq20]. During this era, several key milestones shaped the field. Such as the Perceptron Algorithm - one of the earliest breakthroughs in machine learning - by Frank Rosenblatt [Van86], was an early form of an artificial neural network, it became the building block for future advancements. Where he formulated a serie of machines, each serves to interduce a new concept. It aimed to mimic the way biological neurons process information.

2.1.2 Resurgence and New Techniques (1980s-1990s)

An AI winter started in the 1970s, that was described as a chain reaction much the same as a nuclear winter, that began with pessimisum in the AI community, followed by cutback in fundings, which in turn resulted in end of serious research. After the AI winter, machine learning was resurrected in the 1980s and 1990s. New techniques brought to the light, such as Neural Network (NN) where Researchers were inspired by the human brain's structure, according to Geoffrey Hinton [McD] , often referred to as the "father of Deep Learning,", who made significant contributions during this time.

2.1.3 The Rise of Big Data (2000s)

The 2000s witnessed the rise of big data and the availability of vast amounts of data for training machine learning models. This era witnessed the emergance of new techniques to be able to deal with vast amount of data, such as, Data-Driven Approaches [TR23; AA23], where machine learning algorithms learned patterns directly from large datasets, and Supervised Learning Dominance, where models learn from labeled examples. Techniques like regression and classification gained prominence.

2.1.4 Deep Learning and Beyond (2010s)

The 2010s marked a transformative period for machine learning, where the term Deep learning was first introduced , it it was defined as a subfield of machine learning. other powerful methods were perfected in this era such as Convolutional Neural Networks (CNN) for image recognition and recurrent neural networks for sequential data revolutionized various domains.

2.1.5 State-of-the-Art Machine Learning (Present and Future)

Nowadays machine learning has made far-reaching changes in various applications across industries. Machine learning is applied in healthcare, fraud detection, speech recognition, autonomous vehicles, recommendation systems, and more (Figure 2.1 shows some applications of machine learning). It revolutionizes industries and improves efficiency. However, researchers continue to push boundaries, aiming for AI systems that understand context, and are looking forward for the exciting possibilities ahead.

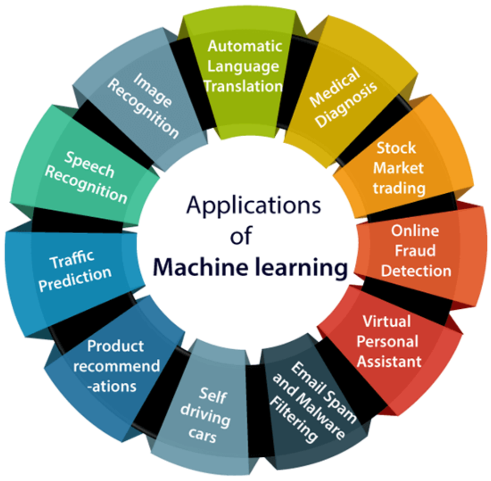
Applications of Machine learning
Automatic Language Translation
Medical Diagnosis
Stock Market trading
Online Fraud Detection
Virtual Personal Assistant
Email Spam and Malware Filtering
Self driving cars
Product recommend -ations
Traffic Prediction
Speech Recognition
Image Recognition

Figure 2.1: Few applications of machine learning.

2.2 The Evolution of MLOps

With the rise of machine learning, so did the need for efficient operations around it. This led to the evolution of MLOps MLOps is an engineering practice that aims at making the process of deploying machine learning models more efficient, reliable, and maintainable. It uses Continuous Integration and Continuous Deployment (CI/CD) and machine learning models to rationalize the monitoring, deployment and maintenance of machine learning systems.

2.2.1 Key Components of MLOps

As the datasets to train machine learning models got bigger and more complex, the realization for the need of ML life cycle grow, as the old techniques weere slow and difficult to scale. Data scientists worked together with IT teams to create and develop an assembly line for each step of training machine learning models. In this chapter I will discuss few steps of michine learning life cycle that are crucial for MLOps process.

- **Data Collection**: Gathering and refining relevant data set from different sources to prepare it for modeling is one of the key components of ML life cycle.
- **Building and training**: This is the step where models are created and trained using data, that were prepared in the previous step. It involves selecting appropriate algorithms and preprocessing the data.

And more as shown in Figure 2.2.

