NTNU
Kunnskap for en bedre verden

Department of Computer Science

IDATT 2900 - Bacheloroppgave

Hyperfetch: Addressing Reproducibility and Environmental Impact in Reinforcement Learning

Author:
Karoline Sund Wahl

May 22nd, 2023

Abstract

In recent years, the field of deep reinforcement learning (DRL) has witnessed remarkable advancements. However, the reproducibility of results in this domain is very advanced and challenging. Henderson et al. [1] shed light on these challenges, highlighting issues related to hyperparameters, random seeds, and the choice of algorithm implementations. These factors can greatly influence the outcome of experiments and hinder the ability to replicate and validate findings if reported wrongly or not in full.

The primary objective of this thesis is to develop an installable tool called Hyperfetch, which is able to train hyperparameters, as well as extract trained hyperparameters for RL projects. Hyperfetch operates in conjunction with a website that visualizes the extracted hyperparameters, alongside their emissions and other relevant metadata. By bridging existing knowledge gaps in emission profiling and contributing factors specific to RL projects, Hyperfetch aims to facilitate reproducibility while providing insights into emission profiles.

Throughout the project, significant emphasis has been placed on creating a functional and dependable optimization module, as well as designing an intuitive interface using user-centered design principles. The development process involved an extensive planning stage, where the website's structure was modeled, and the logic behind optimization and emission tracking was mapped out. Subsequently, a prototype was developed and tested through interviews and user testing. After improvements were made, all components, including the module, database, API, and website, were deployed.

Using the online Hyperfetch system, the impact of different factors were tested when tuning and training RL models. These factors were countries, regions within the United States, cloud providers, and algorithm selection. The test-results were compared and analyzed.

The study revealed variations in emissions among different countries, with European countries generally demonstrating lower emissions compared to other recorded countries. Similarly, emissions varied across different regions within the United States, with Vermont exhibiting the lowest emissions per hour and Kentucky producing emissions approximately 30.8 times higher. Comparisons of different cloud providers consistently positioned Google Cloud as the provider with the lowest emissions per hour, while Azure exhibited the highest emissions. Moreover, the analysis of emissions generated by different reinforcement learning algorithms show that algorithm choice does have an impact on emissions. Specifically, Soft Actor-Critic (SAC) consistently generated higher emissions compared to Proximal Policy Optimization (PPO) and Advantage Actor-Critic (A2C), particularly in regions such as Norway and Germany, where SAC emitted approximately twice as much CO2eq per hour.

These findings underscore the significance of considering factors such as country location, regional location, cloud provider, and algorithm choice when evaluating CO2 emissions during model training. By planning an optimization with these factors in mind, it is possible to reduce the environmental impact associated with model development and contribute to sustainability efforts. This means that by integrating sustainability considerations into the development of RL models, we can work towards minimizing the environmental footprint of these technologies.

Sammendrag

De siste årene har feltet for Deep Reinforcement Learning (DRL) opplevd store fremskritt. Imidlertid utgjør reproduksjon av resultater i dette domenet betydelige utfordringer. Henderson et al. [1] setter lys på disse utfordringene og fremhever problemer knyttet til hyperparametere, Random Seeds, og valg av implementasjon av algoritmer. Disse faktorene kan ha stor innvirkning på resultatene av eksperimenter og gjøør det vanskeligere å gjenskape og validere andres funn.

Hovedmålet med denne rapporten er å utvikle et installerbart verktøy kalt Hyperfetch, som skal kunne trene og utvinne hyperparametere for RL-prosjekter. Hyperfetch fungerer sammen med en nettside som visualiserer de innhentede hyperparameterne, utslippene som ble skapt da modellen ble trent, og annen relevant metadata. Ved å tette eksisterende kunnskapshull innenfor utslippsprofilering, har Hyperfetch som mål å legge til rette for reproduksjon samtidig som programmet gir verdifulle innsikter i utslippsprofiler.

Gjennom hele prosjektet har det blitt lagt betydelig vekt på å skape en funksjonell og pålitelig optimaliseringsmodul, samt å designe et intuitivt brukergrensesnitt ved hjelp av prinsipper for brukersentrert design. Utviklingsprosessen innebar en omfattende planleggingsfase, der nettsidens struktur ble modellert og logikken bak optimalisering og utslippssporing ble nøye kartlagt. Deretter ble en prototype utviklet og testet gjennom intervjuer og brukertesting. Etter forbedringer basert på tilbakemeldingen, ble alle komponenter, inkludert modulen, databasen, API og nettsiden deployert.

Ved hjelp av Hyperfetch-systemet ble virkningen på utslipp gitt av ulike faktorer testet. Dette skjedde under hyperparameter-optimalisering og trening av RL-modeller. Disse faktorene inkluderte land, regioner innenfor USA, skytjenesteleverandører og valg av algoritme. Testresultatene ble sammenlignet og analysert.

Studien avdekket variasjoner i utslipp mellom ulike land, der europeiske land generelt viste til lavere utslipp sammenlignet med andre land i testen. På samme måte varierte utslippene mellom ulike regioner innenfor USA, der Vermont hadde de laveste utslippene per time, mens Kentucky produserte utslipp omtrent 30,8 ganger høyere. Sammenligninger av ulike skytjenesteleverandører plasserte konsekvent Google Cloud som leverandøren med lavest utslipp per time, mens Azure hadde høyest utslipp. Videre understreket analysen at valg av Deep Reinforcement Learning algoritme hadde en betydningen for utslippet som ble produsert. Spesifikt genererte Soft Actor-Critic (SAC) jevnt over høyere utslipp sammenlignet med Proximal Policy Optimization (PPO) og Advantage Actor-Critic (A2C). Dette var spesielt sant i regioner som Norge og Tyskland, der SAC slapp ut omtrent dobbelt så mye CO2-ekvivalenter per time som de andre algoritmene.

Disse funnene understreker betydningen av å ta hensyn til faktorer som lokasjon, skytjenesteleverandør og valg av algoritme ved evaluering av CO2-utslipp under modelltreningsprosessen. Ved å planlegge optimalisering med dette i bakhodet, er det mulig å redusere miljøpåvirkningen knyttet til modellutvikling og bidra til bærekraftige tiltak.

Dette betyr at ved å integrere bærekraftvurderinger i utviklingen av RL-modeller, kan vi jobbe for å minimere miljøavtrykket av disse teknologiene.

Preface

In recent years, the field of Reinforcement Learning (RL) has experienced exponential growth and emerged as a leading sub-field within machine learning. Its potential to tackle complex problems and achieve results in advanced learning problems has garnered significant attention and interest.

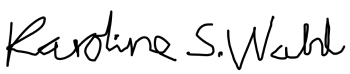
My introduction to RL occurred in 2022 during the elective course IDATT 2502 - Machine Learning with Project. Collaborating with Maiken Louise Brechan, we created on a project that focused on ”Reinforcement Learning using Bio-inspired methods.” Our project aimed to develop an evolutionary algorithm for training Neural Networks. Throughout this experience, I gained a newfound interest for Reinforcement Learning, and its many implications.

As I have learned more about RL, I have recognized its complexity and the challenges associated with tuning hyperparameters, which are important for building models that perform well. Moreover, I have observed a lack of accessible resources and configurations necessary for successful research validation within this field. This realization ignited my motivation to make a meaningful contribution to the RL community by creating a solution that hopefully increases this availability. It is within this context that the problem description explored in this thesis took shape, focusing not only on the reproduction of hyperparameters but also on addressing emissions-related concerns.

I would like to express my gratitude to the following individuals who have supported and guided me throughout this thesis:

- Jonathan Jørgensen for guiding me through the thesis and giving me excellent advice.
- Trond Are Øritsland for giving me feedback on the interface design of the webpage.
- Torkil Marsdal Hanssen for giving me feedback on the user-centered design of the webpage.
- Vibeke Hvidegaard Petersen for teaching me how to use Latex.
- Maiken Louise Brechan for conducting and finishing last years project with me.
- Nora, for staying up late with me and supporting me through this entire process.

May 21st, Trondheim
Karoline Sund Wahl



Problem Description

Reproducibility and understanding of environmental impact are crucial aspects in the field of reinforcement learning (RL). Challenges within the field lead to gaps in knowledge and limits the advancement and wider adoption of responsible and sustainable practices. The first challenge revolves around reproducibility. Reproducing and comparing research findings in RL is stamped by the lack of published hyperparameters alongside research publications. The omission of detailed hyperparameter information is a hurdle that makes it hard for researchers and practitioners to replicate experiments, assess result validity, and understand the impact of specific hyperparameters on performance. Furthermore, the lack of transparent reporting and difficulty in building upon previous research slows down the field's development and hinders knowledge transfer.

The second challenge concerns the environmental impact of reinforcement learning projects. Despite their promising potential, the understanding of emission profiles and their contributing factors remains limited. The absence of comprehensive emission reporting and analysis makes it hard to utilize informed decision-making when training RL models. The field lacks a clear understanding of the factors that significantly contribute to emissions, which hinders the development of strategies to slow down their environmental impact and design environmentally responsible reinforcement learning systems.

To address these critical issues, a system called Hyperfetch will be created. This system will consist of a website, and an optimization module. The websites functionality will be to extract optimized hyperparameters, as well as to compare the environmental impact of different sets of tuned hyperparameters. The optimization module will be used for tuning reinforcement learning models, and persisting their hyperparameters and environmental data. The hyperparameters and environmental data will be available on the Hyperfetch website. This research aims to develop an intuitive and comprehensive tool that efficiently extracts hyperparameters, enabling transparent reporting and reproducibility. In addition, it is thought that insights into contributing factors for CO2 emissions created by RL projects tuning and training can be found by leveraging Hyperfetch. By bridging the gaps in both reproducibility and understanding the environmental impact, we can foster a more robust scientific community and facilitate responsible advancements in RL.

Table of Contents

List of Figures

Glossary

Function Approximators An evaluation environment is a controlled setting or framework where the performance and behavior of a trained model is measured. . 5

Model The algorithm drives the learning process, while the model is the outcome or result of that process. . 6

Parzen estimator The Parzen estimator is a method for estimating the shape of a probability distribution based on a set of observed data points. . 6

SCI Software Carbon Intencity. 9

TOML Toms Obvious Minimal Language. 34

1 Introduction

The field of Artificial Intelligence (AI) has gained tremendous popularity in recent years. AI research has seen a significant surge, with numerous papers being submitted to conferences and conferences focusing on AI advancements [2]. Reinforcement Learning (RL) agents aim to learn the optimal mapping of situations to actions through trial-and-error search, setting RL apart from other areas of Machine Learning (ML) and pushing the boundaries of knowledge [3]. AI has become a forefront solution in various complex domains such as computer vision and natural language processing. It is projected that AI will generate over 15 trillion USD for the world economy and contribute to a 26% boost in GDP by 2030 [4].

Reproducing results in deep reinforcement learning poses challenges. Hyperparameter optimization methods require extensive resources and may not be feasible to use for many researchers and practitioners. Reproducibility issues in deep reinforcement learning include problems with hyperparameters, random seeds, and different baseline algorithm implementations [1]. Obtaining code published by others has also proven challenging, with a low success rate in obtaining code with and without communication with the authors [5].

With the increasing accessibility of powerful GPUs, machine learning practitioners now have greater resources at their disposal. However, contemporary models often require training on more GPUs, utilizing larger neural networks and data sets, and extended training durations. As machine learning models become more computationally intensive [6], understanding their impact on climate change is important. However, estimating the climate change impacts of machine learning research and applications is challenging due to limited reporting and accounting practices.

1.1 Research Questions and Thesis Aim

The primary objective of this thesis is to address these challenges by introducing Hyperfetch [7], a system designed to facilitate result reproducibility in reinforcement learning and simplify research processes. Hyperfetch provides a time-saving website for research and education, where hyperparameters are available for fetching such that users can focus on learning and development. By reducing the need for frequent hyperparameter retraining, the platform contributes to the collective effort of reducing carbon emissions associated with power usage. Additionally, the Hyperfetch system provides an installable module that tracks information regarding produced emissions during hyperparameter tuning [8]. An accurate understanding of this information is crucial in tackling climate change caused by fields such as RL, as emphasized in [9]. Therefore, the thesis will also provide a concrete analysis on server-location and emissions when training reinforcement learning models. The optimization module will be used for this. The thesis aims to answer the following research questions:

1. How can an intuitive and comprehensive tool be designed to efficiently extract hyperparameters?
2. How can the Hyperfetch application be leveraged to gain valuable insights into the emission profile of a reinforcement learning project?

2 Theory

This chapter contains theory essential to solving the research question. The chapter includes theory related to Artificial Intelligence, Reinforcement Learning, Hyperparameters, emissions and design. In addition, deployment and security is quickly briefed.

2.1 Artificial Intelligence

Artificial Intelligence consists of many categories. However, the one relevant to this project is machine learning. More specifically, Reinforcement Learning.

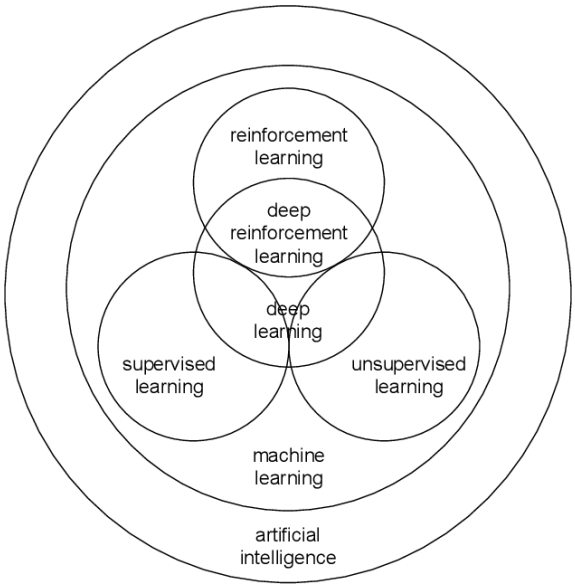


Figure 1: A visualization of the main branches within artificial intelligence with focus on Reinforcement Learning.

Source: [10]

Supervised learning is a type of machine learning where the algorithm learns from labeled training data. The goal of supervised learning is to make accurate predictions or classify new, unseen data. Unsupervised learning involves training machine learning models on unlabeled data, where the input data doesn't have corresponding output labels. The goal of unsupervised learning is to discover underlying patterns, structures, or relationships within the data. Reinforcement learning (RL) is a type of machine learning where an Agent learns to make decisions by interacting with an environment. Deep learning is a subfield of machine learning that focuses on training artificial neural networks with multiple layers, also known as deep neural networks. Deep learning algorithms are inspired by the structure and function of the human brain's neural networks. By using multiple layers, deep learning models can automatically learn hierarchical representations of data, extracting increasingly complex features at each layer. Deep learning can be applied as a technique within the broader frameworks of supervised learning, unsupervised learning, and reinforcement learning.

2.2 Reinforcement Learning

The Agent in RL is the ”decision-maker” within a reinforcement learning framework. The agent’s objective is to learn from its interactions with the environment.

2.2.1 Action space

There are two types of action space for a Reinforcement Learning environment. In a discrete action space, the set of possible actions that the agent can take is finite and countable. Examples of discrete action spaces include selecting from a fixed set of options such as moving up, down, left, or right in a grid world. In a continuous action space, the set of possible actions is uncountable and typically represented by a continuous range of values. Examples of continuous action spaces include controlling the joint angles of a robot arm or adjusting the steering angle and throttle of a self-driving car.

2.2.2 The Agent

The agent is the instance that:

- Interacts with an environment (see figure 2), which can be a simulated or real-world scenario. The environment provides feedback to the agent based on its actions and current state.
- Perceives the environment through observations. All observations in an environment is an observation-space. This observation-space can consist of discrete or continuous values.
- Selects actions to perform based on the observations it makes. The sum of all actions that are available to the agent are called action space.
- The agent follows a policy, which is a strategy or rule that guides its decision-making process. The policies that work best depend on what type of observation space the environment has.
- After taking an action, the agent receives feedback from the environment in the form of a reward or penalty.
- Uses reinforcement learning algorithms like Deep Q-Network or Proximal Policy Optimization to update its policy based on observed rewards and improve its decision-making.
- Has a goal of maximizing cumulative rewards, reaching a specific state, or accomplishing a task specified by the environment.

The illustration in Figure 2 represents the interaction between the agent and the environment, following the principles of a Markov Decision Process (MDP).

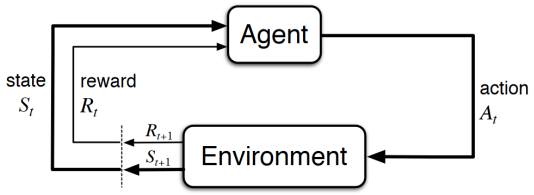


Figure 2: A visualization of interaction between the Agent and the Environment.

Source: [11]

MDP popular due to its ability to model decision-making processes, which uses the Markov property. The Markov Property is a property that says that future states are solely dependent on the current state and encapsulate all relevant information for decision-making. In an MDP:

- The number of possible states is finite, although it can also handle infinite processes with minor adjustments. For time-dependent processes, we assume a finite time period, such as t=0,1,2...,T.
- Each outcome only depends on the outcome of the previous stage, satisfying the Markov property.
- Transition probabilities remain constant over time.
- The system is stationary, implied by the previous property.

While the Markov property generally holds, it's important to consider scenarios where historical information may be relevant, such as a trading agent analyzing past trends.

2.2.3 Deep Reinforcement Learning

In regular, or traditional Reinforcement Learning, function approximators used to estimate action values or policies are often simple; such as lookup tables or linear models. A function approximator is a model that approximates the relationship between input data (observations) and output values (actions or probabilities). These function approximators can handle problems with lower-dimensional state and action spaces. Traditional RL methods are capable of handling relatively simpler problems with smaller state and action spaces. As the complexity of the environment increases, the effectiveness of simpler function approximators might not be very good.

When Deep Learning techniques are applied to Reinforcement Learning (RL), it is to tackle complex sequential decision-making problems. Deep reinforcement learning therefore involves training entire Deep Neural Networks (DNNs) as function approximators to estimate action values or policies in reinforcement learning algorithms. The DNNs look different based on which RL algorithm that is used. An example is the DNN function approximators for Deep Q-Networks (DQNs), which take the state and action as inputs and output the corresponding Q-values. Another example is the Proximal Policy Optimization (PPO) algorithm, which has a different aim than DQNs, namely to directly approximate the policy, which maps states to actions. Therefore, PPOs function approximator DNN has to take the state as input, and output the probability distribution over possible actions.

Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) differ in the type of function approximators used and their capabilities. RL relies on simpler function approximators and is suitable for problems with lower-dimensional spaces, while DRL utilizes deep neural networks to handle complex problems with high-dimensional spaces, allowing for automatic feature extraction and generalization. DRL is particularly advantageous when dealing with raw sensory input, large state and action spaces, and complex environments. Given that the focus of this thesis will be on Deep Reinforcement Learning (DRL), it is DRL specifically that will be referred to as the intended technique in the subsequent chapters when discussing RL.

2.3 Hyperparameters

Hyperparameters are the parameters that configures a learning algorithm or model, and are not being learned from the data itself. They influence the behavior and performance of the algorithm but are not directly estimated from the training data. Hyperparameters are set by the practitioner before the training process begins and are usually tuned and optimized through experimentation and trial-and-error. Hyperparameters can include values such as learning rate, batch size, number of hidden layers, number of units in each layer, regularization strength, dropout rate, maximum tree depth, number of clusters, and many others. The specific hyperparameters depend on the chosen learning algorithm or model. The choice of hyperparameters can significantly impact the performance, convergence, and of a machine learning model. Different hyperparameter configurations can lead to different outcomes, and selecting appropriate values is crucial for achieving the desired results.

2.3.1 Hyperparameter optimization

Tuning hyperparameters involves the process of searching for the optimal combination of hyperparameter values that yield the best performance for a reinforcement learning algorithm on a given problem. This process often involves experimentation, training multiple models with different hyperparameter settings, and evaluating their performance on validation data or by using an Function Approximators.

2.3.2 Search methods

Hyperparameter optimization involves searching through the space of possible hyperparameter values to find the best combination. This space is called the search space. Several search methods can be employed, including grid search, random search, Bayesian optimization, evolutionary algorithms and more. These methods explore different hyperparameter configurations and evaluate their performance on a validation set. The performance of each hyperparameter configuration is evaluated on this validation set, and the configuration with the best performance is selected. Hyperparameter optimization in RL can be computationally expensive, especially when large number of training episodes are required. Techniques such as early stopping or parallelization can be used to improve efficiency and reduce the computational burden of hyperparameter optimization.

Grid Search Systematically explores all possible combinations of hyperparameter values by creating a grid-like search space. Grid search evaluates the performance of the model using each combination and selects the one with the best performance. This method is very computationally expensive, and in cases with a large enough search space, impossible.

Random Search Randomly selects different combinations of hyperparameter values within the algorithms search space and evaluates the Model's performance using these configurations. It is computationally efficient and has been shown to be effective in many cases [12].

