TECHNISCHE UNIVERSITÄT DRESDEN

IFA LS-AT

Fakultät Elektrotechnik und Informationstechnik Institut für Automatisierungstechnik

Diplomarbeit

zum Thema

Application of Saliency Maps for Optimizing Camera Positioning in Deep Learning Applications

vorgelegt von Leonard-Riccardo Hans Wecke
im Studiengang Informationssystemtechnik, Jg. 2016
geboren am 29.06.1997 in Cottbus

zur Erlangung des akademischen Grades eines

Diplomingenieurs
(Dipl.-Ing.)

Betreuer: Dr. rer. nat. Valentin Khaydarov
Dipl.-Ing. Jonathan Mädler
Verantwortlicher Hochschullehrer: Prof. Dr.-Ing. habil. Leon Urbas
Tag der Einreichung: 09.10.2023

TECHNISCHE UNIVERSITÄT DRESDEN

PROCESS CONTROL SYSTEMS | PROCESS SYSTEMS ENGINEERING

Bereich Ingenieurwissenschaften Professur für Prozessleittechnik & Arbeitsgruppe

Aufgabenstellung für die Studienarbeit für
Leonard Wecke, Matr.Nr. 4646001 Studiengang IST

Anwendung von Saliency-Maps zur Optimierung der Kamerapositionierung in Deep Learning Anwendungen

Kontext

Die Platzierung der Kamera hat einen entscheidenden Einfluss auf den Informationsgehalt in Bilddaten und auf die Performance von ML-basierten Bildverarbeitungssystemen, die mit diesen Daten trainiert werden. Diese Fragestellung ist besonders relevant für die Analyse von in verfahrenstechnischen Apparaten durchführenden Prozessen, bei denen die Kamera aufgrund technischer Restriktionen den Prozess nur beschränkt beobachten kann. Die Zielstellung dieser Arbeit ist die Entwicklung eines technischen Systems zur automatisierten Suche nach einer optimalen Position der Kamera beim Lösen von Klassifikationsaufgaben mittels Deep Learning Ansätze. Als Anwendungsfall soll hierbei die Klassifikation von Strömungsregimen bei der Begasung im Bioreaktor betrachtet werden.

Wissenschaftliche Fragestellungen

Welche Ansätze zur Extraktion zusätzlicher Informationen im Feld Explainable AI für Computer Vision Anwendungen liefern sinnvolle Features zur Auswertung der Kameraposition? Welche Blackbox-Optimierungsverfahren sind geeignet, um das Problem der optimalen Kamerapositionierung zu lösen?

Lastenheft

1. Literaturrecherche und begründete Auswahl der Forschungsmethodik zur Bearbeitung der Fragestellungen. Das schriftliche Ergebnis dieses Arbeitspakets dient als Meilenstein.
2. Zielgerichtete Beantwortung der Fragestellung durch systematische Anwendung der ausgewählten Forschungsmethodik
3. Kritische abschließende Bewertung der gewählten Arbeitsweise und der Forschungsergebnisse
4. Die Arbeit ist gemäß der Richtlinie des Instituts für Automatisierungstechnik durchzuführen.
5. Die Datenbeschaffung und Evaluation des Konzepts erfolgen eigenständig im Process-To-Order-Lab.
6. Die geforderte Programmiersprache für Bildverarbeitung und Modellierung ist Python.
7. Für die ML-Experimente ist MLflow oder Tensorboard zu verwenden.

Betreuer:	Dr. rer. nat. Valentin Khaydarov
1. Prüfer:	Prof. Dr.-Ing. habil. Leon Urbas
2. Prüfer:	Prof. Dr.-Ing. Thomas Walther
Datum Arbeitsbeginn:	08.05.2023
Einzureichen am:	09.10.2023

Prof. Dr.-Ing. habil. Martin Wollschlaeger
Vorsitzender des Prüfungsausschusses Informationssystemtechnik

Prof. Dr.-Ing. habil. Leon Urbas
Verantwortlicher Hochschullehrer

TECHNISCHE UNIVERSITÄT DRESDEN

IFA LS-AT

Fakultät Elektrotechnik und Informationstechnik Institut für Automatisierungstechnik

Anwendung von Saliency-Maps zur Optimierung der Kamerapositionierung in Deep Learning Anwendungen

Im Bereich der Prozessleittechnik und Robotik, speziell bei der automatischen Steuerung, treten oft komplexe Optimierungsprobleme auf. Diese Arbeit konzentriert sich auf die Optimierung der Kameraplatzierung in Anwendungen, die Convolutional Neural Networks (CNNs) verwenden. Da CNNs spezifische, für den Menschen nicht immer ersichtliche, Merkmale in Bildern hervorheben, ist die intuitive Platzierung der Kamera oft nicht optimal.

Zwei Forschungsfragen leiten diese Arbeit: Die erste Frage untersucht die Rolle von Erklärbarer Künstlicher Intelligenz (XAI) in der Computer Vision zur Bereitstellung von Merkmalen für die Bewertung von Kamerapositionen. Die zweite Frage vergleicht einen darauf basierenden Algorithmus mit anderen Blackbox-Optimierungstechniken. Ein robotisches Auto-Positionierungssystem wird zur Datenerfassung und für Experimente eingesetzt.

Als Lösungsansatz wird eine Methode vorgestellt, die XAI-Merkmale, insbesondere solche aus GradCAM++ Erkenntnissen, mit einem Bayesschen Optimierungsalgorithmus kombiniert. Diese Methode wird in einer Fallstudie zur Klassifizierung von Strömungsregimen in industriellen Bioreaktoren angewendet und zeigt eine gesteigerte performance im Vergleich zu etablierten Methoden. Zukünftige Forschung wird sich auf die Sammlung weiterer Daten, die Inklusion von verrauschten Daten und die Konsultation von Experten für eine kostengünstigere Implementierung konzentrieren.

Betreuer:	Dr. rer. nat. Valentin Khaydarov
	Dipl.-Ing. Jonathan Mädler
Hochschullehrer:	Prof. Dr.-Ing. habil. Leon Urbas
Tag der Einreichung:	09.10.2023

DIPLOMARBEIT

Bearbeiter: Leonard-Riccardo Hans Wecke

TECHNISCHE UNIVERSITÄT DRESDEN

IFA LS-AT

Fakultät Elektrotechnik und Informationstechnik Institut für Automatisierungstechnik

Application of Saliency Maps for Optimizing Camera Positioning in Deep Learning Applications

In the fields of process control engineering and robotics, especially in automatic control, optimization challenges frequently manifest as complex problems with expensive evaluations. This thesis zeroes in on one such problem: the optimization of camera positions for Convolutional Neural Networks (CNNs). CNNs have specific attention points in images that are often not intuitive to human perception, making camera placement critical for performance.

The research is guided by two primary questions. The first investigates the role of Explainable Artificial Intelligence (XAI), specifically GradCAM++ visual explanations, in Computer Vision for aiding in the evaluation of different camera positions. Building on this, the second question assesses a novel algorithm that leverages these XAI features against traditional black-box optimization methods.

To answer these questions, the study employs a robotic auto-positioning system for data collection, CNN model training, and performance evaluation. A case study focused on classifying flow regimes in industrial-grade bioreactors validates the method. The proposed approach shows improvements over established techniques like Grid Search, Random Search, Bayesian optimization, and Simulated Annealing. Future work will focus on gathering more data and including noise for generalized conclusions.

Tutor:	Dr. rer. nat. Valentin Khaydarov
	Dipl.-Ing. Jonathan Mädler
Supervisor:	Prof. Dr.-Ing. habil. Leon Urbas
Day of Submission:	09.10.2023

DIPLOMA THESIS Author: Leonard-Riccardo Hans Wecke

Contents

List of Figures

List of Tables

Nomenclature

AI	Artificial Intelligence
AUC	Area Under the Curve
CAM	Class Activation Map
CNN	Convolutional Neural Network
EI	Expected Improvement
Fr	Froude Number
Pearson	Pearson Correlation Coefficient
PI	Probability of Improvement
Regret	Absolute difference between the optimal and current objective function values
Regret_t	Regret at iteration t
ROC	Receiver Operating Characteristic
RS	Random Search
SA	Simulated Annealing
maxTemp	Maximum Temperature
minTemp	Minimum Temperature
UCB	Upper Confidence Bound
XaiBO	Explainable Bayesian Optimization

Symbole

α	Acquisition function used in Bayesian Optimization
D_t	Search history up to iteration t
D	Impeller diameter (m)
f	Blackbox function mapping from search space $\mathcal{X}$ to real numbers $\mathbb{R}$
Fl	Flow Number
$f(x)$	Performance measure of configuration x
$f(x^*)$	Optimal objective function value
g	Acceleration due to gravity (m/s^2)
$H(y, p)$	Cross-entropy loss between true labels y and predicted probabilities p
i	Index iterating over each instance in the dataset
I	Total number of runs

μ	Mean of the Best Found Values across I runs
N	Rotational speed (rev/s)
n	Degree of Polynomial
$\mathcal{P}$	Set of possible parameters for the probabilistic model in Bayesian Optimization
π_t	Strategy to propose the next query point based on search history
Q	Flow rate (m^3/s)
r_t	Regret at iteration t
R^2	Coefficient of Determination
R_T	Cumulative regret over T iterations
$\mathcal{X}$	Search space
σ^2	Variance of the Best Found Values
t	Iteration index
x	A particular configuration in search space $\mathcal{X}$
x_i	Best Found Value in the i^{th} run
x_{next}	Next query point in Bayesian Optimization
x^*	Optimal configuration in search space $\mathcal{X}$

1 Introduction

1.1 Motivation

Deep learning technologies have made substantial progress in diverse domains, from healthcare to autonomous driving. In recent years, the process industry has begun showing an increasing interest in intelligent process control as a new frontier for innovation. However, incorporating these advanced algorithms into industry settings is not straightforward, often due to a high-dimensional parameter space, system heterogeneities, and real-time requirements. Hence, there is a pressing need for research to bridge the gap between AI methodologies and their practical application in process control engineering (Khaydarov et al., 2020; Kröger et al., 2022; Markus Esser, 2022).

One specific challenge within process control engineering involves monitoring various types of processes. While some parameters like temperature and pressure can be measured using hardware sensors, others, such as the quality of ongoing chemical reactions, are complex to assess. These intricate processes often require human intervention, leading to potential inconsistencies and inefficiencies.

In recent studies, soft sensor solutions based on Convolutional Neural Networks (CNNs) have been proposed to classify flow regimes in test reactors (Khaydarov et al., 2020). Despite the promise, the generalization of these CNN models has been a stumbling block, particularly when camera positions shift, affecting classification accuracy (Wecke et al., 2022).

Understanding that the camera position significantly affects the quality of image data (Lee Kim & Kim, 2020; Seshadri & Jois, 2019; Torre & Tanimoto, 1996), this thesis seeks to introduce new techniques for guiding camera positioning using insights from Explainable Artificial Intelligence (XAI). Specifically, saliency maps or class activation maps (CAMs) will be investigated for their potential to improve the soft sensor's robustness and adaptability (Simonyan et al., 2014; L. Zhou et al., 2021).

The ultimate objective is to leverage Bayesian optimization and XAI techniques to develop a comprehensive and automated method for optimizing camera positioning (Ozaki et al., 2020; Snoek et al., 2012). The anticipated outcome is to simplify the integration process of camera-based sensors into industrial production systems.

1.2 Problem Analysis

The problem domain of optimizing camera positioning in deep learning applications entails a complex landscape of interconnected challenges. Notably, Convolutional Neural CNNs, despite their proven effectiveness in image classification tasks (Islam et al., 2018), have limitations in terms of generalization across varying camera positions and lighting conditions. These networks, especially those based on architectures like LeNet-5 (LeCun et al., 1989), are highly sensitive to the regions in the input image they are trained on.

One primary complication arises from the difficulty of the physical system in study, such as a bioreactor. Here, gas flow regimes present nontrivial patterns that are not easily captured by a single fixed camera setup. The reactor's design further complicates the matter by introducing occlusions at the window edges, hindering a comprehensive view of the gas flow (Khaydarov et al., 2020). Moreover, an additional layer of complexity is the CNN model's focus on reflection points within the reactor, which are dependent on the internal light source of the reactor and fluctuate with the amount of gas present in the liquid phase. These reflections are challenging for the CNN model, limiting its effectiveness in capturing meaningful features relevant to the flow regime (Wecke et al., 2022).

The Bayesian optimization process, despite its potential, introduces a high degree of customization and intrusiveness into the existing algorithmic framework (Martinez-Cantin, 2019). Interfacing this with class activation maps (CAMs) or saliency maps to improve the camera positioning optimizes one aspect at the cost of introducing noise and complexity into the overall system. Benchmarking these optimization strategies adds another level of complexity. It demands extensive data collection, involving the training and recording of CNNs for multiple camera positions and datasets. This task is not only time-consuming but also resource-intensive.

Furthermore, the task of sliding window implementation poses its challenges in accurately capturing the areas of interest in the reactor (Wecke et al., 2022). Radial positioning of the camera influences the classification capabilities of the CNN, and determining an optimal radial position can be a daunting task given the range of variables at play.

In conclusion, the problem analysis elucidates a series of interconnected challenges that make the problem inherently complex. These challenges range from the limitations of the deep learning models and the complexity of the physical system, to the intricacies involved in integrating Bayesian optimization with saliency maps and other computer vision techniques.

1.3 Research Question

The importance of camera positioning in machine learning based image processing systems is well-established. The placement of the camera significantly affects the information content within image data, which in turn influences the overall performance of these systems. This problem is particularly acute in the realm of process engineering apparatus, where technical limitations often restrict the soft-sensor's observation capabilities. The task of determining an optimal camera position for solving classification tasks using deep learning approaches, therefore, presents a challenging but crucial aspect of enhancing system performance. This study focuses on this task, specifically in the context of classifying flow regimes during gas injection in a bioreactor. To address this challenge, the research proposes to investigate the following key questions:

What are the most effective Explainable Artificial Intelligence (XAI) approaches that can provide valuable features for evaluating camera position in computer vision applications? In particular, this study aims to explore how these XAI approaches can be used to extract additional information from images that can guide the positioning of the camera for improved system performance.

Given the technical constraints in camera positioning, what are the most suitable blackbox optimization methods for solving this problem? The focus here is on identifying methods that can handle the complex search space associated with camera positioning effectively and efficiently, ultimately improving the classification accuracy of the system.

These research questions form the basis of this study, providing the direction for the development of a technical system for the automated search of an optimal camera position in the application of deep learning to classification tasks. The insights and findings from this research have the potential to contribute to the integration of machine learning algorithms into production, particularly in the context of soft-sensor placement.

1.4 Structure of the Thesis

This diploma thesis is structured into nine primary sections, each serving to evaluate distinct aspects of the research methodology.

1. **Introduction**: This opening section furnishes an exhaustive overview of the subject matter, emphasizing the role of camera positioning in machine learning-facilitated image processing, particularly in intricate process engineering setups like bioreactors. The overarching aim of the thesis, as well as the specific case study focusing on the classification of gas flow regimes, is also presented here.

2. **Literature Review**: In this section, a comprehensive analysis of existing scholarly work is carried out. This spans key concepts from flow regime characterization and CNN efficacy metrics to techniques for camera placement optimization. Various optimization algorithms, including but not limited to Random Search, Grid Search, Bayesian Optimization, and Simulated Annealing, are critically evaluated.

3. **Conceptual Framework**: This part elaborates on the theoretical underpinning for the research. It explicates the application of blackbox optimization for determining optimal camera coordinates and details the procedures for image capture and CNN model training at each camera position. Furthermore, it outlines the evaluation metric, which incorporates averaged and normalized Class Activation Maps (CAMs), often termed as saliency maps.

4. **Methodological Approach**: This section is dedicated to outlining the scientific methods employed. It delves into the specifics of the case study concerning flow regime classification in a bioreactor and compares the proposed methodology quantitatively with existing algorithms via a set of well-defined benchmarks.

