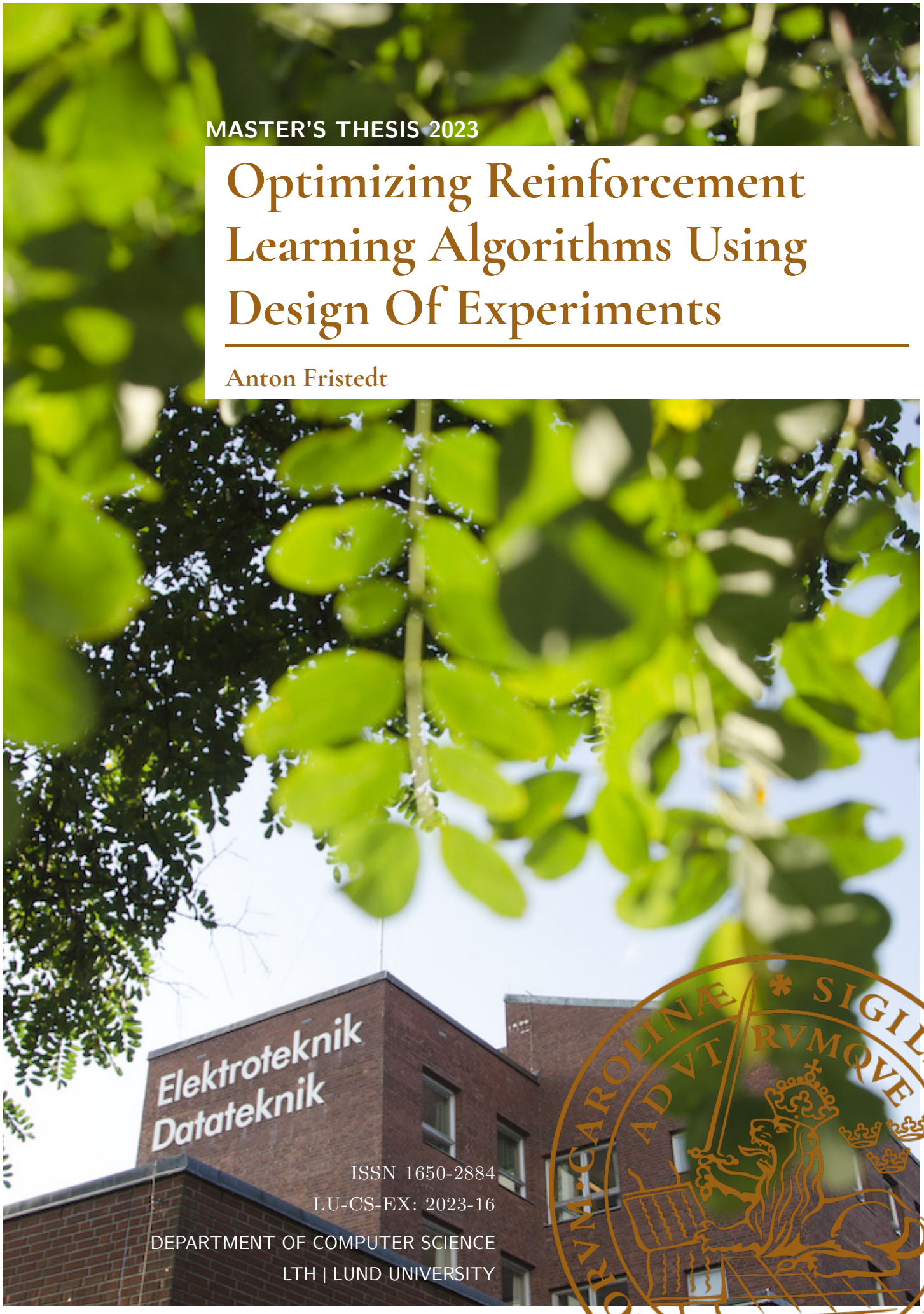


MASTER'S THESIS 2023

Optimizing Reinforcement Learning Algorithms Using Design Of Experiments

Anton Fristedt

ISSN 1650-2884
LU-CS-EX: 2023-16

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2023-16

Optimizing Reinforcement Learning Algorithms Using Design Of Experiments

Optimering av förstärkningsinlärningsalgoritmer med hjälp av försöksplanering

Anton Fristedt

Optimizing Reinforcement Learning Algorithms Using Design Of Experiments

Anton Fristedt
nat13aft@student.lu.se

June 12, 2023

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Linda Hartman, linda.hartman@matstat.lu.se
Examiner: Elin A. Topp, elin_a.topp@cs.lth.se

Abstract

In this project, we conducted design of experiments on the Proximal Policy Optimization (PPO) reinforcement learning algorithm, with the aim of optimizing its performance for robotic learning tasks. We considered different hyperparameters of PPO as factors and used a factorial design to explore their effects on the algorithm's performance. We specifically focused on the CartPole-V1 task in the OpenAI Gym classic control environment and measured the time it took for the learning to reach 95% success as the primary performance metric. Our results show that some hyperparameters have a significant impact on PPO's performance, while others have little effect. Overall, our project demonstrates the usefulness of design of experiments for optimizing RL algorithms for robotic learning tasks and provides insights into the hyperparameter tuning process.

Acknowledgements

This project was carried out on the initiative of Alexander Dürr at the Robotic and Semantic Systems research group at LTH. Thank you Alex for all the help you have provided me with along the way.

Statistics can be difficult and overwhelming at times. Somehow everything is connected in a way, but understanding that pattern can make a good man turn bad. Therefore I would like to express my sincere gratitude to my supervisor Linda Hartman at The Division of Mathematical Statistics. With your continuous support and guidance, my mindset changed from "Help! I am a fraud" to "Statistics is sweet".

I also want to thank my examiner Elin Anna Topp for taking the time to review and provide constructive feedback on this work.

Contents

1 Introduction **1**
- 1.1 Related Work . . . 1
- 1.2 Research Questions . . . 2

2 Theory/Background **2**
- 2.1 Reinforcement Learning . . . 2
  - 2.1.1 Key Elements . . . 2
  - 2.1.2 Proximal Policy Optimization . . . 3
- 2.2 RLBench . . . 3
- 2.3 RLZoo . . . 4
- 2.4 Design of Experiments . . . 4
  - 2.4.1 Factorial Design . . . 5
  - 2.4.2 Center Points . . . 7
  - 2.4.3 Residuals & normality testing . . . 9
  - 2.4.4 Response Surface Method . . . 10
  - 2.4.5 The unreplicated test . . . 11

3 Method **13**
- 3.1 Screening Experiment . . . 13
- 3.2 Optimization Experiment . . . 14
- 3.3 Response Surface Experiment . . . 14

4 Experiments **15**
- 4.1 Environment and Conditions . . . 15
- 4.2 CartPole-V1 . . . 15

5 Analysis of Result **17**
- 5.1 Screening . . . 17
- 5.2 Optimization . . . 18
- 5.3 Response Surface Method . . . 20
- 5.4 Performance with Optimal Hyperparameters . . . 21

6 Discussion and Conclusion **21**
- 6.1 Conclusion . . . 21
- 6.2 Discussion . . . 22
- 6.3 Future Work . . . 23

References **24**

Appendix **26**

1 Introduction

Reinforcement learning (RL) is a popular technique used in robotics to train agents to perform tasks through interaction with their environment. However, optimizing the hyperparameters of RL algorithms can be challenging and time-consuming. In this report, we investigate the use of design of experiments (DOE) to optimize RL algorithms for solving robotic tasks in a more efficient manner. Specifically, we focus on the Proximal Policy Optimization (PPO) algorithm and use fractional factorial design to screen for the most influential hyperparameters on the CartPole-V1 task in RLZoo.

A lot of previous works that utilize RLBench and other benchmarks for testing, either lack statistical analysis or present insufficient evidence. Yet, some still claim their result is significant without providing statistical proof of significance. In a paper by Cedric Colas et al. [1], they started questioning previous papers in this regard. After reproducing the results of a project, they determined that a type-II error had taken place, which means that the hypothesis was not rejected despite being false. They later concluded that the cause of this was wrongful choice of statistical methods and lack of an adequate number of test simulations.

With this in mind, it is important to be critical when reading papers stating significance without proof since it might be incorrect. The lack of statistical analysis in previous work serves as a strong motivation for the current project. By using the principles of design of experiments, we strive to bridge the gap between limited simulations and robust statistical evidence. Our method offers a user-friendly framework and provides straightforward statistical analysis. Additionally, our approach can be adapted to suit different research tasks and is flexible in its application.

The project was conducted at the Robotic and Semantic Systems research group at LTH, in an attempt to contribute to the research directly related to an initial workshop publication [2].

1.1 Related Work

Several studies in the past have compared reinforcement learning algorithms [3][4][5], and more studies are continuously being conducted on the topic. Since RL can be used in a wide range of applications and tested on different benchmarks [6][7][8] there is no guarantee an RL algorithm will have similar performance on different tasks. It is therefore wise to make several assessments of a RL algorithm by fitting the model to different tasks in different environments.

Out of the known RL algorithms, we focus in this thesis on the model-free algorithm Proximal policy optimization (PPO) [9], which has been frequently used in research [3][5]. With this algorithm we investigate its performance for different tasks. This should give a good indication on its performance, and the identified factors guide us to suitable hyperparameter values for optimization and testing [10].

Since our work puts great effort into design of experiments, a lot of methods come from course literature [11][12]. Using experimental design in RL for generating and analyzing data appears to be beneficial but rarely done to standards. We found work using other experimental methods [13][4]. Closely related work can be found in [14] with application on Clustering algorithms. Our project will use factorial design similar to [14]. In addition, we perform the analysis of factor effects, perform a separate screening to first identify important factors and apply response surface methodology. Our factors are in continuous space, allowing center point data in the design.

1.2 Research Questions

The initial plan for this thesis was to explore important relationships when optimizing RL algorithms used for solving real world problems in the benchmark environment RLBench. But due to staff availability issues at the department of Computer Science, this idea had to be buried and instead left for future work.

The new aim of the project was therefore to focus on a more simple task, i.e. investigating, if design of experiments can be used for hyperparameter optimization, which factors matter the most and to use the resulting model to optimize the hyperparameters, allowing for non-linear effects, if needed.

2 Theory/Background

2.1 Reinforcement Learning

Reinforcement learning is a type of machine learning where an agent learns to make decisions by interacting with an environment. It has been successfully applied to a wide range of problems, from playing games to controlling robots. It is a powerful technique for developing intelligent systems that can adapt and learn from experience.