Bayesian Optimization Is a sequential model-based optimization technique that uses a probabilistic model to guide the search process. Bayesian optimization iteratively explores the search space, with a focus on promising regions, and adapts the search based on the model's predictions. It is especially useful when evaluating configurations is computationally expensive [13].

Evolutionary Algorithms Are population-based search methods inspired by natural evolution. This means that they maintain a population of candidate solutions (actually hyperparameter configurations) and iteratively apply mechanisms such as selection, mutation, and crossover to evolve the population over generations. Each candidate solution's performance is evaluated, and selection methods determine which solutions will contribute to the next generation. These algorithms are suitable for complex search spaces, but require more computational resources compared to other methods.

2.3.3 Sampling

Sampling refers to selecting a subset of possible hyperparameter values or generating random or systematic samples to evaluate their performance and determine the optimal configuration. The method chosen for performing this selection plays a crucial role in hyperparameter optimization, as it defines which hyperparameter configurations are explored and evaluated during the optimization -and pruning process. This section describes a few commonly used sampling techniques, and builds on the search methods from the previous section.

Random Sampler: The random sampler is a basic sampling method where hyperparameter configurations are randomly selected from the search space (random search). This sampler does not consider any prior information or knowledge about the performance of different configurations.

TPE (Tree-structured Parzen Estimator): TPE is a Bayesian optimization sampler that models the performance landscape of hyperparameter configurations using a tree-structured Parzen estimator. It maintains separate probability distributions to model good and bad configurations. This method has been shown to be effective in efficiently searching the hyperparameter space, especially in cases where the search space is complex or contains many discrete variables [14].

CMA-ES (Covariance Matrix Adaptation Evolution Strategy): CMA-ES is an evolutionary algorithm-based sampler that employs an evolutionary strategy to optimize hyperparameters. This method is particularly effective in continuous and multimodal optimization problems and has shown good performance in various applications [15].

NSGA-II (Non-dominated Sorting Genetic Algorithm II): NSGA-II is an evolutionary algorithm sampler. NSGA-II applies genetic operators such as selection, crossover, and mutation to iteratively evolve a population of hyperparameter configurations.

2.3.4 Pruning

In the context of hyperparameter optimization, pruning refers to the process of reducing the search space by eliminating or discarding unpromising hyperparameter combinations. Pruning techniques aim to narrow down the set of hyperparameters to explore, making the optimization process more efficient and focused. Pruning as applied to a search method (2.3.2). These are some relevant pruning methods:

Successive Halving Pruner: The Successive Halving Pruner is a pruning technique used in hyperparameter optimization that iteratively eliminates poorly performing configurations. For each round of evaluation, a subset of hyperparameter configurations is sampled (using a sampler) and evaluated. Then, only a fraction of the best-performing configurations from each stage is selected to move to the next stage. This process continues until a single configuration remains.

Median Pruner: The Median Pruner is another pruning technique commonly used in hyperparameter optimization. The Median Pruner prunes configurations that consistently perform worse than the median performance. By discarding under-performing configurations based on the median, it helps focus the search on more promising configurations.

Hyperband Pruner: The Hyperband Pruner is a pruning technique that combines the concepts of the Successive Halving Pruner and Random Sampler (2.3.3. This pruner first samples a set of hyperparameter configurations using a sampler. The pruner then allocates a predefined budget (processing power) to each configuration. It then prunes using Successive Halving Pruner on each set. Configurations with the best performance are promoted, and the budget is increased for those configurations. Underperforming configurations are eliminated.

2.4 Emissions

This section will explain how one can measure energy use, as well as how emissions can be reduced within Reinforcement Learning (RL). Energy accounting is fairly straightforward. The consumption og energy for a system can be measured in Watt-hours (Wh) or Joules (J). For this thesis, kilowatt hour (kWh), which is a measure of how much energy is used per hour, will be utilized. In order to measure, or quantify, carbon emissions,CO2-equivalents (CO2eq) are used, which is a standardized measure that expresses the global-warming potential of various greenhouse gases as the amount of CO2 which would have had the equivalent global warming impact.

Global climate change is a well-recognized phenomenon within the scientific community that seems to be accelerated due to increasing greenhouse gas (GHG) emissions such as carbon dioxide or equivalent, CO2eq [16]. The impacts of global climate change have and safety impacts and are projected to fall disproportionately on the poor and vulnerable. Production of energy is a large factor in GHG emissions, which contributes to about 25% of GHG emissions in 2010 [17]. Because the demands for

computing power require increasingly larger energy demands, many modern machine learning (ML) methods have the potential to significantly contribute to carbon emissions due to their long training times. From an environmental standpoint, there are some important aspects of training a neural network that have a big impact on the quantity of carbon that is emitted. These factors include the location of the server used for training, the energy grid that is used at the server location, the length of the training procedure, and even the make and model of hardware on which the training takes place. This is because the efficiency of the hardware varies with the amount of processing power that is utilized. For the most part, the most efficient use of energy is when the computing power is near maximum utilization [18]. This knowledge is an important factor in optimization, particularly when using large cloud compute systems.

2.4.1 Emissions within Machine Learning

There is a great degree of variability in CO2eq emitted depending on the geographical region of a given server, as can be seen in figure 3. Even within a single region, large amount of variation can be found. For example, servers located in North America can emit anywhere between 20g CO2eq/kWh in Quebec, Canada to 736.6g CO2eq/kWh in Iowa, USA [19]. This difference is a factor of 37, caused by the Quebec location using renewable energy sources, whilst the Iowa location relies on fossil fuels.

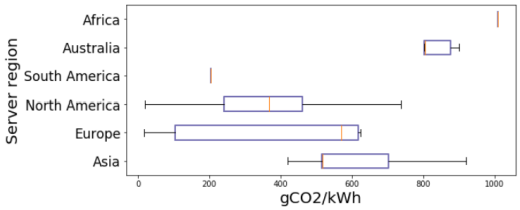


Figure 3: The distribution and variation in carbon emissions depending on geographical region.

Source: [20]

Although the location of the server running the training of a machine learning model is the greatest factor in regards to carbon emissions, other important - although smaller - factors exist. A factor like that is the computing infrastructure used and training time utilized to train a model. Deep learning models such as Deep Reinforcement Learning models, can be computationally intensive and require significant computational resources. With the neural networks residing within these models becoming deeper (more layers added to the network) and thus more complex, recent state-of-the-art models are often trained using multiples GPUs. This training can last for several weeks (or months) to beat benchmark performance, and requires more and more energy.

2.4.2 Reducing emissions within Machine Learning

This section will explain how emissions can be reduced within Machine Learning (ML). This can also be adopted to Reinforcement Learning. To reduce collective emissions within Machine Learning, there are some guidelines that should be followed [20].

Selecting Data Center Location: Although several cloud providers claim carbon neutrality, the carbon intensity of their data centers can vary based on the local power grid they rely on. Some data centers may still heavily rely on carbon-intensive energy sources, while others are powered entirely by renewable energy. Consequently, the choice of data center location for training an algorithm significantly influences its carbon emissions. State-of-the-art models like BERT [21], which is used in natural language processing (NLP), and VGG [22], primarily used in computer vision, require extensive training on multiple GPUs for several weeks. By choosing to train models like these on a server powered by hydroelectricity instead of fossil fuels, it is possible to avoid emitting several hundreds of kilograms of CO2eq.

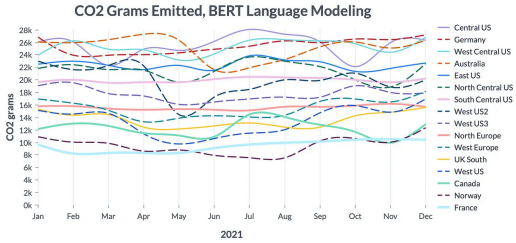


Figure 4: Choosing the appropriate geographic region plays the largest role; selecting it can reduce the Software Carbon Intencity (SCI) by almost 75%.

Source: [23]

Reduce Wasted Ressources: Grid search is still often used in practice, in spite of its low efficience both in terms of model performance and environmental impact. However, it has been shown that random search and other search algorithms are more efficient and provide better optimizations, consequently reducing carbon emissions [24, 25, 26].

When deciding on a training procedure for machine learning architectures, it is important to consider whether training a model from scratch is necessary or if reusing existing models and hyperparameters would suffice for the task. Recent research has demonstrated that fine-tuning pre-trained models specifically for the task at hand achieves comparable performance to training from scratch. This approach has proven effective in image recognition tasks [27] and natural language processing tasks [28]. Fine-tuning involves retaining the weights of the initial layers (the ”base” or ”feature” layers) from the pre-trained model, while updating or replacing the later layers - often known as the ”classification” layers - to suit the target task.

2.5 Design theory

According to the Association for Computing Machinery (ACM), human-computer interaction is the field that encompasses the design, evaluation, and implementation of interactive computing systems for human use. It also involves studying the significant phenomena associated with these systems [29]. When designing a user interface, several important design principles should be considered.

Early focus: This principle emphasizes understanding the users and their tasks. It involves identifying the user demographics, determining the tasks users will perform, and defining the frequency of task execution.

Empirical measurement: This principle involves testing the interface with real users who interact with it regularly. The results obtained from these tests may vary depending on the users' skill levels, and they may not always represent the typical human-computer interaction scenarios.

Iterative design is a process that follows the determination of users, tasks, and empirical measurements mentioned above. This approach involves breaking down the development process into multiple iterations, each encompassing activities such as designing, prototyping, testing, and refining the product incrementally. The main focus of iterative design is to gather feedback from users at an early stage and throughout the development process. This way, iterative design can be seen as the manifestation of the design principles mentioned before this one. By incorporating the user feedback and refining the design, iterative design ensures that the final product aligns closely with user needs and expectations.

An iteration might look like this:

1. Design the user interface
2. Test
3. Analyze results
4. Repeat

According to DeLone and McLean [30], user satisfaction is a key measure of computer system success, if not synonymous with it. The user experience, which involves various aspects such as presentation, functionality, system performance, interaction behavior, and assistive capabilities [31], directly contributes to user satisfaction.

The User-Centred Design is based on the usability of the application, and revolves around finding and catering to the user's needs [32]. This approach involves several points.

- The project should be based on the understanding of users.
- Users should be involved throughout the development process project. User engagement is a valuable source of knowledge about the context of use and should be used to explore solutions.
- The project should be conducted and refined through assessments focusing on the user, which minimizes the risk of the system do not reach requirements that meet the users needs and desires.

- The design process should be iterative. The iterative design process is repeated until a sensible, user-friendly interface is created.
- The design should address the entire user experience.

2.5.1 Gestalt principles

The Gestalt principles, commonly referred to as the laws of perceptual organization, are a set of principles that describe how humans naturally perceive and make sense of visual information [33]. These principles serve as valuable guidelines for creating visually appealing designs that facilitate easy processing and comprehension by the human brain. Among the numerous Gestalt principles are the following:

Proximity: Elements that are close together are perceived as a group or a single unit.

Similarity: Elements that are similar in shape, color, or texture are perceived as a group or a single unit.

Closure: When an object is incomplete or has missing parts, our brains fill in the gaps to create a complete shape.

Continuity: Our brains prefer to perceive continuous and smooth lines or curves, rather than abrupt changes in direction or shape.

Symmetry: Our brains perceive objects that are symmetrical as more visually appealing and balanced.

Figure-ground: Our brains perceive objects as either a figure (the object of focus) or a background (the surrounding area).

Common fate: Elements that move or change in the same direction or at the same time are perceived as a group.

These principles are important to consider when designing interfaces with the aim of being easily understood by the viewer, and work well alongside user-centered design.

2.6 Deployment

Deployment refers to the process of making an application available and operational for end-users or clients to access and use. It involves taking the developed application code and associated resources and setting them up in a production environment where the application can run smoothly and reliably. Some methods for deployment are on-premises hosting, shared hosting, cloud-based deployment. On-premises hosting involves hosting an application on ones own servers or hardware. This option requires self-managed and self-maintained infrastructure, which can be expensive and time-consuming. Shared hosting involves sharing server resources with other websites hosted by the same provider. Shared hosting can be a cost-effective option for small websites with low traffic, but may not provide the same level of performance and customization as other options. Cloud-based deployment involves using Cloud Service Providers such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud

Platform (GCP) to host and manage the application. Cloud providers offer various services, including:

- Infrastructure-as-a-Service (IaaS): The cloud provider takes care of the physical infrastructure, such as servers, storage, and networking.
- Platform-as-a-Service (PaaS): The cloud provider Abstracts away the management of infrastructure and provide developers with a platform to deploy and run their applications.
- Software-as-a-Service (SaaS): The cloud provider manages everything from infrastructure to application deployment and maintenance.

2.7 Security

Both websites with user input and those without can be vulnerable to a variety of security risks. However, websites that accept user input are generally considered to be at higher risk because user input can be used to exploit vulnerabilities in the website's code or infrastructure.

For example, websites with user input may be vulnerable to attacks such as cross-site scripting (XSS), SQL injection, and other types of injection attacks. These attacks can be used to steal sensitive data, compromise user accounts, or take control of the website or underlying infrastructure. However, these attacks can also happen for websites that do not utilize user input. In those scenarios, the URL holds the risk as messages to the API must happen through the use of query string-parameters. Regardless of whether a website accepts user input or not, it is important to take a comprehensive approach to security and to regularly review and update security measures to protect against both known and emerging threats.

Cross-Site Scripting This occurs when an attacker injects malicious code into a website, which can then be executed by users who visit the site. CSS attacks can allow attackers to steal sensitive information, such as login credentials or credit card numbers. This vulnerability occurs when an application does not properly sanitize or validate user input and then displays that input on a web page without proper encoding.

SQL injection This occurs when an attacker injects malicious SQL code into a website's database, which can then be executed by the database server. An attacker does this by identifying an applications input field that is used directly in constructing SQL queries. This could be a user input field, URL parameter, or form submission. SQL injection attacks can allow attackers to extract sensitive information from the database, or even take control of the server. SQL injections can be avoided by using prepared statements, proper input validation and sanitation, following the Least Privilege Principle, and by avoiding Dynamic SQL. The Least Privilege Principle means to grant the database user account or application access only to the necessary database resources and restrict unnecessary privileges. Dynamic SQL is where SQL statements are constructed dynamically based on user input or other run-time values, which is the opposite of using a preparared statement.

HTTP Host Header attacks HTTP Host header attacks, also known as Host header injection attacks, are web security vulnerabilities that exploit the Host header of an

HTTP requests. By manipulating the Host header, an attacker may attempt to hijack a session, gain unauthorized access to sensitive data, or perform other malicious activities. This can be prevented by ensuring that the requested host name matches the expected host name. A way to do this is to hold a predefined list of allowed host names.

Man In The Middle attacks This occurs when an attacker intercepts and potentially alters communications between two parties who believe they are directly communicating with each other. The attacker positions themselves between the communicating parties, allowing them to eavesdrop on the conversation, manipulate data, or even impersonate one or both parties. This occurs when a website does not properly encrypt its communications. Attackers can intercept this data that is not encrypted and use it for malicious purposes. To avoid this attack, a website should use the Hypertext Transfer Protocol Secure (HTTPS), instead of regular HTTP. This protocols encrypt the data being transmitted, making it difficult for attackers to intercept and manipulate it.

3 Method

In order to address the research questions presented in the introduction, this chapter outlines the methodology required to achieve the following research goals:

1. How can an intuitive and comprehensive tool be designed to efficiently extract hyperparameters?
2. How can the Hyperfetch application be leveraged to gain valuable insights into the emission profile of a reinforcement learning project?

The first research goal is addressed in the methodology sections that focus on the planning stage, prototype development, user testing, and the deployment of the final product.

Once the Hyperfetch application is functional, the second research goal takes precedence. Exploring this topic involves utilizing the deployed optimization module using different server locations and RL models for hyperparameter tuning. This will be done by conducting a series of tests.

By doing these evaluations, the study aimed to gain insights into the environmental implications of server placement and algorithm selection during the tuning and training of machine learning models. By undertaking these comparative tests, the study seeks to gather insights into the relationship server placement and algorithm selection has with carbon emissions. Gathering this information would contribute to the broader goal of reducing the environmental footprint of machine learning processes.

3.1 Literature Review

Throughout the research and developmental stage, literature review was systematically done. A comprehensive review of existing literature, research papers, articles, and relevant sources were processes in order to gain a deep understanding of the topic, prior work done within the topic, and determine the path forward. This helped in identifying existing solutions, best practices, and in developing the research questions pertaining to the problem statement. This continuous review was especially helpful as the product was made in a non-group setting and thus had a very limited period of pure research at the beginning.

In the context of software development, research methodology refers to the systematic approach used to conduct research and gather evidence in order to solve a specific problem. Due to the width of the application, this approach has been comprehensive. For this project, *literature review* has been a part of the ongoing process, but especially leading up to the research question. Due to the research question revolving around the creation of a platform and a separate module, the strategy of *design and creation* followed. To explore the development of the application, data needed to be gather throughout and after the development and deployment of the project. To gather this data, the methods *interviews* with professionals and*questionnaires* with end-users were utilized. The interviews gave the developmental process important feedback and adjustments. The questioneers (user-testing), provided great feedback

on the deployed result. In addition, *evaluation* was utilized as a method. This materialized itself as a grand testing of the application, which involved gathering data on emissions based on regions of energy-grids. Combined, the 3 methods for gathering data resulted in a combination of quantitative and qualitative feedback. Qualitative data is non-numerical, descriptive and subjective in nature. This type of data aims to capture rich, in-depth insights, opinions, experiences, and interpretations from participants. Quantitative data is numerical data and based on measurable variables. This data focuses on gathering objective information and statistical patterns, allowing for statistical analysis and generalization.

Figure 5: Illustration of the research methods used throughout development and deployment. The marked boxes are - to a varying extent - present in this thesis.

Source: [34]

3.1.1 Optimization

The pre-project plan aimed at using an evolutionary optimization algorithm for training and tuning hyperparameters, but the research conducted phased out that option due to the discovery of Optuna, which is a framework for hyperparameter optimization. An optimization framework uses various search algorithms and pruners to explore and evaluate hyperparameter combinations, aiming to find the best-performing configuration for a given machine learning model. Using Optuna in favor of a singular evolutionary optimization algorithm was preferable, as it would allow the optimization-module to consist of many mainstream, effective optimization algorithms for the users to choose from. It seemed reasonable that this would increasing the chance that an algorithm might fit with a users specific needs for optimization. Optuna supports optimization algorithms such as Random Search, Tree-structured Parzen Estimator, Covariance Matrix Adaptation Evolution Strategy, and evolutionary algorithms, such as NSGA-II. All these are described in Section 2.3. Additionally, the research highlighted the necessity of reliable algorithm implementations, leading to the discovery of Stable-Baselines3. Stable-Baselines3 is a popular Python library that provides a collection of state-of-the-art reinforcement learning algorithms, and are known for their algorithm implementations being reliable and well-established. Furthermore, the research identified the requirement for suitable training environments to generate models using the chosen algorithms. In this regard, OpenAI's Gym was found to be a a good option, due to its popularity and reputation within the field, as well as its standardized interface. Gym is Python library that provides a collection of environments for developing and testing reinforcement learning (RL) algorithms

3.1.2 Calculating emissions

The focus of the vision document on CO2-tracking is a direct reflection of the problem statements focus on CO2 emissions within machine learning. Specifically, the problem statement specifies the need to create an environmental profile for reinforcement learning projects based on their environmental impact. Therefore, parts of the initial literature review was spent searching for an applicable solution. In the end, CodeCarbon seemed a feature-rich option that would provide the application with the tools required to track and measure the carbon emissions associated with the optimization process.

3.2 Developmental Methodology

While the traditional Agile methodologies are often associated with teams, the principles of iteration, collaboration, and adaptability can still be applied by individuals working independently. The process did not strictly align with a specific Agile methodology like Scrum or Kanban, but it did exhibit key characteristics of iterative development and continuous improvement. The developmental process consisted of iterative design where feedback was incorporated through interviews and user tests, and where necessary adjustments were made based on the insights gained. The process was administrated through a Gantt chart, which was used as a visual tool for the project. Gantt is used in traditional project management to plan and schedule tasks by providing a timeline view of the project. The timeline shows the start and end dates of each task. Gantt charts are useful for visualizing the overall project schedule and tracking progress but are not associated with agile methodologies. Therefore, the development methodology was a mix between agile and traditional development, which is perhaps not unexpected when development was conducted by a singular person. As the process went along, new details were added in order to divide the process into smaller, more managable tasks. When the project had been developed and deployed, the chart had been successfully completed. However, the tasks were often not done or started at the time that had been expected.

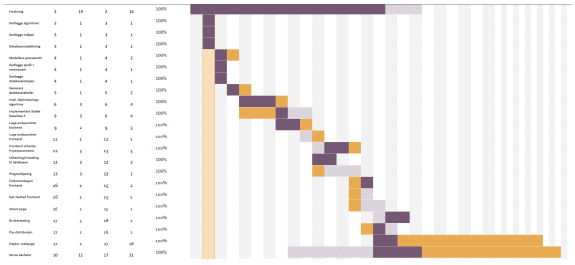


Figure 6: The Gantt-chart at the very end of the project.