5. **Data Collection & Experimentation**: Here, meticulous attention is given to the experimental setup, describing both the camera configuration and the associated process engineering apparatus. Additionally, this section enumerates any technical considerations or challenges that may impact the data collection process.

6. **Implementation and Algorithmic Development**: This crucial segment discusses the custom blackbox optimization strategy specifically developed for this thesis. It dives deep into the mechanics of Bayesian Optimization influenced by manipulated a priori probabilities, and elucidates how these probabilities are altered using a delta vector calculated from CAMs. Comparative benchmarks against traditional optimization methods such as Random Search, Grid Search, and Simulated Annealing are also explained, backed by various performance metrics.

7. **Results and Evaluation**: This portion is committed to a rigorous analysis of the research findings. It scrutinizes observed patterns, contemplates their wider implications, and addresses the inherent limitations or challenges encountered during the study.

8. **Discussion**: In this section, the main results are looked at and compared with what other studies have already found. This section also talks about what these findings mean for the areas of deep learning and process engineering. Both the good points and the weak points of this study are discussed.

9. **Summary and Outlook**: The concluding section synthesizes the main contributions of the thesis and sets the stage for future research avenues, delineating potential enhancements and adaptations that could further optimize camera positioning in similar applications.

2 State of the Art

2.1 Literature Research Methodology

The methodology for the literature review is designed to be systematic and comprehensive, aiming to cover all relevant aspects of the research topic. The primary objective is to identify, evaluate, and synthesize existing research related to the optimization of a camera's position in deep learning applications, particularly in the context of gas flow regimes in bioreactors.

2.1.1 Search Strategy

The search strategy was systematically designed to cover a broad spectrum of research while ensuring relevance to the specific context of this work. An initial search yielded approximately 250 papers from several databases, including IEEE Xplore, PubMed, ACM Digital Library, arXiv.org, ScienceDirect, SpringerLink, and Scopus. A combination of keywords and Boolean operators, as shown in Table 2.1, was employed to maximize the retrieval of relevant articles. Further, a snowball technique was used to identify additional papers by scanning the references of initially selected articles. This technique added an estimated 20 papers to the initial 100, totaling 120 papers identified for potential inclusion. To visualize this search process, the data flow chart 2.1 is provided, illustrating the contributions from each database and the narrowing down of the pool of papers (Runeson & Höst, 2009). The keywords in the keyword table were chosen from analysing the scientific research question.

2.1.2 Inclusion and Exclusion Criteria

This work assumes a foundational understanding of Neural Networks and Convolutional Neural Networks (CNNs) and will not delve into the basics of these concepts, given the widespread use and explanation of these topics. Explaining these foundational elements would not only be redundant but would also exceed the scope and length constraints of this work. The criteria for paper selection was established to maintain rigor and relevance. Papers were included if they met the following criteria:

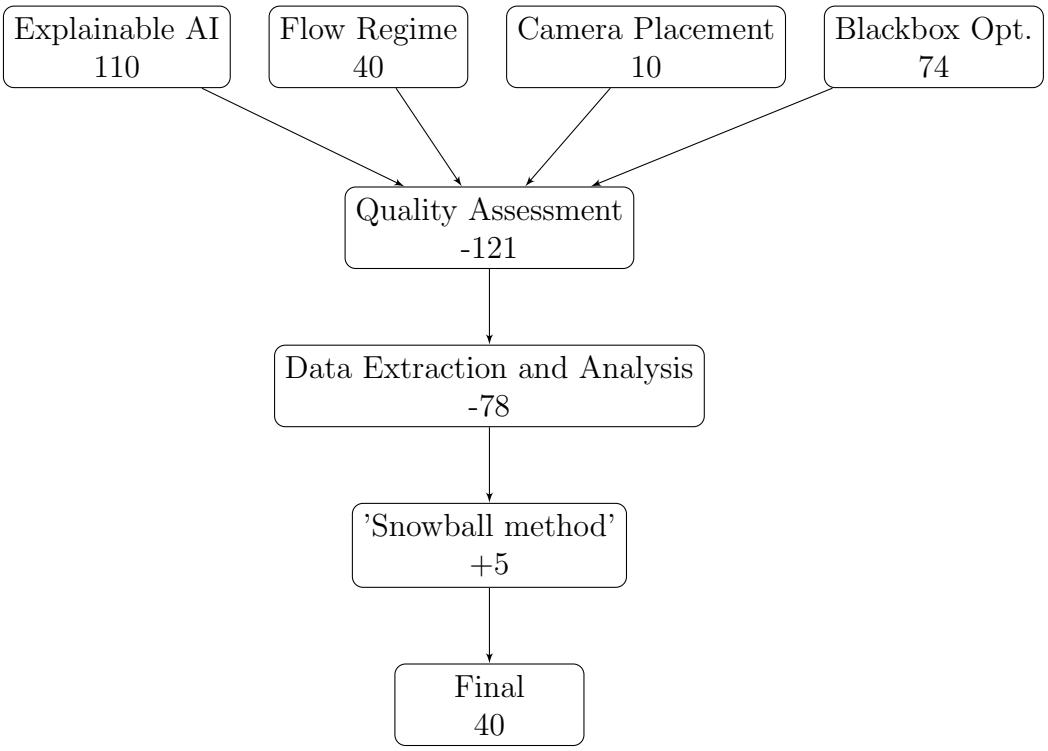


Figure 2.1: Data flow chart illustrating the literature search process for each topic

- Peer-reviewed articles published in reputable journals.
- Articles focused on the optimization of camera positioning in machine learning or computer vision applications.
- Articles that discuss flow regimes, especially in bioreactors.

Conversely, exclusion criteria were also defined to filter out papers that may not add significant value to this work. The criteria for exclusion include papers from outside the institution of TU-Dresden with fewer than 5 citations, as they may lack academic recognition. Papers that are older than 10 years, unless they are seminal works with high citation counts.

Applying these criteria, the initial 234 identified papers were reduced to 60 papers for in-depth review. This approach aligns with Runeson et al.'s guidelines for systematic literature reviews (Runeson & Höst, 2009).

Phrase	and, or
Explainable AI	
Saliency maps	
Object detection	
Class activation maps	
Flowregime Classification	
Stirred vessels flow regimes	
Gas injection flow regimes	
Bioreactor flow regimes	
Bioreactor flow regimes	camera
Optimal Camera Placement	
Camera positioning	
Optimal camera placement	
Optimal camera placement	Flow regime classification
Camera location	Optimal camera placement
Camera angle	Optimal camera placement
Blackbox Optimization	
Camera positioning	
Blackbox optimization	
Derivative free optimization	
Quantitative analysis	
Hyperparameter optimization	

Table 2.1: Literature word combinations for optimizing camera positioning in deep learning applications

Quality Assessment The 60 selected papers underwent a thorough quality assessment to determine their relevance and reliability. Quality indicators such as the rigor of the methodology, relevance to the research questions, and the impact factor of the journals were considered. For assessing the rigor of the methodology, papers that used empirical methods, provided detailed explanations, and performed validations were rated higher. On a scale of 1 to 5, the average methodology score was found to be 3.5 for the reviewed papers.

Relevance to the research questions was another critical metric. Papers that directly addressed issues related to optimizing camera positioning in deep learning applications and gas flow regimes in bioreactors received higher scores. In this context, 30 out of the 60 reviewed papers were found to be directly relevant.

Lastly, the impact factor of the journals in which the papers were published served as another indicator. Papers published in journals with an impact factor greater than 3.0 were given extra weight during the review. About 40% of the papers met this criterion. Collectively, the quality assessment aimed to filter the papers further and to provide a balanced view on the existing literature. This methodology aligns with the standard practices in this field, as suggested by (Runeson & Höst, 2009).

Data Extraction and Analysis After sorting the literature based on the inclusion and exclusion criteria, the remaining papers underwent a detailed review. Each article was read carefully, and crucial data points were extracted. This extraction focused on methodologies employed, key findings, and limitations for each paper, following the best practices in literature reviews in process control engineering (Runeson & Höst, 2009).

A dedicated database was built using Mendeley reference manager, to manage and analyse the extracted information systematically. The database contained columns for title, authors, year of publication, citation count, methodology, key findings, and limitations, among others. A qualitative analysis was carried out on this database to identify trends, gaps, and clusters within the reviewed papers.

Out of the remaining articles, 10 were identified to not be directly related to the research questions posed in this work and sortet out. The remaining papers contributed to forming the theoretical foundation and offered varying solutions to the problem of finding the optimal camera positioning in the context of gas flow regimes in bioreactors. The information extracted from these papers will serve as the base for designing the experimental phase and for comparison with the results obtained in this work. This process was consistent with the recommended guidelines for data extraction and analysis in systematic literature reviews, as suggested by Runeson and Höst (2009).

Synthesis After the quality assessment, a final list of 40 papers was deemed suitable for synthesis. The synthesis phase involved bringing together the extracted data to form a coherent understanding of the current state of research in the domain of camera positioning in machine learning and gas flow regimes in bioreactors. A thematic approach was used to categorize the papers into specific themes, such as explainable AI techniques, flow regime classification methods, and blackbox optimization algorithms. In particular, 16 papers discussed explainable AI, 10 papers were related to flow regime classifications, 11 papers focused on blackbox optimization techniques and 5 papers on CNNs. No paper fulfilled the requirements and was related to a systematic camera positioning concept in the context of CNNs.

By organizing the papers thematically, patterns and trends started to emerge. For example, explainable AI methods like saliency maps and class activation maps were commonly used to interpret CNN decisions in camera positioning. On the flip side, there were gaps in the literature concerning real-time camera adjustments based on changing flow regimes, pointing towards self-implementation.

The synthesis revealed both the strengths and weaknesses of existing solutions and methodologies. It offered insights into the current limits of technology and gave insides into the research question of this diploma thesis.

The synthesis process aligns with the best practices outlined for systematic literature reviews, consistent with guidelines by Runeson and Höst (2009).

2.2 Blackbox Optimization

Blackbox optimization is a technique for optimizing systems where the internal workings are not fully understood or accessible (Jones et al., 1998). This is particularly useful in fields like computer vision, deep learning, and process control engineering (Martinez-Cantin, 2019; Shahriari et al., 2016).

The choice of optimization method often depends on various factors such as the dimensionality of the problem, the cost of function evaluations, and the availability of computational resources (Martinez-Cantin, 2019).

- **Exhaustive Methods (Random Search, Grid Search):** These methods are often used for problems with low dimensionality where an exhaustive search is feasible. However, they may not be efficient for high-dimensional or expensive-to-evaluate functions (Bergstra & Bengio, 2012).
- **Bandit-based Methods (Hyperband, Hyperopt):** Hyperband is a variation of random search, including some explore-exploit mechanisms to find the best time allocation for each of the configurations. They are often used when computational resources are limited but not extremely scarce (Li et al., 2018).
- **Model-based Methods (BOHB):** These methods use statistical models to guide the search and are particularly useful when function evaluations are expensive. They adaptively focus on regions of the search space that are likely to yield better results (Falkner et al., 2018).
- **Probabilistic Model-based Methods (Bayesian Optimization):** Bayesian Optimization is often recommended for problems where each function evaluation is costly in terms of time or computational resources. It builds a probabilistic model of the objective function to guide the search (Brochu et al., 2010; Shahriari et al., 2016).
- **Metaheuristic Methods (Simulated Annealing):** Simulated Annealing is a probabilistic technique often used for finding an approximate solution to an optimization problem. It is particularly useful for problems that have many local minima (Samora et al., 2016).

Advantages and Disadvantages of Methods Here we discuss the pros and cons of each method, providing a more nuanced understanding of when to use each.

- **Exhaustive Methods (Random Search, Grid Search):**
  - *Advantages:* Simple to implement and understand. Suitable for low-dimensional problems (Bergstra & Bengio, 2012).
  - *Disadvantages:* Computationally expensive and inefficient for high-dimensional or complex problems (Bergstra & Bengio, 2012).

- **Bandit-based Methods (Hyperband, Hyperopt):**
  - *Advantages:* Efficiently explores the search space and adaptively allocates resources (Li et al., 2018).
  - *Disadvantages:* May require more initial setup and tuning compared to simpler methods. (Bergstra et al., 2013)
- **Model-based Methods (BOHB):**
  - *Advantages:* Effective when function evaluations are expensive, focuses on promising regions (Falkner et al., 2018).
  - *Disadvantages:* May require a significant amount of data to build an accurate model (Falkner et al., 2018).
- **Probabilistic Model-based Methods (Bayesian Optimization):**
  - *Advantages:* Efficient for costly evaluations, builds a probabilistic model to guide the search (Brochu et al., 2010).
  - *Disadvantages:* May be sensitive to the choice of kernel and hyperparameters (Shahriari et al., 2016).
- **Metaheuristic Methods (Simulated Annealing):**
  - *Advantages:* Capable of escaping local minima, useful for complex landscapes (Samora et al., 2016).
  - *Disadvantages:* Cooling schedule and other parameters may need careful tuning (Samora et al., 2016).

Global vs Local Search In blackbox optimization, the terms 'global' and 'local' search often come into play. Global search algorithms, like Bayesian Optimization and Simulated Annealing, aim to explore the entire parameter space to find the global optimum (Samora et al., 2016; Shahriari et al., 2016). On the other hand, local search algorithms, often employed in methods like Grid Search, focus on a specific region of the parameter space and may get stuck in local minima (Bergstra & Bengio, 2012).

Uninformed Search Uninformed search algorithms, also known as blind search, operate without any additional information about the state space other than how to traverse it. In the context of blackbox optimization, Bayesian Optimization serves as an interesting counter-example. Normally, Bayesian Optimization is an informed search method, but it can be considered without its acquisition function, which is the 'informed' part.

The objective function $f(x)$ is sampled at different points x in the search space $\mathcal{X}$, without any guidance on where the next sample should be. This is akin to uninformed search as the algorithm does not use any external information to decide the next query point x_{next}.

Guided Search Methods Unlike basic search techniques that solely navigate through the search space, guided search methods utilize extra data to influence decision-making during the search. These methods employ a heuristic function, denoted as $h(x)$, in tandem with the actual cost function, $g(x)$, to estimate the total cost $f(x) = g(x) + h(x)$ from the starting point to the final goal via x. The heuristic function $h(x)$ serves as supplementary information steering the search. This heuristic is often precalculated and relies on domain-specific knowledge, providing the algorithm with an estimation of the cost from the current state x to the goal.

2.3 Mathematical Notation

Before discussing each method in detail, it's crucial to establish a common mathematical notation for blackbox optimization. The goal of blackbox optimization is to maximize or minimize an objective function, which in our specific application refers to maximizing the classification accuracy of the deep learning model. The mathematical notations presented below provide a foundation for understanding and comparing various blackbox optimization methods (Jones et al., 1998; Shahriari et al., 2016).

Let $f : \mathcal{X} \to \mathbb{R}$ denote the blackbox function, where $\mathcal{X}$ is the search space, and $f(x)$ is the performance measure of a particular $x \in \mathcal{X}$. In the context of our study, x represents the camera coordinates and $f(x)$ denotes the classification accuracy of the Convolutional Neural Network (CNN) model trained on the images captured at these coordinates. The primary optimization objective can be expressed as finding an $x^* \in \mathcal{X}$ such that:

$$x^* = \arg\max_{x \in \mathcal{X}} f(x) \tag{2.1}$$

Bayesian Optimization, one of the methods used, introduces an acquisition function $\alpha : \mathcal{X} \times \mathcal{P} \to \mathbb{R}$, where $\mathcal{P}$ denotes the set of possible parameters of the probabilistic model learned from the data. This acquisition function guides the selection of the next query point x_{next}, determined as:

$$x_{\text{next}} = \arg\max_{x \in \mathcal{X}} \alpha(x,P) \tag{2.2}$$

In order to further facilitate the understanding and comparison of the various blackbox optimization methods, we introduce the following additional notations:

- **Iteration Index:** Let $t \in \mathbb{N}$ denote the iteration index. Each iteration represents a new trial or search in the optimization space.
- **Search History:** The search history up to iteration t is represented by $D_t = \{(x_1, y_1), \ldots, (x_t, y_t)\}$, where $x_i \in \mathcal{X}$ are the input vectors and $y_i = f(x_i)$ are the corresponding function evaluations.
- **Query Strategy:** Each optimization method has a strategy, represented by $\pi_t : D_t \to \mathcal{X}$, to propose the next query point based on the search history:

$$x_{t+1} = \pi_t(D_t) \tag{2.3}$$

2.4 Bayesian Optimization

Bayesian Optimization (BO) is a global optimization technique designed for optimizing blackbox functions without assuming any functional forms (Shahriari et al., 2016). It has found applications in various domains such as hyperparameter tuning in machine learning, design optimization, and robotics (Brochu et al., 2010).