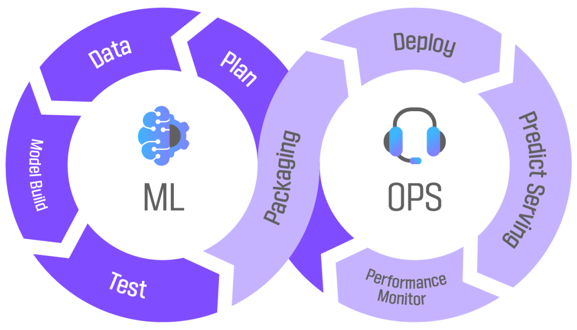

Figure 2.2: machine learning life cycle.

2.2.2 Comparing MLOps and DevOps

DevOps is a method which aims at bringing together software development and operations, it shortens the system's development life cycle and provides a continu delivery to a high software quality.
MLOps does, however, borrow from the DevOps principles of a rapid, continuous approach to writing and updating applications, in that they both have a code-validate-deploy loop. But the MLOps life cycle contains additional steps, that are necessary for building and training the machine learning model as shown in Figure 2.3. The aim in both cases is to take the project to production more efficiently with faster fixes, faster releases and ultimately, a higher quality product that boosts customer satisfaction, whether that's software or machine learning models.

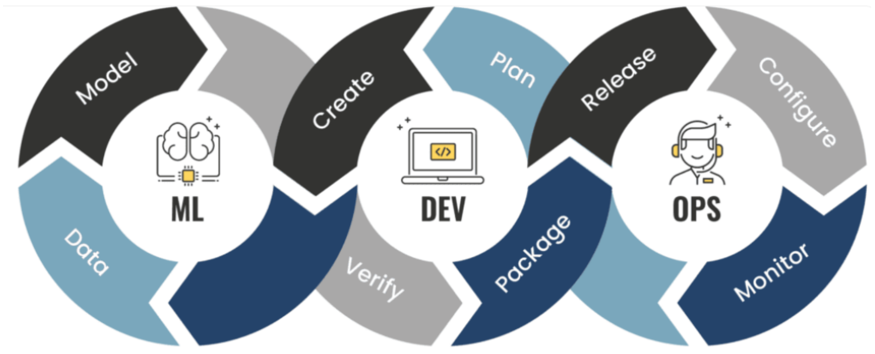

Figure 2.3: The transition from DevOps to MLOps life cycle.

In this thesis, I delve into these aspects, aiming to bridge the gap between local ML implementation and successful code execution in production systems. Let us embark on this journey toward efficient MLOps practices.

3 The Gap: WHy Does It Exists

Before diving into the details of bridging the gap between local code development and cloud execution in MLOps, it is essential to understand how the gap exists in the first place. In this chapter, I will explore a brief explanation of why there is a gap between the two environments.

3.1 Why the Gap Exists

There is a notable distinction between running code locally on a developer's machine and deploying it to a cloud server for execution. This disparity can pose certain challenges in terms of infrastructure, environment, configuration and more, which can make it challenging to ensure seamless functionality across both environments.
I would like to take this opportunity to examine a few of the reasons that have been presented. It is important to consider these reasons in a fair and impartial way:

- Local development environments often have different configurations, dependencies, and resources than cloud environments. This can lead to inconsistencies and incompatibilities when deploying code to the cloud.
- Cloud environments often have different security, compliance, and scalability requirements than local environments. This can lead to additional complexity and overhead when deploying code to the cloud.
- Cloud environments often have different monitoring, logging, and debugging tools than local environments. This can make it difficult to diagnose and resolve issues that arise in the cloud.

In the coming chapters, I will discuss different approaches to overcoming these difficulties and how they can be implemented in practice and put a spurt in the development and debugging process on the cloud.

4 Bridging the Gap

In the previous chapters, I explored the history of machine learning and provided a comprehensive overview of the most critical aspects of the MLOps life cycle, and briefly discussed the reasons for why the gap between running code locally on developer's machine and deploying it in a cloud-based environment for running.
This chapter aims to introduce strategies for overcoming this gap and its associated challanges. Integrating these strategies requires a comprehensive understanding of both the theoretical and practical aspects of cloud computing and local development environments. It is important to point out that each strategy outlined in this chapter is accompanied by a comprehensive explanation of how it works, and include code examples that demonstrate its practical application.

4.1 Containerization

In order to grasp the concept of containerization, it is essential to understand the concept of vertualization and the differences between various virtualization mechanisms.

4.1.1 Virtualization

Virtualization is the process of creating a virtual version or several virtual versions of a piece of computer or a sofware according to cambridge dictionary [Dic]. It is the art of creating a virtual version of somthing that does not exists physically, appearing as though is does.
In computer science, virtualization refers to replicating a physical system or server and its components within a single machine, to mimic their functionality. Basically, it is about sharing capabilities of a single physical machine accross multiple users or virtual settings. This approach empowered cloud service providers to share their physical infrastructure capabilities to user more efficiently.