Overall, the approach can be classified as an individual, iterative development process that valued feedback, adaptation, and incremental improvement.

3.3 Planning stage

During the planning stage, the application's individual components were modeled, sketched, and documented. These components are the optimization module, database, REST API, and the frontend. This process used insights from the literature review and previous knowledge. The goal was to establish a clear understanding of the system's structure and logic, such that this could be used for supporting the subsequent development phase. This stage a solid foundation for the overall application.

3.3.1 User-centered design

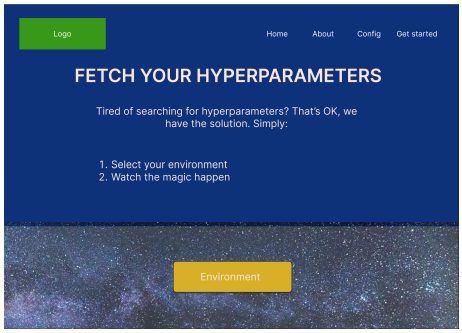
The first stage of designing revolved around creating prototypes, which were designed in the form of wireframes. The design process utilized the iterative design approach specified in Section 2.5. This section can be considered the planning stage for the iterative design. The Requirement documentation was created as a first step in this process, with user-stories being central to the document. The user stories and vision-document build on each other, with both explicitly stating what users might need the application for. The wireframes resembled the expected outlook of the web page, and were created with user-centered-design in mind. This means that the wireframes were created with the user-stories and vision document in mind, with the goal of meeting the users needs and desires. In addition, Gestalt principles were utilized when placing the components on the pages. Examples of principles used were:

- Proximity: The about-page's wireframe features 3 text-boxes; each with their own image in close proximity.
- Symmetry: The page displaying algorithms and runs was made in a way such that the two boxes would be symmetrically aligned over the central y-axis of the page. In addition, the ”environment”-button of the home-page was set in the very center.
- Similarity: The background and navigation-bar is the same on every page. Another example is that the runs displayed on the algorithm -and runs page are made with no grid. However, due to their similar horizontal structure, each row is intended to be seen as an individual component due to the similarity in row-structure.

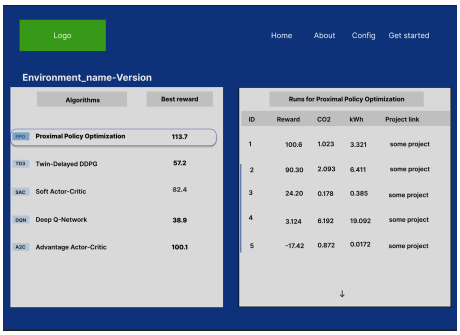
Following is the entire application as it was modelled before development had started.

For the homepage shown in Figure 7a, the button ”Environments” is thought to function as a drop-down menu. This drop-down menu displays the environments that have optimized runs in the database. After selecting an environment from the drop-down menu, the user is routed away from the homepage and onto the selection-page shown in figure 7b. On this page, the user has to select one of the algorithms in the left-hand box. Upon selection, the right-hand box appears. This box displays all the runs available for the environment x algorithm combination.

After the user has selected one of the right-hand ”runs” from Figure 7b on the last page, the user is directed onto a page that displays all the information for the selected run. This information is modelled in figure 8a. This includes environmental



(a) The Home page.



(b) The Selection page.

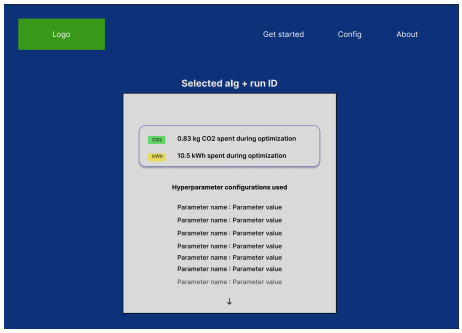
Figure 7: Figma prototypes displaying the Home -and Algorithm Selection page as they were modelled in the planning stage.

parameters, such as CO2-emissions and consumed energy throughout the optimization process, as well as the best hyperparameter configurations for the run. This page is the final destination for the user-flow associated with the fetching of hyperparameters. The About Page shown in figure 8b was designed to provide users with a clear understanding of the motivation behind developing the Hyperfetch application. It aims to highlight the key benefits and features of the application, focusing on three main aspects:

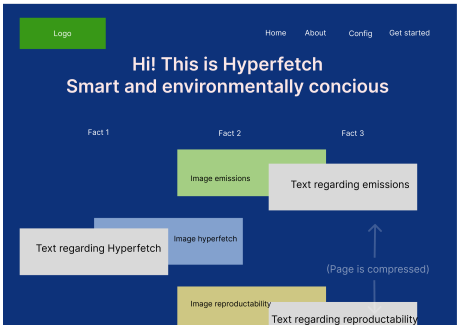
Collective Emissions Reduction: This section explains how the Hyperfetch application contributes to reducing collective emissions by optimizing hyperparameters for reinforcement learning models.

Enhanced Reproducibility: This section explains how the application supports researchers in achieving greater reproducibility of their work.

Ease of Use for Beginners: This section emphasizes that Hyperfetch is designed to simplify the process of utilizing reinforcement learning. It highlights how the application offers beginners a user-friendly interface and readily available pre-configured hyperparameters.



(a) The Run page.

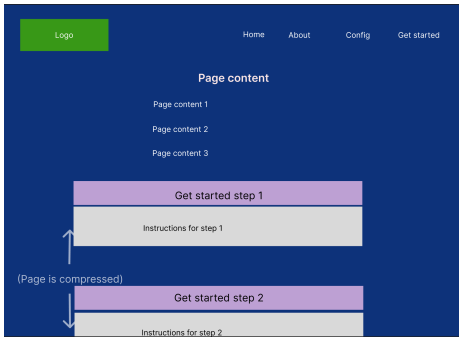


(b) The About page.

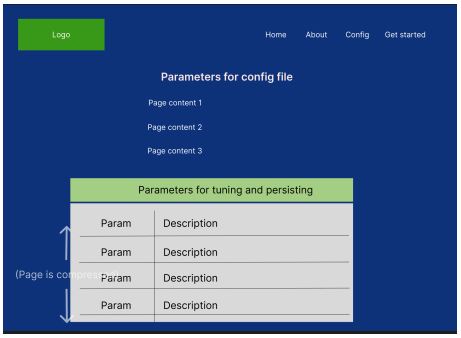
Figure 8: Figma prototypes displaying the Run -and About page as they were modelled in the planning stage.

The Get Started Page shown in Figure 9a serves as a guide for users who are new to the Hyperfetch application. Its primary objective is to assist users in downloading and utilizing the optimization module correctly. The page provides step-by-step instructions on how to set up the module, configure the necessary parameters, and initiate the hyperparameter optimization process. In addition, the page explains to

the user how to use the website to fetch hyperparameters, such that the user can get started properly with all the functionalities of the system. The Config Page shown in figure 9b plays an important role in empowering users to make the most of the Hyperfetch application. This page serves as a central hub for all configurable parameters available in the optimization module. It provides explanations and guidelines for each parameter, which enables users to make informed choices based on their own specific requirements.



(a) The Get Started page



(b) The Config page

Figure 9: Figma prototypes displaying the Get Started -and Config page as they were modelled in the planning stage.

3.3.2 Database structure

This section describes the modelling of the database. Initially, the database was modelled as a relational database.

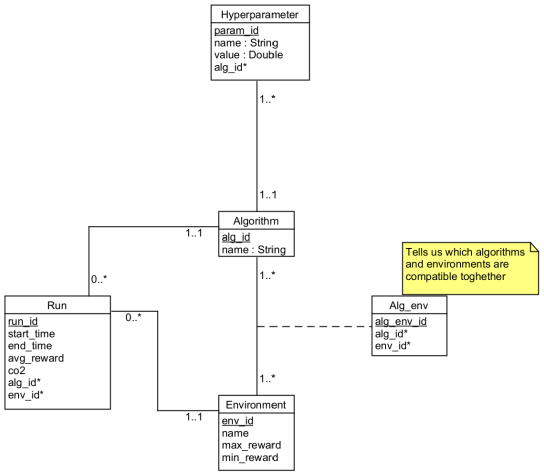


Figure 10: The intial plan was to create a relational database with the entity ”Run” as the central entity.

However, the plan was shortly ruled out in favor for a non-relational (NoSQL) solution. NoSQL databases allow for fast development due to their smaller demand for upfront planning. They are alos optimized for write-heavy workloads and handling large volumes of data. However, that is not why NoSQL ended up being chosen. As the structure of the relational database was being modelling, the need arose for

a different solution due to problems in modeling the hyperparameters. The Hyperparameters vary for each reinforcement learning algorithm, both in attributes and numbers. This causes a problem due to how relational databases demand structure, because storing key-value pairs is difficult when the keys are not generalizable. NoSQL databases, however, are designed to handle flexible data structures, and can store dictionaries (also known as maps or key-value pairs) as an attribute. These non-relational databases also support dynamic schemas, which means that fields can easily be added or removed from the database without needing to update the entire database schema. These features seemed befitting for the project, especially since it was likely that the database would receive new attributes at later stages due to the project not being easy to define at that level for this stage of development. Thus, the new non-relational database model became much more simple as the entire logic fit within a single document.

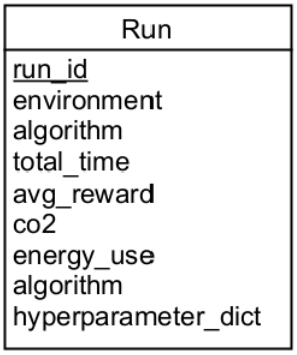


Figure 11: When switching the relational database out for a NoSQL database, the entirety of the entities could be persisted as a single document.

3.3.3 Resulting Stack

As the application as a whole will consist of a database, a module for optimization, a REST API, and a frontend, the project can be considered full stack. This section will cover the technologies used for each part of the application.

The Database The choice of a suitable database management system was an important decision in the project's development. After careful consideration, MongoDB, a popular NoSQL database, was selected to meet the project's requirements. The selection of MongoDB was motivated by its suitability for fast development, reduced upfront planning, and its flexibility in storing dictionaries. In addition, it supports dynamic schemas, which aligned well with the project.

Optimization module As the optimization module will take advantage of Optuna, Stable-Baselines3, CodeCarbon and OpenAI environments, which are all pip-installable Python packages; it was logical to write the module in Python. As Python is a multi-paradigm programming language, meaning it supports several programming styles, it makes for a good choice when aiming to produce Object oriented code. In general, Python is a preferred language for machine learning and artificial intelligence. This is because the finished package can be distributed to Python Package Index (PYPI) when finished.

REST API framework There were many options for selecting the Rest API framework. The requirements were that it had to be fast to provide the best user experience, reasonably efficient to implement and easy to containerize with Docker for deployment. A framework that checked of all the boxes was FastAPI. FastAPI supports type annotations to reduce boilerplate code, automatically generates interactive API documentation and is easy to deploy using Docker. The choice of FastAPI as the REST API framework was driven by its support for asynchronous programming. Synchronous APIs process requests sequentially, leading to longer response times and limiting the number of concurrent requests that can be handled. However, an asynchronous REST API, as implemented in FastAPI, allows multiple requests to be processed concurrently without waiting for each response before handling the next request. This asynchronous architecture can handle a larger number of requests simultaneously.

Frontend framework It would be of little value to have a REST API optimized for speed accompanied by a slow frontend. Therefore, the choice of frontend framework highly relied on speed and performance. The choice eventually fell on Vue.js. Vue is a lightweight framework that requires minimal setup and has a small footprint, making it ideal for building small to medium-sized applications, such as this one. In addition, Vue is designed with performance in mind. This is evident in its virtual Document Object Model (DOM) implementation, which makes rendering updates faster and more efficient than traditional DOM manipulation. Lastly, Vue has the option of creating reusable components, which will make the code cleaner and more concise.

3.4 Prototype

Developing the prototype consisted of creating a database, creating an optimization module, a REST API with all its logic, and then creating the frontend to display the website. The project was built from the bottom up, starting at the database, and ending at the website.

3.4.1 Database

The creation of the MongoDB database was not very complicated. Firstly, the MongoDB Community Server was downloaded from the official MongoDB website and installed on the computer. This installation included essential features such as the MongoDB server (*mongod*) and the MongoDB Compass. Next, a designated data directory was created to store MongoDB's data files. The MongoDB shell was launched by executing the "*mongo*" command in the terminal. The admin database was entered, and an admin user was set up. This also involved assigning relevant privileges, and used the following command:

```
db.createUser({ user: "admin_username", pwd: "admin_password",
    roles: [ { role: "userAdminAnyDatabase", db: "admin" } ] })
```

3.4.2 Optimization module

The Optimization Module holds many features. Firstly, it enables the user to provide input, granting the user the ability to select an environment and an algorithm to be trained within the specified environment. Additionally, users can specify parameters related to how they want the algorithm's hyperparameters to be tuned. Secondly, the module performs input validation to ensure that the provided user input is accurate and appropriate. Next, the module optimizes the hyperparameters for the configurations selected by the user. Furthermore, the module generates data pertaining to CO2 emissions and energy usage. This data helps quantify the environmental impact associated with the optimization process, and is essential for the problem statement. Finally, the module persists the tuned or optimized hyperparameters along with other relevant data for future reference and analysis.

To ensure proper code organization and facilitate the user input, validation, optimization, and persistence processes, an Object-Oriented Programming (OOP) paradigm was adopted for the module. In this paradigm, a Manager-instance was introduced as the central ”controller” or ”facade” class. The Manager-instance plays a pivotal role in coordinating the behavior of other objects within the system. It achieves this by calling methods on other objects, facilitating data transfer between them, and controlling the overall program flow.

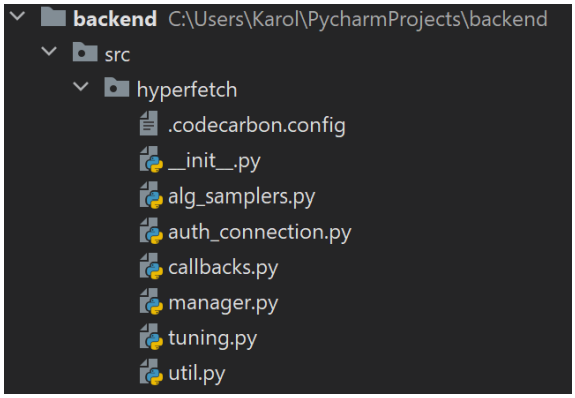


Figure 12: The project structure for the code that optimizes hyperparameters for Reinforcement Learning algorithms.

Designing a user input solution that accommodates users with varying levels of programming knowledge requires careful consideration. In this application, a configuration file was chosen as the medium for user input. Another medium that could have been chosen would have been input through command-line arguments. The decision to use a configuration file instead was driven by the desire to maintain flexibility. This also excludes the limitations that arise when handling numerous command-line arguments, especially as the application scales with additional parameters. Among the various types of configuration files available, YAML (YAML Ain't Markup Language) emerged as the most suitable choice. YAML, being a text-based format, offers readability and ease of writing. By utilizing a configuration file, users are enabled to more easily identify and rectify errors before running the application. In addition, YAML's support for a wide range of data types enhances the flexibility stated earlier.

The development of the validation -and optimization processes occurred in parallel, as determining the appropriate configurations to include in the validation process relied

on understanding which configurations were involved in the optimization. Initially, only a few parameters were configurable in the configuration file, but as the development process progressed, additional parameters were incrementally introduced based on evolving needs.

Validation Ensuring that users understand the functionality of the parameters they configure in the file is crucial, as these parameter values can significantly impact performance. The validation procedure serves the following purposes:

1. Detecting missing required parameters: If a required parameter is omitted, the validation process returns an error message and halts the execution.
2. Handling invalid parameter values: If an invalid value is assigned to a parameter, the Manager assigns a default, valid value instead. An informational message is returned, providing details on valid values and the default value assigned. The execution is allowed to continue.
3. Resolving conflicting parameter values: In cases where a value assigned to a parameter contradicts other values, the Manager assigns default values to the least critical parameters. For example, if the NSGAII-sampler is selected, no pruner can be chosen. In such cases, the Manager sets the Pruner parameter to ”None” if the user had assigned it a different value. An informational message is provided to communicate the changes made.

The validation process operates efficiently by reading the configuration file as a dictionary and validating the presence of each possible parameter using a switch-like function (multiple if-statements). If a non-mandatory parameter is absent, the Manager adds this parameter to the YAML file, and informs the user of the change by logging it in the terminal. This informs the user that the parameter is needed to run the optimization, and that a default value was assigned to it. By adding it to the YAML file for the user, the user will be able to visualize the parameters that were used by the application in the previous run, and alter them later on if they want to. In addition, the application will not log that the parameter is missing, as it is already there the next time the user runs the YAML file.

For validation, the switch-like approach is chosen due to its favorable algorithmic time-complexity compared to other options.

Option 1 (Chosen): The validation function follows these steps with their respective time complexities:

1. **O(n)**: Reading the *n* parameters from the configuration file into a dictionary.
2. *n* if-statements: Each if-statement checks for the existence of a specific parameter in the dictionary.
    - **O(1)**: If a required parameter is missing, an error message is returned.
    - **O(1) + O(1)**: If a non-required parameter is missing, it is inserted into the configuration file with a default value.

Total complexity: O(n) + O(1) + O(1) = **O(n)**

Option 2: (Not chosen) The validation function follows these steps with their respective time complexities:

1. **O(n)** Reading the *n* parameters from the configuration file into a dictionary (*a*).
2. Creating a list (*b*) containing all the implemented parameters.
3. **O(n)** A for loop that iterates over the list *b*
4. Each iteration includes an if-statement that checks if the current value exists as a key in the dictionary *b* (**O(1)**).
    - **O(1)**: If the current value is a required parameter, an error message is returned.
    - **O(1)**: If a non-required parameter is missing, it is added to the configuration file with a default value.

Total complexity: O(n) + O(n) + O(1) + O(1) + O(1) = **O(2n)**

Based on the time complexities, Option 1 is chosen as it offers a more efficient solution with a complexity of O(n), compared to Option 2 with a complexity of O(2n).

Optimization After validating user input, the Manager proceeds to create the environment using the values from the configuration file. This involves tasks such as stacking frames for vectorized environments, running environments in parallel, and setting a random seed for the Random Number Generator (RAG). Random number generators (RNGs) are algorithms or programs that produce a sequence of seemingly random numbers. A random seed is a starting point or an initial value used in an RNG. An RNG is used to introduce randomness and variability into the environment's behavior, whilst a random seed is used to ensure that the same sequence of random numbers is used across different training runs in the environment. To ensure reusability, a function was developed specifically for environment creation. The environment is what the Agent will interact with, as specified in section 2.2.2. As specified in the literature review, environment will created using Gym for this project. Gym is a toolkit for developing and comparing reinforcement learning algorithms. It provides a diverse suite of environments that range from easy-to-use, simulated games, to more complex, real-world scenarios. Gym is developed by OpenAI, which is a research company that focuses on advancing artificial intelligence in a safe and beneficial way.

Since hyperparameters - as mentioned earlier - are algorithm-specific, a separate file was dedicated to holding dictionaries of hyperparameters. The Manager retrieves the relevant hyperparameters based on the chosen algorithm specified by the user in the configuration file.

To facilitate optimization with Optuna, the environment creation and hyperparameter retrieval were integrated into an objective function. An objective function defines the optimization problem, taking a set of hyperparameters as input and returning a score representing the quality of the solution. In this project, any rewards deemed unworthy by the configured pruner are not returned.

To tie everything together, the Manager implements a method called ”run” that orchestrates the optimization process. The run method handles the creation of the Optuna study, which takes the objective function, the number of parallel jobs, and the number of trials as parameters. The study is the entity optimized by Optuna, and the

objective function is executed for each trial. If the objective function does not yield a reward that is good enough, the trial is pruned. Additionally, the run method incorporates functionality for early stopping, allowing optimization to end once a user-defined reward threshold is reached. Custom callbacks imported from a separate script are utilized for evaluating trials and stopping the optimization process accordingly. The reward threshold, like other user-defined values, resides in the configuration file.

Persistence After optimization, the tuned hyperparameters had to persisted. Considering the preferences of machine learning practitioners who often prefer managing dependencies within their own projects, it was decided to incorporate the persistence functionality within the module itself. This approach allows practitioners to easily download the package into their project and configure it in a single file, rather than navigating external websites and dealing with multiple parameter text fields. As a result, an additional function was added to the Manager. This function is responsible for delivering a JSON object containing the best-performing hyperparameters, along with the average reward achieved by those hyperparameters, to the MongoDB instance. Other relevant parameters from the configuration file are also included in the JSON file.