2.1.1 Key Elements

Agent-Environment Interaction

One of the most important key components in reinforcement learning involves the interaction between a learner, also known as an agent, and its surrounding environment [15][16]. Based on actions taken by the agent in the environment, it receives feedback in the form of rewards. The agent's goal is to learn a policy that maximizes the cumulative reward over time.

To achieve this goal, the agent performs a series of time steps t as illustrated in Figure 1. First, the agent observes the environment's state $s_t \in \mathcal{S}$, for $\mathcal{S}$ possible states. Second, it selects an action a_t based on the available actions $\mathcal{A}(s_t)$ in the current state. Third, it receives a reward R_{t+1} based on the action taken and ends up in a new state s_{t+1}. Finally, it updates its policy based on the reward received and the new state of the environment.

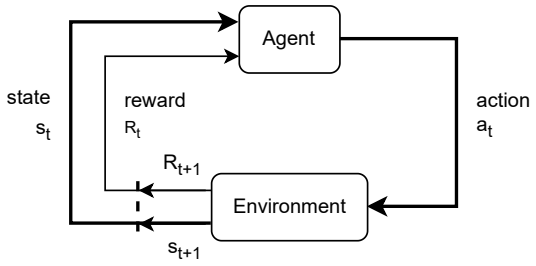


Figure 1: General interaction between agent and environment [15] in reinforcement learning

Policy

Policy is the strategy or rule applied by the agent and is denoted by $\pi_\theta(a_t|s_t)$, which maps the probability of selecting a certain action a_t given the current state s_t at time step t.

A big challenge in reinforcement learning is the exploration-exploitation trade-off. The agent needs to explore new actions to discover the optimal policy, but it also needs to exploit actions that have already been tried and tested to maximize the cumulative reward.

Reward- & Value Function

As mentioned, the goal is to maximize the cumulative reward over time. A reward function solves this problem by determining which action generates the highest reward at each time step. Different from the reward function, which focuses on

immediate success, the value function predicts the expected rewards in the long run. This allows the agent to compare future actions and helps in deciding which action to take in each state. By combining the reward and value functions, the agent can learn to make decisions that maximize the expected cumulative reward in the long run.

2.1.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO) [9] is one of the most popular and widely used reinforcement learning algorithms since it is simple to use and reliable.

On-Policy

PPO is an on-policy method which means that it learns by updating its policy based on the actions it takes while following that same policy. This approach is more time consuming than alternative methods since the updates are more conservative but with the upside of being more stable.

Policy Gradient Method

Policy gradient method directly learns a policy by optimizing its parameters through stochastic gradient ascent. It works by first computing an estimate of the policy gradient and then passing it to a stochastic gradient ascent algorithm. The policy is updated iteratively to maximize rewards. One frequently used gradient estimator is given by

$$\hat{g} = \hat{E}_t[\Delta_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{1}$$

where π_θ is a stochastic policy and $\hat{A}_t$ is an estimator of the advantage function at timestep t. Furthermore, $\hat{g}$ in equation (1) is obtained by differentiating the objective function

$$L^{PG}(\theta) = \hat{E}_t[\log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{2}$$

The advantage function $\hat{A}_t$ can be generalized as

$$\hat{A}_t = \delta_t + (\gamma)\delta_{t+1} + ... + \gamma^{T-t+1}\delta_{T-1} \tag{3}$$

$$\text{where } \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{4}$$

and γ is the discount factor, timestep t in the interval [0,T] and the estimated value V in state s_t.

Clipped Surrogate Objective

PPO utilizes a surrogate function, denoted $L(\theta)$ in eq. (5). The function puts a constraint on the policy in order to avoid unnecessarily large policy updates, causing the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ being carried to far away from 1. The function is given by

$$L(\theta) = \hat{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{5}$$

where ϵ the clip range. The probability ratio is clipped so it stays within the interval $[1-\epsilon, 1+\epsilon]$.

2.2 RLBench

RLBench [8] is a benchmark and learning environment for robot learning, designed to, among other things, be able to compare and evaluate different reinforcement learning algorithms. The benchmark offers 100 unique tasks covering different skills such as reaching, pushing and placing objects. These tasks are supposed to resemble real world problems including realistic physics, with the ambition to advance the development of vision-guided manipulation.

Environments

Tasks are performed by a Franka Emika Panda arm with 7 movable joints, attached to a wooden table [8]. It is supplied with a stereo- and a monocular wrist camera that provides visual information about the scenery in the form of rgb, depth and segmentation mask data as shown in

Figure 2. Proprioceptive data such as joint angles, velocities, torques and the end-effector pose are available as well.

In order to improve training and task completions, one or more variations are added to each task. This shifts the focus away from specific objects and instead emphasizes the task itself. It would not be particularly useful if the agent would be really good at grasping a red ball but fail miserably if the ball was blue. A reward of +1 is only given upon task completion, making the reward system completely sparse for each task.

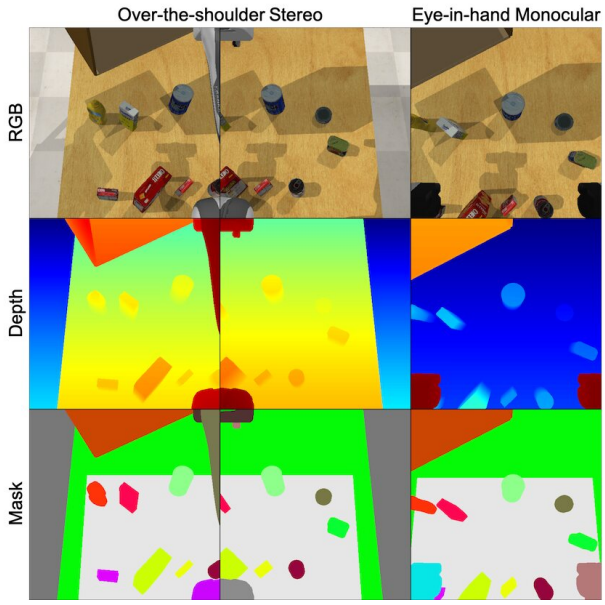


Figure 2: Observation space [8] in RLBench

2.3 RLZoo

RLZoo [17] is a versatile library that can be used for a wide range of RL applications, including robotics and game playing. It is designed to encourage the development, testing, and comparison of RL algorithms by offering a selection of pre-implemented RL algorithms, which can be customized according to the users needs.
RLZoo supports a collection of environments, such as OpenAI Gym [6], which makes it easy to benchmark different algorithms and compare their performance. Figure 3 shows an image of the CartPole-V1 task available from OpenAI Gym environment, where the objective is to balance the pole.

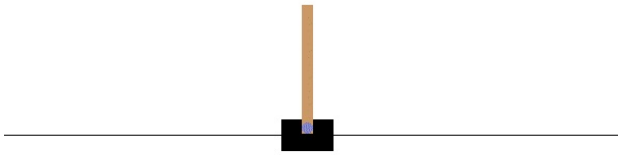


Figure 3: CartPole-V1 from OpenAI Gym [6]

2.4 Design of Experiments

When striving to achieve a model's desired performance, we often resort to trial and error by tuning hyperparameters and observing the results of running the program. The success of our choices is determined by the higher or lower performance achieved, but it can be difficult to tell whether any improvements are due to luck or a genuinely better choice of parameters. This can result in a confusing search process and relying on getting a lucky run that meets a desirable threshold.

By conducting an **experimental design** at the beginning of a survey instead, the researcher can plan, conduct and analyze all tests in a structured manner. With help from statistical methods, the results are easy to interpret and provides strong statistical evidence. The purpose of DOE is to understand the relationship between the inputs and outputs, which are referred to as **factors** and **response**, and to determine the optimal combination of inputs that lead to the best solution. Finding these relationships can be explained as a

series of processes as in Figure 4, where controllable factors can be set for each experiment while uncontrollable factors act more like noise.

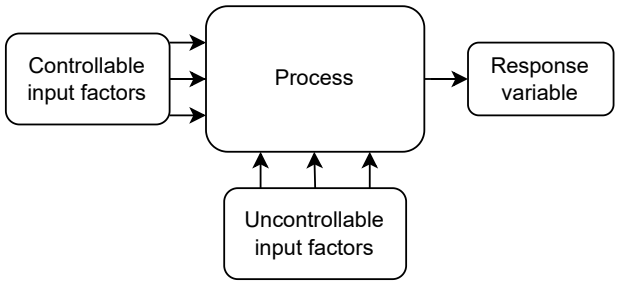


Figure 4: The experimental process in design of experiments [11]

2.4.1 Factorial Design

When conducting DOE, the first step is usually a **screening experiment**. The idea is to analyze several factors in a cost efficient way. Factors and interactions with significant influence on the response are then selected for a new, more detailed experiment. One of the most common and widely used screening methods is factorial design [11].
A factorial design with two levels is represented by 2^k, where k refers to the number of factors. The two levels are defined as low (-1) and high (+1), which can take on either quantitative- or qualitative values. The specific values assigned to the levels depend on the factor being studied.

Tables 1 and 2 provide an example of how to set up a 2^3 design with three factors. First, the low and high levels of each factor are identified, and then a design matrix is created. Each row in the design matrix represents a single experimental run and specifies the levels at which each factor should be set. This specific design can be visualized in the shape of a cube as in Figure 7a, where each corner represent one experiment. The completion of all experimental runs is considered as one replication of the test, and the data collected from these runs are referred to as the response.

	Factors		
Level	**A**	**B**	**C**
-1	x_1	u_1	z_1
+1	x_2	u_2	z_2

Table 1: Level of the three factors

	Factors			**Response**
Exp no.	**A**	**B**	**C**	**y**
1	-1	-1	-1	$(1) = y_1$
2	+1	-1	-1	$a = y_2$
3	-1	+1	-1	$b = y_3$
4	+1	+1	-1	$ab = y_4$
5	-1	-1	+1	$c = y_5$
6	+1	-1	+1	$ac = y_6$
7	-1	+1	+1	$bc = y_7$
8	+1	+1	+1	$abc = y_8$

Table 2: 2^3 Design Matrix

Calculating the (fixed) main effect of factor **A** for instance, is then

$$A = \bar{y}_{A^+} - \bar{y}_{A^-} = \frac{(a+ab+ac+abc)}{4n} - \frac{((1)+b+c+bc)}{4n} = \frac{a+ab+ac+abc-(1)-b-c-bc}{4n} \tag{6}$$

where $\bar{y}_{A^+}$ is the mean value of all high level operations of **A**, $\bar{y}_{A^-}$ the low level and n number of replications.
Similarly, the factor effect of **B** is given by

$$B = \bar{y}_{B^+} - \bar{y}_{B^-} = \frac{b+ab+bc+abc-(1)-a-c-ac}{4n} \tag{7}$$

Lastly, the interaction effect between A and B is given by

$$AB = \frac{abc + ab + c + (1) - a - ac - bc - b}{4n} \tag{8}$$

i.e. the effect of A is different for different levels of B (and vice versa). For the interested reader, further information on factor effect computation

is available in Montgomery's *Design and Analysis of Experiments* [11].