4.1.2 Containers vs. Virtual Machines

We can distinguish between two primary forms of virtualization [Ale24; BUC; Tea22]:

1. **Virtual Machines (VMs)**: VMs virtualize an entire machine down to the hardware layers. They mimic the behavior of the underlying physical hardware/computer - like CPU, Disks and Networking devices - using a hypervisor, thus creating several virtual machines. Each virtual machine runs its Operating System (OS) required for the respective applications, libraries, and functions separately from the other VMs. Different VMs can be run on the same physical host. Virtual machines may also include a complementary software stack to run on the emulated hardware. These hardware and software packages combined produce a fully functional snapshot of a computational system. As shown in Figure 4.1

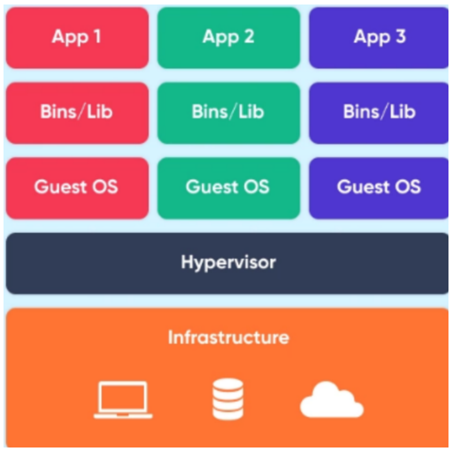

Figure 4.1: Structure of vertual machines.

Advantages

- **Complete Isolation**: VMs operate in total isolation, effectivly wording as independent systems. this isolation ensures that these machine are protected from any harmful exploits, or interference from other VMs sharing the same host. While it is still possible for a virtual machine to be vulnerable to some exlpoits, the contaminated virtual machine will not affect the neighboring VMs due to this isolation, preventing any cross-contamination amoung virtual machines.
- **Dynamic Development**: VMs offer more dynamic development environment. Starting from basic setup, virtual machine can be treated like individual computers. this adaptability enables the direct installation of software, enabling the hands-on development process. Additionally, virtual machines allow for the creating of images, capturing their configurations at any given time.

Disadvantages

- **Cost of Storage Space**: It is worth noting that virtual machine take up a lot of space, due to the fact that they virtualize the entire machine. This expansion can lead to potential disk storage issues as the virtual machines keep expanding. Therefor it is crucial to keep monitoring and managing the consumtion to ensure optimal performance and to avoid any disruption on the virtual machine.
- **Speed of Iteration**: Creating and maintaining a virtual machines can be a complex and time-consuming process, as it involves setting up an entire system stack. Modefiying a snapshot of a virtual machine can require significant efforts to rebuild and ensure its expected functionality.

2. **Containers**: Containers take an alternative approach, by bunding the application with its necessary binaries, libraries and dependencies into a single package. Unlike traditional techniques, containers mimic the host operating system, giving each container the illusion of being isolated and that it is interactiting with its own virtual kernal, while the truth is, that all containers on host are sharing the same kernal. As demonstrated in Figure 4.2. This shared kernal design alloes one OS instance to run several isolated containers, making them lightweight and faster to boot compared to virtual machines. This efficiency is resource management and the accelerated application delivery highlights the advantages of containers over VMs.

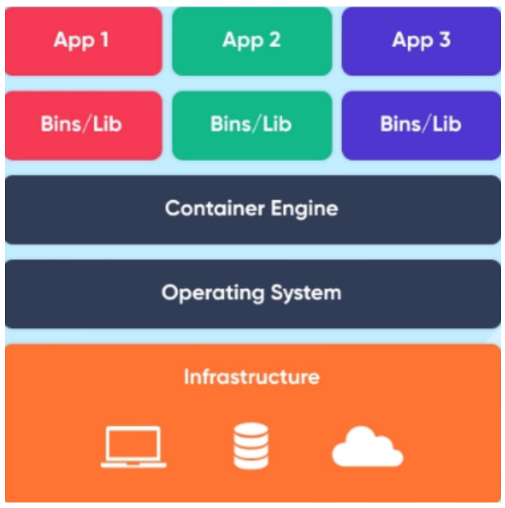

Figure 4.2: General structure of containers.

Advantages

- **Speed of Iteration**: Due to the fact that containers are lightweight and include only the essential high-level software, as we saw earlier, they possess a remarkable capability for rapid development and iteration. Their basic design and focus on minimalism allow efficient modification capabilities.
- **Robust ecosystem**: Container runtime system offer developers access to pre-defined repositories. These repositories provide an extensive selection of popular software applications, such as databases and messaging systems, that are fast to restore and execute. This setup can help development teams to save time and focus on their core tasks.