During the implementation of this method, it was discovered that there was no existing mechanism in the database structure to connect a persisted run with the corresponding project. Connecting hyperparameters to specific projects is an important feature for increasing the reproducibility of results. In response to this discovery, modifications were made to the configuration file and the validation method, such that users could enter their project name and link within the YAML file. At this stage, persisting a run would result in a database document structured as shown in Figure 13

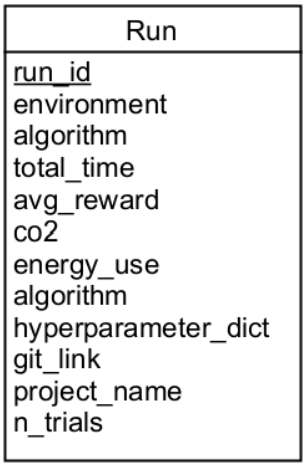


Figure 13: The structure for documents being persisted to the database after having made alterations to accommodate for connecting the run to existing projects.

Emissions and energy usage A key aspect of this project is to quantify the CO2 emissions associated with the process of optimizing hyperparameters. The design of the database includes fields to store CO2 emissions and energy usage values. To track these features, the CodeCarbon emission-tracker was integrated into the module.

CodeCarbon collects data on computer energy consumption, and converts energy consumption into carbon emissions. This is accomplished by gathering information about

the energy consumption of the CPU and GPU, as well as the location of the computer or server. Different CPUs and GPUs have varying energy consumption rates, and the location is considered as it is assumed that energy is sourced from the nearest energy grid from the computer or server running the optimization. The data CodeCarbon generates is entered into a Comma-Separated Values (CSV) file. This data consists of the values for the previously mentioned factors (GPU, CPU, location), as well as relevant cloud provider data is used. Incorporating this CSV data into the persistence method enhances the comparative analysis of runs that can be found on future website, thereby improving the accuracy of result reproduction. Given the non-relational nature of the database, integrating these features into the persistence process proves to be a time-efficient approach. As a result, the revised document structure, as seen in Figure 14 contains more parameters.

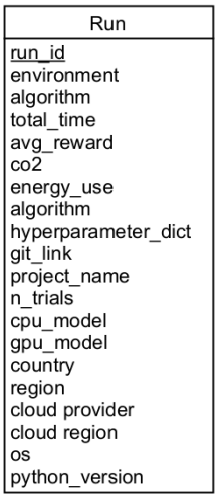


Figure 14: The structure for documents being persisted to the database after having made alterations to accommodate for more environmental parameters.

Testing The module underwent incremental testing throughout its development, allowing for real-time identification of issues. With the addition of new features, such as new configurable parameters in the YAML file, the module was repeatedly executed for short optimization runs to target and address any arising errors.

To ensure comprehensive testing of the modules validation of user input, a final test was conducted. This test involved performing optimizations using all possible combinations of environment types, algorithms, samplers, and tuners, while keeping other configuration parameters set to their default values. Despite the extensive nature of this testing method, it was feasable, as there were relatively small numbers of options available for each parameter-type. The practical implementation of this test involved a nested 4-fold for-loop iterating over environments, algorithms, samplers, and tuners. This nested for-loop was executed twice; once for discrete environments and corresponding algorithms, and once for continuous environments. This exhaustive testing process, with a time complexity of $O(2n^4)$, required a total of 34 hours to complete.

3.4.3 Developing the REST API

The REST API was developed directly after the optimization module. Documentation-Driven Development (DDD) was utilized as a method to drive the process. This method involves starting the development process by creating thorough API documentation, before following that up with Test-Driven Development (TDD) and then starting to code. This process was incremental, meaning that the API documentation was used for guiding the test-writing, and that the coding was guided by the test-writing. As new features and demands were revealed, the documentation and tests were improved, followed by the code. While this might seem like an unnecessary step, it helped keep the progress on track and avoided lengthy detours into unplanned terrain. In summary, the DDD approach helped ensure that the API was well-documented from the start, and that the development process was focused on meeting the documented requirements. For further details on the content of the API documentation, see the system documentation's Section 6 in the appendix.

The process of using DDD and TDD was as follows. To start with, A DDD file was created, defining endpoints for fetching environments, algorithms, top-performing runs, selected runs, creating runs, and deleting runs. Following the principle of Test-Driven Development (TDD), a test was written for each endpoint. These tests were initially designed to fail and later passed as the code became functional. Once the tests were in place, the endpoints themselves were implemented. As the server for the REST API, Uvicorn was used. Uvicorn is a lightweight and fast ASGI server that serves as a runtime for web applications written in Python. It provides high-performance asynchronous request handling, which makes it well-suited for serving APIs. A FastAPI router was also instantiated for routing between the endpoints. Finally, a database connection was established to facilitate testing. The connection proved to be functional, and after further code modifications, the tests were passing at this stage.

Although the application was functional at this point, further enhancements were needed to incorporate proper exception handling and achieve a more modular architecture. Improvements were made to enhance the robustness and modularity of the application. First, the Domain-Driven Design (DDD) was updated to incorporate error-handling. Logical error names were added to the tests to provide clear and descriptive error messages. To handle these errors effectively, a file containing custom exceptions was created. These exceptions were imported and integrated into the router endpoints, ensuring proper and readable error-handling. Once the tests were passing, the focus shifted towards the creation of the Data Access Layer (DAL). The DAL served as an intermediary layer between the persistence logic and the upper layer, which was the FastAPI Router. This architectural decision was made to adhere to object-oriented principles such as encapsulation and modularity.

The app at this point was functional, but not secure. The next phase of development focused on enhancing the security aspects of the application. To achieve this, Pydantic models were introduced into the tests. These models were also designed using object-oriented principles. The Pydantic models played an important role in defining, validating, and serializing/deserializing data within the application. By validating incoming request data and automatically parsing and serializing data, these models minimize the risk of data validation and sanitization attacks, such as Cross-site scripting (XSS) and SQL injection attacks. In addition to the implementation of Pydantic models, endpoints related to creating and deleting a run were removed from the ap-

plication. Although these endpoints had been initially included, they were found to be unnecessary for the webpage and increased the attack surface of the application. Even if the endpoints were not actively utilized, their accessibility posed a potential safety risk. Therefore, to minimize potential vulnerabilities, these endpoints were eliminated. Once the tests were successfully passing, Cross-Origin Resource Sharing (CORS) middleware was integrated into the application. This middleware IS responsible for facilitating communication between the future and the API. As the frontend was not up at this point, CORS could not be properly configuring CORS yet, but it would not prove hard to add the remaining frontend address to the middleware's empty list of allowed URLs.

3.4.4 Developing the website

The website was developed last, directly after the FastAPI backend. As a base for prototype-development, the Wireframes presented in Section 3.3.1 were used. This is the second stage of the iterative design that the website has gone through. After this stage, the prototype will be functional and ready for the third stage, which is testing.

During the development of the web server, the website was created following a systematic order of operations. To establish a connection with the backend, the Service layer was implemented first. This layer consisted of JavaScript functions that utilized Axios to make HTTP requests to the API endpoints. It served as the bridge between the frontend and the backend. Next, the Store was developed using Vuex. This involved defining methods and attributes for storing state, actions, mutations, and getters. The Store interacted with the Service layer to fetch data from the API, handling the response by modifying its state or returning appropriate error messages. The basic views were then created to serve as containers for the various components. In Vue.js, a view refers to a component that represents a specific page or section of a web application. Although initially empty, these views provided a structural foundation for further development. To enable navigation between different views, the Vue Router was implemented. This routing mechanism facilitated seamless switching between different sections of the website. To enhance the visual aesthetics of the website, a logo and smaller images such as symbols and algorithmic abbreviations with background coloring were created. To ensure proper placement and timing of images within grids, renderers were developed. A renderer is a mechanism or component that is responsible for taking data and rendering it into the appropriate format for display on a web page. The goal in doing this was to maintain maintained code cohesion by separating the image rendering logic from the main component code.

The landing page, designed as the homepage, was the first fully developed view. It played an important role as the starting point from which component logic branched out.

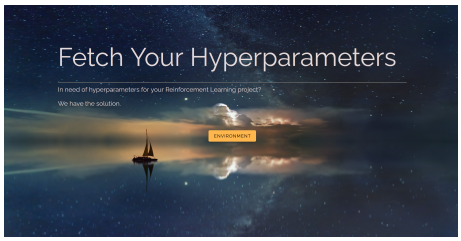
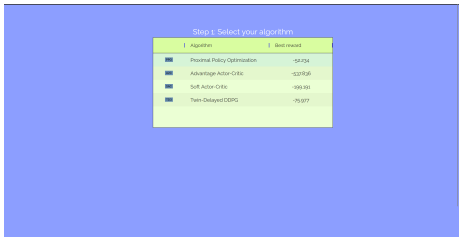
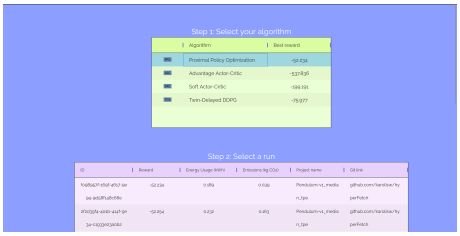


Figure 15: Home page for the prototype

Creating the landing page first proved beneficial, as it allowed for easy validation and testing of other components and their functionality when navigating to them from the homepage. Subsequently, the individual components were designed and implemented. These included displays for algorithms and their associated runs, a view for displaying detailed information about a selected run, and a navigation pane. After selecting an environment, the page shifts onto the algorithms, as shown in Figure 16a. An example-set of algorithms is shown in Figure 16a. Renderers were used to render each algorithm's abbreviations as images in the left column. After selecting an algorithm, the optimization runs conducted using the selected algorithm appears as shown in Figure 16b.



(a) Algorithms available



(b) Runs for a selected algorithm

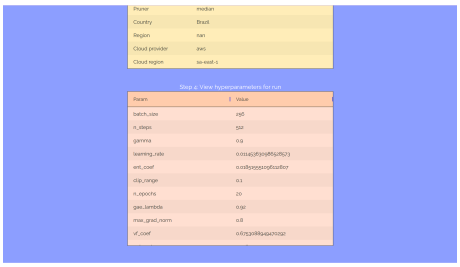
Figure 16: Upon selecting an environment from the Landing page, the user is presented with algorithms to choose from. The user selects PPO, and the runs conducted using the selected environment and PPO are revealed.

Upon selecting a run, the run is loaded onto the page. Renderers were used to display the icons here as well. The page is designed to be compartmentalized into three different components. The first component is the reward and emission-profile for the run. These parameters are presented first, as they are the factors that the users are thought to use when selecting hyperparameters. After that, the second box highlights other parameters relevant to the tuning of the hyperparameters, such as server location, hardware, software, pruner and sampler. Lastly, the users that are content with the results and configurations used may scroll down to reveal the hyperparameters, as seen in Figure 17b.

Lastly, the supporting views were created. These views are not part of the user-flow related to fetching and comparing tuned hyperparameters, but are there to support the user. The development of these supporting views involved careful consideration of the target audience and their specific needs. User-centric design principles, such as the gestalt principles described in the planning stage in section 3.3.1, were applied to create intuitive pages and to convey the information to the users.

While the supporting views differ in function from the rest of the application, a conscious decision was made to maintain visual cohesion with the other views. This was

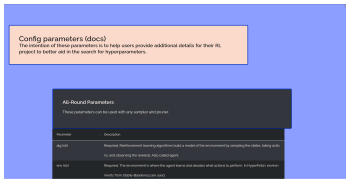
(a) The selected run - upper page



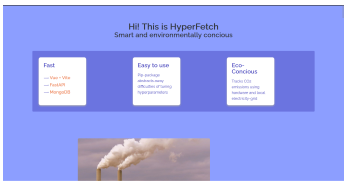
(b) The selected run - lower page

Figure 17: Upon selecting a run, the user is presented with the data for the selected run. This involves average reward, CO2 emitted, hyperparameters and more.

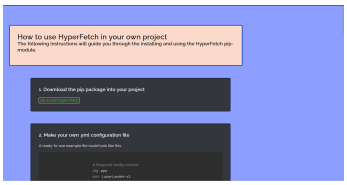
achieved by utilizing the same color palette and overall box-style layout as the main user-flow views related to fetching hyperparameters. In particular, the supporting views share the same background color as the other views that are part of the regular user-flow. The primary color chosen for the application was blue. Blue was selected based on its associations with trust, reliability, and professionalism [35], which align well with the goals for the project.



(a) Config page



(b) About page



(c) Get started page

Figure 18: The supporting views of the application. These views are created to support the user experience of the application by granting the users the resources to understand how to use the website and the optimization module.

By following this systematic approach, the website was progressively built, resulting in a functional website-prototype that was ready for testing.

3.5 Testing

After the previous step, the full stack application functioned as a Minimum Viable Product (MVP). As the third stage of the iterative design of this website, testing had to be conducted. To get feedback, suggestions, and recommendations for improving the prototype, interviews with professionals and end-users were conducted. Because professionals in the field can provide advice and insights based on their knowledge and experience, their input can bring a broader perspective, and provide guidance based on industry standards and best practices. By involving users in the testing process, insights can be gained into their needs, preferences, and usability challenges. Gaining these insights into usability, user experience, and the real-world perception of the website was important when design with a user-centered approach. By combining professional interviews and user testing, design choices were tested and great feedback was received. This helped in making improvements, such that the prototype could be ensured to be on the right track before moving forward with further development and eventual deployment.

Interviews were conducted in person, and provided a valuable methodology for ex-

ploring the topics of interface design and user-centered design in detail with the professionals. The personalized nature of interviews allowed for clarification of complex topics such as gestalt principles, as well as users needs.

User testing was conducted through questionnaires, which is a methodology for obtaining feedback from end-users. This methodology was used in order to assess the usability, effectiveness, and overall user experience of the website and optimization module. The use of these questionnaires were to ensure consistency in data collection, such that the data could be analyzed easier.

The following parts will describe the process pertaining to interviews and user-testing.

3.5.1 Interview 1

During the interview with an expert in the field of interaction design, feedback was obtained regarding the web interface. The expert highlighted specific gestalt principles that were lacking in the design. *Continuity*, which refers to the smooth flow and absence of abrupt changes, was identified as an area for improvement. Additionally, the principle of *common-fate*, where elements that belong together should move together, was emphasized. Other suggestions for enhancement were also provided, including the need to center the home-page button in the middle of the yellow stream to achieve symmetry. Symmetry is a Gestalt Principle, that had been implemented from the start, but one that had not been achieved according to the experts standard. The expert also recommended making the logo the pathway to the home page instead of having a separate home button as had been the case with the navigation bar from the prototype.

Figure 19: The navigation bar that was created as part of the Prototype.

Reordering the navigation bar's items to create a more logical user-flow was suggested as well. The expert explained that the eyes follow an F-pattern, where they start looking at the upper left (the logo) and move towards the right. After that, the eyes move back to start, move down, and repeat the movement until the bottom of the page is reached. Therefore, the "Get Started" tab was recommended to be moved to the left of the other tabs, as it was beneficial and logical for the viewer's eyes to see that tab first. Furthermore, the expert identified the importance of applying gestalt principles to ensure appropriate spacing between elements on the Config page and the Get Started page. It was also noted that the absence of a footer might lead users to believe that the page has not fully loaded. Regarding the About page, several pointers were given. It was recommended to maintain a controlled mess by placing boxes on the same side and in the same color, while ensuring images are of the same size. A Wireframe describing his intent with this suggestion can be seen in Figure 20.

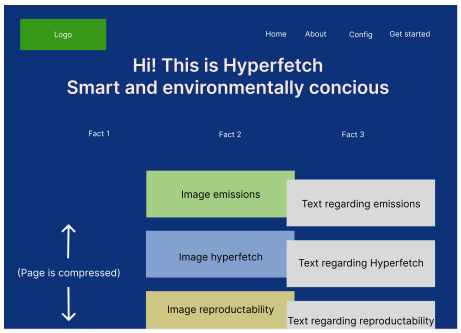


Figure 20: The revised About-page as suggested by an expert in interaction design.

Additionally, centering everything was discouraged, and the suggestion was made to remove the three horizontal boxes and present the text directly on the background for simplicity. The interview provided valuable qualitative feedback, contributing to the iterative improvement of the web interface.

3.5.2 Interview 2

The second interview was conducted directly after the first interview. This time, to provide a different perspective, the interviewee was a communication advisor. The communications advisor could provide insights into the expected user experience and response. Regarding the Config page, it was recommended to remove the beige title-boxes and instead include the title-box content into the boxes containing the configurations. This change aimed to improve user comprehension by emphasizing cohesive parts and aligning with natural reading patterns. Additionally, the need to enhance contrast for better readability on links presenting page content was highlighted. For the About page, specific suggestions were provided. Firstly, the small segment showcasing the stack, titled ”fast,” should either be positioned last among the initial segments or moved to the bottom of the page. This adjustment aimed to prevent users from being directed away from the main content. Secondly, it was suggested that the links within the ”fast” segment open in a new tab to indicate that the webpage remains the center of attention. Currently, external links open in the same tab as the webpage. Another recommended improvement was to have the scroll-up button persistently visible as users scroll up and down the page. Currently, the button only appears when the user reaches the very bottom of the page. The interview with the communication advisor also provided qualitative feedback.

3.5.3 Improvements

Following the feedback received from the professionals during interviews, specific areas for improvement in the prototype were identified. Due to limited resources and a focus on developing and deploying a functional application, it was not possible to implement all suggested improvements within the given time frame. However, efforts were made to prioritize the feedback from the initial interviewee, while leaving the feedback from the second interviewee for future work. The insights and recommendations provided by the expert in interaction design were prioritized as the experts profession aligned more directly with the specific goals and requirements of the

application. However, feedback from both interviews were highly relevant. As the development team consists of a single person, resource allocation played a significant role in deciding to not improve on all the feedback.

One notable improvement was the reordering of elements in the About page as suggested by the expert. Another improvement is the implementation of the gestalt principle of ”common fate.” This principle was applied by assigning appropriate transitions to components that belonged together as a cohesive group. By doing so, the visual coherence and intuitiveness of the interface were enhanced. In addition, the navigation bar was edited, as recommended by the expert.

Figure 21: The improved navigation bar -created using feedback from an expert in interaction design.

A final example of improvement is the footer seen in Figure 22 that was added upon feedback that the page did not appear to have loaded fully without the presence of a footer. This is an example of the brain functioning in the way specified in the gestalt principle called *closure*. This principle states that when an object is incomplete or has missing parts, our brains fill in the gaps to create a complete shape.

Figure 22: The footer created after receiving feedback from an expert in interaction design about needing one.

These improvements, along with others, have contributed to the development of the final web interface.

3.5.4 User tests

Once the necessary improvements had been made based on the professional feedback, user tests were conducted. The goal of the user test was to provide additional insights on website improvements from actual end-users. User tests were conducted using a questionnaire. The spreadsheet showing questions asked and responses received can be found as an appendix. The feedback obtained through user tests were used to identify areas for improvement, validate the platform’s functionality, and guide future enhancements to ensure that Hyperfetch meets the needs and expectations of its users. Due to time constraints, this valuable feedback has been implemented in a sense, but not to its full extent. Results from the user tests can be found in in the Result Chapter.

3.6 Final Product

After incorporating feedback from both professionals and users, the process of deploying the application could commence. The developmental phase consisted of deploying the database to a MongoDB Atlas Shared cluster, the backend and frontend to Azure,

and the optimization module to PYPI. Azure is provided by Microsoft and offers a wide range of tools and services for building, deploying, and managing applications. Thus, an Azure account was created before deploying both instances. Using the account, an Azure resource group was created to hold the front -and backend as their own individual resources.

3.6.1 Publishing the optimization-module

A natural way to distribute a python package is by uploading it to the Python Package Index (PYPI). This makes it available for use by a wider audience. While the project could have been cloned and installed by each user, distributing the project ensures a simpler process regarding installation and dependency management for the user. To install a distributed Python module, the user simply has to type *"pip install <package-name>"*. The setup.py and pyproject.toml files are alternative configuration files for Python packaging, and they are designed to work with the newer Setuptools and pip toolchains. The benefit of using pyproject.toml is that it allows for defining all dependencies, build system, metadata, and other options in a single file. When deploying the Hyperfetch pip module, Toms Obvious Minimal Language (TOML) was used.

To distribute the Hyperfetch package, a systematic approach was followed to ensure a smooth and reliable dissemination process. First, a .TOML configuration file was created, containing essential metadata, required dependencies, and scripts necessary for importing and utilizing the package. The scripts are the functions that the users of the module call on to use it, after having installed it. The hyperfetch module contains scripts for running the module through the command line or inside a Python script. To provide users with detailed instructions and guidance, a thorough README file was crafted. This README document explained the intricacies of using the Hyperfetch package by providing examples of usage. To conduct the distribution, the Twine -and Build packages were used. Twine is a tool for publishing Python packages, and Build is a tool that, as the name suggests, builds the package. To ensure the integrity of the distribution files, a verification process was implemented. The command *"twine check dist/*"* was executed to validate the format of the distribution files, confirm the inclusion of all required files, and ensure the accuracy of the package metadata. Finally, the package was distributed using the command *"twine upload dist/*"*. This step involved entering PYPI login and password credentials to securely upload and publish the package to the designated repository. By following this comprehensive approach, the Hyperfetch package was effectively distributed, making it readily available to users [8].