Probabilistic Modeling with Gaussian Processes: The core component of BO is a probabilistic model, often a Gaussian Process (GP), used to represent the unknown objective function (Rasmussen & Williams, 2006). GPs provide a distribution over functions and are updated as new data points are observed. They are particularly useful because they can compute the posterior distribution over functions consistent with observed data.

Acquisition Function: The acquisition function plays a crucial role in BO. It guides the selection of the next point to evaluate based on the current GP model. Common choices include Expected Improvement (EI), Probability of Improvement (PI), and Upper Confidence Bound (UCB) (Jones et al., 1998).

Iterative Approach: BO operates in an iterative manner, focusing on two main components: the GP model and the acquisition function. Initially, the GP model starts with a prior distribution that represents the initial beliefs about the objective function. As new data points (hyperparameter settings and their corresponding performance metrics) are collected, the GP model is updated to compute the posterior distribution, which combines the prior beliefs with the new data. This posterior distribution serves as a surrogate model for the actual objective function, capturing the algorithm's updated beliefs about the function's behavior across the hyperparameter space.

In each iteration, the acquisition function takes this posterior distribution as input and selects the next point to evaluate. The acquisition function aims to balance exploration and exploitation based on the current state of knowledge captured by the GP model. For exploration, it may choose points where the model is uncertain, aiming to learn more about the objective function. For exploitation, it may focus on areas where the model predicts high performance, aiming to fine-tune the hyperparameters.

The iterative process continues, with each new evaluation serving to update the GP model's posterior distribution. This, in turn, informs the acquisition function for the next iteration. The cycle repeats until a stopping criterion is met, such as reaching a maximum number of iterations or achieving a performance metric that satisfies predefined criteria.

Kernel Function: The choice of kernel function in the GP model is important, as it determines the function's smoothness and other properties. The kernel function $k(x, x')$ defines the covariance between any two points x and x' in the input space. Mathematically, the prior over functions $f(x)$ is modeled as a multivariate Gaussian distribution with mean zero and covariance matrix K, where $K_{ij} = k(x_i, x_j)$. The formula for the covariance matrix is given by:

$$K = \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \cdots & k(x_n, x_n) \end{pmatrix}$$

- $k(x, x')$ - Kernel function that calculates the covariance between points x and x'.
- x, x' - Two points in the hyperparameter space.
- K - Covariance matrix representing the relationships between all pairs of points in the dataset.
- K_{ij} - Element at the i-th row and j-th column of the covariance matrix K, calculated using $k(x_i, x_j)$.
- $x_1, x_2, \ldots, x_n$ - Points in the dataset for which the covariance matrix is being calculated.
- n - Number of points in the dataset.

Different kernels can capture various types of function behavior, making the choice of kernel an important aspect of BO. The kernel function essentially shapes the prior beliefs about the function to be optimized, influencing how the GP model generalizes from observed data to unobserved points.

Limitations and Extensions: BO assumes that the objective function is noise-free and smooth, which may not always be the case. It is also sensitive to the choice of kernel and acquisition function. Various modifications have been proposed to address these limitations, such as handling noisy observations and optimizing the acquisition function more effectively (Snoek et al., 2012).

Application in This Work: In the context of this work, BO can be used for optimizing camera coordinates in a bioreactor setup. The method would evaluate the performance of a CNN model at different camera positions, update the GP model, and then use the acquisition function to select the next position. This allows for efficient optimization, especially when the cost function is expensive to evaluate.

Algorithm 1 Bayesian Optimization for Camera Positioning

Require: Blackbox function $f : \mathcal{X} \to \mathbb{R}$
Require: Acquisition function $\alpha : \mathcal{X} \times \mathcal{P} \to \mathbb{R}$
Initialize data $D = \{\}$ and a probabilistic model M
for $t = 1, 2, 3, \ldots, n$ **do**
Select the next query point by optimizing the acquisition function:

$$x_{t+1} = \arg\max_{x \in \mathcal{X}} \alpha(x, M(D))$$

Query the blackbox function at x_{t+1} to obtain $y_{t+1} = f(x_{t+1})$
Augment the data with the new observation: $D = D \cup \{(x_{t+1}, y_{t+1})\}$
Update the probabilistic model M based on the data D
end for
Return the best configuration $x^* \in \mathcal{X}$ that maximizes $f(x)$ based on the data D and model M

This pseudocode describes a standard Bayesian Optimization algorithm. It starts by initializing the data and a probabilistic model. Then it enters a loop for a predefined number of iterations (N). In each iteration, it selects the next query point by optimizing the acquisition function, queries the blackbox function at the next query point to obtain the corresponding value, augments the data with the new observation, and updates the probabilistic model based on the augmented data. Finally, it returns the best configuration that maximizes the blackbox function based on the data and model. The specifics of the acquisition function, the probabilistic model, and how they are updated would depend on the actual implementation.

The use of a grid, whether discrete or continuous, is generally to approximate the objective function within a specified domain. In Bayesian optimization, the grid of test points serves as candidates where the algorithm estimates the expected improvement. It then selects the point that maximizes this expectation to be the next evaluation point.

2.5 Simulated Annealing

Simulated Annealing (SA) is an optimization algorithm that finds its roots in the annealing process used in metallurgy. The algorithm is designed to find a low-energy state in a system, similar to how annealing removes defects in materials. While SA has a broad range of applications, it has also been adapted for specific tasks such as mineral exploration and real-time pressure process plant control (Biswas, 2016; Devan et al., 2022).

The algorithm starts with a random initial solution and explores the neighborhood solutions through small changes. The acceptance of a new solution is determined by an objective function and a probability function influenced by a parameter known as temperature. Over time, the temperature decreases, reducing the likelihood of accepting suboptimal solutions . Advanced variants like Very Fast Simulated Annealing (VFSA) have been developed to speed up the computation time (Biswas, 2016). The temperature parameter plays a crucial role in balancing exploration and exploitation. High temperatures enable the algorithm to explore new areas in the solution space, while low temperatures focus the search on local optimization. Some implementations introduce randomness to help the algorithm escape local minima (Yang et al., 2020). Key hyperparameters in SA include the initial temperature, cooling schedule, and the number of iterations at each temperature level. The choice of these hyperparameters can significantly affect the algorithm's performance (Yu et al., 2022). Some approaches even modify traditional SA parameters based on other optimization algorithms to improve convergence speed (Wang et al., 2018). SA's performance is commonly evaluated using metrics like convergence time, solution quality, and computational overhead. It is often compared with other optimization algorithms to assess its effectiveness (Wang et al., 2018). In the field of process control engineering, SA is frequently employed for tasks like parameter tuning and control strategy optimization. It has also been adapted for specific control applications, such as real-time pressure process plant control (Devan et al., 2022; Gaspar & Oliveira, 2011). The algorithm can be computationally expensive, particularly for problems with large search spaces. The choice of hyperparameters is often problem-specific, requiring multiple runs for optimal tuning (Wang et al., 2018).

2.6 Random Search

Random Search (RS) is a straightforward optimization method that doesn't require gradient information, making it suitable for blackbox optimization problems. It has been applied in various domains, including hyper-parameter optimization in machine learning. RS operates by randomly sampling points in the search space and evaluating them using an objective function. The algorithm keeps track of the best solution found during its run. The search can be either structured, following a specific pattern, or truly random (Anderson, 1953).

Random search is particularly useful when the objective function is not well-understood, non-continuous, or non-differentiable. However, its effectiveness diminishes in high-dimensional spaces due to the randomness of the search (Anderson, 1953). In this work, RS is applied to find optimal camera positions. Despite its simplicity, RS serves as a baseline for comparing more sophisticated optimization techniques.

2.7 Gridsearch

Grid search is a straightforward method for hyperparameter optimization that exhaustively searches through a manually specified subset of the hyperparameter space (Bischl et al., 2021).

Given a set of possible values for each hyperparameter, Gridsearch constructs a grid of configurations and evaluates the model's performance for each. The best-performing configuration is selected as the optimal solution. In this study, the aim is to optimize camera coordinates x, y to maximize a CNN's classification accuracy. Gridsearch would involve specifying possible values for x and y, constructing a grid of x, y pairs, and evaluating the CNN for each pair.

Mathematically, Gridsearch can be represented as:

$$\text{For each } x \in \mathcal{X} \text{ and each } y \in \mathcal{Y}: \tag{2.4}$$

$$\text{Evaluate } f(x, y) \tag{2.5}$$

$$x^*, y^* = \arg \max_{x \in \mathcal{X}, y \in \mathcal{Y}} f(x, y) \tag{2.6}$$

Gridsearch is simple but can be computationally expensive, especially for high-dimensional hyperparameter spaces (León et al., 2020). Compared to Random Search, it may be less efficient as it doesn't focus on regions of

better performance (Bergstra & Bengio, 2012).

2.8 Explainable A.I. and Saliency Maps

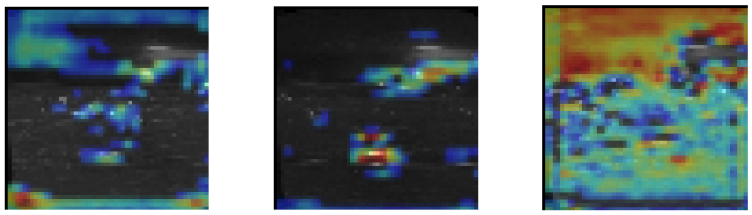


Figure 2.2: GradCAM++ and original image computed from single images and a trained LeNet5 CNN.

In recent years, machine learning models, particularly CNNs, have demonstrated exceptional performance in tasks like image recognition, segmentation, and various other computer vision applications. Due to the focus in this work being the optimization and feature integration, a detailed explanation of the fundamental workings of CNNs can be referred to in related work of (Khaydarov et al., 2020; Kröger et al., 2022; Markus Esser, 2022; Wecke et al., 2022). While these models excel in terms of accuracy, their internal operations are often considered unknown, making them difficult to interpret. This lack of interpretability poses challenges in understanding model behaviour, diagnosing errors, and gaining insights into the features that the model finds important. Thus, explainability has emerged as a significant field to make machine learning, especially CNNs, more transparent and understandable. Saliency maps are visualization techniques that provide insights into which regions of an input image are instrumental for a CNN to arrive at a specific classification. Essentially, they highlight the 'important' pixels or regions that contribute most to the model's decision-making process (Selvaraju et al., 2017).

One popular approach for generating saliency maps are techniques like Gradient-weighted Class Activation Mapping (GradCAM) and its extension, GradCAM++, that aim to provide visual explanations for model decisions. These methods generate what is commonly referred to as saliency maps (Selvaraju et al., 2017). The most basic form of a saliency map is a heatmap where each pixel's intensity corresponds to the importance of that particular pixel in classifying the image (Chattopadhay et al., 2018). It allows a graphical interpretation of how the neural network makes its decisions and can be helpful in applications where understanding model behaviour is crucial. GradCAM uses the gradients of class evaluations, flowing into the final convolutional layer to produce a low-resolution map. This map is then upsampled to match the input image resolution, thereby highlighting the important regions.

Figure 2.2 illustrates a GradCAM++ heatmap representing three different saliency maps for a specific classification generated by a trained CNN model. This heatmap shows areas in the input image that the model considers important for making its decision, thereby providing a way to qualitatively analyze the model's behaviour. It is essential to note that saliency maps are generated based on a specific input image and a pre-trained CNN model. Therefore, they serve as a qualitative image analysis tool, highlighting the regions that are critical for the model's classification decision. Other techniques like Network Dissection (Bau et al., 2017) also exist, which aim to interpret networks by identifying and labeling the units in the network that match human-defined concepts.

GradCAM++ a detailed explanation: GradCAM is a technique developed to interpret the decisions made by CNNs by visualizing the regions in the input image that contribute significantly to the output prediction. The primary goal is to create a heatmap-like overlay that signifies the important regions in the image as recognized by the CNN. In the GradCAM technique, gradients play a critical role in explaining the output of the CNN model. This process can be broken down into simpler steps:

1. The first step involves executing the forward pass of the CNN model to classify the input image. During this pass, activations are recorded at each convolutional layer. The final layer's output determines the most probable class to which the input image belongs (LeCun et al., 1989).

2. The gradients of the output class, often referred to as the target class, are calculated concerning the feature maps of the last convolutional layer. Mathematically, these gradients are partial derivatives. They quantify how minute changes in each feature map can affect the class score (Zeiler & Fergus, 2014).

3. The computed gradients are then subjected to a global average pooling process. This results in a single scalar value for each feature map. These scalar values serve as weights and indicate the importance of each corresponding feature map for the target class (Selvaraju et al., 2017).

4. A weighted sum of these feature maps is calculated using the weights obtained from the global average pooling. The resultant weighted sum produces a coarse heatmap, which shows the importance of different regions in the original image with respect to the target class (L. Zhou et al., 2021).

5. This coarse heatmap is resized to match the resolution of the input image. Generally, interpolation methods are employed for the resizing (Dhillon & Verma, 2020).

6. The resized heatmap is finally superimposed on the original input image. This superimposition helps to visualize which areas in the image were significant in classifying it into the target class which can be seen in Figure 2.2 (Selvaraju et al., 2017).

In Figure 2.3, the feature maps (A1, A2, A3) can be considered as the final feature maps from the last convolutional layer. The gradients for these feature maps are computed and used to generate the GradCAM heatmap. The weighted combination of these feature maps forms the basis for the heatmap, which is then upscaled to the resolution of the input image for a side-by-side comparison and interpretation.

In Figure 2.3, the process starts with an input image, where darker regions highlight the presence of particular objects or features. This input image is then transformed through several convolutional layers. Each convolutional layer, as depicted, focuses on extracting different features, breaking them down into individual feature maps (A1, A2, A3). It is these feature maps that serve as the foundation for generating the GradCAM heatmap.

Turning attention to GradCAM++, it is an extension of GradCAM and aims to offer a more nuanced visualization. One primary improvement is the capability to focus on more localized regions, often smaller but significant in context, which might be ignored or underweighted by GradCAM. This is achieved by incorporating higher-order gradients into the calculation, offering a more fine-grained attribution of importance to specific regions in the image (Chattopadhay et al., 2018).

This difference is visually represented in the saliency maps in Figure 2.3. In GradCAM++, the greyscale values are notably higher in smaller feature regions compared to GradCAM, highlighting that these regions are also important in decision-making. Further illustration is provided in Figure 2.2, where the red-to-blue spectrum is utilized to represent the degree of importance in the GradCAM++ heatmap.

To summarize, GradCAM provides a base understanding of CNN attention on a broad scale, capturing larger, more evident features. However, if the focus is to hone into more localized, perhaps subtler, but significant features in an image, GradCAM++ serves as a more suitable technique. Therefore, for detailed and nuanced interpretability, opting for GradCAM++ would be a well-considered choice (Chattopadhay et al., 2018).

In summary, saliency maps offer a compelling way to understand and interpret the behaviour of CNN models by visually highlighting the regions in the input data that are key for classification.