Disadvantages

- **Shared host exploits**: Containers are built on a shared hardware infrastructure below the operating system layer. That means it is possible that an exploit in one container could break out of the container and affect the hardware that is shared with other containers. This poses a security risk when using one of these public images as they may contain exploits or be vulnerable to being hijacked by nefarious actors.

4.1.3 Docker

Docker[1,2] [BBA17] is the leading containerization technology in the market taht provides open-source tools to simplify the creating, deployment, and management of containerized applications. It serves a wide range of users, from open-source enthusiasts to large-scale enterprises with its versatile versions.
Docker's containerization technology is predicated upon capabilities of the linux kernal, basically exploting mechanisms such as Control Groups (cgroups) and kernal namespaces, in order to create and manage environments that are somewhat isolated, allowing for efficient resource management. These cgroups are equiped with the capabilities for monitoring and managing resources such as CPU, disk IO, network, and memory for a collection of processes. Additionally, namespaces play a critical role is ensuring that process groups and resources are isolated from ane another. In docker, each container is assigned a unique set of namespaces and processes, which allows it to have its own view of the system, including its own network interfaces and file systems.

[1]https://docs.docker.com/
[2]https://hub.docker.com/

4.1.4 Why Use Docker in Machine Learning

The lifecycle of machine learning involves a series of steps aimed at leveraging ML and AI technologies to meet business goals. This include the initial setting of both business and project-specific goals, acquiring and analyzing relevant data, applying different algorithms to model the data, interpreting the outcomes, and sharing them with stakeholders, and finally deploying and sustaining the project. Given the rapid pace of technological and market evolution, it is essential to go through multiple iterations and experiments in order to swiftly adapt to the variuos changes and to ensure a seccessful deployment of reliable project that satisfy business expectations [BCG17].
Docker encapsulate the entire dependencies of a project down to the host OS. This helps prevent issues, where projekts function smoothly on one machine and but preforms horribly on another (Figure 4.3). By using Docker, we are essentially sharing our entire environment rather than just the dependencies and source code. Doing so, allows for seamless collaboration across machine learning projects while ensuring consistency, portability, and effective dependency management. Compared to VMs docker is lightweight, which in turn inhances the speed of each iteration and allows for faster deployment of the project.

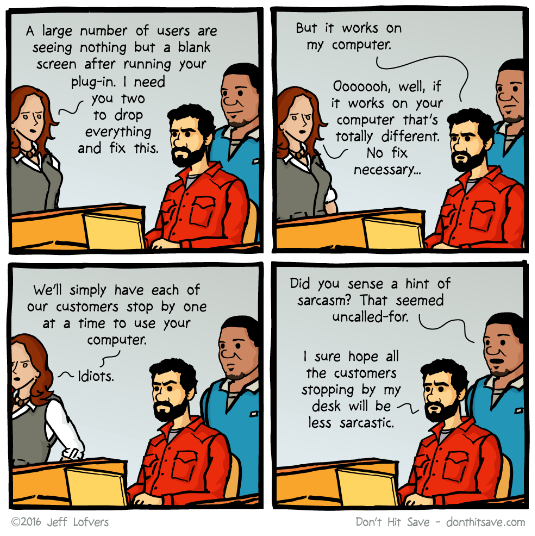
A large number of users are seeing nothing but a blank screen after running your plug-in. I need you two to drop everything and fix this.
But it works on my computer.
Ooooooh, well, if it works on your computer that's totally different. No fix necessary...
We'll simply have each of our customers stop by one at a time to use your computer.
Idiots.
Did you sense a hint of sarcasm? That seemed uncalled-for.
I sure hope all the customers stopping by my desk will be less sarcastic.
©2016 Jeff Lofvers
Don't Hit Save - donthitsave.com

Figure 4.3: It worked on my machine meme.