3.6.2 Deploying the database

To deploy the local MongoDB instance to the shared Atlas cluster, the following steps were taken. First, a MongoDB Atlas account was created, and a new project and cluster were set up. An admin user with appropriate permissions was then configured for security. The cluster's IP access was adjusted to allow all IPs. Finally, the connection was established through MongoDB Compass using the cluster's URL, enabling visualization and data management. This deployment ensured a reliable and accessible database environment for the Hyperfetch application.

3.6.3 Deploying the REST API

The deployment process for the REST API involved a series of steps to ensure its successful deployment. First, a Dockerfile was created, specifying the required dependencies, defining the exposed port, setting the working directory, and configuring the server startup. Using the Dockerfile, a Docker image was built. A Docker image is an executable software package that includes everything needed to run a piece of software, including the code, runtime environment, system libraries, and dependencies. The Docker image containing the REST API was then pushed to DockerHub, which is a container registry for Docker images. Within the Azure Portal, the Deployment Center was accessed to retrieve the Docker image from DockerHub and deploy its contents as a Linux Web App.

During this deployment process, a security breach was identified as the application's Git repository had inadvertently exposed the URL for the database. This vulnerability falls under the category of an Insecure Direct Object Reference (IDOR), posing a risk of unauthorized access to the database and potential manipulation of sensitive data. To address this issue, the database was thoroughly reviewed to ensure data integrity, the administrator's password was changed, and a new URL was generated. The exposed URL was removed from the code. To prevent further exposure, the repository's privacy settings were adjusted to make it private while these measures were taken. To solve the issue, the URL was added to Azure Portal's configuration section for the deployed API, such that the environment variable could be read in to the deployed Docker image. This incident was a valuable lesson in reinforcing the importance of keeping sensitive information safe when developing, as well as deploying.

3.6.4 Deploying the Website

This section contains the process for deploying the website as a static web app. The choice to deploy the frontend as a static web app instead of as a regular web app, is that static web apps support modern front-end frameworks like Vue.js out of the box, which makes it easy to deploy and host the Vue app without the need for additional configuration. Alongside the deployment of the Azure Static Web Application a continuous integration and deployment pipeline (CI/CD) was utilized. To do this, a GitHub Actions workflow file has been incorporated into the project, enabling the automatic building and deployment of the Static Web App upon each push to the GitHub repository [36]. By automating the deployment process, the likelihood of manual errors during deployment is reduced, ensuring a more secure and efficient deployment workflow.

To deploy the Vue app as an Azure Static Web App, the Azure Portal was used. Since the Vue project utilized Vite, a build tool and development server, a Vite configuration file had to be appropriately set up to serve the application. Next, the frontend project was compiled into a deployment package which included all essential files, such as HTML, CSS, JavaScript, and image files. Compilation was achieved through the ”*npm install*” command, resulting in the packaging of files within a ”dist” folder. Within the web app settings, Vue.js was chosen as the build preset to ensure proper handling of the project. The previously created ”dist” folder, generated during the build step, was specified as the ”static content directory” for the web app, ensuring the appropriate files were served. Other configurations not worth mentioning were also configured.

Upon completing the configurations, the application was deployed. Finally, a URL for the Static Web App was provided by Azure, confirming the deployment.

3.6.5 Domain name

To improve the accessibility and reflect the content of the webpage, a new domain name was acquired for the Azure-hosted website. Considering cost-effectiveness, a domain name with the ".online" top-level domain (TLD) was chosen. Once the domain was purchased, DNS settings needed to be configured both with the domain provider and within the Azure portal.

Within the created account with the domain provider, a new DNS record of the type Canonical Record (CNAME) was added. CNAME records associate one domain name with another, eliminating the need to use IP addresses directly. For the CNAME record, "www" was set as the host name, the Azure Static Web App URL as the value, and a suitable time to live (TTL) value was set.

Figure 23: Configuring a CNAME record to point to the Azure-generated URL provided by an Azure Static Web App.

Within the Azure portal, the same configuration had to be done. Another CNAME record was added, specifying "www" as the hostname, and the purchased domain ("hyperfetch.online") as the value. After verification, the custom domain was added [7].

Following these configurations, the website became accessible through the new user-friendly domain name, enhancing the overall user experience.

3.6.6 Security

The deployed REST API and wbsite utilizes Hypertext Transfer Protocol Secure (HTTPS), which is an essential security measure for protecting the communication between the client (website) and server (REST API). HTTPS provides encryption and authentication, ensuring confidentiality and safeguarding against eavesdropping, packet sniffing, and session hijacking, which are commonly exploited in Man-In-the-Middle (MITM) Attacks. In contrast, regular HTTP does not protect against MITM attacks. Compliance with regulations such as the General Data Protection Regulation (GDPR) further mandates the use of HTTPS to protect sensitive user data. Therefore, ensuring the security of the web page is of paramount importance, especially when considering potential future user bases.

In addition to HTTPS and the Pydantic models, additional security measures have been implemented. This includes the use of custom middleware, which restricts access to endpoints exclusively for the website. This is achieved by validating the 'Origin' header. Middleware has also been implemented to verify the 'host' header, providing protection against HTTP Host Header attacks.

3.6.7 Bug Handling

After deploying the website, a bug was encountered during the interaction between the frontend and backend components. While both the frontend and backend were successfully deployed with assigned SSL/TLS certificates, an issue arose when the frontend made requests to the backend's API. The problem can be explained through the following flow:

1. The frontend sent a request to the backend over HTTPS using a URL like "https://backend/api" (simplified).
2. The backend's router received the request and responded with a 307 Temporary Redirect status code. This code indicates a temporary move of the requested resource to the URL specified in the Location header. However, the Location header contained an HTTP URL like "http://backend/api".
3. The frontend followed the specified location, requesting the same URL but over HTTP instead of HTTPS.
4. The backend received the request, fetched data from the database, and sent it back to the frontend over HTTP.
5. As a result, the browser issued a "Blocked loading mixed active content" message, preventing the frontend from receiving the data.

This issue was challenging to debug, as it involved the concept of mixed active content. Mixed active content is content that has access to all or parts of the Document Object Model (DOM) of the HTTPS page[37]. In this case, the mixed content was caused by the response being received through HTTP instead of HTTPS, which is considered insecure. The "Blocked loading mixed active content" message is triggered to protect against potential MITM attacks and such.

The reason for the unexpected redirect was eventually traced back to a bug in the built-in router of the FastAPI framework. It was discovered that the router had difficulties handling URLs with trailing slashes ("/"). The solution to this issue was to create a custom router that inherited the FastAPI router, make the required corrections to it, and use the custom router in the API instead. After deploying a new Docker image addressing this issue, the bug was successfully resolved.

3.7 Emission profiling

In the final stage of this chapter, the distributed optimization module underwent a comprehensive series of comparative tests to analyze the CO2 emissions generated during the optimization of hyperparameters. The tests aimed to assess the impact of different factors such as geographical location, algorithm selection, and cloud providers on the emissions. These tests were important in gaining insights into the environmental implications of the optimization process. To ensure the reliability and consistency of the results, all tests were executed twice, validating the accuracy of the outcomes. Additionally, the tests employed the same hardware configuration for fairness and comparability. To ensure comprehensive data and experimentation, each optimization test consisted of 1000 optuna trials. Ideally, virtual machines (VMs)

should have been used for these tests. However, due to their cost, I opted to use CO2 constants associated with each provider's regions [38]. All constants used are properly cited and credible. Nonetheless, to demonstrate the preferred method in case of funding, I created a VM using the existing Azure profile from the previous deployment in Section 3.6. Upon creation, this profile was granted some starting credits for free. These credits were used in creating the VM, which was then accessed via SSH (Secure Shell). Within the VM, I installed Python and the necessary prerequisites for the optimization module [8]. Following this, I installed Hyperfetch, created the required configuration file for optimization, and initiated the optimization process.

Furthermore, I moved the VM from one region to another in order to show the ease of this approach if funding had been available. The complete process is explained in the *VM Creation* appendix. The appendix illustrates this process using screenshots and some explanation. Additionally, the same process of VM creation and server-location migration is outlined for Google Cloud Platform (GCP) and Azure in the same file.

Following are the comparisons that were executed:

Country Comparison: This test compared the CO2 emissions generated by optimizing the hyperparameters in five different countries. These countries were from the continents of Europe, North America, Asia, and Oceania. Specifically, two countries from Europe were selected to investigate intra-continental differences.

Regional Comparison within the United States: To examine the impact on CO2 emissions between regional differences, the United States was selected. the optimization module performed optimizations within four different regions.

Cloud Provider Comparison: The emissions produced by the three biggest cloud providers, namely Google Cloud Platform (GCP), Azure, and Amazon Web Services (AWS), were compared. The optimization runs were conducted in the same geographical regions whenever possible, enabling a comprehensive evaluation of the environmental performance of each provider.

Algorithm Comparison: Alongside the already mentioned tests, emissions generated by three different algorithms were also compared. These algorithms were Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C), and Soft Actor-Critic (SAC). The objective was to investigate the potential impact of algorithm choice on resulting emissions.

4 Results

This chapter will present the results of the project. Results tied to the product, its features and the process will be given. The system documentation can be used as a supplementary resource.

4.1 Findings from Product testing

User tests were conducted as a last step before deploying the application. The full report can be found in the user testing spreadsheet. The proposed system, Hyperfetch, consists of two core components: a web page and an optimization module. The optimization module is designed for tuning and persisting hyperparameters, while the web page provides users with access to their previously optimized configurations.

The user testing conducted for Hyperfetch yielded several important findings. Participants expressed a favorable view of the separation between the application and the module, as Python is commonly used in machine learning applications. The users stated that this division allowed for efficient client-side processing of complex tasks. As seen in Figure 24, users also believed that the module would aid in recreating previous projects within reinforcement learning.

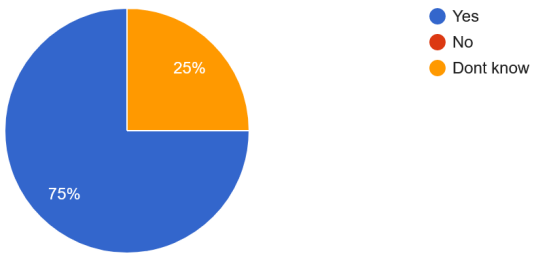


Figure 24: Pie chart visualizing the percentages of users who thought the module would aid in recreations of previous projects within reinforcement learning.

Only 5 out of the 9 testers tried to download the optimization module, as shown in 25. This is likely because they were given the choice to test one or both as the testing took some time. However, for those that did install it, some challenges were identified during the installation. Users on Windows and macOS encountered difficulties due to dependencies, specifically Swig and Box-2D. These issues have been addressed, and the module is now downloadable on macOS and Linux, although further testing on Windows is needed. In addition, users expressed the need for a way to fetch hyperparameters through the pip-module. They suggested having a lite-version of the web application's results available in the terminal, providing a more streamlined and convenient experience.

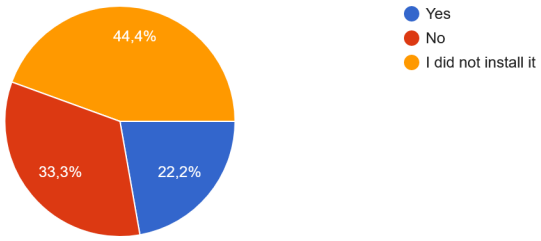


Figure 25: Pie chart visualizing the percentages of users who tried to install the optimization module during the user tests.

Feedback regarding the web page was generally positive, with users finding it clear and easy to navigate. However, some users experienced slow response times on the endpoint that fetches environments, which was subsequently resolved by enhancing the cloud computing power of the distributed REST API and optimizing the middleware.

When asked if the website lacked any information, 3/8 testers mentioned that it could benefit from additional content. They suggested improving the beginner guides by reducing repetition and including links to the source code (GitHub). They also suggested including links to the TOML file containing the specific dependencies for the Optimization module. The reasoning was for help with versioning and transparency in installing. This comment was likely in relation to the issues with Swig and Box-2D. Concerns were also raised about the website's compatibility with the optimization-library. Efforts have been made to update and align the documentation for both parts accordingly. Users also suggested including more information on the front page, drawing from the "About" page content, to provide a clearer overview of Hyperfetch's purpose.

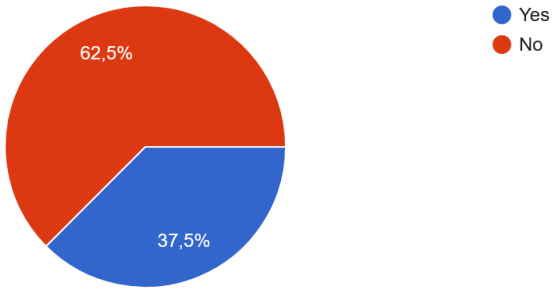


Figure 26: Pie chart visualizing the percentages of users who stated that the website held too little information.

Similarly, some users felt that the website had an excessive amount of information. They suggested making components like samplers and pruners collapsible, or even collapsed by default, to improve the overall user experience. This would allow users to focus on specific aspects without feeling overwhelmed.

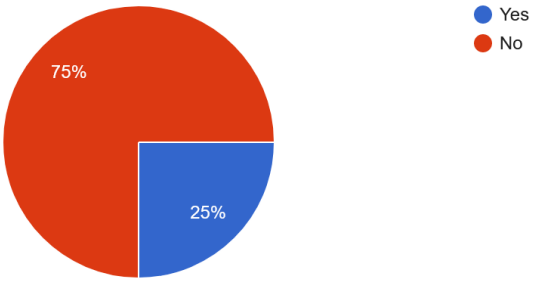


Figure 27: Pie chart visualizing the percentages of users who stated that the website held too much information.

Feedback also highlighted the readability of page-navigation links, with some users finding them difficult to read. The feedback provided pointed out that the orange text on a pinkish background should be changed to ensure compliance with accessibility guidelines. This feedback is a reiteration of what the second expert stated in the second interview from Section 3.5.2.

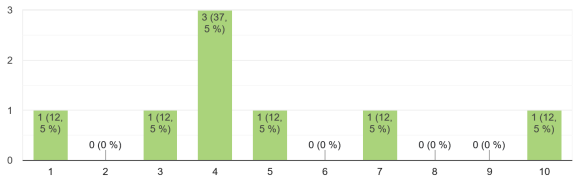


Figure 28: Bar chart visualizing if the users thought the links that displayed page-content were hard to read (10 being very hard to read).

When asked to provide an overall design rating on a scale of 1-10, the ratings presented in Figure 29 reflected a functional website. Participants appreciated the simplistic and neat design of the webpage, which aided their understanding of the system. They found the text to be well-written and easy to comprehend, and thought the design was seemed to target an audience already familiar with machine learning concepts. A user pointed this out, and stated that that is allowed for focused and concise information tailored to their needs. Another user with less experience within the field found the webpage initially confusing and intimidating for newcomers. After exploring the site and reading the ”About” page, the user wrote that they thought it became more logical and understandable.

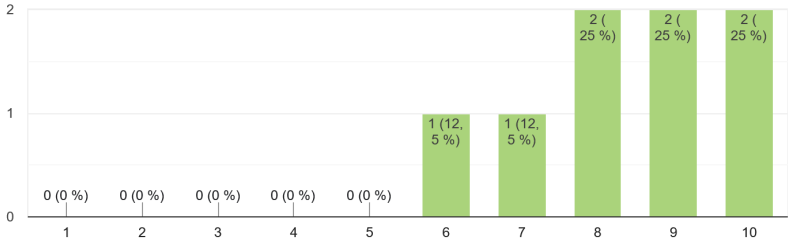


Figure 29: Bar chart visualizing the users’ overall thoughts on the design of the web page.

However, other design-related feedback highlighted issues with color combinations and the speed of animations. To enhance accessibility, participants recommended improving the color contrast, and increasing the speed of the animations. Recommendations were also made to optimize box sizing on larger monitors, and ensure accessibility compliance through tools like Firefox Developer Edition. Additional recommendations included enabling result sorting for parameters to facilitate accessing the ”top 10 configs” and addressing design issues such as providing a favicon and a more explanatory website title for better user recognition. Currently, the basic ”Vue + Vite” is the website title. These recommendations highlight areas for improvement within the website. It is apparent that the website currently faces challenges in effectively conveying its purpose and functionality to a diverse audience. This observation is supported by the findings presented in Figure 30, where users were asked about their level of understanding when using the website. Looking at this chart, some users did not grasp the concept of the web page.

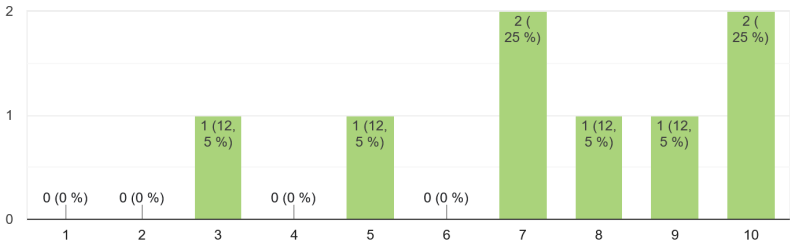


Figure 30: Bar chart visualizing the users’ opinions on whether or not they understood how to use the website. A score of 10 reveals a very good understanding.

4.2 Final Product

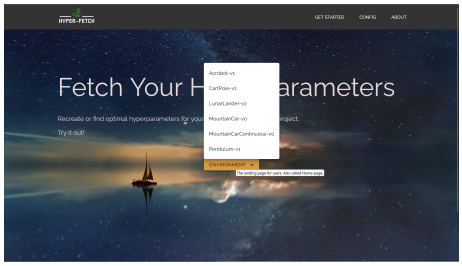
In this section, the final product will be discussed. All parts of the stack pertaining to the website, as well as the module are deployed and distributed. The Hyperfetch system provides a way for users to fetch hyperparameters online [7], as well as for users to post their own hyperarameters [8] using the module. The website displays hyperparameters, along with other parameters such as CO2-emissions, energy usage, time utilized, hardware, cloud-region and provider (if utilized), operating system, and python version. A user can download the pip module and run it, either to optimize hyperparameters and/or to post them to the website.

4.2.1 Website

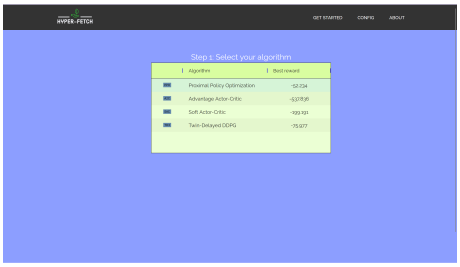
After gathering feedback on the web interface from both expert advice and end user testing, improvements were implemented, and the website was deployed[1]. The resulting website is presented in this section.

When the user enters the web page, they arrive at the landing page. The landing page holds a dropdown menu that appears when the user hovers the mouse over the button. In the example shown in figure 31, the user clicks on the ”Pendulum-v1” environment, which is based on the classic pendulum problem in control theory. This redirects the user to the algorithm selection, which displays the available algorithms

[1] https://www.hyperfetch.online/



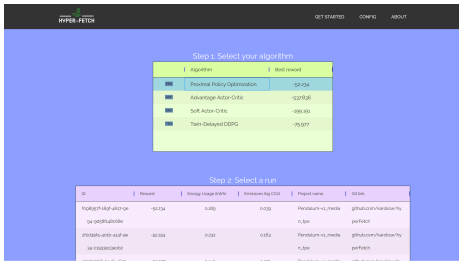
(a) Home page



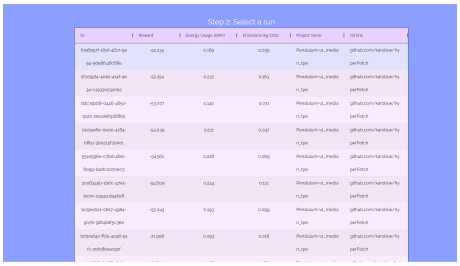
(b) Algorithm selection

Figure 31: The finished Home -and Algorithm selection page.

for the Pendulum-v1 environment. As is highlighted, the user hovers their mouse over Proximal Policy Optimization (PPO) as their selected algorithm.