In a **full factorial design** as exemplified in Table 2, all permutations are tested which can be cumbersome when increasing the number of factors. A 2^5 design requires 32 experiments, add one additional factor and suddenly 64 experiments are needed. Beside from being a costly procedure, performing this many experiments usually serves little purpose. In reality we are most interested in factor's main effects A, B & C and their two-way interactions with each other AB, AC & BC. The reason behind neglecting third or higher factor interactions is simply because they rarely have an impact [13]. In total, 7 experiments are sufficient to estimate 3 main effects, 3 two-way interactions and the model intercept (grand average from all tests).

	Base Fac.			**ABC**
Exp no.	**A**	**B**	**C**	**=D**
1	-1	-1	-1	-1
2	+1	-1	-1	+1
3	-1	+1	-1	+1
4	+1	+1	-1	-1
5	-1	-1	+1	+1
6	+1	-1	+1	-1
7	-1	+1	+1	-1
8	+1	+1	+1	+1

Table 3: 2^{4-1} Design Matrix

The natural solution for reducing the number of experiments is **fractional factorial design**, denoted as 2^{k-p}, where p is the size of the fraction. When reducing a design this way, some combinations are lost in the process, compared to a full factorial design where all combinations are present. In order to minimize the information loss, p independent factors are generated by taking the product of the base factors.

For example, in Table 3, a 2^{4-1} design is represented. Instead of conducting 16 runs to test 4 factors, this fractional design requires only 8 runs. $(k-p)$ factors, in this case A, B and C, are considered as base factors and their levels are structured as usual. The p independent factors however, factor D in this example, need to account for the missing combinations. Selecting which combinations to include is done by letting D be the product of the base factors A, B, and C. In general, the last p independent factors are generated by taking products of the first $(k-p)$ base factors.

As a side effect of this, higher order interactions are now assumed to be negligible when estimating main effects. Table 4 shows the alias structure of this, all main effects are confounded with three-factor interactions, thus, these effects must be ignored. Furthermore, all two-factor interactions are confounded with another interaction, making it difficult to estimate them separately.

A = BCD
B = ACD
C = ABD
D = ABC
AB = CD
AC = BD
AD = BC

Table 4: Alias structure of a 2^{4-1} design with factor generator D=ABC

Factor effects can be established by methods such as analysis of variance (ANOVA) and linear regression.

In a screening experiment, these effects can be analyzed with a normal probability plot and a Pareto chart. The normal plot, depicted as in Figure 5, represents the factor effects assuming normal distribution. A normal line is fitted to align with as many points as possible. Points deviating

from this line are considered as outliers, indicating significant influence, which is of great interest for further analysis. Additionally, the magnitude of the calculated factor effects can be presented in a bar diagram known as a Pareto chart, as shown in Figure 6. This chart provides further clarity regarding the size of the effects.

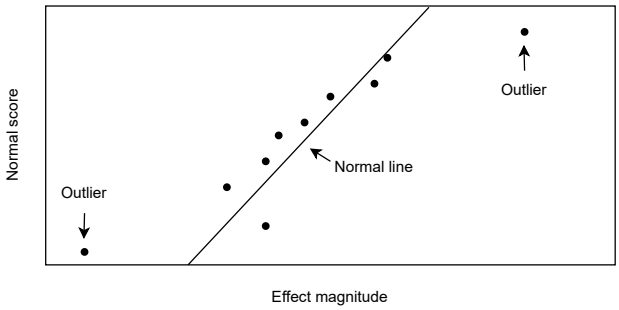


Figure 5: Normal probability plot. Points following the line can be considered normally distributed, outliers indicate significant effect

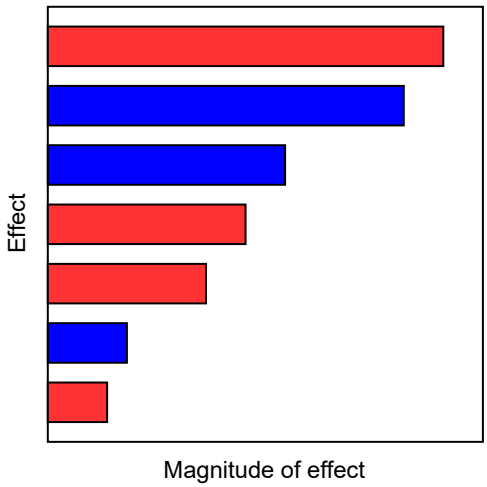


Figure 6: Bar diagram of the magnitude of factor effects. Positive effects in blue and negative in red

Another useful output parameter from ANOVA or linear regression is the **p-value**, which usually tests a null hypothesis H_0 against an alternative hypothesis H_1 such as

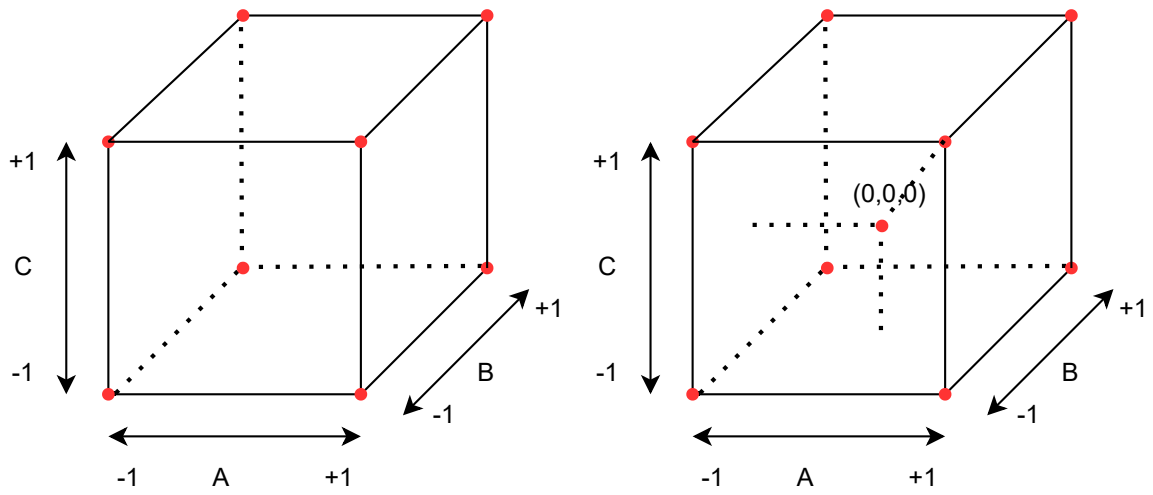
$$H_0 : \mu_1 = \mu_2 = ... = \mu_k$$
$$H_1 : \mu_1 \neq \mu_2 \neq ... \neq \mu_k$$

Here H_0 states that there is no difference in mean μ for k factors, while H_1 states that there is a significant difference. The p-value is then used to determine if an hypothesis can be rejected or not. If p-value $> \alpha$, the hypothesis H_0 can not be rejected, which is inline with that the samples can be considered coming from the same distribution. If p-value $\leq \alpha$ we reject the null hypothesis on level α and conclude that there is a significant difference between observations. Typically α is set to test at 0.05 significance level, meaning that in each test the risk is 5% to reject a true null hypothesis.

2.4.2 Center Points

A risk when running factorial design is missing out on non-linear relationships between factors and the response [18]. An unreplicated design has even bigger flaws since replications are necessary in order to estimate experimental error.

By introducing **center points** to the factorial design, the model gains the ability to detect second-order effects and adds well needed replications for error estimate. The idea is very simple, a 2^3 factorial design can be visualized as a cube as in Figure 7a, where each corner represents one experiment. A center point is then placed in the absolute center of the cube, creating a point between all levels. Important here is that all factors have to be quantitative and continuous, where a center value between two levels can be found. Complementing the design matrix from Table 2 with center points results in Table 5. It is recommended to spread the center points out during the experiment, preferably at the beginning, in the middle and in the end. If possible, center points should always be included in a factorial design.



(a) 2^3 **(b)** 2^3 with center point

Figure 7: Factorial design cubes

	Factors		
Exp no.	**A**	**B**	**C**
1	0	0	0
2	-1	-1	-1
3	+1	-1	-1
4	-1	+1	-1
5	+1	+1	-1
6	0	0	0
7	-1	-1	+1
8	+1	-1	+1
9	-1	+1	+1
10	+1	+1	+1
11	0	0	0

Table 5: Design Matrix with added center points

Apart from being able to estimate the experimental error, and for checking drift in the experimental setup, an additional benefit with center points is that they make it possible to check if there is a linear relationship between factor levels. By drawing a line between the high- and low levels, we can compute the distance from the drawn line to the center points as shown in Figure 8. If the distance is significant, curvature between points can be assumed.

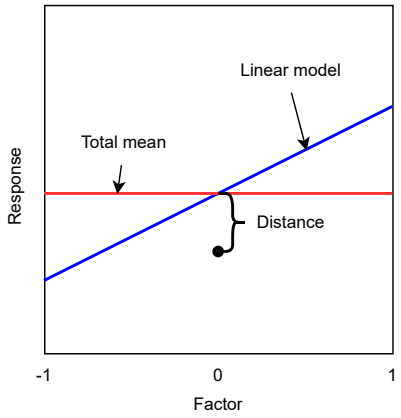


Figure 8: Using center points to assess the deviation from linearity

To confirm this, we can use a statistical tool called a permutation test. This test does not make any assumptions about the distribution of the data and is useful for testing curved relationships. How permutation tests work can be explained with an example. Figure 9 shows a group receiving treatment in red and a control group in blue. This is the observed data and a

test statistic (such as mean) can be calculated by *Test statistic* $= \mu_T - \mu_C$, where μ_T & μ_C are the mean of the treatment and control group, respectively. This value will be important for later and therefore saved in the second image in Figure 9.