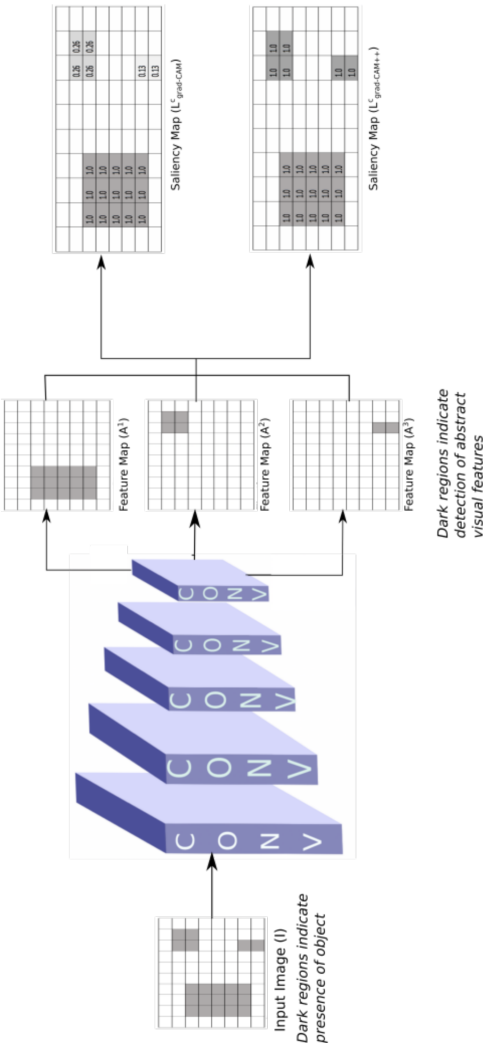


Figure 2.3: Comparison between GradCAM and GradCAM++. GradCAM++ allows for more intricate feature identification (Chattopadhay et al., 2018).

2.9 Flowregime Classification in Stirred Vessels

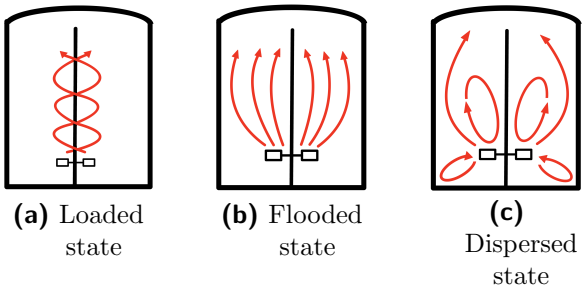
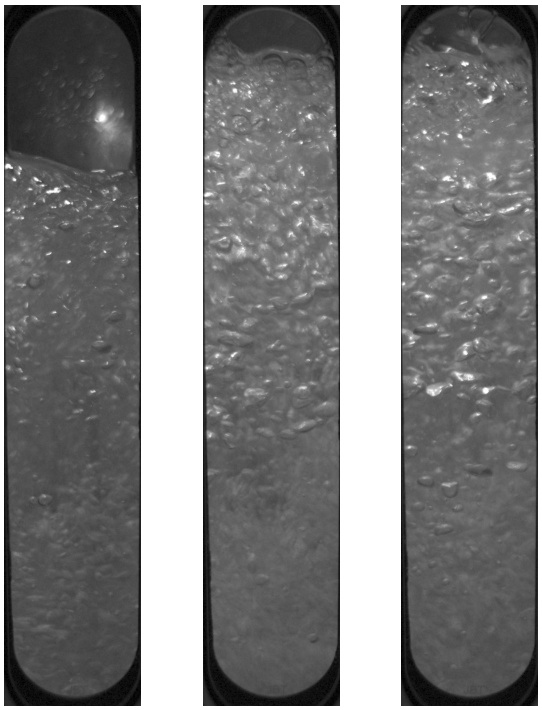


Figure 2.4: Bubble distribution in different flowregimes (Markus Esser, 2022)

The classification of flow regimes in stirred vessels is a task in process control engineering that is normally done through human labour. Using machine learning is the subject of the case study in this thesis. This task is not only important for ensuring efficient mixing but also crucial for maintaining optimal reaction conditions. The Explainable AI methods discussed in the previous section could provide valuable insights for this classification task. For the remainder of this work, the term 'flow regime' will refer to the manner in which the gas phase is dispersed throughout the reactor (Troniewski & Ulbrich, 1984).

Flow regime classification has been studied using various techniques. One notable approach is the use of CNNs as employed by Korinna Kröger (Kröger, 2019). Additionally, Marcus Esser has explored the integration of multiple data sources through multimodal approaches for enhanced classification performance (Markus Esser, 2022). Valentin Khaydarov and his colleagues have made contributions by combining CNNs with flow regime classification in stirred vessels, significantly influencing the field (Khaydarov et al., 2020). The student thesis of Leonard Wecke has focused on utilizing explainable AI methodologies, such as saliency maps, to qualitatively examine the flow regime reactor for areas of interest for the CNN. The results show that the CNN concentrates on fix points that it finds very useful for classification that are not easily observable by the human eyes. Scanning the reactor for such areas of interest has the potential of increasing CNN performance (Wecke et al., 2022). His work aligns closely with the intent of leveraging explainable AI for improved visualization and understanding of flow regimes.



(a) Loaded state **(b)** Flooded state **(c)** Dispersed state

Figure 2.5: Visual differences between flow regimes in the same dataset

The body of existing work sets the stage for the investigation presented here. The main focus of this work is to capitalize on the advancements in explainable AI for visualizing areas of interest, and subsequently, applying blackbox optimization methods to achieve automatic repositioning of sensors or measurement instruments. In stirred vessels, different flow regimes can be observed, mainly influenced by stirrer speed and gas supply. These regimes have been broadly categorized into three types: Flooded, Loaded and Dispersed that are visualy shown in figure 2.6 (Kröger, 2019). Each regime has its unique characteristics, primarily influenced by bubble formation and their spatial distribution within the reactor. Accurate classification of these regimes is vital for efficient reactor operation.

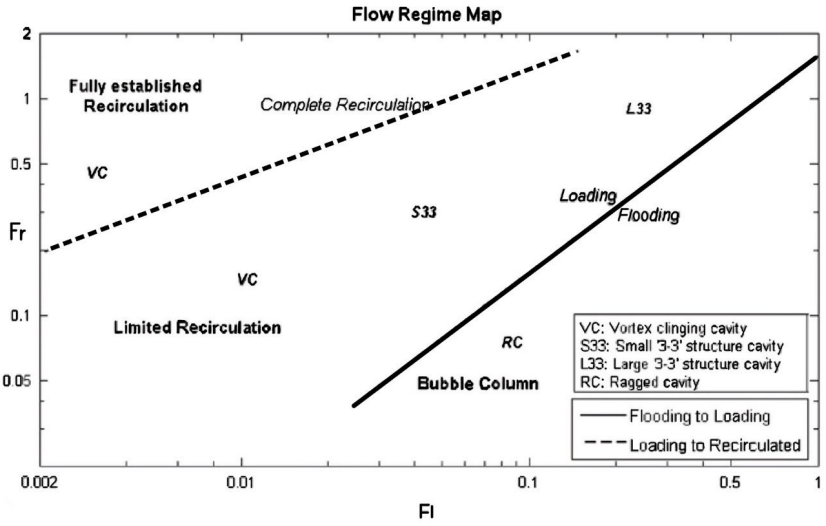


Figure 2.6: Original flowchart with logarithmic scale. Is used to set the reactor in a particular flowregime by calculating the gasflow and rpm map as explained (Kröger, 2019).

Contrary to human observation, which might naturally focus on bubble shape and size and dispersion, CNNs in this work have demonstrated a different approach which is most often the case (Das et al., 2017). According to studies, the CNNs focus on specific reflection points learned during the training process (Wecke et al., 2022). These reflection points are highly specific and extremely sensitive to a variety of factors such as lighting conditions, camera position, and the reactor's own characteristics, as well as the fluid it contains.

This raises challenges for the generalization of the trained model to other use cases. If any of these conditions change, the CNN must be retrained to accurately classify the flow regimes. Hence, each use case essentially becomes a unique scenario requiring a customized CNN training process. This reality underscores the need for a careful setup and training regimen when using CNNs for flow regime classification.

The state of the flow regime is a complex function of numerous variables. These include but are not limited to, flow velocity, stirrer speed, and various geometrical and thermodynamic parameters (Kröger, 2019). Two dimensionless numbers, namely, the Flow Number (Fl) and the Froude Number (Fr), have been identified to characterize these regimes more comprehensively.

$$\text{Flow Number (Fl)} = \frac{Q}{N \cdot D^3} \tag{2.7}$$

$$\text{Froude Number (Fr)} = \frac{N^2 \cdot D}{g} \tag{2.8}$$

The major challenge in this classification task is the transitional states between different regimes. These states are hard to classify even by human experts.

The flowchart in Figure 2.6 serves as a computational tool for determining the reactor's current flow regime. Here, the x-axis and y-axis may represent process variables such as Rotations per Minute (RPM) and Gasflow number, respectively. The plot uses a logarithmic scale, which helps in capturing the non-linear behavior of the system. Based on these two variables, it becomes feasible to classify the reactor into a specific flow regime, thus enabling more precise control (Kröger, 2019). In the context of process control engineering, the state of the flow regime in a reactor is influenced by various factors. The Flow Number Fl and the Froude Number Fr serve as key dimensionless numbers to encapsulate this complexity.

- Q - Flow rate in m^3/s represents the volumetric rate at which fluid flows through the reactor.
- N - Rotational speed in rev/s indicates the speed at which the stirrer rotates.
- D - Impeller diameter in meters signifies the size of the impeller facilitating the stirring.
- g - Acceleration due to gravity in m/s^2, a constant value affecting the motion of fluids.

The reactor controls the state of the flow regime primarily through the adjustment of RPM (Revolutions Per Minute) and gas flow rate. These settings influence N and Q, which in turn affect Fl and Fr.

For the analysis of image data captured from the reactor, these dimensionless numbers are employed to compute labels for each state of the flow regime. The resulting labels can be categorized as, Flooded, Loaded and Dispersed. These labels are essential for further data analysis and can be used to train machine learning models that aim to recognize and manage different flow states. The non-linear nature of the flowchart suggests the complexity of the relationship between the reactor's operating conditions and its flow regime. Furthermore, the classification task is closely tied to the reactor's specifications, emphasizing the need for a tailored approach when setting up process controls. This flowchart allows for the reactor to be set in a particular flow regime by calculating the gasflow and RPM map as explained.

Through these visual aids and computational tools, the study aims to improve the understanding and control of flow regimes in stirred vessels. This could lead to more efficient and stable reactor operation. The complexity and high dimensionality of this task make it a suitable candidate for applying Explainable AI methodologies. The feature importance methods discussed in the previous section can help in understanding which parts of the image are informative for classifying a given flow regime. This could be particularly useful for understanding transitional states, which are typically the most challenging to classify.

In summary, flow regime classification in stirred vessels is a complex yet crucial task in the realm of process control engineering. While there are existing methods to characterize these regimes, they are not without limitations. Neural network-based methods, coupled with Explainable AI techniques, could provide a robust solution to this challenging problem.

2.10 Performance Metrics

2.10.1 R^2 Score and Polynomial Regression for Experiment Data Analysis

The R^2 score is selected as the primary metric for evaluating the correlation in the experimental data for this work. Also known as the coefficient of determination, the R^2 score measures the extent to which the independent variable can explain the variance in the dependent variable. The R^2 value can range from 0 to 1 with a higher value indicating a better model fit to the observed data.

While other metrics such as the Pearson correlation coefficient exist, they are typically suited for linear relationships between variables. Given that this study anticipates more complex shapes in the objective function, Pearson correlation is not deemed suitable.

To better capture the expected complexity, polynomial regression is used. This approach fits a polynomial equation to the data, allowing for curves instead of just straight lines. The polynomial's degree (n) is a parameter that can be adjusted to achieve different curve fits. However, selecting an appropriate degree n is critical to avoid overfitting, which occurs when the model performs very well on the training data but poorly on unseen or new data. To strike this balance, n is varied and the R^2 score is observed. The goal is to find a degree that maximizes the R^2 score without causing overfitting.

In summary, polynomial regression is used to account for the potentially complex relationships in the data, and the R^2 score provides a quantitative metric for evaluating the fit. The challenge is to optimize n to achieve a high R^2 score.

2.10.2 Blackbox Optimization Performance Metrics

In assessing the performance of blackbox optimization algorithms for optimizing camera positions in deep learning applications, a few key metrics stand out as particularly informative. These metrics offer quantitative ways to compare how effectively different algorithms perform, especially in scenarios like gas flow regimes in bioreactors where the objective function is often complex and non-intuitive.

Best Found Value: One of the principal metrics in this context is the 'Best Found Value' on the optimization curve. Essentially, after running an optimization algorithm, the output consists of a sequence of candidate solutions with their associated objective function values. The 'Best Found Value' is the most optimal (either maximum or minimum, depending on the problem) value that the algorithm was able to find during its run. When dealing with stochastic algorithms, the process is typically repeated multiple times, and the 'Best Found Values' are averaged to get a more reliable estimate (Shahriari et al., 2016). This averaged 'Best Found Value' serves as a primary quantitative metric for benchmarking the performance of various blackbox optimization algorithms in this work.

Variance: Another metric that offers valuable insight is the variance in the 'Best Found Values' across multiple runs. Mathematically, the variance σ^2 for the 'Best Found Values' $x_1, x_2, \ldots, x_I$ across I runs is given by:

$$\sigma^2 = \frac{1}{I} \sum_{i=1}^{I} (x_i - \mu)^2 \tag{2.9}$$

In Equation 2.9:

- σ^2 is the variance of the 'Best Found Values'.
- x_i is the 'Best Found Value' in the i^{th} run.
- μ is the mean of all 'Best Found Values' across I runs.

A lower variance signifies that the algorithm consistently finds near-optimal solutions across different runs, making it more reliable. On the other hand, a higher variance implies that the algorithm's performance can be inconsistent. The variance metric is used in this work to provide a deeper understanding of an algorithm's robustness (Eggensperger et al., 2018).

Regret: While the 'Best Found Value' serves as an essential measure for assessing the quality of the solution found by an optimization algorithm, the regret metric provides an alternative perspective on algorithmic performance. The regret measures the absolute difference between the optimal objective function value and the objective function value at a specific iteration. The mathematical definition of regret is:

$$\text{Regret} = |f(x^*) - f(x_i)| \tag{2.10}$$

In Equation 2.10:

- Regret is the absolute difference between the optimal objective function value ($f(x^*)$) and the objective function value at the current iteration ($f(x_i)$).
- $f(x^*)$ is the optimal value of the objective function.
- $f(x_i)$ is the value of the objective function at the current iteration i.

A lower regret value suggests that the algorithm is closer to the optimal solution at that particular iteration, rather than over its full run (Shahriari et al., 2016).

Cumulative Regret: Cumulative regret is an essential metric for evaluating the speed at which the algorithm converges. It is computed as the sum of regrets at each iteration:

$$\text{Cumulative Regret} = \sum_{t=1}^{I} \text{Regret}_i \tag{2.11}$$

In Equation 2.11:

- Cumulative Regret refers to the total sum of individual regrets up to iteration I.
- i represents the current iteration.
- I is the total number of iterations.
- Regret_i is the regret at the i^{th} iteration.

Unlike the 'Best Found Value', which only gives a snapshot of the algorithm's best performance at a specific point, the cumulative regret offers a more comprehensive view of the algorithm's effectiveness over its entire run (Eggensperger et al., 2018). In the setting of optimizing camera positions for deep learning applications, a lower cumulative regret suggests that the algorithm identifies optimal or near-optimal camera positions more efficiently. Conversely, a high Cumulative Regret might imply a more explorative approach by the algorithm.

Number of Function Evaluations, Iterations, and Sampling: The term 'Number of Function Evaluations' refers to the computational cost required by an optimization algorithm, essentially indicating the number of times the algorithm samples the objective function to find an optimal or near-optimal value. In the context of optimization, each function evaluation is equivalent to an iteration, which involves sampling the objective function.

For different optimization methods, this concept varies:

- In Grid Search, each function evaluation corresponds to a single grid point in the parameter space.
- In Random Search, each iteration involves randomly selecting a point in the parameter space to evaluate the objective function.
- In Simulated Annealing, a function evaluation is tantamount to a state transition, which involves a potential move to a neighboring solution.
- In Bayesian Optimization, each function evaluation corresponds to a decision point where the algorithm's probabilistic model recommends the next sample to evaluate.