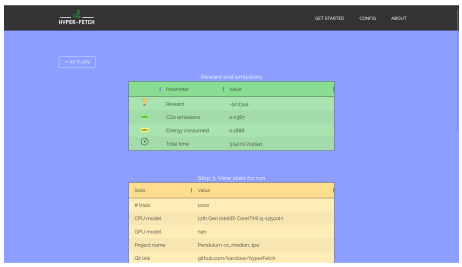
(a) Run selection activated



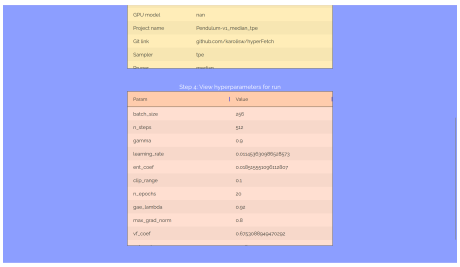
(b) The mouse hovers over the best run

Figure 32: Runs are activated upon algorithm click, and a run is selected

Upon selecting PPO as the algorithm, the runs that are conducted using Pendulum-v1 and PPO appear on the screen. These runs are ranged from best to worst. Pagination is applied, such that only ten runs are displayed. However, there is no option to see any runs other the ten first at this time. As seen in Figure 32, the runs appear, and the mouse hovers over the best performing run.



(a) The Selected run (top of the page)

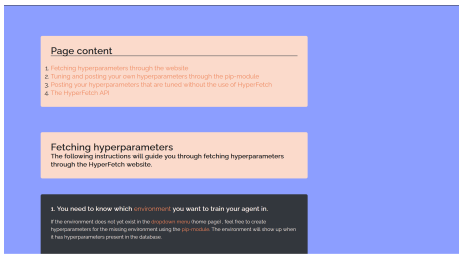


(b) The selected run (bottom).

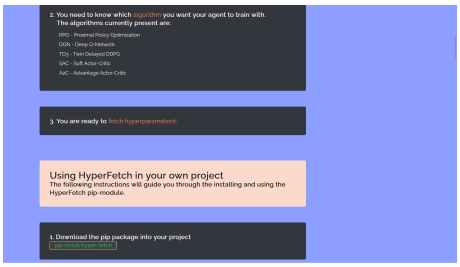
Figure 33: The selected run is displayed. The page contains three separate boxes in their own distinct colors. The last box contain the optimized hyperparameters.

Upon selecting the run that the mouse hovers over in Figure 32b, the user is redirected onto a page that displays hyperparameters, environmental parameters, hardware configurations, and software used for the tuning process. As can be seen in figure 33, the page is divided into the same three components as within the prototype. This remained due to receiving positive feedback on this page and the way it was sectioned.

For users that have not used the Hyperfetch web page, or optimization module before, the *Get Started page* provides a natural starting point. As get started is the natural



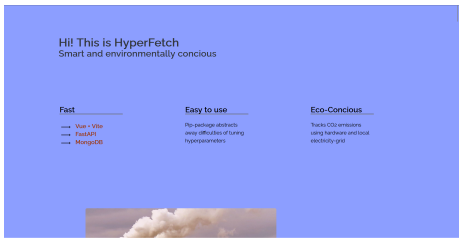
(a) Get Started page (at the top)



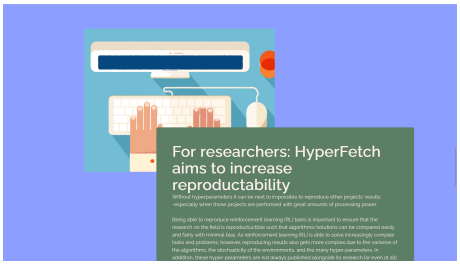
(b) Get Started page (step-by-step)

Figure 34: The Get Started page displays instructions for the user on how to use the web page and optimization module.

first step, the navigation- tab for the page is displayed furthest to the left on the navigation bar, as specified by the expert interview in section 3.5.1. The Get Started page, as displayed in Figure 34b, provides step-by-step instructions on how the user can utilize the website and the installable module.



(a) The top of the About page



(b) Format of About page content

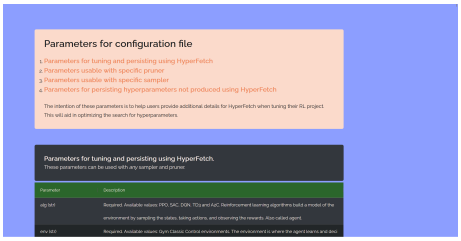
Figure 35: The about page explains the motivation and ulterior motive for the application as a whole.

The About page, as visualized in Figure 35, has been improved according to the feedback from the first interview. All sections of the page follow the same text -and image combination pattern as is shown in Figure 35b. In addition, the gestalt principle of closure has been applied by sectioning the three paragraphs shown in Figure 35a and removing the boxes that were originally there. Lastly, the Config page as visualized in figure 36, displays the available configurations for use within the configuration YAML-file that users have to provide before using the pip-module. This page received criticism from the expert about being to cluttered originally, and was sectioned further to avoid being messy. Due to the vast amount of configurable parameters, the headers were colored green to make the sectioning within the page more visible, and to avoid making the headers look like parameters.

4.2.2 Architecture

The resulting architecture when excluding the module, consists of two separate applications and a database. The optimization module is excluded from this section as it is a free-standing project. The website provides the GUI for users that want to fetch hyperparameters for their models.

In Figure 37, a user requests to enter https://hyperfetch.online. When a user types in a domain name in their web browser, their computer sends a Domain Name System



(a) Top of the Config page

(b) Some configurations

Figure 36: The Config page explains to the user which parameters are available for use within the configuration file that has to be supplied to the optimization module when using it.

(DNS) query to a DNS server in order to find the IP address associated with that domain name. The DNS system is organized in a hierarchical manner, the root DNS servers are at the top of the hierarchy, followed by top-level domain (TLD) servers (.com, .org, .net, etc.), and then authoritative DNS servers that hold the specific domain names. When a specific domain name like ”hyperfetch.online” is typed into the web browser, the browser needs to know the corresponding IP address to establish a connection with the web server hosting that website. DNS helps in this process by translating the domain name to its associated IP address, which is actually the one associated with the messy Azure-provided website URL. This process is illustrated through the dotted line. DNS returns the IP address, such that the user can make the request to the actual web page. The user then interacts with the web page such that the endpoint requests are made. This is illustrated through the bold line. Data is returned to the frontend and displayed for the user.

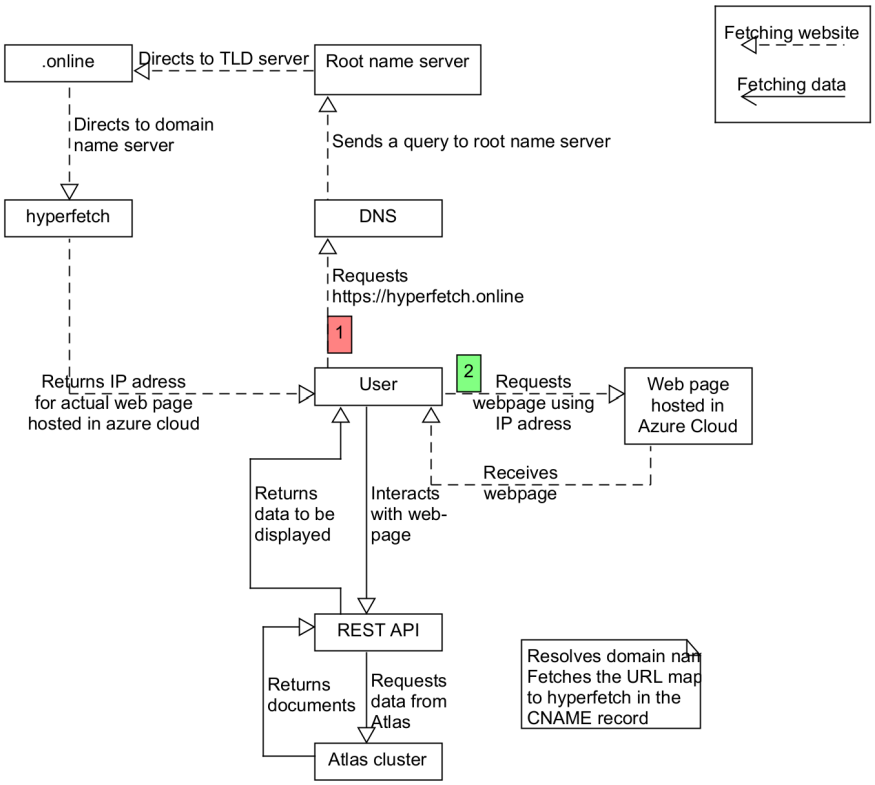


Figure 37: Illustration of the process that is activated when the user enters the web page.

4.2.3 Layers

Figure 38 displays the layered design of the web application, the server, and the database. The user interacts with the interface, causing the VueX store to reach out to the service and tell it to send a request over the correct endpoint. This call is sent to the REST APIs client. The client's middleware validates the request, before the client forwards the validated request to the router, which forwards it to the Data Access Layer (DAL). The DAL interact with the database through the use of input-sanitized Pydantic models. The database sends the correct information back to the DAL. The response is forwarded recursively back to the Vue-store, which handles and mutates the data, before returning it to the component that called on the store in the first place.

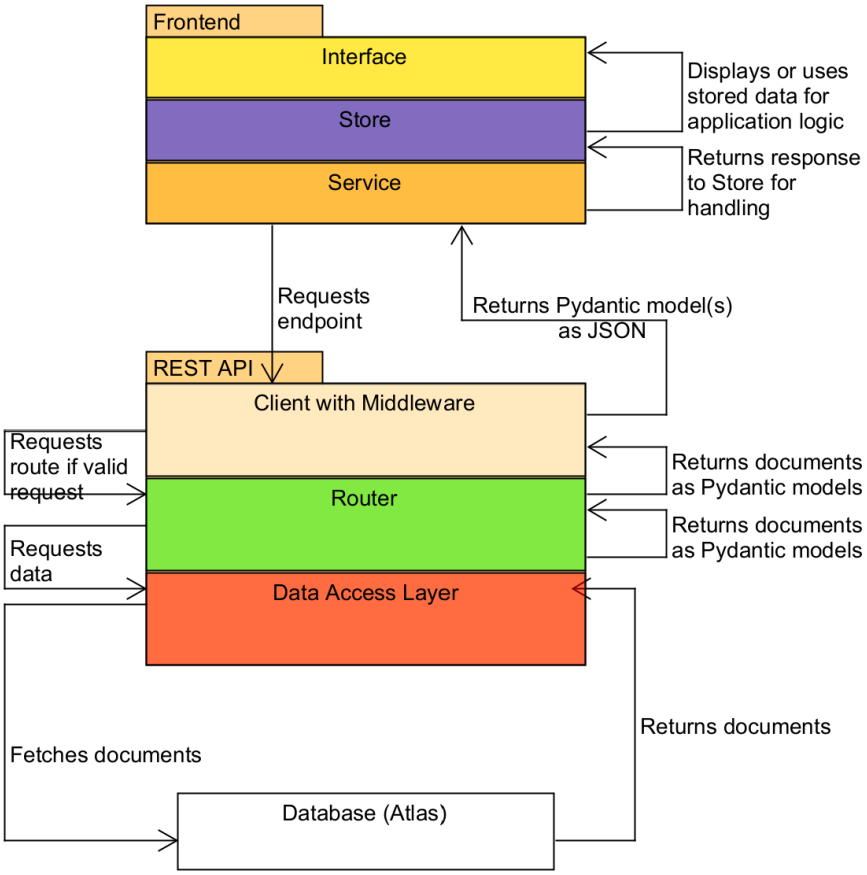


Figure 38: The layers of the frontend and server, as well as their communications between each other and the database.

4.3 Emission Profiling

To present the test results and their implications, I delve into the findings obtained from the series of comparative tests. These tests were introduced and explained the background for in Section 3.7. There, it is specified that VMs could not be used due to the expenses, although a free trial was used to create a single VM. To visualize and read more about that process, read the VM Creation appendix. As a last remark on the VMs, the CodeCarbon *Offline Emission Tracker*, alongside the verified carbon constants were used as a substitute for the VM. This allowed for database persistance with cloud providers and varying countries.

The tests were conducted to evaluate the impact of different factors when evaluating CO2 emissions during the tuning and training of machine learning models using the optimization module. The different factors tested were countries, regions within the United States, cloud providers, and algorithm selection. All test results provide Co2 emissions per hour as a means of normalizing results, due to the variation in training time. It is important to mention that the CO2 emitted per hour is actually not only CO2, but CO2 equivalents (CO2eq). To ensure data accuracy, two separate runs were conducted for each country and region. This approach enabled more reliable and

precise emission measurements for each specific region.

4.3.1 Emissions by country

In figure 39, I present the calculated hourly rate of emissions from five different countries worldwide. Notably, the emissions per hour of training appear to be lower in the European countries, with the worst performing country being Australia, which emits 2.5 times as much CO2eq as Spain.

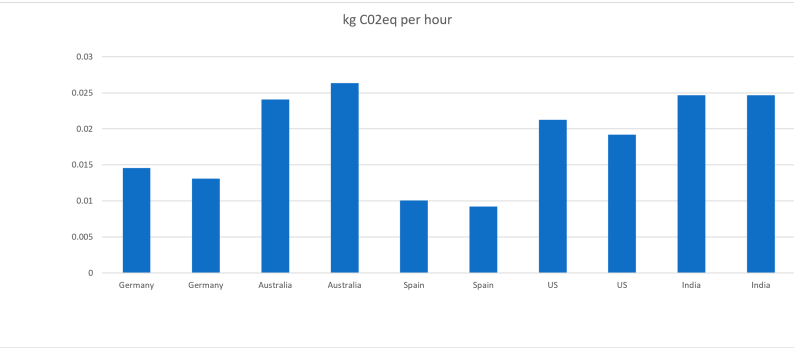


Figure 39: CO2 emissions when training the model in different countries.

4.3.2 Emissions by Region

Figure 40 displays the calculated emissions for four distinct regions within the United States. In the state of Vermont, an average of 0.0013 kg C02eq are emitted per hour of model training, whilst 0.04kg CO2eq are emitted per hour in Kentucky. The difference is a factor of 30.8. These numbers are fetched from the spreadsheet containing the data fetched from the database. This spreadsheet is called ”co2 emissions” and can be found as an appendix.

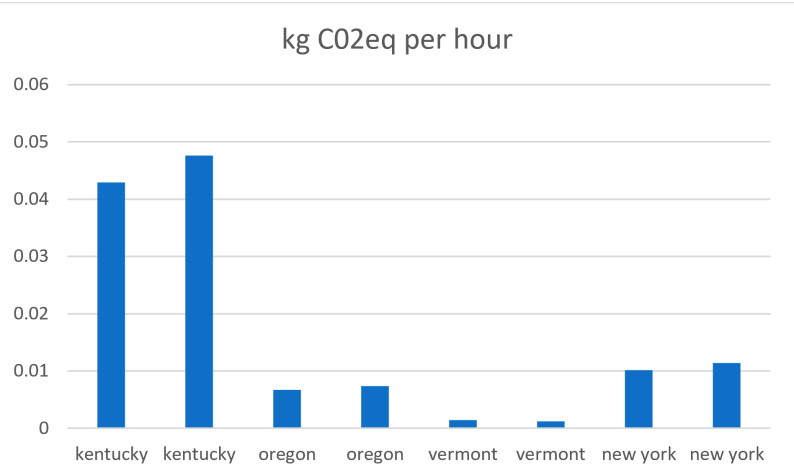


Figure 40: CO2 emissions when training the model in different regions within the United States.

4.3.3 Emissions by Cloud-provider

The graph depicted in Figure 41 provides a comparative analysis of CO2 emissions among the three leading cloud providers: GCP, AWS, and Azure. To acquire emissions data, two runs were conducted for each provider across multiple countries. This was done using the Offline Emission Tracker as an alternative to the preferred VM solution. The graph showcases the average emissions from these runs for each combination of cloud provider and country. It should be noted that if a particular cloud provider did not provide data for a specific region, it was excluded from that particular comparison, resulting in limited entries for certain testing regions, as indicated in the figure.

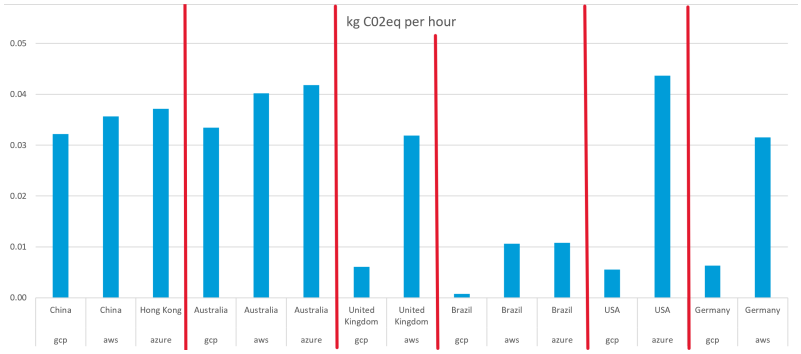


Figure 41: CO2 emissions when training the model in different countries with different cloud providers.

The obtained results reveal significant variations in emissions across different regions, particularly in the United States. Similar patterns are observed in the United Kingdom and Germany. In contrast, countries like China and Australia exhibit relatively stable emission rates, attributed to their heavy reliance on fossil fuels [39, 40]. However, both nations have outlined plans to increase their utilization of renewable energy sources in the future. Notably, it is interesting to observe that the rate of emissions, from least to most, consistently places Google Cloud as the provider with the lowest emissions and Azure as the provider with the highest emissions.

4.3.4 Emissions by RL model

Even more interestingly, the module can be used to compare the emissions produced by different reinforcement learning algorithms. In this analysis, emissions per hour for optimization runs using PPO, A2C, and SAC were compared in 5 different countries.

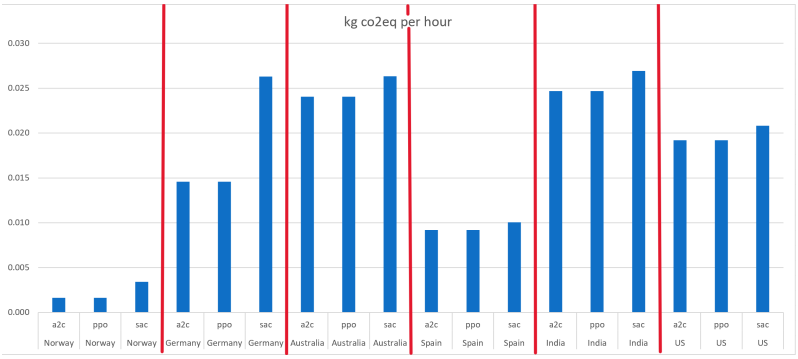


Figure 42: CO2 emissions when training models in different countries using
.

As observed in the other results, the choice of server location plays a significant role. The countries included in this test are the same as those in the country-comparison, with the addition of Norway. It remains consistent that European countries generally exhibit lower emissions compared to other recorded countries, except for Germany in this test. When comparing emissions produced by individual algorithms, SAC consistently demonstrates higher emissions. In regions like Norway and Germany, SAC generates approximately twice as much CO2eq as the other algorithms per hour. In the other regions, SAC emits more, but by a smaller factor.

4.3.5 Findings

The series of comparative tests conducted in this study aimed to evaluate the impact of various factors on CO2 emissions during the tuning and training of machine learning models using the optimization module.

The test results reveal significant variations in emissions across different countries, with European countries generally exhibiting lower emissions compared to other recorded countries. For example, Spain has significantly lower emissions per hour of training compared to Australia, which emits 2.5 times as much CO2eq. This indicates that the choice of country for model training has a notable influence on emissions. Within the United States, emissions also vary across different regions. Vermont has the lowest emissions per hour, while Kentucky has emissions that are approximately 30.8 times higher. Comparing different cloud providers, the results consistently position Google Cloud as the provider with the lowest emissions per hour, and Azure as the provider with the highest emissions. Furthermore, the analysis of emissions produced by different reinforcement learning algorithms demonstrates that the choice of algorithm also impacts emissions. SAC consistently generates higher emissions compared to PPO and A2C, particularly in regions like Norway and Germany, where SAC emits approximately twice as much CO2eq per hour.

4.4 Administrative results

The administrative results for this thesis consisted of a comprehensive timetable showcasing the hours dedicated to various project activities, a Gantt chart and completion in regards to functional features. The timetable is organized on a weekly basis and is accompanied by corresponding status reports. Throughout the course of this project, a total of 600 hours were invested in research, design, development, testing, writing, and troubleshooting. The complete timetable, including detailed breakdowns, can be found in the appendix. The same goes for the Gantt-chart. The system's functional features, are presented accordance with the plans outlined in the vision document. Each functional feature is described below, along with its corresponding status of completion. Additionally, the progress plan, featured in the appendix, illustrates the originally planned completion dates alongside the actual dates of completion. It is important to note that the progress plan was created alongside the pre-project plan, prior to the completion - and mostly also the start - of the literature review. Consequently, not all tasks outlined in the progress plan align with the current problem description.