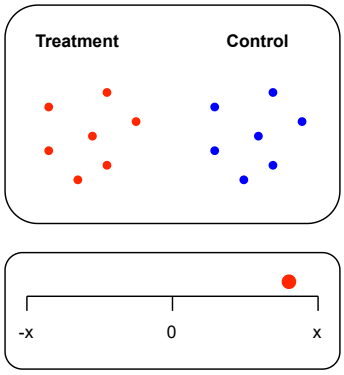


Figure 9: Observed data. Treatment and control group above and test statistics below

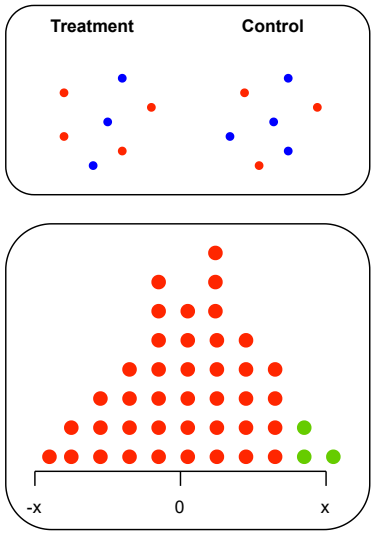


Figure 10: Permutation distribution. Resampling data above and permutation distribution below

Then in Figure 10, a permutation distribution is created by re-sampling the data many times. Each time, the treatment and control group are randomly shuffled, and a new test statistic is computed for the re-sampled data. After enough collected samples we are left with a, in this case one-sided, permutation distribution as the bottom image in Figure 10. Here the green dots represent the number of test statistics that are more extreme than the observed value. Based on this number, the p-value can be calculated to determine whether it can be explained by chance or not.

2.4.3 Residuals & normality testing

When performing ANOVA, we make the assumption that our data is normally distributed. Before any conclusions can be drawn, it is important to verify if this assumption is true. A convenient way to do so is by computing the difference between the measured values and the predicted values, also known as **residuals**. Noting the residuals as ϵ, are then calculated by (9)

$$\epsilon_{ij} = y_{ij} - \hat{y}_{ij} \tag{9}$$

where $\hat{y}_{ij}$ is the estimated value of observation y_{ij} for treatment i and observation j.

Residuals are then analyzed through graphical visualization. A normal probability plot is an effective approach for visual assessment of the distribution of the data, where the residuals are plotted against the theoretical values, i.e. it helps determine whether the residuals follow a normal distribution.

The advantage of using residuals to check normality opposed to raw data is that potential noise and errors can be seen as normally distributed if residuals are. A second tool is the residuals vs fitted values plot, used to assess the linearity and constant variance assumptions of a linear regression model. It plots the residuals against the fitted values, i.e. the expected values. A random scatter of points with no clear pattern indicates that the assumptions are met, while an apparent pattern, such as a heteroscedastic or quadratic

shape, suggests that the model needs further attention.

2.4.4 Response Surface Method

In a scenario where the assumption of linearity is violated, quadratic effects can be assessed using Response surface methodology (RSM). RSM is a technique for modeling and analyzing how a response variable of importance is influenced by different factors. The aim is to find the relation between factors and response that optimizes the response.

Describing the result of such relation for a linear function, a first-order model can be represented as the response function (10)

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \epsilon \tag{10}$$

where y is the response, x_j factors, β_j coefficients and an error term ϵ. However, this representation only include main effects, it is therefore wise to extend the equation with the interaction term $x_i x_j$.

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \sum \sum_{i<j} \beta_{ij} x_i x_j + \epsilon \tag{11}$$

Since assuming perfect linearity is often problematic, this addition gives more flexibility because first-order interactions can be approximated. At times when the quadratic effect is too prominent to fit equation (11), a second-order response function is more suitable.

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \sum \sum_{i<j} \beta_{ij} x_i x_j + \sum_{j=1}^{k} \beta_{jj} x_j^2 + \epsilon \tag{12}$$

After choosing the appropriate response function, the surface is represented graphically and analyzed.

Calculating the coefficients $\beta_i, i = 0...k$ to fit the regression model is related to equation (6), by taking 1/2 of the (fixed) effects

$$\beta_0 = \overline{y} \qquad \beta_1 = A/2$$
$$\beta_2 = B/2 \qquad \beta_{12} = AB/2$$

with exception of the intercept β_0 that is the average of all responses. Center points only contribute to the intercept and will not interfere when calculating normal factor effects.

Central Composite Design

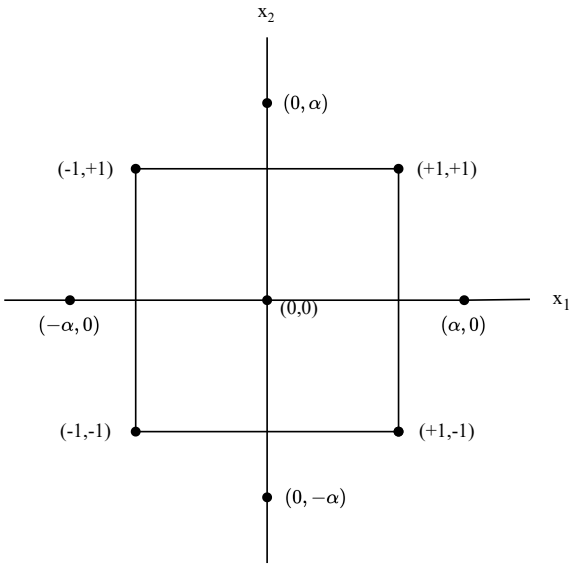
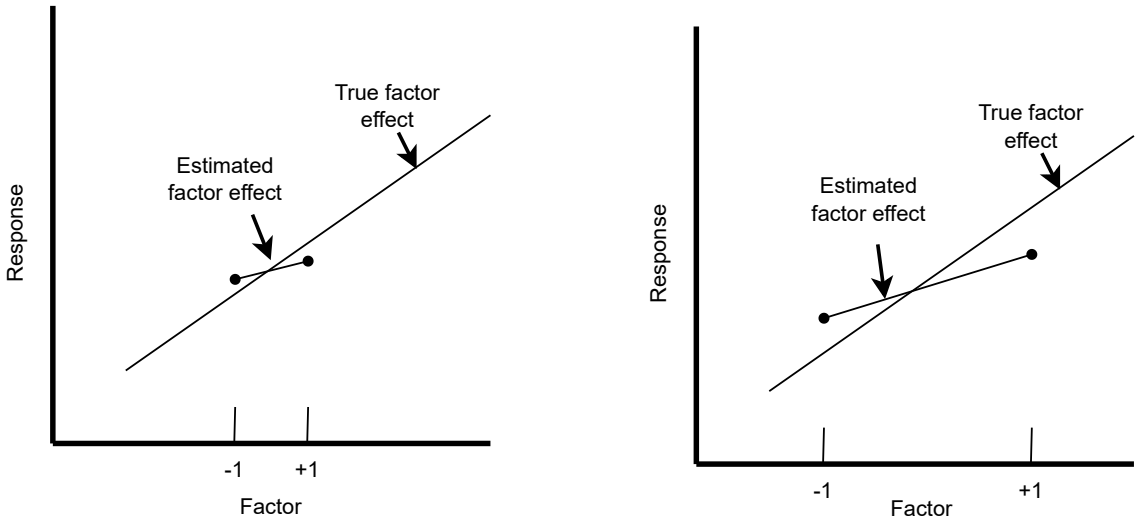


Figure 11: CCD with 2 factors [11]

When conducting RSM, it is common to use central composite design (CCD) to add additional data points that allow for building a second-order response surface. It usually consists of a 2^k factorial design with 3-5 center points and $2k$ so called axial points. In Figure 11 we see an example of a 2^2 design with the regular corner- and center points. The axial points are placed at $(\pm\alpha, 0)$ and $(0, \pm\alpha)$, where the choice of α depends on how we want to model the design.



(a) Narrowly spread out factors

(b) Aggressively spread out factors

Figure 12: Effects of spreading out factors [11]

2.4.5 The unreplicated test

In reinforcement learning, the number of tests has gradually decreased over the years due to the increasing computational expense [19]. However, the importance of replications was demonstrated by Colas et al. [1], who reviewed several papers on deep RL and made a concerning discovery. None of the surveys used more than 5 seeds, which resulted in a 51% chance of false negatives (i.e. failing to reject a false null hypothesis). Furthermore, inappropriate statistical methods were used when evaluating. Although cutting corners may be unavoidable due to practical limitations, assuming that any method works for analysis is a dangerous practice. Instead, it is important to choose an appropriate method based on the specific situation.

In terms of DOE there are several helpful tools if resources are limited that still allow for interesting observations given few tests. When performing an unreplicated factorial design during screening, a smart trick is to aggressively spread out factors as depicted in Figure 12. This increases the chances of detecting important effects that might have been missed otherwise. Imagine doing the factorial design in Table 2 without replications, i.e. only one test for each combination. The clear risk of this is introducing undesirable noise to the system, which in turn can cause misleading output. By embracing the uncertainty and using aggressive factor spreading, we can make the most of limited resources and still obtain valuable information from unreplicated factorial designs.

Secondly, factorial design cannot estimate experimental error unless some runs are replicated. To provide these replications without replicating all permutations in the design, some center points are added to the test. This does not influence factor effects β_i ($i > 0$) but adds to the grand average β_0 in Eq. (10).