In scenarios involving computationally intensive tasks like image processing for deep learning applications, a lower number of function evaluations indicates a higher efficiency of the algorithm.

2.10.3 CNN Performance Metrics

Within the context of this work, which centers on optimizing camera positioning using saliency maps generated by deep learning techniques, understanding Convolutional Neural Network (CNN) performance metrics becomes crucial. These metrics facilitate the calibration and generalizability of the CNN models, which is highly relevant in challenging scenarios like the monitoring of gas flow regimes in bioreactors.

Cross-Entropy as a Measure of Accuracy: In the binary classification problem, the cross-entropy loss $H(y, p)$ is calculated using the following equation:

$$H(y, p) = -\sum_{i}[y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \tag{2.12}$$

For multi-class classification, the formula can be generalized to:

$$H(y, p) = -\sum_{i}\sum_{j} y_{ij} \log(p_{ij}) \tag{2.13}$$

In Equation 2.12 used for binary classification, H denotes the cross-entropy loss. Here, y_i signifies the true label for the i^{th} instance, which can be either 0 or 1. The predicted probability of the i^{th} instance being in the positive class is represented by p_i, and i serves as an index for each instance in the dataset. On the other hand, Equation 2.13 pertains to multi-class classification. In this case, y_{ij} indicates the true label for the i^{th} instance in the j^{th} class. The p_{ij} symbolizes the predicted probability that the i^{th} instance belongs to the j^{th} class, and j functions as an index for each class.

Utilizing cross-entropy as a form of 'accuracy' provides a more detailed assessment of model performance compared to traditional accuracy, which only counts the fraction of correct classifications. Cross-entropy accounts for both the correctness and confidence level of each prediction, making it a more comprehensive measure for performance evaluation in the context of camera positioning optimization.

ROC and AUC: The Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) are other metrics that are frequently employed in machine learning literature for performance assessment. The ROC curve illustrates the performance of a binary classification model by plotting the true positive rate against the false positive rate at various threshold settings. AUC, which is the area under the ROC curve, provides a single scalar value that sums up the overall performance of the model.

Advantages:

- ROC and AUC are less sensitive to class imbalance, making them suitable for datasets where one class significantly outnumbers the other.
- They provide a comprehensive view of the model's performance across various classification thresholds.
- These metrics are good for comparing multiple models as they summarize performance in a single scalar (AUC) or a curve (ROC).

Disadvantages:

- These metrics may not be as interpretable as accuracy or cross-entropy, especially for those not familiar with the intricacies of machine learning evaluation metrics.
- In situations where class distribution is balanced and where the performance at specific classification thresholds is not a primary concern, ROC and AUC may offer more information than is necessary.

In various scenarios, ROC and AUC have been considered valuable metrics for evaluating model performance (Z. Zhou et al., 2017). However, for balanced datasets and tasks where interpretability and the confidence of each prediction matter, metrics like cross-entropy are often preferred.

3 Methodology

3.1 Requirement Analysis and Research Hypothesis

The focus of this study is on designing and testing an optimization algorithm that incorporates additional information from the CNN model to optimize camera positioning. This approach aims to provide the algorithm with advantages over other popular optimization methods applicable in the given context of the use case study. The case study will offer a reference scenario used to benchmark the algorithm against other optimization methods. The primary variables under investigation include camera position, bioreactor parameters, and CNN model accuracy.

Drawing upon the extensive literature review, which covers topics ranging from blackbox optimization methods like Bayesian Optimization, Simulated Annealing, and Grid Search, to Explainable AI and performance metrics, the study aims to bridge the gap between these diverse areas. Specifically, the literature on Bayesian Optimization and Explainable AI provides a foundation for the development of an informed optimization algorithm that leverages saliency maps for better performance. Therefore, the study aims to answer the following specific research questions:

Specified research question:

1. From these algorithms, which is most suitable for external knowledge incorporation?

2. How can saliency maps be used, to create a heuristic for an informed search algorithm?

3. Which other novel optimization algorithms are most suited for a quantitative comparison?

4. What quantitative metrics can be used for comparison?

5. How does the algorithm compare to other optimization methods in terms of these metrics?

6. What are the limitations of using saliency maps for camera position optimization in the context of bioreactors?

Research Hypotheses:

1. Features extracted through Explainable AI (XAI) approaches, particularly saliency maps, are effective in evaluating the quality of camera positions in Computer Vision applications within bioreactors.

2. The integration of features extracted through XAI methods into optimization algorithms can be successfully implemented as a proof-of-concept for solving the problem of optimal camera positioning within bioreactors.

3.2 Research Approach: Case Study

This thesis conducts a case study approach to design, benchmark analyze, and explore the use of XAI information. The rationale behind this choice stems from the gap in existing literature, which lacks an appropriate solution for single-camera positioning for CNN classification tasks. Therefore, this research will perform a design study to design and benchmark a proof-of-concept algorithm and conduct quantitative and qualitative analysis to compare its performance against popular gradient-free optimization methods. The implementation of this research incorporates a hybrid methodology, combining both quantitative and qualitative approaches, aiming to provide an assessment of a successful exploration and implementation of XAI features in blackbox optimization.

In this work, both quantitative and qualitative methods are employed to analyze the CNN model to explore the impact of Class Actication Maps (CAM) on model accuracy in different camera positions. The use of qualitative analysis is particularly important for feature selection. The reason being that features highlighted in saliency maps have similarities to what human vision considers important, thus making them appropriate for qualitative evaluation. The research approach involves the following stages:

Design: The initial phase focuses on designing a blackbox optimization method in the area of informed search and defining the research objectives. This stage includes literature research to find the most suitable solution for the usecase of flowregime classification and camera placement. Selecting appropriate architectural configurations, finding useful approaches in CAM processing, considering relevant parameters and determining the scope of the study.

Implementation and Benchmarking: Once the model is designed, it is implemented in a suitable programming framework or platform. The implementation involves coding the CNN architecture, configuring hyperparameters, and preparing the dataset. Rigorous testing is conducted to ensure the correct functioning of the model and to establish a baseline for performance evaluation.

Qualitative Analysis: Qualitative research plays a critical role in the initial exploratory phase of the study, providing valuable insights into the functioning and implications of different methodologies, algorithms, and approaches. In this thesis, qualitative methods are employed to achieve the following objectives:

Understanding Complex Phenomena: Qualitative research methods aid in comprehending the intricate mechanisms through which camera positioning affects image data and, consequently, the performance of deep learning systems. Through qualitative analysis, a deeper understanding of the underlying processes and interactions can be obtained, enabling the identification of nuances and subtleties that may not be captured by quantitative measures alone. This is specificly usefull when evaluating CAMs.

Analyzing Class Activation Maps: Qualitative methods are utilized to analyze CAMs and identify key features, or 'points of interest', that correlate with the accuracy of the model. By closely examining the CAMs, insights can be gained into the visual patterns and characteristics that contribute to the model's performance. Qualitative analysis allows for a nuanced interpretation of these visual cues, facilitating a deeper understanding of the model's behavior.

Quantitative Analysis: In this work, quantitative research methods serve as the cornerstone for evaluating the implemented solutions. Specifically, the CNN model's performance is assessed using metrics such as training time, prediction accuracy, and model complexity. These metrics offer objective data that allows for an analytical understanding of the model's effectiveness. Alongside this, blackbox optimization techniques are applied to empirically determine the optimal camera positions. The resulting numerical data is subjected to statistical analysis to rigorously compare the effectiveness of different camera positions in the given use case. Additionally, Class Activation Maps (CAMs) are used to pinpoint key features within images. The relationship between these features and the model's accuracy is quantified through appropriate metrics, which in turn offers insights into what influences the model's performance. To ensure the reliability of these quantitative evaluations, variance-reducing methods are employed. This involves performing multiple runs of the optimization algorithms with different random initializations and applying statistical measures such as the mean and standard deviation. This approach allows for a stable and reliable comparison of the proof-of-concept against other prominent algorithms in the specific scenario under investigation.

Improving the model: Based on the evaluation results, qualitative and quantitative analyses are made to enhance the performance and effectiveness of the model. This may involve adjusting hyperparameters, refining the architecture, or modifying the model. These improvements aim to address any identified weaknesses and optimize the model's performance and stability.

Uncommon Methodology: Due to the lack of multiple different Bioreactors with different sizes and window constellations, the optimization algorithm will be implemented as a proof of concept in the given use case study. Therefore, the quantitative analysis is limited to the provided reactor only.

3.3 Data Collection

Data collection will involve gathering multiple accuracy metrics from multiple trained CNNs models. Every CNN model will be trained in a different position. The training positions are defined by a gridsearch mechanism. The grid will be defined in a circular horizontal motion around the reactor window, which can be identified with a yaw value. A robotic arm will position the camera in the fine grid to find the global optimum. This data will then be used in the benchmarking phase to compare different solutions against each other with a reduced amount of sampling possibility. The data will serve as a knowledge base and serve for creating the simulation for running the optimization algorithms. The accuracy will serve as the objective function $f(x)$ for optimization, where x represents the yaw or camera position. The experimental setup includes a bioreactor, an industrial camera and a robotic arm for precise camera positioning as seen in figure 5. It should be clearly stated, that the experiments are structured to provide an objective function for later optimization with an intelligent strategy. Variables like revolutions per minute and gas flow can be controlled via orchestration scripts, and the CNN model structure and parameters are based on previous research findings.

3.4 Evaluation and Justification

The focus here is to critically assess the performance of the self-implemented algorithm, particularly in terms of its ability to integrate explainable AI (XAI) features into blackbox optimization. Several metrics will be employed to ensure a thorough qualitative analysis, enabling a multi-faceted view of the algorithm's capabilities.

For the CNNs, R^2 correlation will serve as the quantitative metric for assessing accuracy within the yaw grid, as explained in section 2.10. The objective function $f(x)$ will be subject to optimization through the self-implemented algorithm. This will be compared against other gradient-free optimization algorithms to provide a comparative perspective. Tools such as Tensorflow and Tensorboard will be utilized for overseeing machine learning experiments.

The choice of a simulation-based approach was made after careful consideration of factors like efficiency and accuracy, as well as identified gaps in existing literature. This approach facilitates both efficient data gathering and expedient algorithm testing, thereby aligning well with the study's objectives.

Obstacles and Solutions: The study encountered several obstacles that posed significant challenges to its objectives. One primary concern was the difficulty in selecting an algorithm capable of incorporating external data. To compound this issue, identifying external data that adhered to scientific guidelines proved to be a cumbersome task. The integration process for this data was far from straightforward, necessitating multiple rounds of trial and error to achieve the desired outcome. Data collection posed another challenge. Initially, it was assumed that a large dataset would be essential for establishing a reliable objective function. However, collecting such an extensive dataset would have been time-consuming, extending into weeks or even months. This led to a reduction in dataset size, introducing noise into the training data. To counteract this, variance reduction methods were employed to maintain the integrity of the dataset. Lastly, the study faced inconsistencies in model performance upon retraining, which introduced additional noise. This was mitigated by conducting multiple training runs to stabilize performance. These challenges were addressed through meticulous planning and iterative improvements, thereby setting the groundwork for a more reliable analysis in this work.

4 Concept

This chapter provides an overview of the fundamental components and data flows in the system developed for camera positioning around a bioreactor. The main elements include a bioreactor for conducting the chemical processes, a camera for image data collection, and a robotic arm responsible for adjusting the camera's position. In addition to these primary elements, machine learning models and optimization algorithms are integrated to refine the camera placement iteratively.

The chapter is structured to deliver a clear understanding of how these various components collaborate to achieve the aim of optimal camera positioning. Details on the interaction between these elements, the role of machine learning models, and the functioning of optimization algorithms will be explained.

4.1 System Overview

In the system overview shown in Figure 4.1, a variety of elements are displayed, and arranged to highlight their interconnectivity and data flow. The main elements are a bioreactor, a camera and a robotic arm, positioned vertically on the left side of the figure. Supplemental components include a GradCAM++ image and an XAI-informed Bayesian optimization surrogate model. There are also three separate boxes labeled 'Images Training Dataset', 'Images Validation Dataset', and 'Trained LeNet5-CNN Model'. Data flow is visualized using arrows colored in green, red, purple, and black. Green arrows represent forward data flow, while red arrows denote backward flow. The center of this data flow is the camera, which captures images for the training and validation datasets.

The steps in the data flow are as follows:

- The camera records images for the training and validation datasets.
- Labels from the bioreactor are used by the LeNet5 CNN model.
- The validation dataset helps to evaluate the model's accuracy and to calculate GradCAM++ information.

After completing these steps for one camera position, the accuracy of the CNN model is sent to the XAI-informed Bayesian optimization surrogate model. GradCAM++ information in the form of a vector pointing towards the point of interest, is also sent to this surrogate model to provide additional insights.

The surrogate model then guides the robotic arm to move the camera to a new Yaw value, which sets the next camera position in a circular pattern around the bioreactor. The loop of steps repeats until either the Bayesian optimization algorithm stops improving or a set number of iterations (for example 20) are completed.

Finally, the camera is placed at the optimal position as determined by the iterative optimization process (Brochu et al., 2010; Selvaraju et al., 2017; Shahriari et al., 2016; Snoek et al., 2012).

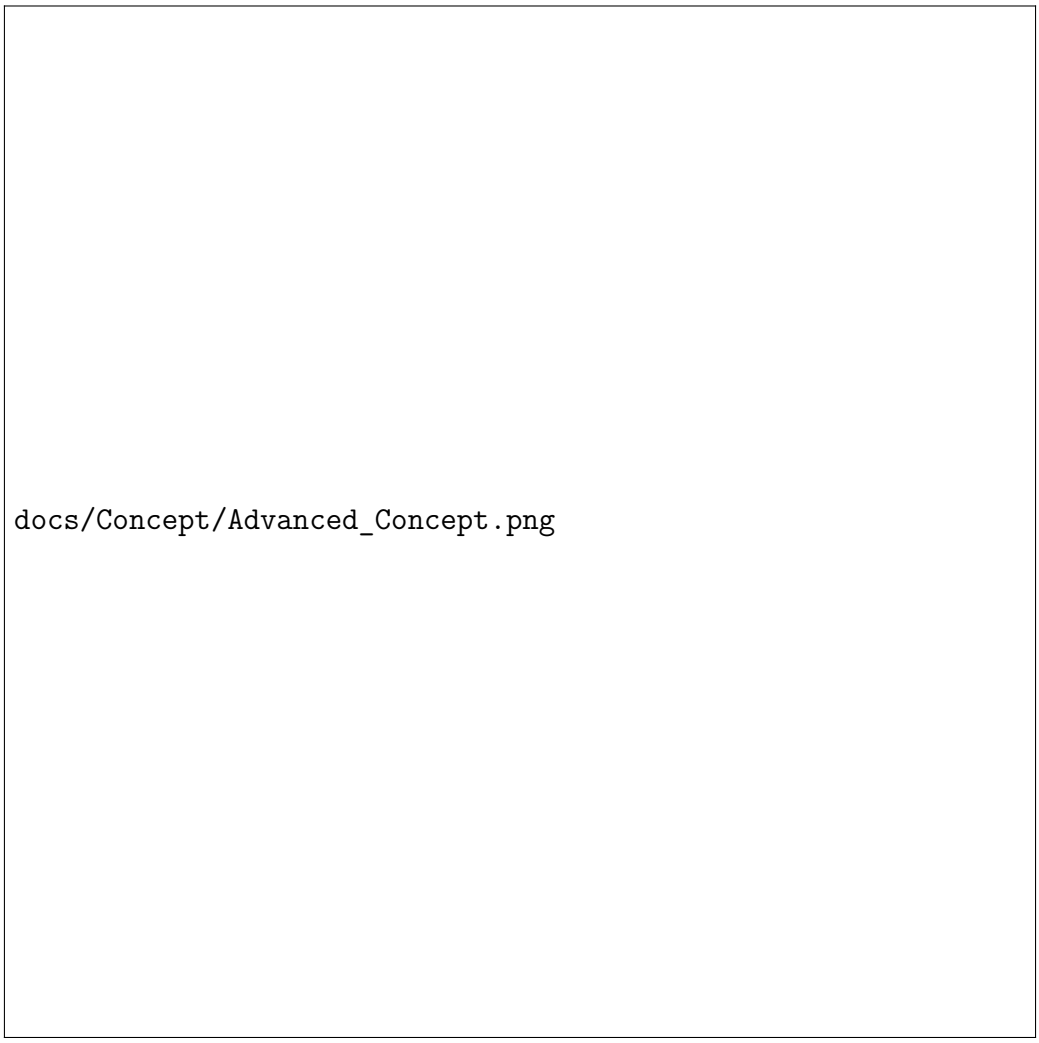


Figure 4.1: XAI enhanced automatic positioning system using GradCAM++

4.2 Data Flow

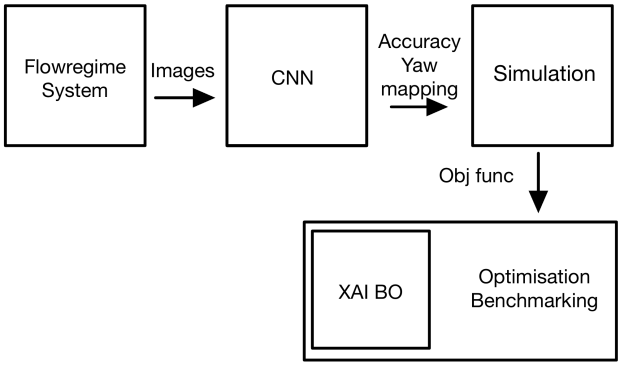


Figure 4.2: Data flow diagram detailing the interactions between various system components. This illustrates derivative-free optimization in the use case of process control engineering.