Feature	Description	Status
Algorithm selection	Ability to choose between several algorithms when optimizing and/or posting (pip module), or when fetching (website)	Completed
Environment selection	Ability to choose between several environments when optimizing and/or posting (pip module), or when fetching (website)	Completed
Visualize hyperparameters	Ability to visualised hyperparameters for a chosen environment -and algorithm combination using the website	Completed
Interface	Interface that can retrieve hyperparameters	Completed
Sustainability	The interface has information regarding sustainability (CO2 savings)	Completed
Algorithm	A population based evolutionary algorithm that is able to optimize all the algorithms available for selection during an optimization	Completed
Emissions function	A function that converts time spent for each run to CO2 usage	Completed
Endpoints	Endpoints for GET, UPDATE and more.	Completed

5 Discussion

This Chapter will interpret and analyze the results provided in section 4. The research questions for this thesis, as presented in the introduction, were:

1. How can an intuitive and comprehensive tool be designed to efficiently extract hyperparameters?
2. How can the Hyperfetch application be leveraged to gain valuable insights into the emission profile of a reinforcement learning project?

The most important findings in relation to the research questions were the results from the user testing, as well the results from the emission profiling tests that were conducted.

The findings from product testing can be used to reflect on the first research question to some extent. However, it does not provide much meat to the bone for the second question. This is because the first research question to a great extent demands qualitative answers, as it revolves around designing an intuitive and comprehensive tool to efficiently extract hyperparameters. This question requires qualitative feedback to respond to, as it is hard to answer using numbers. The user tests supply such qualitative answers. The second research question however, which pertains to gaining insights into the emission profile of reinforcement learning projects, demands quantitative data in order to reflect on it. Therefore, the first research question will be debated in section 5.1, which debates the product testing, whilst the second research question will be debated in section 5.2, which debates the results from the emission profiling tests. Elements pertaining to the developmental process will be discussed in section 5.3. A reflect on the process of conducting can be found as an appendix.

5.1 Product testing

With the goal being to design an intuitive and comprehensive tool to efficiently extract hyperparameters, it is important to analyze whether the user testing adequately reflects this objective. Overall, the product was usable and of an acceptable standard, but did not comply on all aspects.

The user testing revealed both positive feedback and areas for improvement. Participants expressed appreciation for the separation between the application and the module, as it facilitated efficient client-side processing of complex tasks and aided in recreating previous projects within reinforcement learning. However, challenges were identified during the installation of the pip-module on Windows and macOS, highlighting the need for further testing. Feedback on the web-page was generally positive, with users finding it clear and easy to navigate. However, some users experienced slow response times on certain endpoints, which was addressed by increasing the cloud computing power and optimizing the middleware. Although, it is worth mentioning that the cloud computing power had to be turned back to its original (free) state after some time, as the Azure account ran out of free credits. Therefore, slow endpoints can be expected again at this time.

Suggestions were provided for improving beginner guides and reducing information overload on the website, such as implementing collapsible sections and clearer links

to external resources. Additionally, a user provided feedback on the necessity of the module providing a lite-version of the web application's results in the terminal. Design-related feedback highlighted issues with color combinations, link readability, and animation speed, leading to recommendations to adopt a cohesive color scheme, improve readability, optimize box sizing, and ensure accessibility compliance. While the issue with dependencies was resolved to some extent by editing the dependencies such that macOS users could download the module, Windows-use is still not fully implemented.

In addition, with only 8 user testers, and only 5 attempting to install the module, it is evident that the module was not tested enough by the users in comparison to the website. In addition, most of the test-questions pertained to the websites functionality and design, which does not cover enough ground to fully answer the research question. This is because the website is only one half of the application.

The 8 testers consisted of 3 professionals, whereas 2 were from within the fields of machine learning, data analysis and environmental RL, whilst one did not work with machine learning. The testers also consisted of a PhD student within optimization, and 4 students between 3rd and 5th grade. These students and professionals all had experience with UX/UI, machine learning, or both. The 15 questions asked revolved around the topics of:

Experience and background, which helps assess the suitability of the tool for different user backgrounds.

Usage of Pip-Module and Website, which gathered feedback on the effectiveness and convenience of the approach (dividing the two instances). This information helps evaluate the tool's efficiency in extracting hyperparameters.

Focus on CO2-Emissions and Project Recreation, which addressed the tool's comprehensiveness and alignment with the research question. Their elaborations provided insights into the effectiveness of the tool in achieving these objectives.

Ease of Installation and Usage, which assessed the tool's accessibility and user-friendliness. The Participants responses provided insights into challenges or improvements needed in the installation process, especially considering dependencies and different operating systems.

Understanding of Website and Features, which assessed the clarity and intuitiveness of the tool's interface.

Overall Design and Feedback, which provided participants with the opportunity to provide general feedback and suggestions.

Regarding sample size, meaning the number of testers, there are many opinions regarding how many users are appropriate for a test to be considered extensive enough to be valid. It is widely assumed that 5 participants suffice for usability testing [41, 42]. However, in a study from 2003 [43], the risks of using only 5 participants are demonstrated, and the benefits of using more are shown. 60 users were tested, and random sets of 5 or more users were sampled from the whole. Some of the randomly selected sets of 5 participants found 99% of the problems, while other sets found only 55%. With 10 users being tested in a sample, the lowest percentage of problems revealed by any sample set was increased to 80%. With 20 users, the lowest percentage of problems revealed were 95%.

With this application being tested by 2 experts and 8 users, the problems detected could be 80% or more, if counting the interviews as tests. However, this statement might not hold, as the questions could have been formulated better. While participants were asked about their experience with machine learning and UX/UI, it would likely have been beneficial to gather more detailed information about their level of expertise in these areas, especially for the students. Understanding the participants' depth of knowledge to a wider extent could have helped better assess the tool's suitability for different user profiles. In addition, the questionnaire primarily focuses on the usability and user experience aspects of the tool. However, it would have also been beneficial to gather feedback specifically related to the tool's effectiveness in extracting hyperparameters. For example, participants could have been asked if they found the tool to be efficient in delivering accurate and reliable hyperparameters, or if they encountered any limitations in its functionality. Another example question could have been to ask the participants if they thought the hyperparameters were displayed in a good manner, or if they thought they should have been displayed differently. If such a question had been asked, there would likely be more feedback on the page displaying hyperparameters and environmental parameters for an optimized model. This would have been benefitital, as the page is the one that directly relates the most to the research questions focus on hyperparameter extraction. To add on that, it would have been helpful to have asked the participants about their experience with features such as parameter selection, optimization settings, or data retrieval. This would have made it easier to detect if the sample set of users accurately represents the end-users. Lastly, to gain a deeper understanding of the tool's efficiency and intuitiveness, it would have been beneficial to include questions that allow for comparative analysis. As an example, the participants could have been asked if they have used other hyperparameter extraction tools before and how this tool compares in terms of ease of use, functionality, and overall user experience.

As a final remark on the user sample, answers were anonymous, meaning that answers are not connected to a single persons identity. The users were told to be honest and that their answers would be anonymous. However, all participants were asked in person or through mail about testing the application, and are not anonymous in that sense. Although some test subjects were unknown and contacted through mail, others are more familiar with the developer. This could have added bias to the results, as the subjects could have felt obliged to give better ratings than what they felt the application deserved.

5.2 Emission Profiling

This section contains an analysis of the findings from the emission profiling in section 4.3. The emission profiling tests were conducted in order to research the contributing factors to the emission profile of a reinforcement learning project. This is needed, because the absence of comprehensive emission reporting and analysis makes it hard to conduct informed decision-making when deciding which location and algorithm to train a reinforcement learning model in/with. The tests aimed to evaluate the impact of server location, cloud provider and algorithm selection.

5.2.1 Location of Server

The results show a variability in CO2eq emissions when comparing locations and regions. This data is consistent, as it is shown in both test runs for each factor. Calculated emissions from different countries during model training is illustrated in Figure 39. From the model, it can be observed that the European countries show lower emissions per hour of training, suggesting potential differences in energy sources or efficiency measures compared to the other countries. Emissions calculated for training models in various regions within the United States is showcased in Figure 40. The graph reveals significant variations in emissions across the regions of the country. With a factor of 30.8 separating the best and worst performing state, it is logical to assume that the energy source differs greatly in green quality between these regions as well. As mentioned briefly in the theory-chapter Lacoste et. al. [20], performed a study in 2019 where the distributions and variations in emissions were outlined between the continents. The results is shown in Figure 43.

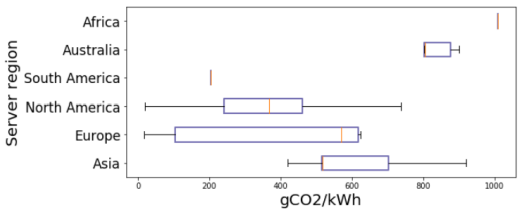


Figure 43: The distribution and variation in carbon emissions depending on geographical region.

Source: [20]

When looking at Australias numbers in the figure, they are higher than the other countries. To compare the data from the figure with the data from these tests, it is interesting to compare the emissions per kWh. This does requires some math, as the optimization module only reports total kWh spent. The numbers presented on the website are emissions per run. Using the Hyperfetch website [7] to find the run, as shown in Figure 44, the numbers can be compared.

Figure 44: The emissions produced by training a model in Australia using Hyperfetch and a VM.

Source: [7]

The training process for the model spent 0.2436 kWh. This produced 0.1211 kg CO2eq. Per kWh, the energy used would have to be multiplied by 4.1050. If it assumed that energy used and emissions produced form a linear graph, then the emissions would be

$$0.1211\,\text{kg CO2eq} \times 4.1050 = 0.497\,\text{kg CO2eq}$$

This number is not totally comparable to graph in the Figure, as Australia in the graph would be set to emit almost double the amount recorded through Hyperfetch. This could be due to factors such as different hardware being used, different amount of computing power being utilized, or even due to a decrease in use of fossil fuels between 2019 and now (2023). The latter could be the case, as Australia had begun installing renewable energy (solar and wind) 4-5 times faster per capita than the EU, USA, Japan and China in 2019 [44]. In addition, the numbers presented in the regional emission comparison within the United States closely lines up with the study performed by Brander et. al. in 2011 [19], where similar tests were conducted. This study was also mentioned briefly in the Theory Chapter (Section 2.4.1). In the study it is stated that servers located in North America can emit anywhere between 20g CO2eq/kWh in Quebec, Canada to 736.6g CO2eq/kWh in Iowa, USA. This difference is a factor of 37, caused by the the Quebec location using renewable energy sources, whilst the Iowa location relies on fossil fuels. The line can be drawn from these variations, and onto the variations between locations in this thesis, where the difference between Vermont and Kentucky had been a factor of 30.8.

5.2.2 Cloud Provider

The results revealed interesting findings, demonstrating significant variations in emissions between the providers. Notably, Google Cloud consistently exhibited the lowest emissions, while Azure displayed the highest emissions. Although specific information regarding energy sources based on regions for the providers was no where to be found, the observed patterns in this comparison seem to align with results from several articles.

According to the 2017 Clicking Clean report by Greenpeace, Google Cloud Platform (GCP) ranked as the most environmentally friendly cloud infrastructure, surpassing Amazon Web Services (AWS) and Microsoft Azure [45]. This finding is further supported by articles published in 2020 and 2023, which trained models using these three providers and yielded similar results [46, 47]. However, the situation is not as straightforward as it may seem. A heatmap from 2022 displaying the carbon intensity of data centers by region reveals that GCP does not always outperform its competitors [48]. For example, in Canada, AWS's server emits 0.308 g CO2eq/h, while GCP emits 0.497 g CO2eq/h. Similarly, in England, AWS's server emits 0.539 g CO2eq/h, Azure's server emits 0.578 g CO2eq/h, and GCP's server emits 0.893 g CO2eq/h. These numbers from the heatmap challenge the notion that GCP consistently outshines other providers. It appears that various regional nuances play a role in the quantity of emissions.

Nevertheless, the findings from the testing results and the mentioned articles support the overall pattern where GCP appears to be the leading cloud platform in terms of emissions. However, it is important to acknowledge that more comprehensive data is

needed to draw definitive conclusions. The heatmap also highlighted the observation that GCP tended to be positioned in colder regions of the heatmap compared to the other providers, particularly Azure.

In summary, the comparison of CO2 emissions among cloud providers reveals variations, where GCP generally demonstrates lower emissions. These emissions seem to be backed by a multitude of articles. However, regional differences and the limited availability of data indicate the need for further investigation as it was hard to find research papers within the topic.

5.2.3 Algorithm selection

In the algorithm-comparison results, variations in emissions were observed among the tested algorithms. Soft Actor-Critic (SAC) consistently created a higher rate of emissions compared to the other algorithms. Specifically, SAC produced approximately twice the amount of emissions as Proximal Policy Optimization (PPO) and Advantage Actor Critic (A2C) in Norway and Germany. In Australia, Spain, India, and the United States, SAC only marginally emitted more than the other models. Although a pattern is evident, its interpretation is not entirely clear.

The influence of location on the emission rate between the models seems illogical, given that all runs utilized the same hardware and all algorithms within a specified region were trained using the same carbon constants. It would seem more logical for each country to display the same pattern for their algorithm-tests. However, there is one logical factor to consider. SAC is known to have a significantly slower training process, resulting in time constraints for this study. Consequently, only the initial two runs (Norway and Germany) were trained for 1000 trials, while the remaining four runs were trained for only 70 trials, which still provided a few hours of training. It could be coincidental that the the runs in Norway and Germany, with 1000 SAC trials, displayed a pattern between each other, whilst the runs with 70 SAC trials also displayed a pattern between each other (at a different scale). If SAC had been tested to have a linear energy consumption per hour for an extended amount of time, then the graphs would likely be accurate. However, that has not been tested in this thesis, leaving no proof.

Therefore, it is possible that the data for SAC in the four countries where it was trained with fewer trials may be inaccurate. Conversely, PPO and A2C consistently exhibited similar results as they were trained for 1000 trials in all countries. It is noteworthy that the graph depicts emissions per hour, indicating that SAC consumed more energy per hour. However, due to the absence of proper scientific methodology in the testing process, it is difficult to definitively conclude whether SAC indeed produces twice the emissions of PPO and A2C. Nevertheless, it is apparent that SAC emits more CO2eq in comparison to the others.

5.2.4 Summary

The emission profiling analysis in this study discusses the findings from the Results. Variability in emissions was observed when comparing different countries and regions, with European countries generally showing lower emissions per hour of training. This suggests potential differences in energy sources or efficiency measures in these re-

gions. The results align with previous studies that have demonstrated variations in emissions based on geographical location. Additionally, the comparison of emissions among different cloud providers revealed significant variations, with Google Cloud Platform (GCP) consistently exhibiting the lowest emissions and Azure displaying the highest emissions. These findings are consistent with previous reports ranking GCP as the most environmentally friendly cloud infrastructure. However, it is important to note that further investigation is needed. Lastly, variations in emissions were observed among the tested reinforcement learning algorithms, with Soft Actor-Critic (SAC) consistently producing higher emissions compared to Proximal Policy Optimization (PPO) and Advantage Actor Critic (A2C). However, the interpretation of these results is limited due to potential biases and the need for further testing.

Overall the analysis of the results emphasizes the importance of factors such as country, region, cloud provider, and algorithm choice when evaluating CO2 emissions during machine learning model training. By planning an optimization with these factors in mind, it is possible to reduce the environmental impact associated with model development and contribute to sustainability efforts.

5.3 Final product

One notable drawback of the development process and the final product is the inadequate test coverage. Particularly, this concerns the frontend and the standalone optimization module. Although end-to-end tests were implemented for the RestAPI, the other two remain untested, leading to insufficient coverage overall. Initially, there was a plan to adopt a more comprehensive testing approach to ensure the correctness of all application components. However, due to the limitations of developing without a team and the constraints of time, it was challenging to achieve the desired level of test coverage. Consequently, the focus predominantly centered on the RestAPI, overlooking the frontend and the standalone optimization module, which are essential elements of the application's functionality and would have been thoroughly tested with more available resources.

The absence of tests for the frontend introduces an increased risk of undetected bugs, usability issues, and unexpected behaviors. However, it is worth noting that user testing provided valuable insights and served as a compensatory means of evaluating the frontend, albeit not as systematically or as dedicated as automated tests would have been.

Similarly, the lack of testing for the standalone optimization module poses a potential risk to its accuracy, which is critical in achieving optimal results and fulfilling the application's primary objective. Inadequate testing in this area leaves room for potential errors, inconsistencies, or sub-optimal performance that may undermine the reliability of the application's optimization capabilities. Nevertheless, real-world testing conducted during emission profiling tests and the comprehensive example optimization runs described in Section 3.4.2 served as valuable sources of validation and error detection, even if it was through use of a nontraditional approach.

However, the introduction of the Data Access Layer (DAL) in the project brought several benefits to the codebase. By containing the database-interaction functions within the DAL, the code became more modular. This abstraction improved code cohesion and reduced coupling, making it easier to modify specific components without im-

pacting others. The separation of concerns achieved through the DAL also makes it easier to conduct future integration -and unit tests, because the increased modularity offered by the DAL would enable easier testing of individual components.

5.3.1 Functional Features

The functional features and their state of completion were presented in the results. All functional features were successfully implemented within the application. However, in retrospect, it would have been beneficial to break down these features into smaller components. Additionally, it is important to acknowledge that the list of features could have been longer. There are several features that contribute to the application's functionality and value that are not mentioned. These include the ability to

- Select different samplers and pruners for optimization
- Enabling parallel processing ('jobs' and 'frame stack' in the configuration file) for improved efficiency
- Distributing the optimization module as a downloadable pip package
- Implementing error handling and logging mechanisms for validation
- Connecting a run to a project on GitHub for version control
- Persisting trained reinforcement learning models to the database, such that it can be viewed on the Hyperfetch website.

Most of the functional features that were mentioned align with and contribute to addressing the research questions posed in the beginning of the chapter. Below is a description of the two most important features and how they contribute to making the application able to answer the research questions.

Providing an interface to retrieve hyperparameters: This feature plays an important role in addressing both research questions. By offering an interface to retrieve hyperparameters, the application enables users to access and utilize the extracted hyperparameters efficiently. This functionality promotes the analysis of hyperparameter configurations and their implications, which fosters informed decision-making.

Displaying sustainability information: This feature directly aligns with the second research question. By incorporating sustainability information, such as CO2 emissions calculated for the optimization process, the application provides information about the emission profile of a reinforcement learning project. This allows users to evaluate the environmental impact of their algorithms and make informed choices to reduce their carbon footprint.

5.3.2 Effect goals

In addition, the project contribution towards reaching the effect goals outlined in the pre-project document is explained in detail. Because the pre-project plan was written before the literature review had been conducted, the effect-goals do not entirely reflect the problem statement, nor the research questions. Following are the effect goals and a description of the degree to which they are met in the final product.

Effect goal 1: Create a platform that solves the problem regarding reproducing of earlier projects/results within reinforcement learning.

The core aim of Hyperfetch, as stated in the problem description, is to create a comprehensive platform for displaying hyperparameters belonging to a Reinforcement Learning project (RL), as well as gain insight into the emissions profile of said RL project. While complete resolution of this challenge may require ongoing development and refinement, Hyperfetch lays the foundation for advancing reproducibility and offers a promising lead towards achieving this effect goal.

Effect goal 2: Make it easier to do research within reinforcement learning.

Hyperfetch is designed to make research within reinforcement learning more accessible and efficient. By offering a user-friendly interface and many readily available hyperparameters[2], the Hyperfetch website empowers researchers and students to delve into the depths of the field with greater ease. While the full realization of this effect goal may require more work, Hyperfetch provides a valuable tool set that encourages analysis and exploration of reinforcement learning projects.

Effect goal 3: Make tuning hyperparameters for a given problem (algorithm x environment) easier. This will allow for improvement in models if Hyperfetch can find a better combination of hyperparameters than what was already present.

This effect goal is successfully addressed by Hyperfetch. By enabling users to optimize hyperparameters for specific problem instances through a YAML configuration file, Hyperfetch offers the users a solution to this goal.

Effect goal 4: Create an environment for researchers and students to save time by fetching optimized hyperparameters (or tuning effectively) such that RL can be more effective.

Ultimately, the creation of Hyperfetch establishes an environment where researchers and students can save valuable time and resources by fetching pre-tuned hyperparameters, as well as aid in combating the ongoing replicatin-crisis by being able to publish ones own hyperparameters and access others' projects through their linked information alongside their runs.

5.3.3 Security

In terms of the security section related to deploying the REST API, it is essential to acknowledge that while precautions have been taken to prevent attacks, no specific security testing has been conducted. As a result, it is important to recognize that the application's security cannot be definitively proven. Although steps have been taken to implement security measures, the absence of security testing means that there may still be unidentified security risks or vulnerabilities. Therefore, conducting thorough security testing is crucial to ensure a more robust and secure deployment of the REST API.

Security breach While the website only utilizing GET requests decrease the attack surface-area as there are no input fields, there are still some vulnerabilities. For ex-

[2] https://www.hyperfetch.online/

ample, the website is vulnerable to SQL injection attacks and Cross-Site Scripting (XSS). This is due to the reliance on query string parameters in the URL for passing user input. However, it is important to discuss the potential impact of these vulnerabilities. XSS attacks can lead to unauthorized access to sensitive data, manipulation of website content, and the potential compromise of future user accounts. However, the application does not hold user accounts at the current stage. SQL injection attacks can result in the unauthorized retrieval, modification, or deletion of data stored in the database. These vulnerabilities can have bad consequences, including data breaches and privacy violations, undermining the trust and integrity of the application. For the time being, the database holds no private data, meaning that this is purely hypothetical and intended as a debt ate regarding safe implementation of new features with the application being in its present state.