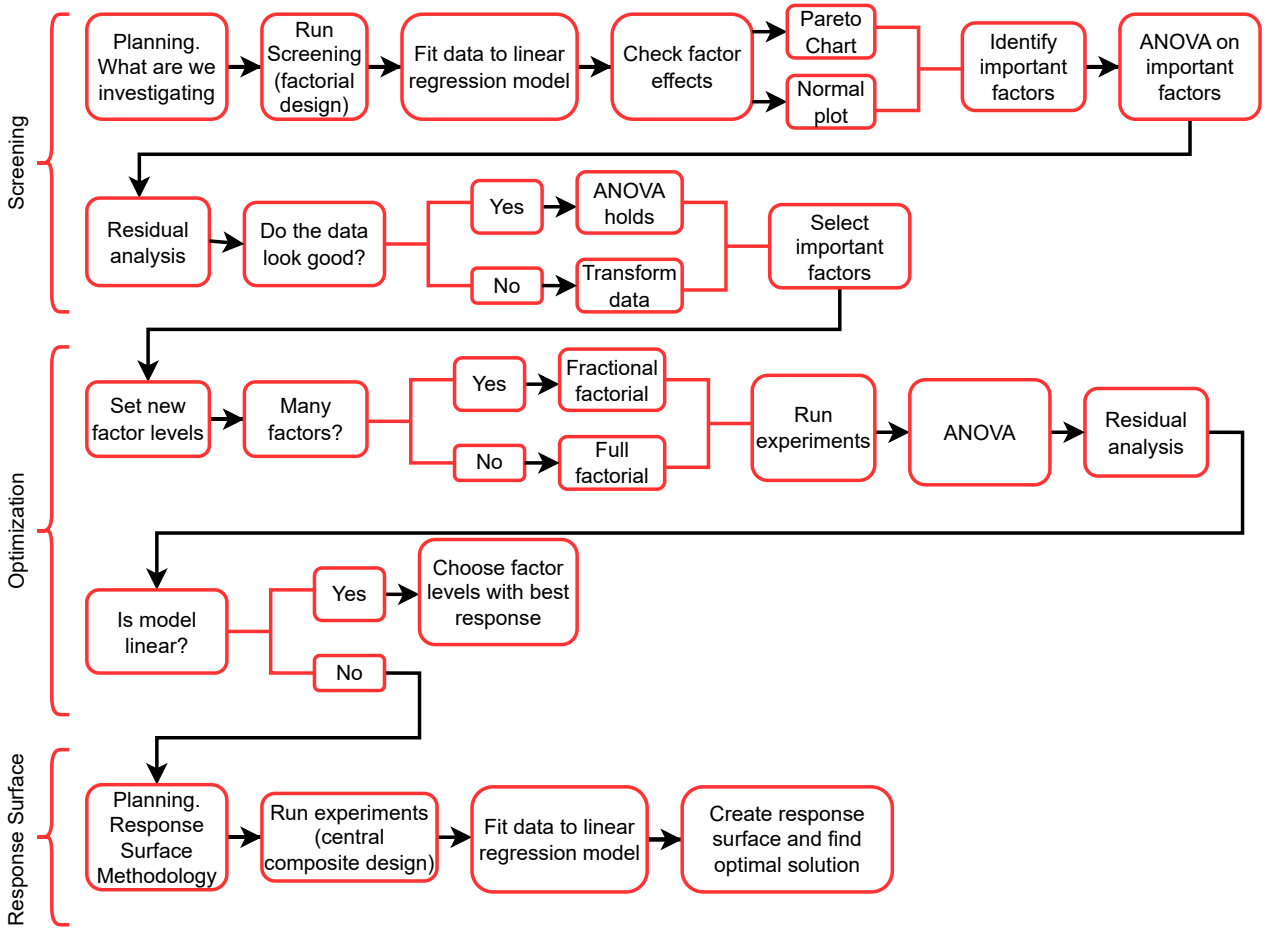


Figure 13: A diagram showcasing the method for the planned execution of experiments

3 Method

The method for this thesis can be divided into three separated parts, namely screening followed by an optimization experiment and lastly an optional response surface for optimization. Figure 13 shows a general approach on how this thesis was conducted in order to answer the research questions.

3.1 Screening Experiment

The experiment began by deciding which factors to investigate and the most appropriate response to measure. Testing multiple factors at a low cost is key, therefore a fractional factorial design was appropriate. Factors were spread out to increase the chance of detecting significant effects as well as including center points.

The sampled data were fitted to a linear regression model and factor effects were visualized using a normal probability plot and Pareto chart, as previously shown in Figure 5&6. In the normal probability plot, all effects are spaced as if they were normally distributed, closer near the center and with increasing distance towards the tails. If no effect is present, the effect estimates would come from a normal distribution with mean 0, and would thus form a straight line in this diagram. Interesting observations that can be made here are points that deviate from this line, so called outliers. Outliers are the primary subjects when identifying the most important factors, and assumptions can be verified further by a Pareto chart, which plots the magnitude of all effects in a descending bar graph. The best candidates from these graphs were analyzed further using ANOVA, where factors showing statistical significance would be selected.

The last step after reducing the model to only include significant factors was residual analysis, to analyze if the assumptions of the ANOVA and linear regression are fulfilled, i.e. that error terms are normally distributed, independent and with constant variance. Normal distribution was assessed by illustrating the residuals in a normal plot, e.g. Figure 14, which compares the observed quantiles of the residuals to the expected quantiles of a normal distribution. Residuals following a straight line indicate normality. Also present in the plot is a shaded area, it represents the range of values that would be expected if the residuals followed a normal distribution.

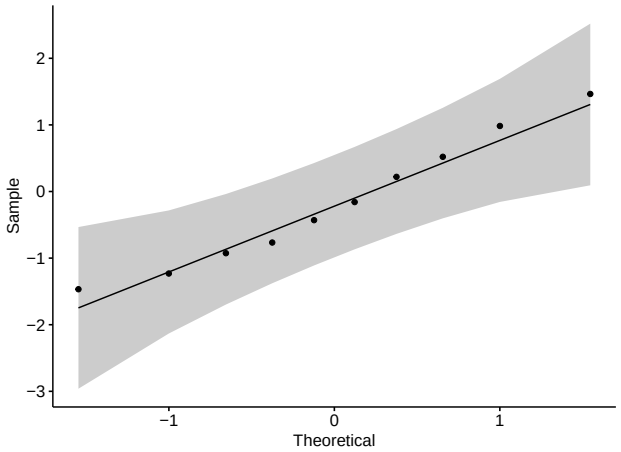


Figure 14: Normal plot for assessing the normality of the residuals

Independence and constant variance was assessed by plotting residuals against their expected values. Ideally all data points are scattered randomly around zero with no visible pattern, behaving as they are expected to. In a scenario where the residuals did not behave linearly nor normally, log-transformation of the data was necessary.

If everything looked good we could proceed to the following step which was a more detailed experiment on the significant factors. New levels were determined based on factor levels with best performance.

3.2 Optimization Experiment

Depending on the number of factors, either a fractional or a full factorial experimental design would be used here. A difference from the screening experiment is that multiple replications were required here in order to get a good and trustworthy estimate of factor effects. Ideally the model has been reduced so it will not be too computationally expensive running all replications. Given the number of replications, there is no longer any need of preselecting factor effects through normal probability or Pareto plots, like in the screening. Thus we can jump straight to ANOVA and analyze the result and residuals.

After conducting all the necessary analyses, we assess whether the data exhibited linearity. If that was the case, we would have reached the end of the survey and chosen the factor levels yielding the best response. But if the data showed signs of non-linearity, we performed a permutation test to confirm its significance. If the permutation test yielded a significant outcome, we proceeded to the next phase, which involved implementing response surface methodology.

3.3 Response Surface Experiment

Optionally, if there is evidence of the model being non-linear, more testing is required in order to determine optimal settings. Response surface method was applied using central composite design. Data collected from the optimization was complemented with new runs in the so called axial points. By fitting all data to a second order response function it could be analyzed the exact same way as before and determined which factors caused non-linearity to occur. The best solution could be obtained through modeling response surfaces graphically. When done and optimal hyperparameters were found, a comparison was made with the default hyperparameters to see if any improvements were made.

4 Experiments

4.1 Environment and Conditions

In this thesis, simulations were carried out using RLZoo which is a collection of reinforcement learning algorithms, frameworks and applications [17]. In this case, the implementation from OpenAI Gym [6] was used and experiments were conducted on the CartPole-V1 task from the classic control environment which are written in Python.

For all experiments, the agent was trained using the algorithm PPO with clipping. The available default hyperparameters for this algorithm are shown in Table 6 which comes from RLZoo's PPO documentation. Some of these hyperparameters were then used as factors for the experiments.

Hyperparameter	Value
train episodes	1000
max steps	200
save interval	50
gamma	0.99
batch size	32
actor update steps	10
critic update steps	10
clip range	0.2
actor learning rate	1e-4
critic learning rate	2e-4

Table 6: Default hyperparameters PPO

4.2 CartPole-V1

In the Cartpole task, an agent tries to balance a pole, earning rewards upon success. The measured response for the experiments was the time it takes for the agent to reach a success rate of 95 % (episode reward 190 of 200), with a running average over the last 100 episodes.

The experiment began with a screening in order to detect the most prominent factors in a cost efficient way. From Table 6, the five hyperparamters batch size, actor update steps, critic update steps, actor learning rate, and critic learning rate were selected as factors. Their respective factor levels are visible in Table 7, which were based on their default settings and then aggressively spread apart.

	Levels		
Factors	**-1**	**+1**	**0**
A: batch size	50	300	175
B: a. update steps	10	150	80
C: c. update steps	10	150	80
D: actor lr	1e-4	1e-3	5.5e-4
E: critic lr	1e-4	1e-3	5.5e-4

Table 7: Selected factor levels for screening

	Factors				
Exp no.	**A**	**B**	**C**	**D**	**E**
1	0	0	0	0	0
2	-1	-1	-1	1	-1
3	1	1	-1	-1	1
4	-1	1	1	-1	1
5	-1	-1	-1	-1	1
6	1	-1	-1	-1	-1
7	1	-1	1	1	-1
8	1	1	-1	1	-1
9	-1	1	-1	-1	-1
10	0	0	0	0	0
11	-1	-1	1	1	1
12	-1	-1	1	-1	-1
13	-1	1	-1	1	1
14	1	-1	-1	1	1
15	1	-1	1	-1	1
16	-1	1	1	1	-1
17	1	1	1	1	1
18	1	1	1	-1	-1
19	0	0	0	0	0

Table 8: 2^{5-1} Design matrix with added center points

In order to estimate all main effects & first order interactions, a minimum of 16 experimental runs were required. Therefore a 2^{5-1} fractional factorial design with 3 center points was the most suitable choice and can be spotted in Table 8. The design matrix was fully randomized so chances of order influencing the result could be eliminated.

Upon completing the screening, the method from Figure 13 was followed closely. The choice of the next model and factor levels depended on our findings from analyzing the result and the number of factors to be included for further testing. Factors that did not show any signs of significance were set to their default values in Table 6 for the remainder of the project.

The new experiment with reduced number of factors differentiates a little bit from the screening. The factor levels were narrowed down to simplify the search of optimal settings. Secondly, multiple replications of each data point allowed for good error estimate and a more reliable ANOVA test. In a scenario were assumptions of linearity are violated, a permutation test on the center points would be required in order to determine if further analysis was necessary. In that case the sampling was extended to allow for response surface estimation.