In the context of process control engineering, the optimization of operational variables is crucial for enhancing both efficiency and performance. This work utilizes a data-driven approach that incorporates Convolutional Neural Networks (CNN) and derivative-free optimization algorithms in a simulated environment. The focus is on optimizing the positioning of cameras in a bioreactor. Figure 4.2 provides a detailed overview of how the various units in the system interact, thereby driving the optimization process. Four core units are discussed: the Flowregime System Unit, CNN Unit, Simulation Unit, and Optimization and Benchmarking Unit.

Flowregime System Unit: This unit acts as the foundational element of the study. It aims to optimize the camera positions in an industrial-grade bioreactor, specifically in the Biostat D-DCU. A camera controlled by a UFactory xArm 5 Lite captures images and has a yaw movement range from yaw_{-45° to yaw_{+45°. The positioning of the camera is a central variable in the study, and the images captured serve as the primary data for training the CNN model.

CNN Unit: Responsible for feature extraction and classification, this unit utilizes images from the Flowregime System Unit for its training. It generates raw performance metrics like accuracy and additionally provides saliency maps. These maps are incorporated into the optimization algorithms and offer insights into the features that are most important, aiding in the formulation of the objective function in the Simulation Unit.

Simulation Unit: Situated between the CNN and Optimization and Benchmarking Units, this unit plays a vital role as an intermediary. It generates an objective function using metrics from the CNN unit along with other parameters from the bioreactor. A grid of sample points is established and fitted with a polynomial function to form a derivative-free optimization landscape. This allows for quicker evaluation cycles, particularly given the high computational costs associated with real-world experiments. Bayesian Optimization and Simulated Annealing are among the optimization methods used here.

Optimization and Benchmarking Unit: This unit contains various optimization algorithms, including a self-designed XaiBO. These algorithms query the objective function in the Simulation Unit and suggest the next data point, either a camera position or a yaw value, for evaluation. The algorithms' performance is then evaluated using quantitative metrics, serving as the final step in the optimization cycle.

The following will elaborate on the data flow and interactions between these units, providing insights into how they collectively contribute to the optimization process.

Interplay: The interaction between the different units can be best understood by referring to Figure 4.2. Initially, the Flowregime System Unit captures images within the range of $yaw_{-45°}$ to $yaw_{+45°}$, which are then used for training the CNN Unit. Post-training, the CNN Unit outputs raw performance metrics and saliency maps, which become integral to the formation of the objective function in the Simulation Unit.

The Simulation Unit, acting as a blackbox, creates an objective function based on these metrics and additional parameters from the bioreactor. This objective function serves as a queryable interface for the Optimization and Benchmarking Unit. Since the internals of this objective function are abstracted, the Optimization and Benchmarking Unit remains uninformed about its detailed workings. The optimization algorithms in this unit suggest the next data point, such as camera position or yaw value. These suggestions could either be implemented in a simulated environment or applied directly in the real-world setup for iterative refinement.

Importantly, the self-designed XaiBO is hosted within the Optimization and Benchmarking Unit and undergoes evaluation against other optimization algorithms. By facilitating this structured data flow, the system allows for a more controlled and repeatable process, which is particularly useful for design iterations, hyperparameter tuning, and performance evaluation. Multiple instances can even be run in parallel for efficient benchmarking.

4.3 Experimental Setup

The following elaborates on the hardware components visible in Figure 4.3. The primary elements are the Biostat D-DCU bioreactor and the uFactory xArm 5 Lite robotic arm, which is equipped with a camera.

Bioreactor: The Biostat D-DCU bioreactor serves as the main hub for generating and observing flow regimes. A transparent window on the vessel becomes the focus point for image capture, ensuring a centralized area for data collection.

Camera Positioning System: Around the bioreactor, a circular horizontal line is marked, indicating possible camera positions. These positions range from $yaw_{-45°}$ to $yaw_{+45°}$, and the camera aligns with the transparent window of the bioreactor for centralized image capture.

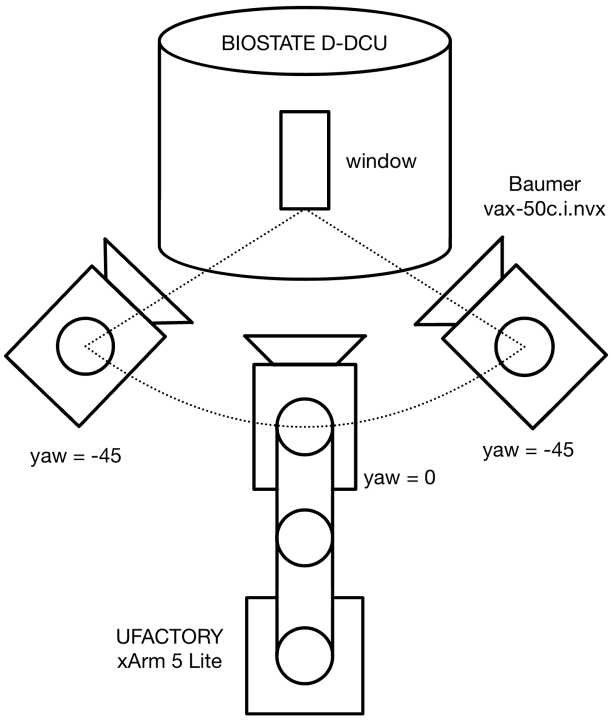


Figure 4.3: Experimental setup featuring the biostate D-DCU bioreactor and camera positions. Details how the camera collects data along a predetermined curve for CNN training.

Data Collection Strategy: At each yaw angle, images are taken and form distinct datasets. The more datasets captured, the less noise is introduced into the training data for the CNN. The number of datasets thus balances with the runtime constraints of the experiment.

The captured datasets from these specific yaw points contribute to the overall model accuracy. These discrete yaw angles are then transformed back into a continuous space through polynomial regression, aiming to optimize the camera's positioning while reducing noise in the data.

4.4 Optimization Challenges and Approaches

Optimizing camera positions for flow regime classification using CNNs brings about unique challenges. One major challenge is the computational and time expense involved in evaluations. This necessitates optimization methods that can efficiently manage the trade-off between exploration and exploitation, while keeping the number of evaluations to a minimum. In situations where the objective function is either not explicitly known or computationally demanding, derivative-free optimization methods gain importance (Audet & Kokkolaras, 2016). These methods, such as Bayesian Optimization and Simulated Annealing, are well-suited for real-world applications where obtaining derivatives is not practical. In the context of this work, the objective function is defined within the Simulation Unit and incorporates CNN performance metrics and other factors essential for flow regime classification.

Benchmarking is another critical aspect of this work, aimed at gauging the efficacy of the optimization algorithms. The work employs a self-designed Explainable XaiBO, which is compared against other standard optimization methods like Bayesian Optimization, Grid Search, Random Search, and Simulated Annealing (Eggensperger et al., 2018; Klein et al., 2017; Samora et al., 2016; Shahriari et al., 2016). These algorithms are evaluated using various metrics, including but not limited to, the rate of convergence, the total number of evaluations required and the overall quality of the obtained solutions.

5 Data Collection and Experimental Setup

The experimental research design plays a crucial role in this study, enabling the manipulation of variables, specifically the camera position, and the measurement of their impact on outcomes, such as model accuracy. This design allows for the establishment of robust evidence regarding cause and effect relationships, which is particularly valuable in the context of this experimental investigation. The main steps involved are Pre-Experiment phase for setting up the hardware and designing and configuring the parameters and scripts that are needed for a successful run. This chapter starts with a detailed explanation of the hardware, then takes the reader in chronological order through the taken steps in the experiment phase. In addition to design criteria from this work, also design criteria from related work are mentioned that also conducted experiments using similar hardware (Kröger et al., 2022; Markus Esser, 2022; Wecke et al., 2022).

Design Requirements for Experiments: The design of experiments in this work addresses multiple criteria to ensure validity and generalizability. Essential requirements include:

- **Temporal Scope of Experiments:** The experiments should fit within a manageable time frame to ensure timely results without sacrificing thoroughness.
- **Adequate Data Points for Optimization Curve:** Sufficient data points should be collected to construct a meaningful optimization curve, serving as a basis for benchmarking optimization algorithms.
- **Representation Space of Datasets:** The dataset should include measurement series covering the entire potential representation space. All possible classes and various ranges within those classes should be represented (Kröger, 2019; Markus Esser, 2022).
- **Visibility of Reactor Interior:** The camera should aim to capture a large area of the reactor interior to maximize information density in the training image set.

- **Lighting Conditions:** Lighting during image acquisition should be constant to prevent the CNN model from being misled. Use of artificial light on top of the reactor is advised (Kröger, 2019).
- **Camera Positioning:** A mechanical arm should be employed for precise camera positioning, contributing to repeatability and data stability.
- **Uniform Distribution of Labels:** Data points should be uniformly distributed over rpm and gas flow rate to avoid unwanted correlation in the training data.
- **Data Preprocessing and Augmentation:** Proper techniques for data preprocessing and augmentation must be applied before training the CNN model to enhance its generalizability.

5.1 Hardware Components

Bioreactor BIOSTAT® D-DCU The BIOSTAT® D-DCU is a compact bioprocess system with vessel options ranging in working volume from 10 to 200 litres. The BIOSTAT® D-DCU is equipped with numerous desirable and cutting-edge features to accommodate virtually any bioprocess application requirement. The bioreactor is considered a stirred vessel with a capacity of 30 liters and is the primary component of the experiment. The reactor is equipped with an integrated step stirrer and interior lighting. The stirrer (turbine) is placed vertically in the centre of the reactor, visible through a viewing window. The reactor is connected to a module for gas supply, through which the incoming gas flow can be configured. The reactor is filled with water. Air is used as a gas phase. The viewing window in the upper portion of the reactor is used to capture images in this study. (Sartorius stedim biotech AG, n.d.)

Camera Baumer vax-50c.i.nvx The experiment is conducted with the *Baumer vax-50c.i.nvx* and *Kowa Lens LM6HC F1.8 f6mm 1.* The camera, as seen in figure 4.1, is an industrial camera with the use case of A.I in mind. It uses *2448 x 2048 px, 2/3" CMOS* camera chip. Specifications are a *6-Core NVIDIA Carmel ARM CPU with 384 Core NVIDIA Volta GPU* and *8 GB 128-bit LPDDR4x RAM.* It is Gigabit Ethernet capable over a USB 3.0 connection. Its main purpose is Image recording with more complex data formation. It is capable of running a Linux OS.

In this paper, this camera is used for collecting image series on an external SD card. The camera parameters are derived from findings of previous work, especially from the findings in (Wecke et al., 2022). In the experiments, specific camera settings were used based on prior work to ensure optimal image capture. An exposure time of 5.653 ms was used consistently across all datasets. Both the objective focus and aperture were set to their maximum values, ∞ and 1.8, respectively Markus Esser; Wecke et al. (2022, 2022). Upon initial testing and dataset sampling, it was found that the chosen exposure time was also suitable for the unique distance of 23 cm between the camera and the reactor. This distance is considerably larger than the 4 cm used in the study by (Wecke et al., 2022). The reason for this increase in distance was the need for free camera movement in a 90-degree arc around the reactor. A closer distance would have resulted in the camera making contact with external reactor components, limiting its range of motion. Therefore, a compromise was made between the optimal camera-to-reactor distance and the need for greater camera mobility, with the latter being prioritized.

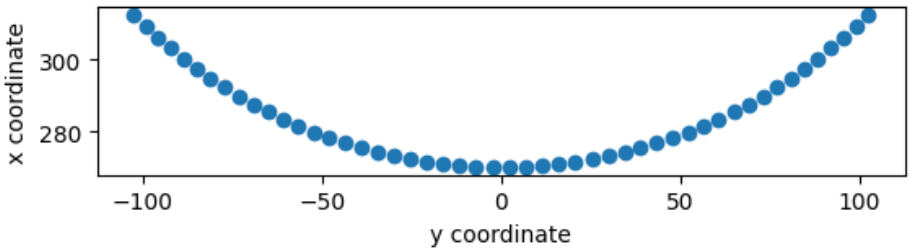


Figure 5.1: Sampled camera positions; 90 yaw angles displayed in the x-y-plane; 0 yaw equals 0 in the y-coordinate

Robotic Arm UFACTORY xArm 5 Lite The camera positioning system is deployed via the UFACTORY xArm 5 Lite is an industrial robot that is mainly used for automating plant production processes. It enables an open-source SDK for beginner-friendly configuration in Python or C++. It provides six degrees of freedom (rotational axes). In this work, the robot is able to hold the camera in a particular place for millimetre-precise positioning and repeatability. The figure denoted by 5.1 illustrates the positions sampled on the x-y-plane, with the camera situated at the midpoint of the radius.