4.1.5 Usage Scenario

Before I start showcasing the usage of docker in machine learning, a machine learning project has to be created first. Under this github repository[3] you can find a CNN project that takes a 2D picture of hand-written numbers and tries to prodict the number written on that picture. The structure of the is as shown in below.

```
.
├── app.yaml
├── main.py
├── ml_model.py
├── mnist_model.pkl
├── requirements.txt
├── templates
│   ├── homepage.html
│   └── prediction.html
└── README.md
```

I used the MNIST library to train the model, which is a dataset of 70,000 images of hand-written numbers. The dataset is split into 60,000 training images and 10,000 testing images. The model is trained for 50 epochs with a batch size of 512.
Now that the foundation is set, we can start using docker to encapsulate the project.

```

FROM python:3.9-slim

WORKDIR /app

COPY . /app

RUN apt-get -y update
RUN apt-get update && apt-get install -y python3 python3-pip
RUN pip3 install --no-cache-dir -r requirements.txt

CMD ["python", "./main.py"]

```

Listing 4.1: The structure of a Dockerfile

Before building the container imgae it is import to prepare the environment first as we can see in the figure above I am using a python image which is imported from

[3]https://github.com/linuxUser10111/thesis

docker hub. Next we have to copy all necessary files in the project's directory for the project to execute successfully and package them in the newly create working directory and defined above. After that we have to install all the necessary dependencies for the project to run. This include updating the environment, and installing the required libraries declared in the requirements file. Finally we have to define the command that will be executed when the container is run.
Now that the dockerfile is definde correctly, it is time to start building the image, Therefor we use the following command:

```

    docker build -t ml-docker-app .

```

Listing 4.2: Building docker image

This line of code will trigger docker to execute the file definded earlier (Listing 4.1), and if every thing is configured accurately, the image will be build and stored in the local docker registry.And finally we can start running the docker image, as the code below demonstrates:

```

     docker run ml-docker-app

```

Listing 4.3: running the built docker image

% TODO rest of Demo will be done later %

4.2 Cloud Computing

4.3 Hybrid Approaches

5 Evaluation

6 Conclusion

Abbreviations

ML	Machine Learning
AI	Artificial Intelligence
MLOps	Machine Learning Operations
NN	Neural Network
CNN	Convolutional Neural Networks
CI/CD	Continuous Integration and Continuous Deployment
VMs	Virtual Machines
OS	Operating System
cgroups	Control Groups

List of Figures

Listings

List of Tables

Bibliography

[AA23] F. Akram and A. Abbas. "Data-driven Decisions: Exploring the Power of Machine Learning and Analytics." In: (Nov. 2023).

[Ale24] M. Aleksic. *Containers vs Virtual Machines (VMs): What's the Difference*. Jan. 2024. URL: `https://phoenixnap.com/kb/containers-vs-vms`.

[BBA17] B. Bashari Rad, H. Bhatti, and M. Ahmadi. "An Introduction to Docker and Analysis of its Performance." In: *IJCSNS International Journal of Computer Science and Network Security* 173 (Mar. 2017), p. 8.

[BCG17] G. Bhatia, A. Choudhary, and V. Gupta. "THE ROAD TO DOCKER: A SURVEY." In: *International Journal of Advanced Research in Computer Science* 8 (Aug. 2017), pp. 83–87. DOI: `10.26483/ijarcs.v8i8.4618`.

[BUC] I. BUCHANAN. *Containers vs. virtual machines*. URL: `https://www.atlassian.com/microservices/cloud-computing/containers-vs-vms`.

[Dic] C. Dictionary. *definition of Virtualization*. `https://dictionary.cambridge.org/dictionary/english/virtualization`.

[Jac22] A. Jacinto. *Machine Learning History*. 2022. URL: `https://www.startechup.com/blog/machine-learning-history/`.

[McD] M. McDonough. `https://www.britannica.com/biography/Geoffrey-Hinton.`. Accessed: 2024-5-7.

[Naq20] A. Naqvi. "Rise of Machine Learning." In: (Aug. 2020), pp. 51–67. DOI: `10.1002/9781119601906.ch4`.

[RVK24] C. Rahal, M. Verhagen, and D. Kirk. "The rise of machine learning in the academic social sciences." In: *AI & SOCIETY* 39.2 (Apr. 2024), pp. 799–801. ISSN: 1435-5655. DOI: `10.1007/s00146-022-01540-w`.

[Tea22] E. Y. Team. *Containers vs Virtual Machines Differences, Pros, & Cons*. Mar. 2022. URL: `https://www.engineyard.com/blog/containers-vs-virtual-machines-differences-pros-cons/`.

[TR23] M. Torkjazi and A. K. Raz. “Data-Driven Approach with Machine Learning to Reduce Subjectivity in Multi-Attribute Decision Making Methods.” In: *2023 IEEE International Systems Conference (SysCon)*. 2023, pp. 1–8. DOI: `10.1109/SysCon53073.2023.10131094`.

[Van86] C. Van Der Malsburg. “Frank Rosenblatt: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.” In: *Brain Theory*. Ed. by G. Palm and A. Aertsen. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 245–248. ISBN: 978-3-642-70911-1.