5 Analysis of Result

5.1 Screening

Using the outputs measured in the screening experiment of Table 8, we began by estimating all factor effects (including two-way interactions). These were visualized in the normal probability plot in Figure 15. Factor A (batch size) and B (actor update steps) look like outliers since they deviate more from the rest, which makes them good candidates for further testing. The first order interaction between B and D (actor learning rate) is a little bit in the gray zone, close but not too close to the rest and showing a fairly large effect. It is worth noting that the data had to be log-transformed after an initial analysis, in which the data exhibited signs of non-linearity as well as non-normality.

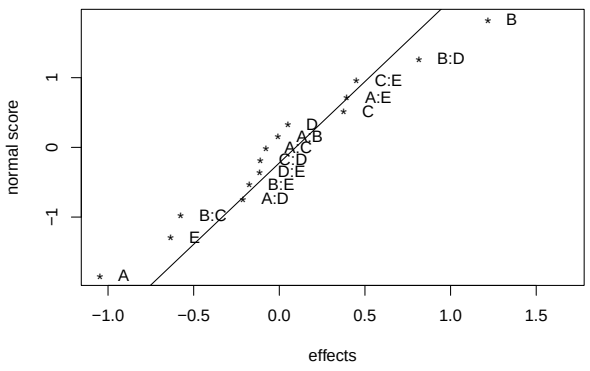


Figure 15: Normal probability plot of factor effects in the screening experiment

The effect magnitude of A, B, and B:D is made more clear after plotting a Pareto chart in Figure 16. Factor A and B stand out a lot from the rest which is in line with our assumptions. As for the interaction B:D, the large factor effect can not be ignored and the same goes for Factor E (critic learning rate). As a consequence of including B:D, factor D should be analyzed as well.

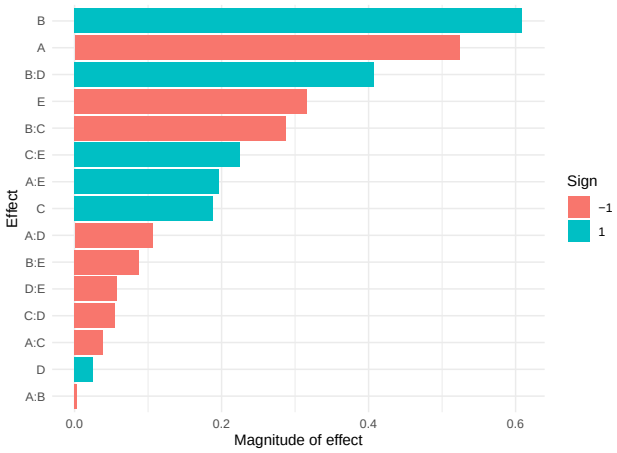


Figure 16: Pareto Chart of factor effects in the screening experiment

We performed ANOVA on a reduced model only containing factors A, B, D, E and the interaction B:D. This works to some extent since enough degrees of freedom are available to estimate errors, compared to before where the model was saturated. Keep in mind that since no replications of the data points were made except for the center points, the result may not be conclusive. However, they will give additional information whether or not our expectations about the factors are correct. The obtained p-values for the factors were

Factor	Effect	p-value
A: batch size	10.533	**0.00638**
B: a update steps	14.245	**0.00232**
D: actor lr	0.024	0.87848
E critic lr	3.850	0.07153
B:D	6.377	**0.02535**

Table 9: Output from ANOVA in the screening experiment

Both A and B are significant with p-value < 0.05 just like we suspected, while the evidence of an effect of E is very weak (p-value$>$ 0.05). Since there is not enough evidence to include factor E further, we neglect it along with all factors whose

effects are smaller than the effect of E in Figure 16. Interestingly, the interaction B:D is significant but since factor B has a very large effect and factor D is almost non-existent, it is possible that B completely dominates and would require more testing to say for sure. However, given the negligible influence of factor D, as indicated by its large p-value of 0.87848, we find it of little interest to include it in subsequent analysis and therefore also the interaction B:D. The aim is always to reduce the number of factors if possible, which is why we choose to only proceed with factor A and B.

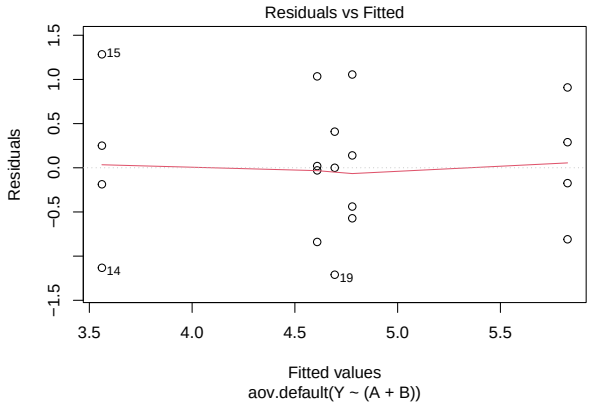


Figure 17: Residuals plotted against fitted values in the screening experiment

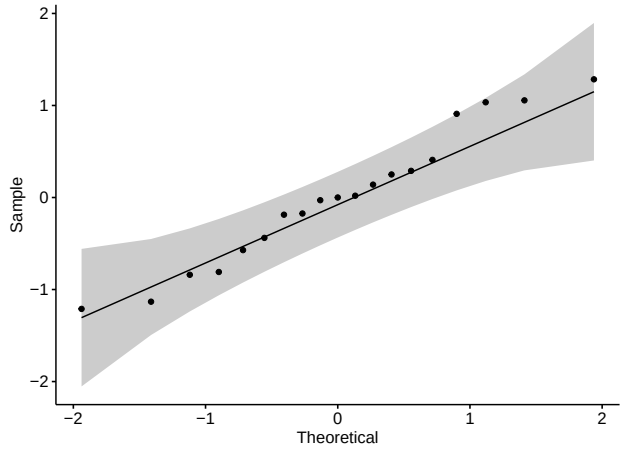


Figure 18: Normal probability plot of residuals in the screening experiment

Residual analysis was performed on factors A and B, as depicted in Figure 17, which shows the residuals against the fitted values. It looks good with data points scattered randomly around zero, behaving as expected, and the assumption of linearity holds. The normal probability plot in Figure 18 looks fine as well, the residuals follow the line and stays within the shaded area. The sampled data can be considered normally distributed and trustworthy.

5.2 Optimization

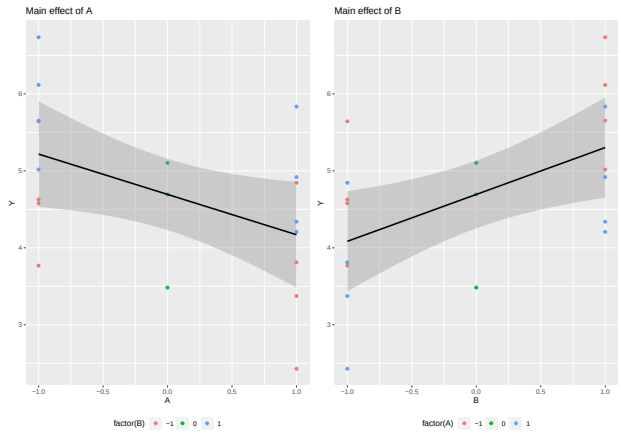


Figure 19: A and B main factor effects from the screening experiment

For the optimization experiment, new factor levels were selected based on the main effects of A and B from the screening in Figure 19. Low run time was desired, which occurred when factor A was high and B was low. The new levels were therefore narrowed down and set according to Table 10.

	Levels		
Factors	**-1**	**+1**	**0**
A: batch size	200	300	250
B: a update steps	10	50	30

Table 10: Selected factor levels in the optimization experiment

With only two factors to investigate, a 2^2 full factorial design was chosen as in Table 11. Each test was replicated 5 times and conducted in a fully randomized order, resulting in a total of 25 experimental runs of which 5 were center points.

	Factors	
Exp no.	**A**	**B**
1	-1	-1
2	1	1
3	1	-1
4	-1	1
5	0	0

Table 11: 2^2 full factorial design in the optimization experiment

Neither factor A or B showed any significance after performing ANOVA, with p-values of 0.975 and 0.131, respectively. This lack of significance probably depends on our newly set levels being too close to each other. This means that it does not matter which values we choose but before drawing conclusions we need to analyze the data. The sampled data looks fairly normally distributed in Figure 20.

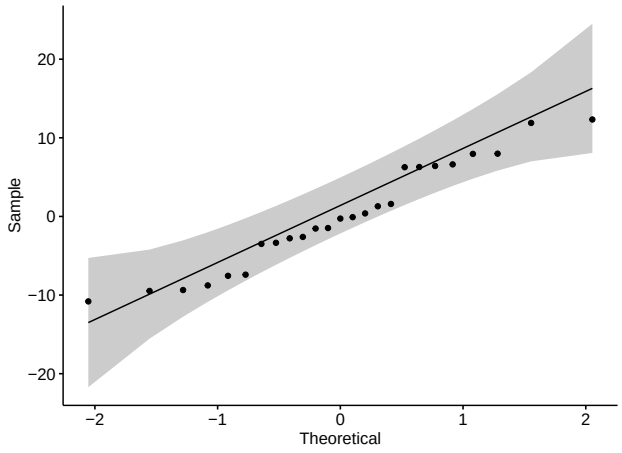


Figure 20: Normal probability plot of residuals in the optimization experiment

More concerning are the residuals versus the fitted values in Figure 21. A clear visible pattern is formed and the center points are not behaving as expected if the model were to be linear. Log-transformation of the data had no effect here, so we suspect there is a potential curvature in the model. To verify if the deviation from linearity in Figure 21 can be explained by chance, we performed a permutation test. The test measures the distance from the mean value of the center points, marked with a blue cross in Figure 22, to the red line which is the mean of all corner points. The plot shows the main effect of factor B since it had the most noticeable effect of the two.