5.2 Data Recording and Design of Experiments

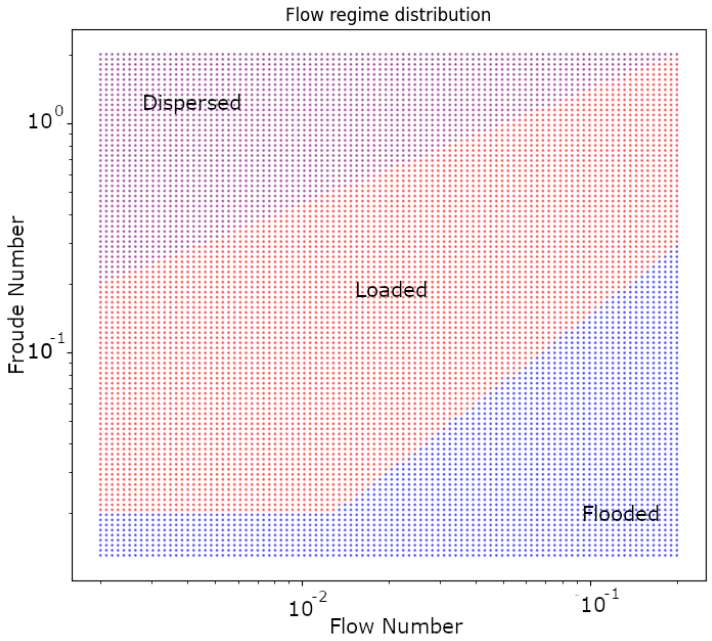


Figure 5.2: Best possible sampling rate of process control state Distribution of Flow Regimes in Experiments

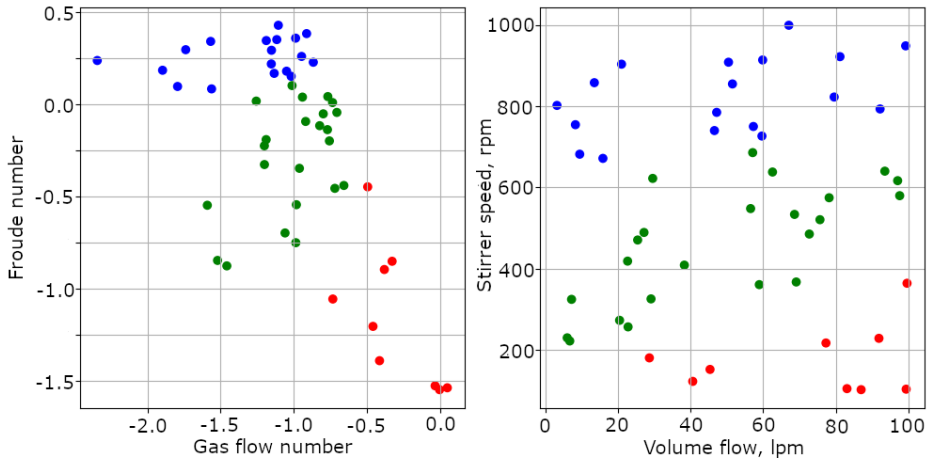


Figure 5.3: Sampled process control state distribution of Flow Regimes in Experiments. Flooded: red; Loaded: green; Dispersed: blue

5.3 Data Collection

Before proceeding to the full experiment, preliminary tests are conducted to validate essential design factors like hardware parameters and orchestration scripts. During this phase, a multiple small dataset where gathered as a trial run.

The robotic arm, a uFactory xArm 5 Lite, is programmed via a Python script to move the camera based on predetermined yaw points around the reactor. This is handled by the orchestration machine 5.4. A flowchart shown in Figure 5.3 outlines reactor states for CNN classification. Reactor states are randomly and uniformly distributed across the volume flow and RPM domains and are randomly selected during the experiment to avoid unintended assumptions that the CNN might develop (Kröger et al., 2022). For accurate image labeling, synchronization between camera timestamps and reactor states is implemented. During the main experiment, 200 images per position are captured at each yaw point. Reactor states are varied by adjusting RPM and gas flow, and images are subsequently labeled as Flooded, Loaded, or Dispersed. Non-uniformity in data distribution is actively corrected, as shown in Figure 5.3, by employing a random distribution of dots in the volume flow and RPM dimensions. This strategy ensures a balanced dataset and limits model bias.

Limitations due to time constraints regarding the experiment duration resulted in only 50 different reactor configurations being used. These constraints impact the CNN's ability to accurately classify the borders of the flow regime states (Kröger et al., 2022). Camera and robotic arm are covered with a light-blocking blanket to avoid external reflections. The hardware parameters can be found in the hardware section 5.1. The CNN model selected is LeNet5, following recommendations from previous works Kröger (2019). An additional dropout layer was added to the model to combat overfitting issues, discussed further in Section 5.4. In total, 56,000 images amounting to 189GBs of data are collected. This dataset is divided into two groups: one for designing the optimization method, including hyperparameter tuning (Optimization training dataset), and another for benchmarking (test dataset). This separation follows common practices in model development and testing, ensuring robust results. Due to the considerable time required for the reactor to stabilize between different RPM and gas flow settings, camera position quantization and the number of sampling points in the flowchart were decided accordingly.

5.4 Post-Experiment

Following the experiment, the obtained results from the small test datasets are compiled and qualitatively analysed to identify correlations and gain insights, including the relationship between class activation map features and model accuracy. Based on these findings, improvements are made to the model and experiment settings. The experiment is then repeated using the updated model and settings to validate the effectiveness of the improvements.

By integrating this qualitative assessment, this research aims to achieve its objectives in a comprehensive and effective manner. The experimental research design allows for the systematic exploration and analysis of the relationship between camera positioning, model accuracy, and the underlying class activation map features, contributing valuable insights to the field.

Training Data Understanding In this section, the focus is on the specific data that feeds into the CNN for each unique position around the reactor. Unlike the complete experimental dataset, which contains 56,000 images, here the term 'dataset' refers to the images captured at a single camera position. Each of these position-specific datasets undergoes its own training phase in the CNN, and the entire pipeline is executed individually for each position. This method helps in understanding the nuances in the data at each position, which is vital for the effective training and performance evaluation of the CNN.

For each of these datasets, the standard practice of splitting the data is followed: 60% for training, 20% for validation, and another 20% for testing, also known as training data split. This equates to around 1000 images that are allocated to each CNN dataset to track the model's performance during training (LeCun et al., 1989).

In the pipeline, the training set is mainly used for the weight adjustments in the CNN. The validation set contributes to the fine-tuning and offers an unbiased metric for performance evaluation. Lastly, the test set is deployed to assess the fully trained model's effectiveness Zhang and Wang (2013).

Training Data Augmentation: Data augmentation strategies are utilized to bolster the CNN model's robustness. By applying various transformations like rotation, scaling, and flipping to the original training samples, new data points are generated Zeiler and Fergus (2014). This procedure accomplishes two things: it enlarges the training dataset and assists the model in better generalizing to unfamiliar data.

Specifically, in this work, standard data augmentation techniques such as random cropping, horizontal flipping, and brightness adjustments were performed on the training images. Additionally, the data was converted to greyscale and resized to 56x56 pixels to conform to the LeNet5 architecture requirements as described in the work of Kröger; Markus Esser; Wecke et al. (2019, 2022, 2022). The images were also subjected to rotations and flipping operations. Collectively, these augmented datasets had a positive impact, enhancing both the performance and robustness of the CNN model.

Various factors are considered when evaluating the training results and preprocessing stages in optimization. These factors include the risk of overfitting or underfitting the model, divergence in the validation curve, the reduction of noise in experimental data, and the analysis of data correlation and data regression. All of these criteria will be analysed in this section.

Training Evaluation with Tensorboard: The performance and generalization of the Convolutional Neural Network (CNN) were assessed over a span of 90 epochs. Key metrics include accuracy and loss for both the training and validation datasets. In addition, test accuracy served as an auxiliary gauge of model behaviour. The issues of overfitting and underfitting are monitored using Tensorboard, which provides learning curves for the model. These curves are useful for detecting whether the model is learning too much from the training data (overfitting) or too little (underfitting). Tensorboard also helps in identifying any divergence in the validation curve. To mitigate the risk of such divergence, early stopping mechanisms, and 'restore best weights' are utilised.

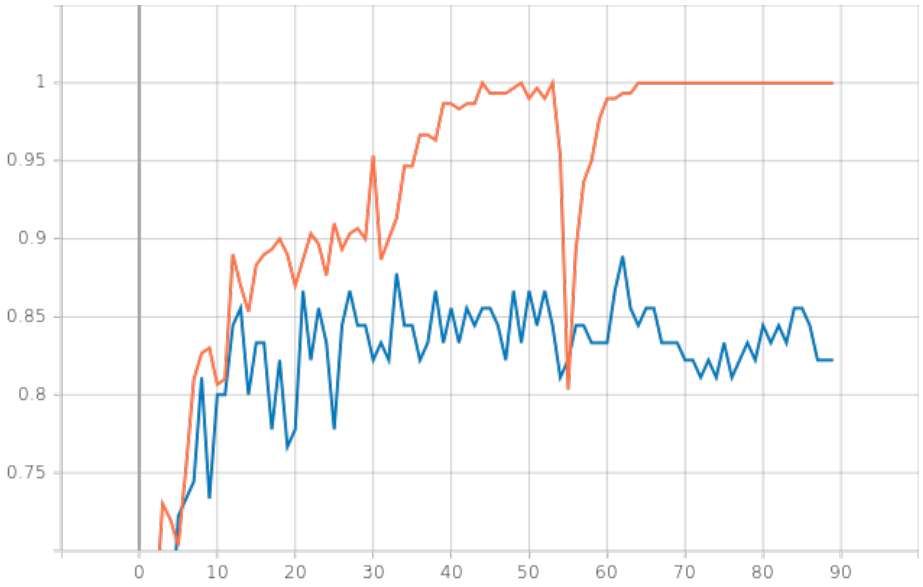


Figure 5.4: Progression of Training Accuracy Over Epochs. Y-axis Accuracy, X-axis Epochs. Training accuracy: orange; Validation accuracy:blue

Accuracy Trends: The training accuracy shown in figure 5.4 displays an initial rise to a level of around 0.9 until the 15th epoch. After the 10 Epoch, the validation accuracy ocilates around an accuracy value of 0.83 . The Training accuracy, escalated to reach a value of 1 between the 30th to 50th epoch. At epoch 55 the dropout layer randomly omitted a crucial value for the training accuracy curve while the validation accuracy seemed mostly unaffected. This might show the value of employing a dropout layer.

Shortly after the training accuracy recovers while the validation accuracy also gains a short boost so closely under 0.9 accuracy. Soon after, this spike decreases. The training process did not stop from early stopping in this case, due to the dropout event. Best weights are restored after training, which was at epoch 63, where the validation accuracy gained has its maximum. No overfitting can be seen due to the dropout implementation, which also can be seen from the oscillation of the training and validation curve as well as by the plato the training curve experiences from epoch 10 to epoch 30. This rise would be much faster without a dropout layer, which resulted in overfitting in previous experiment runs.

Data Correlation and Noise Reduction in Experimental Data: The process of diminishing noise in the experimental data and evaluating data correlation was undertaken. The correlation was quantified using the R^2 score. To improve this score and thereby enhance data correlation, variance reduction techniques were employed. Specifically, methods like Monte Carlo simulations were applied, which elevated the R^2 score from 0.156 to 0.3339. Additionally, datapoint regression techniques were utilized to transform discrete datapoints into a continuous function, thus enabling a more robust analysis of inherent data patterns.

Data Regression: Data regression is essential for reducing the noise introduced during the data collection phase. While a noisy landscape may realistically represent production scenarios when investigating the influence of XAI features on optimization algorithms, it significantly increases the complexity of the task. Although optimization algorithms can handle a certain level of noise through algorithmic modifications, such adjustments are beyond the scope of this study. Therefore, the benchmarking will focus on a noise-free environment, fully acknowledging the limitations that come with this approach. Further discussion on these limitations can be found in the Discussion chapter 8.

In this context, the regression model of degree 11 is scrutinized for its performance and robustness. The R^2 score serves as the performance metric, in line with the criteria set out in section 2.10 and corroborated by Eggensperger et al. (2018).

Upon transforming the data into a higher-dimensional feature space, Linear Regression is applied. The coefficients, determined by the `fit` method, reflect how well the model can capture nonlinear dependencies such as those between the yaw angle and weighted average accuracy. These coefficients are mapped back into the original feature space, rendering a polynomial function of degree 11.

The obtained R^2 score was 0.33, indicating a moderate level of data fit. This suggests that the model captures a reasonable amount of variance but may not fully encapsulate all the nuances of the dataset. The polynomial degree of 11 was purposefully selected to achieve a balance between model complexity and fit, mitigating the risk of overfitting.

Computation: All ML operations and bulk data formation were done on a dedicated machine containing a Quadro RTX 4000 GPU. It was applied to data preprocessing, augmentation, neural network training, hyper-parameter optimization, evaluation (inference), and visualization (salience maps). The programming was done in Python 3 using mainly Keras and its backend, Tensorflow.

6 Implementation

This implementation section aims to provide a comprehensive understanding of the code structure and the computational methodologies employed to implement the intelligent optimization algorithm that combines Bayesian Optimization and XAI. It will explain how the Gradient-Based Class activation maps (GradCam++) are generated and utilized to enhance the Bayesian optimization method. Also, the benchmarking of the algorithm in comparison to other prominent algorithms in the field is implemented. Therefore, the recommendation scalar is introduced that is computed from the GradCam++ images. Also, the integration of the recommendation scalar

6.1 Simulation Unit

The Simulation Unit is fundamental to the development and benchmarking of the intelligent optimization algorithm. It serves as a surrogate for real-world experiments, enabling efficient data collection and iterative algorithm testing without the time constraints and complexities associated with physical tests.

Simulation Creation The purpose of creating this simulation is to facilitate the gathered information persistently and reusable in a Postgres SQL database layer. It enables the development and benchmarking of the optimization algorithm intended for robotic camera placement effectively. The simulation also mimics the robotic arm's movement across a grid of potential camera positions by querying the data collected in the data collection chapter 5. Therefore, the robot arm was moved systematically in a fine grid to sample a wide range of possible positions, as explained in chapter 5. This was done by a Python script that interacts with the UFACTORY Xarm6 lite, to create a comprehensive Postgres SQL database with all knowledge that is computed and is needed for the simulation.

Benchmarking Interface To standardize the testing environment and to facilitate algorithm comparisons, a Python interface for the simulation is designed. This interface allows various optimization algorithms, such as Bayesian Optimization, Simulated Annealing, and others, to access the simulation's data. This collection of data serves as the objective function for these algorithms, effectively simulating robotic movement through the grid of potential camera placements. A pseudo-code snippet of this is depicted below:

```
# Pseudo-code to demonstrate the benchmarking interface
def objective_function(position):
  return simulation_data[position]

# Access objective function from different algorithms
bayesian_optimization(objective_function)
simulated_annealing(objective_function)
```

Creating the Objective Function In this work, the objective function plays a vital role, as it serves as the measure that the optimization algorithm seeks to improve. This function is constructed based on simulation data and it mimics the behavior of the real-world robotic arm and camera system. To achieve a continuous objective function akin to what would exist in a real scenario, regression techniques were employed. Notably, the simulation was designed to eliminate data noise, a decision that comes with its own set of trade-offs.

This intentional omission of noise simplifies the implementation and has several advantages. Most notably, a noise-free environment streamlines the optimization landscape, making it easier to navigate and analyze. Such a simplified landscape aids in building a more foundational understanding of the optimization problem and implemented XAI features at hand, thereby offering valuable insights into the theoretical aspects of system behavior.

Moreover, the results derived from optimization in a noise-free environment are more foundational, as they represent the best-case performance of the system under ideal conditions. These 'noise-free' results can serve as a benchmark against which real-world performance can be compared, allowing for a clearer assessment of the system's limitations and potential for improvement, which is out of scope for this thesis.

It is also worth noting that optimization algorithms tend to produce results faster in a noise-free setting. This is because each evaluation of the objective function is deterministic, thus eliminating the need to perform multiple evaluations to average out the noise.

Overall, while the absence of noise in the simulation might detract from the realism, it does facilitate a more efficient and fundamental investigation into the system's performance. This decision to eliminate noise was made carefully, taking into account both its advantages and disadvantages.

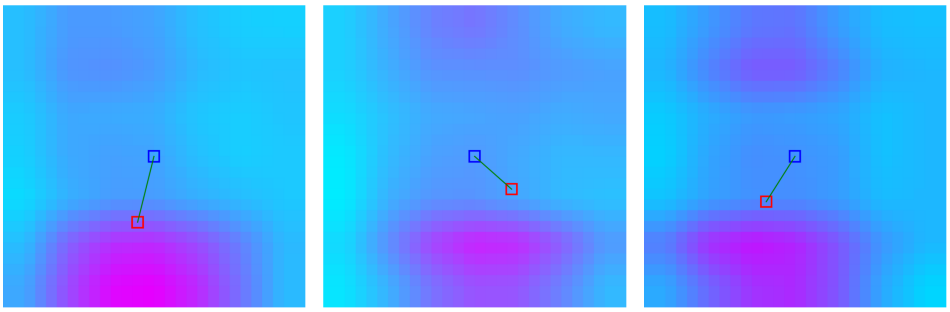
```
# Pseudo-code for data utilization
def construct_objective_function(simulation_data):
  return lambda position: simulation_data[position]
  ['quality_metric']
```

Quality Metric The quality metric for optimization is the accuracy value associated with each potential yaw value, as explained in chapter 4.1. The optimization algorithm aims to find the position that maximizes the accuracy.