Looking ahead, it is important to consider security testing. Exploring the possibility of integrating automated security testing tools or engaging security professionals can provide an extra layer of assurance in maintaining the application's security. While the application currently lacks security testing, it is worth acknowledging the importance of proactively addressing potential security risks. Incorporating Pydantic models as a data validation measure was a proactive step to strengthen the codebase's security. However, the current measures do not make the application robust and secure enough. Validating and sanitizing all user input, including query string parameters, can significantly reduce the risk of XSS and SQL injection attacks.

Following is an example to illustrate the potential risk of an SQL injection attack. In this scenario, the application fetches run details from a MongoDB collection based on the run ID passed as a parameter from the FastAPI REST API.

```
run_id = request.query_params.get("run_id")
runs = db.runs.find({"id": run_id})
```

If an attacker were to pass in a run ID parameter like this:

```
run_id = '1"; db.runs.deleteMany({}); db.runs.find({"name": "admin"}) //'
```

The resulting MongoDB query that would be executed would be:

```
db.runs.find({"id": "1"; db.runs.deleteMany({}); db.runs.find({"name": "admin"})
})
```

This query first fetches the run with ID 1, but then it injects a second command that deletes all documents from the runs collection and finds the user with name ”admin”. This can be a very dangerous attack that results in a data breach and potential loss of data.

5.3.4 Document Driven Development

In the context of Document Driven Development (DDD), the primary goal is to establish a shared understanding of the problem domain by employing detailed documentation and engaging in collaboration with domain experts. While the prosess did involve extensive documentation and collaboration with experts from the domain, the creation of domain models was not pursued. Furthermore, it should be noted that while DDD aims to influence the entire development process, including analysis, design, and implementation, in this project, its application was limited to the development of the REST API. As a result, the extent to which DDD influenced the overall development process was only partial.

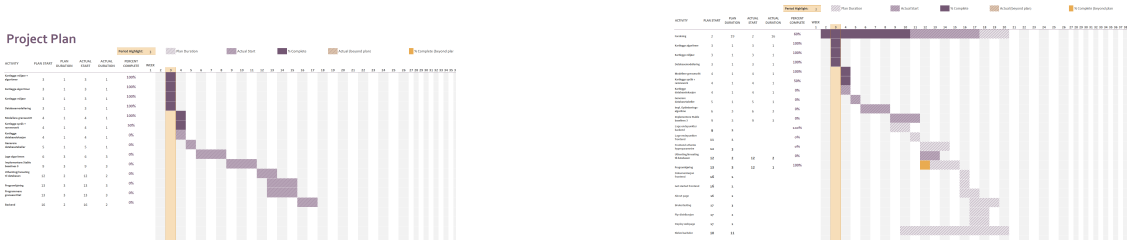
To summarize, there were certain deviations from the complete implementation of DDD principles in this project. Examples are the absence of domain models, as well as the limited application of DDD beyond the REST API. Nevertheless, the project did benefit from detailed documentation and valuable collaboration with domain experts, contributing to the finished application.

5.3.5 Alterations

The development process of the prototype underwent some alterations, leading to variations in the project’s Gantt plan. These alterations included tasks being completed earlier or later than initially planned, with durations deviating from the expected values.

The discrepancies suggest that the initial estimates for task durations were not accurate. This was due to insufficient information, unexpected challenges, and changes

in requirements. As there was no previous knowledge present from the developer regarding technologies present in the system, it proved hard to correctly estimate time use. In addition, the Gantt-chart was likely not accurate enough, which is likely because the tasks were too big to accurately measure beforehand. This highlights the importance of continuously refining and updating task estimates as more insights are gained throughout the project. This was done once, as can be seen when comparing the Gantt diagrams in figure 45, but should have been done more often as the tasks were too large to accurately measure. In addition, the change from using a single evolutionary algorithm to using a whole optimization framework was not reflected in either Gantt charts. The Gantt chart should have been edited and divided further to account for the additional complexity and validation requirements.



(a) Gantt january (b) Gantt march

Figure 45: The Gantt diagram's evolution between januray and march.

In addition, it was evident from the Gantt charts that the ordering of tasks to be completed was in need of improvement. The interdependencies between tasks and resource availability can affect their durations. Because the planning of the database took longer than initially planned - due to switching from a relational to a non-relational database - all database-dependent tasks were affected. This led to tasks taking longer or finishing earlier than anticipated, which indicates that the sequencing of work needed some improvement.

As mentioned earlier in this section, Optuna was adopted as a replacement for a single evolutionary optimization algorithm. This change significantly enhanced the capabilities of the application, offering a broad array of features that offered a wider range of tuning options and increased overall flexibility. The incorporation of Optuna brought forth a comprehensive framework dedicated to hyperparameter optimization and reinforcement learning algorithms. By using Optuna, the application benefited from advanced algorithms for pruning and sampling that enabled more effective exploration and exploitation of the hyperparameter search space. This resulted in improved performance and the ability to identify optimal configurations for reinforcement learning models in more ways than just one.

However, it is important to note that the integration of Optuna also introduced additional complexity to the application. This complexity had an unintended consequence of making the website appear more cluttered than initially intended. Despite this visual trade-off, the inclusion of Optuna outweighed the clutter concerns due to the substantial benefits it brought in terms of optimization capabilities.

6 Conclusion and Future Work

This section will describe the projects contribution to the field by answering the research questions.

These are the research questions:

1. How can an intuitive and comprehensive tool be designed to efficiently extract hyperparameters?
2. How can the Hyperfetch application be leveraged to gain valuable insights into the emission profile of a reinforcement learning project?

6.1 Research question 1

This thesis aimed to address the first research question by designing an intuitive and comprehensive tool for efficient hyperparameter extraction. The findings and discussions based on user testing demonstrate that the application exhibits efficiency in extracting hyperparameters. However, user feedback revealed areas that require improvement to enhance the application's intuitiveness and user-friendliness.

Specifically, users encountered difficulties in understanding certain elements of the application, indicating the need for simplification, clearer instructions, and overall design refinement. By incorporating these improvements and considering the valuable feedback received, the application has the potential to become a more accessible and comprehensive tool for hyperparameter extraction.

Several areas for improvement have been identified, including the installation dependencies, website design, and responsiveness. Users also provided suggestions for additional features, such as the inclusion of a lite-version of the web application's results in the terminal for the module. By incorporating these improvements and conducting further testing, the potential exists for the deployed application [7, 8] to become a more comprehensive and intuitive tool for efficiently extracting hyperparameters.

However, it is important to note that the answer to the research question remains non-conclusive at this stage. The suggested improvements and modifications need to be implemented and tested in a new iteration of the application. This will provide a clearer understanding of whether the changes successfully lead to a more comprehensive and intuitive solution for hyperparameter extraction. Therefore, further testing and evaluation will be necessary to validate the effectiveness of the implemented changes and their impact on the application's overall usability.

By continually incorporating user feedback, refining the design, and enhancing the application's features, it is anticipated that the final product will offer an improved user experience and effectively address the research question regarding the development of an intuitive and comprehensive tool for efficient hyperparameter extraction.

6.2 Research question 2

The second research question aimed to leverage the Hyperfetch application to gain valuable insights into the emission profile of a reinforcement learning project.

In this study, I conducted an emission profiling analysis to examine the factors contributing to CO2 emissions during model training for Reinforcement Learning (RL) projects. The analysis of the results underscores the importance of considering factors such as country, region, cloud provider, and algorithm choice when evaluating CO2 emissions during optimization and training. By strategically planning and optimizing these factors, it is possible to reduce the environmental impact associated with model development and contribute to sustainability efforts.

Overall, this research contributes to the understanding of emissions in the context of reinforcement learning and highlights the significance of considering environmental factors in decision-making processes. In addition, two methods have been shown to leverage Hyperfetch such as to gain insights into the factors that play a role in creating the emission profile for an RL project. The first is through use of a Virtual Machine, which is explained and visualized throughout the thesis and in the ”VM Creation” appendix. The second method is through the use of the Offline Emission Tracker provided by CodeCarbon.

The results show that by integrating sustainability considerations into the development of RL models, we can work towards minimizing the environmental footprint of these technologies. Further research and collaboration are needed to enhance our understanding and develop more comprehensive frameworks for sustainable machine learning practices.

6.3 Limitations

Time constraints during the development process limited the scope and thoroughness of testing. As a solo developer, the available resources were limited, resulting in non-optimal test coverage. Time limitations also impacted the implementation of user feedback, requiring a selection of which feedback to prioritize. Furthermore, adherence to the Web Content Accessibility Guidelines (WCAG) was not fully achieved, except for elements such as ALT text. In addition, the sample size for user testing was small, with only 8 potential end-users and 2 experts interviewed. A larger sample size would have increased the probability of identifying potential issues and provided more comprehensive feedback. Given more time, incorporating the valuable feedback received could have led to further development and improvement of the website and module. Unfortunately, time constraints prevented additional testing and implementation of the suggested enhancements.

Limitations were also encountered in regards to funding. The researchs inability to utilize a virtual machine (VM) for conducting the tests related to the second research question: ”How can the Hyperfetch application be leveraged to gain valuable insights into the emission profile of a reinforcement learning project?” was primarily driven by financial constraints, as using a VM typically incurs additional costs. By not using a VM, I might have missed out on the opportunity to explore the emission profile in a more controlled and isolated environment, which would have likely provided more accurate insights. Despite this limitation, the study still provides meaningful insights

into the emission profile of reinforcement learning projects.

6.4 Future Work

To further enhance the Hyperfetch application, several key areas of development have been identified.

Firstly, the behavior of the scroll-up button should be improved to provide a more seamless user experience, ensuring it follows along as users scroll up and down the page. Additionally, a user-base feature can be implemented, granting users their own personalized dashboards with comprehensive data on their individual runs. This could include interactive charts showcasing CO2 emissions and average rewards per trial, as well as the ability to track the progression of rewards over time. Another design detail is the rounding of numbers. The current rounding of numbers in the runs and run section is very messy, due to inconsistent amount of decimal numbers. To address this issue and improve readability, the numbers in the runs and run sections should be rounded using scientific notation and a set amount of decimal numbers. Regarding hyperparameters, it would increase the accessibility and ability to recreate projects if the hyperparameters for a run were downloadable through the push of a button. This feature could, for example, have consisted of downloading the hyperparameters and other configurations in a YAML file format ready for use.

For configuration and validation within the optimization module, verbosity (the level of detail) should be added to the logger. This would be especially useful for when values are not present in the configuration file, and the module prints out continuous messages specifying which default settings were applied, as well as which configurations were changed if applicable. Thus, the verbosity would be an option to disable logging of default values for non-mandatory parameters. In addition, comprehensive testing should be prioritized in the future, especially in the frontend and optimization module. Implementing dedicated tests, such as unit tests and integration tests, would increase the application's stability, correctness, and reliability. Additionally, the incorporated security measures, such as HTTP, middleware, and Pydantic models, should be accompanied by thorough security testing.

Looking ahead, there is also a possibility to extend the Hyperfetch platform beyond reinforcement learning, including other machine learning paradigms as well. This expansion would offer a comprehensive overview of emissions associated with various machine learning approaches. Furthermore, the application's logo should be aligned with the page and pip-modules, by adopting the name ”Hyperfetch” as a single word, instead of the current logo name, which is ”Hyper-Fetch”.

7 Societal Impact

The study focuses on evaluating and understanding the CO2 emissions associated with model training in the field of Deep Reinforcement Learning (DRL). By identifying the factors that contribute to emissions, such as country location, regional location, cloud provider, and algorithm choice, the research highlights the importance of considering sustainability and environmental impact in the development of RL models. This awareness can lead to more informed decisions and practices that aim to reduce the carbon footprint of these technologies. In addition, the project emphasizes the need to integrate sustainability considerations into the development of RL models. By raising awareness about the environmental impact of model training and providing insights into emission profiles, the research encourages the adoption of sustainable machine learning practices. This hope is that the research will motivate practitioners to optimize algorithms and hyperparameters with the goal of minimizing emissions, selecting eco-friendly cloud providers, and consider the geographical context of training.

Overall, the societal impact of this project lies in promoting environmentally conscious practices, improving reproducibility, and fostering collaboration within the field of reinforcement learning. By considering the environmental impact of model training and providing tools for reproducibility and knowledge sharing, the research contributes to sustainable machine learning practices and the advancement of the field, benefiting both the scientific community and the broader society.

Bibliography

[1] Peter Henderson et al. 'Deep reinforcement learning that matters'. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[2] Odd Erik Gundersen. 'The Reproducibility Crisis Is Real'. In: *AI Magazine* 41.3 (2020), pp. 103–106. doi: 10.1609/aimag.v41i3.5318. url: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/5318.

[3] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[4] Anand S. Rao and Gerard Verweij. *PwC's Global Artificial Intelligence Study: Exploiting the AI Revolution*. 2017. url: https://www.pwc.com/gx/en/issues/analytics/assets/pwc-ai-analysis-sizing-the-prize-report.pdf.

[5] Christian Collberg and Todd A Proebsting. 'Repeatability in computer systems research'. In: *Communications of the ACM* 59.3 (2016), pp. 62–69.

[6] Dario Amodei and Danny Hernandez. *AI and compute*. 2018. url: https://openai.com/research/ai-and-compute.

[7] Karoline S. Wahl. *Hyperfetch*. May 2023. url: https://www.hyperfetch.online/.

[8] Karoline S. Wahl. *Hyperfetch*. Version 1.0.14. May 2023. url: https://pypi.org/project/hyperfetch/.

[9] David Patterson et al. *The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink*. 2022. arXiv: 2204.05149 [cs.LG].

[10] Yuxi Li. 'Deep Reinforcement Learning'. In: *ArXiv* abs/1810.06339 (2018).

[11] Changxi You et al. 'Advanced Planning for Autonomous Vehicles Using Reinforcement Learning and Deep Inverse Reinforcement Learning'. In: *Robotics and Autonomous Systems* 114 (Jan. 2019). doi: 10.1016/j.robot.2019.01.003.

[12] Horia Mania, Aurelia Guy and Benjamin Recht. 'Simple random search provides a competitive approach to reinforcement learning'. In: *ArXiv* abs/1803.07055 (2018).

[13] Jasper Snoek et al. 'Scalable Bayesian Optimization Using Deep Neural Networks'. In: *International Conference on Machine Learning*. 2015.

[14] Yoshihiko Ozaki et al. 'Multiobjective tree-structured parzen estimator for computationally expensive optimization problems'. In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (2020).

[15] Xiaoyu He and Yuren Zhou. 'Enhancing the Performance of Differential Evolution with Covariance Matrix Self-adaptation'. In: *Applied Soft Computing* 64 (Dec. 2017). doi: 10.1016/j.asoc.2017.11.050.

[16] Thomas J. Crowley. 'Causes of Climate Change Over the Past 1000 Years'. In: *Science* 289.5477 (2000), pp. 270–277. doi: 10.1126/science.289.5477.270. url: https://www.science.org/doi/abs/10.1126/science.289.5477.270.

[17] IPCC. *Global Warming of 1.5 ºC —*. url: https://www.ipcc.ch/sr15/.

[18] Luiz André Barroso, Urs Hölzle and Parthasarathy Ranganathan. 'The Datacenter as a Computer: Designing Warehouse-Scale Machines, Third Edition'. In: *Synthesis Lectures on Computer Architecture* (2018).

[19] Matthew Brander et al. 'Technical Paper| Electricity-specific emission factors for grid electricity'. In: *Ecometrica, Emissionfactors. com* (2011).

[20] Alexandre Lacoste et al. 'Quantifying the Carbon Emissions of Machine Learning'. In: *ArXiv* abs/1910.09700 (2019).

[21] Jacob Devlin et al. 'Bert: Pre-training of deep bidirectional transformers for language understanding'. In: *arXiv preprint arXiv:1810.04805* (2018).

[22] Karen Simonyan and Andrew Zisserman. 'Very Deep Convolutional Networks for Large-Scale Image Recognition'. In: *CoRR* abs/1409.1556 (2014).

[23] Will Buchanan and Jesse Dodge. *Measuring and Mitigating AI Carbon Intensity*. 2022. url: https://blog.allenai.org/measuring-and-mitigating-ai-carbon-intensity-8624bc805c1a.

[24] Lisha Li et al. 'Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization'. In: *J. Mach. Learn. Res.* 18 (2016), 185:1–185:52.

[25] Liam Li et al. 'Massively Parallel Hyperparameter Tuning'. In: *ArXiv* abs/1810.05934 (2018).

[26] Stefan Falkner, Aaron Klein and Frank Hutter. 'BOHB: Robust and Efficient Hyperparameter Optimization at Scale'. In: *International Conference on Machine Learning*. 2018.

[27] Nima Tajbakhsh et al. 'Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?' In: *IEEE Transactions on Medical Imaging* 35 (2016), pp. 1299–1312.

[28] Jeremy Howard and Sebastian Ruder. 'Universal Language Model Fine-tuning for Text Classification'. In: *Annual Meeting of the Association for Computational Linguistics*. 2018.

[29] Thomas T. Hewett et al. *ACM SIGCHI Curricula for Human-Computer Interaction*. Tech. rep. New York, NY, USA, 1992.

[30] W.H. DeLone and E.R. McLean. 'Information systems success revisited'. In: *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*. 2002, pp. 2966–2976. doi: 10.1109/HICSS.2002.994345.

[31] ISO. *ISO 9241-210:2010 Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*. 2010. url: https://www.iso.org/standard/77520.html.

[32] Adriana Chammas, Manuela Quaresma and Claudia Mont'Alvão. 'A Closer Look on the User Centred Design'. In: *Procedia Manufacturing* 3 (Dec. 2015), pp. 5397–5404. doi: 10.1016/j.promfg.2015.07.656.

[33] Dejan Todorovic. 'Gestalt principles'. In: *Scholarpedia* 3.12 (2008), p. 5345.

[34] Briony J. Oates. *Researching Information Systems and Computing*. Sage Publications, 2006.

[35] Lixu su, Annie Cui and Michael Walsh. 'Trustworthy Blue or Untrustworthy Red: The Influence of Colors on Trust'. In: *Journal of Marketing Theory and Practice* 27 (July 2019), pp. 269–281. doi: 10.1080/10696679.2019.1616560.

[36] Karoline S. Wahl. 2023. url: https://github.com/karolisw/hyperfetch/actions.

[37] Mozilla Firefox. 'Mixed content - Web security | MDN'. In: (2023). url: https://developer.mozilla.org/en-US/docs/Web/Security/Mixed_content#mixed_active_content.

[38] Jupyter Notebook. 2023. url: https://github.com/mlco2/codecarbon/blob/master/codecarbon/data/cloud/impact.csv.

[39] Salihu D Musa et al. ‘China’s energy status: A critical look at fossils and renewable options’. In: *Renewable & Sustainable Energy Reviews* 81 (2018), pp. 2281–2290.

[40] Hong Xian Li et al. ‘A review on renewable energy transition in Australia: An updated depiction’. In: *Journal of Cleaner Production* 242 (2020), p. 118475.

[41] Robert A. Virzi. ‘Refining the Test Phase of Usability Evaluation: How Many Subjects Is Enough?’ In: *Human Factors* 34.4 (1992), pp. 457–468. doi: 10.1177/001872089203400407. url: https://doi.org/10.1177/001872089203400407.

[42] Jakob Nielsen and Thomas K. Landauer. ‘A Mathematical Model of the Finding of Usability Problems’. In: *Proceedings of the INTERACT ’93 and CHI ’93 Conference on Human Factors in Computing Systems*. CHI ’93. Association for Computing Machinery, 1993, pp. 206–213. isbn: 0897915755. doi: 10.1145/169059.169166. url: https://doi.org/10.1145/169059.169166.

[43] L. Faulkner. ‘Beyond the five-user assumption: Benefits of increased sample sizes in usability testing’. In: *Behavior Research Methods, Instruments, & Computers* 35 (2003), pp. 379–383.

[44] Andrew Blakers, Matthew Stocks and Bin Lu. ‘Australia: the renewable energy superstar’. In: 2019.

[45] Gary Cook. ‘CLICKING CLEAN: WHO IS WINNING THE RACE TO BUILD A GREEN INTERNET’. en. In: *Greenpeace* ().

[46] Alex Poulin. *Which Cloud Computing Service Is the Most Environmentally Friendly?* en. 2020. url: https://medium.com/sustainable-finance/which-cloud-computing-service-is-the-most-environmentally-friendly-a163d5b7ddc7.

[47] 2023. url: https://www.xomnia.com/post/ai-carbon-footprint/.

[48] Maryam Arbabzadeh. *Clouding the issue*. url: https://www.climatiq.io/blog/cloud-computing-amazon-google-microsoft-helping-companies-go-green.