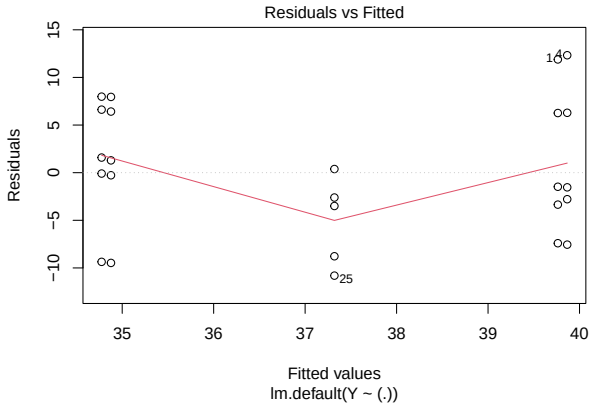


Figure 21: Residuals plotted against fitted values in the optimization experiment

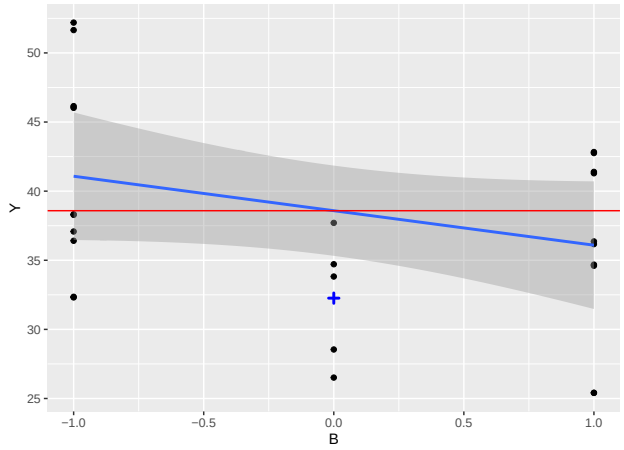


Figure 22: Factor B main effect with center points to assess deviation from linearity

Figure 23 shows the (absolute value of) two-sided permutation distribution with our sample in red located quite close to the tail. The obtained p-value from the test was 0.03715 which is significant on a 0.05 significance level, we conclude that the model is non-linear. More experiments were required in order to determine what the curvature looks like.

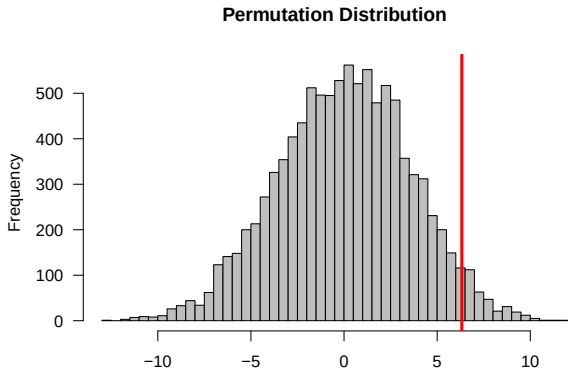


Figure 23: Permutation test for investigating quadratic effects

The reason we experience curvature now and not during the screening, can be due to the increased sample size. Even a model with significant curvature can appear almost linear when examined within a small area of a point. Therefore, we believe that the increased sample size, both overall and specifically for the center points, enables us to detect the deviation from linearity. This suggests that the larger sample size provides a more accurate representation of reality.

5.3 Response Surface Method

When performing response surface method, a central composite design was used. All previous collected data was reused and complemented with the additional axial points from Table 12, resulting in the design matrix in Table 13. All axial points were replicated 5 times just like previously sampled data.

	Levels	
Factors	$-\sqrt{2}$	$\sqrt{2}$
A: batch size	179	321
B: a update steps	2	58

Table 12: Axial points in central composite design

	Factors	
Exp no.	**A**	**B**
1	-1	-1
2	1	-1
3	-1	1
4	1	1
8	$-\sqrt{2}$	0
9	$\sqrt{2}$	0
10	0	$-\sqrt{2}$
11	0	$\sqrt{2}$
12	0	0

Table 13: Central composite design matrix

The results of the ANOVA on a second-order regression model (see Table 14) indicate that there is a significant main and quadratic effect caused by factor B. We can conclude the assumption of non-linearity to be correct. Analyzing the residuals after accounting for the curvature, the residuals vs fitted plot in Figure 24 looks much better than Figure 21 in the optimization experiment, with points randomly scattered around zero.

Factor	**Effect**	**p-value**
A: batch size	-1.80005	0.4968
B: a. update steps	-11.68966	**6.877e-05**
A:B	-0.08205	0.9825
A^2	-3.08060	0.4832
B^2	20.42495	**3.277e-05**

Table 14: ANOVA output in response surface experiment, significant in bold

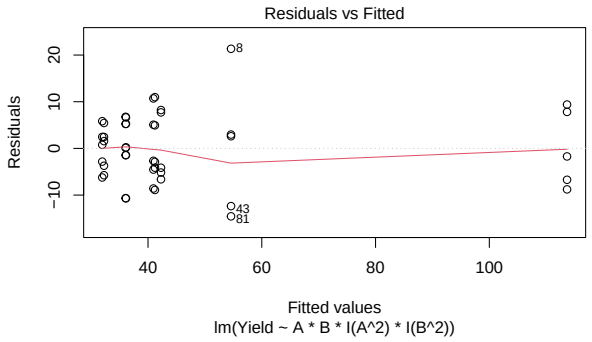


Figure 24: Residuals vs fitted values in the response surface experiment

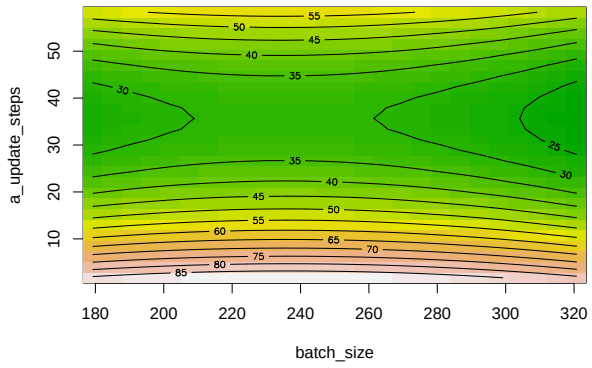


Figure 25: Contour plot in the response surface experiment

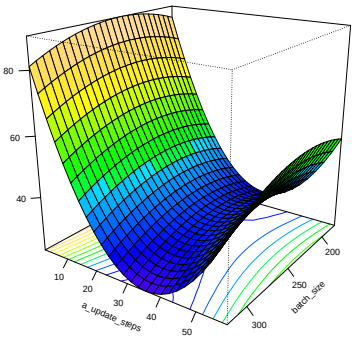


Figure 26: 3D plot in the response surface experiment

A response surface of the result was modeled both as a contour plot in Figure 25 as well as in 3D, Figure 26. A saddle like shape was formed with a stationary point when batch size is 235 and actor update steps is 36. Since batch size has no significant impact here, we can consider this point as the optimal solution to the problem.

5.4 Performance with Optimal Hyperparameters

With the optimal hyperparameter values found for this specific case, a final evaluation was done by comparing our experimental hyperparameters and the default hyperparameters. The final result can be seen in Figure 27, where the blue line is our optimized hyperparameters, the red line is from default settings and shaded area is the standard error. Both were simulated 13 times and the mean value from the runs are shown here. Our settings performed a little bit better by the looks of it, reaching the threshold of 95% success rate (when episode reward hits 190) after 38,361 seconds compared to the default model's 109,037 seconds, making it 2.85 times faster.

6 Discussion and Conclusion

6.1 Conclusion

The aim of this project was to investigate if design of experiments could be implemented for optimizing reinforcement learning algorithms used for solving robotic tasks.

Based on the final comparison between our optimized hyperparameters and default hyperparameters in figure 27, we can conclude that the use of design of experiments was successful. A noticeable difference can be observed, as our model achieved the desired success 2.85 times faster than the default model.

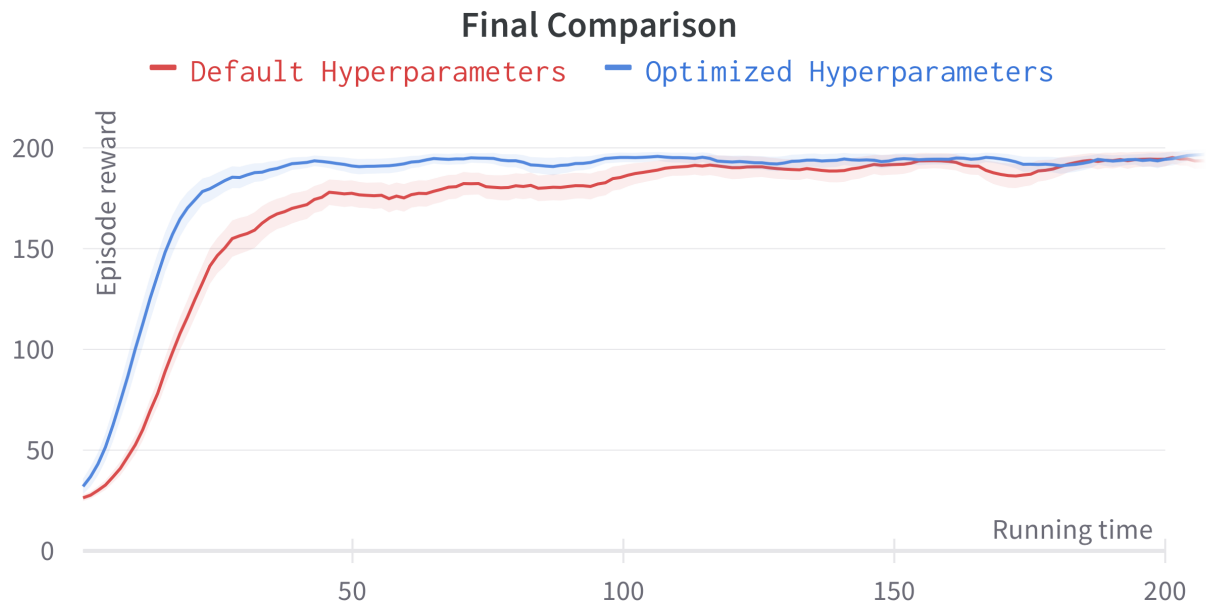


Figure 27: Comparing performance from simulations between optimized hyperparameters and default hyperparameters

The chosen method, as depicted in Figure 13, effectively detected the significant factors batch size and actor update steps which allowed for model reduction and optimization. We were able to identify the non-linearity caused by the actor update steps factor and assess the curvature, using response surface methodology.

In conclusion, our project successfully investigated the implementation of design of experiments for optimizing reinforcement learning algorithms in robotic tasks.

6.2 Discussion

In this study, different hyperparameters from the RL algorithm PPO were chosen as factor for a screening experiment using fractional factorial design. Experiments were conducted on the CartPole-V1 task from the benchmark environment RLZoo, where the response was measured as the time it takes until success rate reached 95%. The goal of the screening was to analyze which factors had the biggest effect on the response in a less computationally expensive way. By including only the important factors for subsequent testing, the model could be reduced, requiring fewer experiments for optimization.

Batch size and actor update steps were the most influential factors from the screening experiment, and therefore selected for the following optimization experiment. In the resulting residual analysis, the factors formed a non-linear relationship, which was also verified through a permutation test. In order to account for the quadratic effect, additional experiments were conducted and a response surface was created. It could be determined that actor update steps was the cause of said second order effects. Through response surface modulation, optimal hyperparameters were found and a final comparison was made between default- and experimental hyperparameters. The

importance of adding center points was put on display here, as without them, we would have likely missed the curved relationship.

After analyzing the final comparison between our model and the default one in Figure 27, we can see a raise in performance. As a result of this, we can consider the use of design of experiments as successful. The choice of factors and their levels yielded significant effects. Spreading out factors aggressively during the screening was really helpful in reducing the risk of missing effects.

Deciding on a good way to measure the response was a bit more challenging. Since not all experimental runs experienced the same success, there were difficulties finding an equally fair way to evaluate and compare the outcomes across different runs. Ultimately, we found a performance metric that fulfilled the criteria of consistency, but figuring out better alternatives could produce a more accurate representation of the factor effects and elevate the result further.

Obviously, the training can be optimized further. In our experiment we neglected factors without statistical significance, this does not mean these factors are completely irrelevant. They can still be tuned to enhance the performance to a higher degree, but that is not what design of experiments is all about. Our goal was to achieve as much as possible with minimal effort by systematically selecting what we consider important.

Design of experiments is most commonly used in real-world problems such as manufacturing processes, pharmaceutical testing, and agriculture. In a simulated environment as in this project, methods for optimizing hyperparameters exist but there is always room for quality improvements. Using this approach might offer a refreshing perspective on how optimization can be done, and in combination with existing optimization methods, allow for interesting discoveries as well as deeper understanding that would not be possible without design of experiments.

The principles and methods demonstrated can be generalized beyond the scope of hyperparameter optimization and any researcher can use them in a way they see fit. All in all, experimental design is relatively unknown in general and deserves some recognition since it is versatile and applicable in numerous fields.

6.3 Future Work

The natural continuation of this work is to conduct the RLBench experiments that never took place. The limitation with the design of experiments I conducted was the requirement of quantitative and continuous factors. It would therefore be interesting with a comparative study on different qualitative settings. In my planned RLBench experiment I would only test for the same algorithm and action- & observation abstraction on one task. Making a statistical analysis and comparing different algorithms and action- & observation abstractions for multiple tasks in an efficient way would result in some interesting findings.

References

[1] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. "How many random seeds? statistical power analysis in deep reinforcement learning experiments". In: *arXiv preprint arXiv:1806.08295* (2018).

[2] Alexander Dürr, Volker Krueger, and Elin Anna Topp. *Towards a comprehensive study to identify the relevant I/O to support or complement Reinforcement Learning.* RL-CONFORM workshop. Oct. 2022.

[3] Thomas Nakken Larsen m.fl. "Comparing Deep Reinforcement Learning Algorithms' Ability to Safely Navigate Challenging Waters". In: *Frontiers in Robotics and AI* 8 (2021). ISSN: 2296-9144. DOI: 10.3389/frobt.2021.738113.

[4] L. Eschapasse Q. He J.L. Liu. "A comparison of reinforcement learning models of human spatial navigation". In: *Sci Rep* 12 (2022). DOI: 10.1038/s41598-022-18245-1.

[5] Ching-An Wu. "Investigation of Different Observation and Action Spaces for Reinforcement Learning on Reaching Tasks". In: (2019). URL: urn:nbn:se:kth:diva-271182.

[6] Greg Brockman et al. *OpenAI Gym.* 2016. DOI: 10.48550/ARXIV.1606.01540. URL: https://arxiv.org/abs/1606.01540.

[7] Markus Giftthaler et al. "The control toolbox — An open-source C++ library for robotics, optimal and model predictive control". In: *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR).* IEEE, May 2018. DOI: 10.1109/simpar.2018.8376281. URL: https://doi.org/10.1109%2Fsimpar.2018.8376281.

[8] Stephen James et al. "RLBench: The Robot Learning Benchmark & Learning Environment". In: *CoRR* abs/1909.12271 (2019). arXiv: 1909.12271. URL: http://arxiv.org/abs/1909.12271.

[9] John Schulman et al. *Proximal Policy Optimization Algorithms.* 2017. DOI: 10.48550/ARXIV.1707.06347. URL: https://arxiv.org/abs/1707.06347.

[10] Luigi Nardi, David Koeplinger, and Kunle Olukotun. *Practical Design Space Exploration.* 2018. DOI: 10.48550/ARXIV.1810.05236. URL: https://arxiv.org/abs/1810.05236.

[11] Douglas C. Montgomery. *Design and Analysis of Experiments, Eighth edition.* John Wiley & Sons, 2012. ISBN: 978-1-118-14692-7.

[12] George E. P. Box. *Statistics for Experimenters: Design, Innovation, and Discovery, Second Edition.* Wiley, 2005, pp. xvi+481. ISBN: 978-0471-71813-0.

[13] Torbjörn Lundstedt et al. "Experimental design and optimization". In: *Chemometrics and Intelligent Laboratory Systems* 42.1 (1998), pp. 3–40. ISSN: 0169-7439. DOI: https://doi.org/10.1016/S0169-7439(98)00065-3.

[14] Natália Maria Puggina Bianchesi et al. "A Design of Experiments Comparative Study on Clustering Methods". In: *IEEE Access* 7 (2019), pp. 167726–167738. DOI: 10.1109/ACCESS.2019.2953528.

[15] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. ISBN: 0262039249.

[16] Olivier Buffet, Olivier Pietquin, and Paul Weng. *Reinforcement Learning*. 2020. DOI: 10.48550/ARXIV.2005.14419. URL: https://arxiv.org/abs/2005.14419.

[17] Zihan Ding et al. *Efficient Reinforcement Learning Development with RLzoo*. 2020. DOI: 10.48550/ARXIV.2009.08644. URL: https://arxiv.org/abs/2009.08644.

[18] Sourav Das and Solomon Tesfamariam. *State-of-the-Art Review of Design of Experiments for Physics-Informed Deep Learning*. 2022. DOI: 10.48550/ARXIV.2202.06416. URL: https://arxiv.org/abs/2202.06416.

[19] Rishabh Agarwal et al. "Deep Reinforcement Learning at the Edge of the Statistical Precipice". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 29304–29320. URL: https://proceedings.neurips.cc/paper/2021/file/f514cec81cb148559cf475e7426eed5e-Paper.pdf.

[20] Antonin Raffin et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

Appendix

RLBench Experiment

In this experiment, the benchmark environment RLBench was used together with the algorithm implementations from Stable-Baseline3 [20]. The measured response was the number of environment steps required for the accounted episode rollout reward to converge.

Task	Reach target
Algorithm	PPO
Action abstraction	Cartesian control end effector with planning
Observation abstraction	Proprioceptive task information

Table 15: Test setup

	Levels		
Factors	**-1**	**+1**	**0**
A: gamma	0.5	0.999	0.7495
B: gae_lambda	0	0.98	0.49
C: max_episode_timesteps	1	101	51
D: learning_rate	$1 * 10^{-4}$	$1 * 10^{-3}$	$5.5 * 10^{-4}$
E: clip_range	0.1	0.3	0.2
F: n_episode_rollout	50	450	250

Table 16: Selected factor levels for screening. Batch size was kept static at 50

Exp.	A	B	C	D	E	F
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	-1	-1	-1	1	1	1
4	1	1	-1	1	-1	-1
5	1	1	1	1	1	1
6	1	-1	-1	-1	-1	1
7	0	0	0	0	0	0
8	1	-1	1	-1	1	-1
9	-1	1	-1	-1	1	-1
10	-1	1	1	-1	-1	1
11	-1	-1	1	1	-1	-1
12	0	0	0	0	0	0
13	0	0	0	0	0	0

Table 17: 2^{6-3} design matrix for screening experiment with 5 center points

The intended design matrix in Table 17 shows a $2^6 - 3$ design, which is the smallest possible design for 6 factors. This design would then be complemented with additional runs, depending on which factors showed signs of significance after an initial factor effect analysis. A technique called folding, which has not been discussed in this work, would be applied. Basically, we perform the inverse of already performed experimental runs, adding new combinations. This is important in order to reduce confounding effects between main effects and higher order interactions.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2023-06-01

EXAMENSARBETE Optimizing Reinforcement Learning Algorithms Using Design Of Experiments
STUDENT Anton Fristedt
HANDLEDARE Linda Hartman (LTH)
EXAMINATOR Elin A. Topp (LTH)

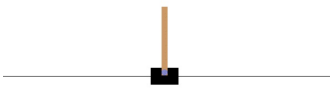
Kan försöksplanering fylla ett tomrum inom förstärkningsinlärning?

POPULÄRVETENSKAPLIG SAMMANFATTNING **Anton Fristedt**

Försöksplanering är vanligt förekommande vid optimering av processer inom verkliga problem, så som tillverkning, hälsovård och jordbruk. I detta arbete undersöktes huruvida försöksplanering kan användas vid optimering inom en datorsimulerad miljö, mer specifikt, optimering av förstärkningsinlärningsalgoritmen Proximal Policy Optimization (PPO).

Inom förstärkningsinlärning finns det ingen enskild inställning som löser alla problem. Att hitta optimala hyperparametrar (en algoritms justerbara värden vid inlärning) är därför en ständig utmaning. Optimering är en tidskrävande process där det finns utrymme för kvalitetsförbättringar mellan ett begränsat antal simuleringar och tillräckligt med statistiskt underlag.

I försöksplanering observerar vi hur en förändring i ingångsvärden orsakar en förändring i utgående värden, i jakt efter de mest betydelsefulla ingångsvärderna, även kända som faktorer. Med hjälp av statistiska modeller kan vi på ett enkelt sätt avgöra vilka faktorer som har en signifikant effekt, och på så sätt minska komplexiteten genom att bara fokusera på dessa.



Experimenten genomfördes på uppgiften Cartpole från OpenAI Gym, där en enhet försöker lära sig att balansera en stång. Vid lyckade försök erhåller denna enhet en belöning, med det långsiktiga målet att samla ihop så hög belöning som möjligt. Vi använde hyperparametrar från algoritmen PPO som faktorer i experimenten och mätte tiden det tog för enheten att nå den önskade tröskeln på 95% framgång.

Våra resultat visade att användningen av försöksplanering var framgångsrik för att förbättra prestandan hos förstärkningsinlärningsalgoritmen. Signifikanta faktorer optimerades, och utvärderades sedan mot en modell med standardinställningar. Resultatet av denna utvärdering visade att vår modell nådde den önskade tröskeln för framgång 2,85 gånger snabbare.