Objective Function Properties In this work, the objective function is tailored to be continuous but not necessarily differentiable. This design choice is particularly relevant for optimization techniques that rely on gradient-free methods. One characteristic of blackbox functions is that they do not provide any system-specific information, including gradients. This absence of gradient information limits the use of gradient-based optimization algorithms, which are among the most widely used and effective methods for optimization (Shahriari et al., 2016).

6.2 Recommendation Scalar from Saliency Maps

In this section, the focus is on the generation and utilization of saliency maps, specifically GradCAM++ for calculating the Recommendation Scalar, which serves as the information that will be fed into the optimization algorithm in addition to the objective function. The blackbox is defined as finding the optimal camera position for the classification task, so using information from the CNNs models is not 'breaking' the blackbox.



(a) yaw = 40 **(b)** yaw = -3 **(c)** yaw = 10

Figure 6.1: Guidance Mechanism of Saliency Maps in the Optimization Process

Intuition: In Figure 6.1 three exemplary GradCAM++ images are shown, labelled as (a), (b), and (c), which were obtained from different yaw positions.

Each of these images is a composite, calculated by arithmetically averaging GradCAM++ outputs across the **entire** validation dataset, offering an averaged GradCAM++ (AGCAM) and view of the area of interest for a given camera position. The GradCAM++ images serve as a visual aid in identifying regions of interest within the camera's field of view. Image (a), for instance, displays a Gaussian-shaped purple zone located at the bottom, signifying the area of greatest interest. Additionally, a blue rectangle marks the center of the image, while a red square represents the image's center of gravity. A line connecting the blue and red squares indicates the direction, which is crucial for the calculation of the Recommendation Scalar.

GradCAM++ Generation The computation of GradCAM++ saliency maps serves as a pivotal component in the task of classification for this work. These saliency maps, as illustrated in figure 6.1, reveal critical regions within the images. These visual cues are instrumental in later determining the 'recommendation scalar', a metric that influences the direction of the intelligent optimization algorithm during its search phase.

The algorithmic steps to compute the GradCAM++ maps can be mapped to specific lines in the Python implementation, as shown below:

```
# Corresponds to Lines 10-24
 1:  conv_output, predictions
     = model_forward_pass(image, model)
# Corresponds to Lines 25-42
 2:  grad1, grad2, grad3
     = compute_gradients(conv_output, predictions)
# Corresponds to Lines 44-54
 3:  alpha, weights
     = calculate_alpha_weights(grad1, grad2, grad3)
# Corresponds to Lines 56-61
 4:  grad_CAM_map
     = get_gradCAM_map(alpha, weights, conv_output)
# Corresponds to Lines 63-67
 5:  heatmap
     = normalize_heatmap(grad_CAM_map)
```

To clarify these steps:
1: The forward pass of the model is conducted to obtain both the convolutional output and the model's predictions.
2: Gradients of first, second, and third orders are computed from the retrieved convolutional output and predictions.
3: Subsequently, alpha values and weights are determined from these gradients.
4: The GradCAM++ is synthesized by integrating the alpha values, weights, and convolutional output.
5: Finally, this GradCAM++ is subject to normalization, resulting in the finished heatmap.

More information on GradCAM++ can be found in the state-of-the-art chapter section 2.8. The GradCAM++ maps are made for all images in the validation set. After going through all images in the validation set, this total sum is divided by the number of images. This gives an average heatmap. This average shows what parts of the images are usually important for the model when it is making its decisions. This helps to make better choices when tuning the model and gives a more solid basis for evaluating its performance.

Next, the average heatmap is normalized. This means its values are changed to be between 0 and 1. This makes it easier to compare the heat maps.

6.3 Saliency Map Features as Guidance Mechanism

The GradCAM++ saliency maps are generated for each camera position, providing insights into the areas of interest within the images. These insights are then utilized in the intelligent optimization algorithm to guide the search in a more informed manner. How this is computed is explained in the following. GradCAM++ images form the foundation for calculating what is termed the 'recommendation scalar'.

Computation of Recommendation Scalar The recommendation scalar is a critical value used to approximate the center of gravity from the averaged GradCAM++ outputs. It aids in making decisions about where to position the camera in the stirred vessel for the next capture. This scalar is calculated by computing the weighted average of grayscale pixel values, giving more weight to pixels that are considered more important by the GradCAM++ algorithm.

In the Python implementation, the `compute_rec_scalar` function plays a key role in calculating this scalar. It starts by scaling the image size with an upscale factor of 100, applying antialiasing supersampling. After resizing, the function converts the image to grayscale. Weighted averages are calculated for x-coordinates based on the intensity of each pixel. Finally, the function finally returns the relative x-coordinate of the center of gravity, which is used by the Bayesian Optimization algorithm to decide the next camera position.

The formula for the recommendation scalar is given as:

$$\text{Recommendation scalar} = \frac{\sum_{i=1}^{n} w_i \cdot x_i}{\sum_{i=1}^{n} w_i} \tag{6.1}$$

where w_i are the pixel intensities and x_i are the pixel x-coordinates. The n describes the width of the image in a squared format. The formula aims to capture the essence of what parts of the image the model considers important, guiding the optimization algorithm to move the camera effectively.

The following Python code shows the core logic of the `compute_rec_scalar` function:

```
 1:  def compute_rec_scalar(img):
# Antialiasing supersampling
 2:  |  UPSCALE_FACTOR   = 100
 3:  |  upscaled_img     = cv2.resize(
                            img,
                            (img.shape[1] * UPSCALE_FACTOR,
                            img.shape[0] * UPSCALE_FACTOR),
                            interpolation=cv2.INTER_NEAREST)

 4:  |  grayscale_img    = cv2.cvtColor(upscaled_img,
                                        cv2.COLOR_RGB2GRAY)
# Normalize the grayscale image
 5:  |  min_val, max_val = np.min(grayscale_img),
                           np.max(grayscale_img)
 6:  |  normalized_grayscale_img
     |                   = (grayscale_img - min_val) /
     |                     (max_val - min_val)
 7:  |  total_weight     = np.sum(normalized_grayscale_img)
 8:  |  avg_x            = np.sum(normalized_grayscale_img *
     |                     np.arange(grayscale_img.shape[1])) /
     |                                        total_weight
 9:  |  center_x         = (grayscale_img.shape[1] - 1) / 2
 10: |  center_of_gravity_x = avg_x - center_x
 11: |  return center_of_gravity_x
```

The center of gravity in the saliency maps serves as a feature for Explainable Artificial Intelligence (XAI) in this work. The choice of using the center of gravity is motivated by its ability to offer a comprehensive summary of the saliency map. By calculating this singular value, the algorithm can acquire a balanced view that incorporates all aspects of the saliency features in the image, rather than focusing on individual hotspots. This approach aims to furnish the optimization algorithm with a generalized, yet nuanced signal for making subsequent decisions on camera positioning within the stirred vessel.

Therefore, the center of gravity acts as an effective feature for summarizing complex saliency data, streamlining the optimization process, and enhancing the decision-making capabilities of the system. The recommendation scalar is a valuable asset in this work, aiding in effective camera placement by guiding the Bayesian optimization algorithm.

6.4 GradCam++ Enhanced Bayesian Optimization

This section details the implementation of the Bayesian Optimization algorithm integrated with an XAI Feature. This feature is the recommendation scalar that was computed as center of gravity from the averaged GradCam++. The focus is on integrating the recommendation scalar into the posterior probability kernel in Bayesian Optimization. More information on Bayesian Optimization is given in chapter 2.4.

In the implementation of Bayesian Optimization, two key components are set up: the Gaussian Process model and the acquisition function. The Gaussian Process model predicts the performance of different settings, and Scikit-learn is used for this task. The acquisition function, specifically Expected Improvement (EI), guides the selection of the next camera position. The EI is a unitless value that is computed from the mean curve *mu* and variance band of the GP model. A simplified pseudo-code snippet is shown below:

```
# Compute Expected Improvement
 1:  EI      = calculate_EI(x_test, mu, sigma)
# Select next point based on max EI
 2:  x_next = select_next_point(EI)
```

The `calculate_EI` function computes the EI for each possible next point, referred to as x_{test}. It uses the mean (μ) and variance (σ) of the model's predictions for these points. The goal is to find the point that has the highest potential to improve the objective function.

The variable `EI` stores these potential improvements for each test point. The function `select_next_point` picks the next point to evaluate (x_{next}) based on the maximum value in `EI`. In summary, the EI serves as a criterion to select the most promising next point.

Gaussian Process model In Bayesian Optimization, the Gaussian Process (GP) model plays a critical role in estimating the underlying function that

we aim to optimize. The GP model is fully described by a mean function, often denoted as μ, and a covariance function, or kernel. In this work, the GP model is used to predict the mean ($\mu_\star$) and variance ($\text{var}_\star$) of the test points based on the training data.

The key computation steps for $\mu_\star$ and $\text{var}_\star$ are as follows:

```
# Compute the predictive mean (mu_star)
 1:  inv_K    = np.linalg.inv(K)
 2:  mu_star  = K_star @ inv_K @ y_train + offset_kernel
# Compute the predictive variance (var_star)
 3:  temp     = K_star @ inv_K
 4:  var_star = self.kernel(x_test, x_test) -
        np.einsum('ij,ij->i', temp, K_star)
```

Here, K is the kernel matrix for the training points, and $K_\star$ is the cross-kernel matrix between the test points and the training points. The symbol '@' represents matrix multiplication, and 'np.linalg.inv' computes the inverse of a matrix. The y_{train} contains the observed objective values at the training points, and offset_kernel is an offset that can be applied to the mean function.

The predictive mean $\mu_\star$ is computed as a linear combination of the observed values (y_{train}) weighted by the inverse kernel matrix and the cross-kernel matrix. The addition of offset_kernel adjusts the mean and **serves as the interface for the recommendation scalar**. The predictive variance $\text{var}_\star$ is calculated as the difference between the diagonal of the kernel of the test points and a term dependent on $K_\star$ and K. The term ensures that the variance takes into account the similarity between the test points and the training points.

In essence, the GP model provides both an estimate of the function value at a new point ($\mu_\star$) and the uncertainty of this estimate ($\text{var}_\star$). These are then used in the EI calculation for selecting the next point to evaluate. The pseudo-code for the calculation of the kernel function is shown below.

```
# Python pseudo-code for Bayesian Optimization implementation
of the kernel function
```

```
# Define the Gaussian Process kernel
def bluekernel(self, a, b):
|  return np.exp(-0.5 * ((a - b) ** 2 /
|                          self.KERNEL_SCALE ** 2))
```

In the given pseudo-code, the function 'predict' is a method of a Gaussian Process model and serves to predict the mean and variance at test points given some training data. Here is a breakdown of the variables involved: `a` and `b` are the input vectors for which the kernel value is computed. `KERNEL_SCALE` is a hyperparameter that controls the scale of the Gaussian kernel. The custom kernel function that incorporates the recommendation scalar is called offset kernel because it offsets the Gaussian model's $\mu_\star$ and $\text{var}_\star$ in a particular way, as shown in figure 6.2. This function is utilized to modify the posterior distribution in the GP model, steering the optimization algorithm towards specific regions. The function essentially acts as an informed guidance mechanism for optimization, which takes into account the outputs of XAI methods through the recommendation scalar. The offset scalar by itself is just a value that represents the center of gravity in the GradCam++ image, therefore it must be transformed into a usable data structure for the $\mu_\star$-vector. $\mu_\star$ contains the predicted mean curve, $\text{var}_\star$ contains the probabilistic band that surrounds the mean curve. To manipulate these curves, at the evaluated positions x, a perturbation function will be added. One perturbation function is spawned at every evaluated point and is accumulated in the offset-kernel that ranges over the entire input space X. This results in a probabilistic model shown in figure 6.2. The skew of the mean value and the GP is clearly visible. At -7 and 12 yaw, almost no offset of the GP can be seen due to the recommendation scalar being close to zero.

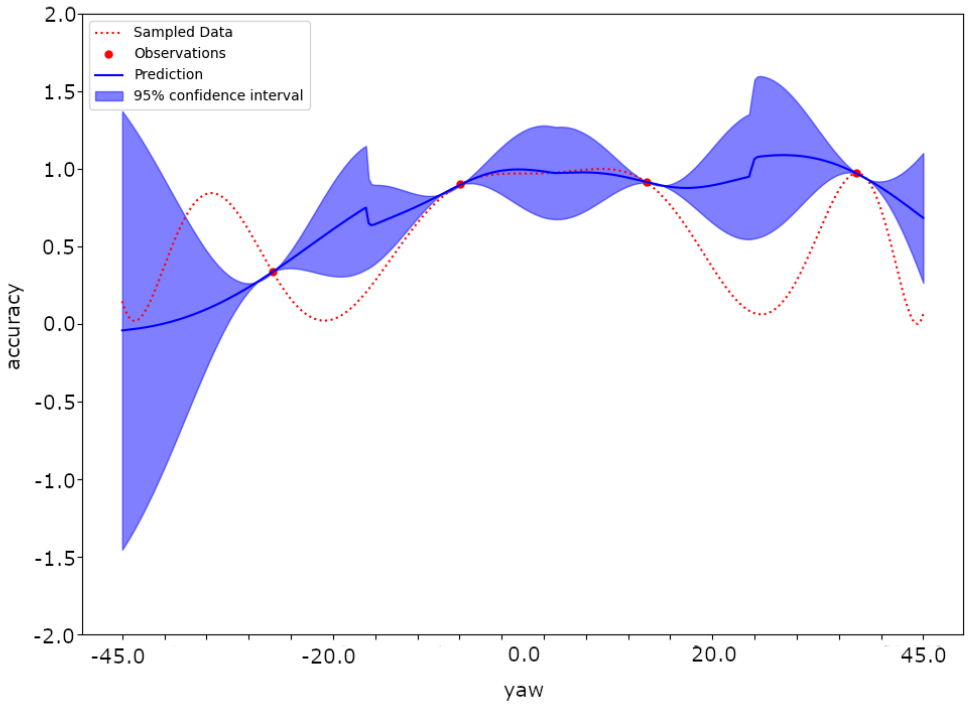


Figure 6.2: Manipulated GP; Perturbation is added to the gaussian model for every sampled point; the height of the offset depends on the recommendation scalar and sign

The function can be expressed in mathematical terms as follows:

$$\text{perturbation}(x, \mu, \text{width}, a, \text{y\_offset}) = a \times \frac{x - \mu}{\text{width}} \times e^{-\left(\frac{x-\mu}{\text{width}}\right)^2} + \text{y\_offset}$$

Here, the variables serve the following roles:

- x: The input sample point.
- μ: The mean around which perturbations occur.
- width: A scale factor that determines the width of the Gaussian term.
- a: Amplitude of the perturbation, typically derived from the recommendation scalar.
- y_offset: A constant offset applied to the perturbation function.

The function multiplies the Gaussian term with a factor $\frac{x-\mu}{\text{width}}$, thereby skewing the Gaussian distribution based on the value of x. This allows the function to be more sensitive to variations around μ, which is generally the point of interest. This tailored sensitivity is crucial for leading the optimization algorithm to explore specific regions of the function space, particularly those regions that are considered important based on the insights provided by the XAI methods.

This approach can be seen as a fusion between traditional optimization techniques and modern XAI, facilitating a more focused and informed search strategy.

Hyperparameter Optimization To make the GP model effective, a systematic search is carried out for optimizing its hyperparameters. These include the length-scale and the constant multiplier of the RBF kernel.

Hyperparameter tuning Hyperparameters are fine-tuned to regulate the impact of the recommendation scalar on the optimization, the kernel size, and the perturbation range variable. The hyperparameter search was done by a grid search method. No normalization was employed to the recommendation scalar, due to a hyperparamter of the Bayesian Optimization algorithm, that by itself scales this value. This scalar was also optimized in hyperparameter optimization, as explained in the implementation section 6.4, therefore normalizing the recommendation scalar. A pseudo-code snippet of this is shown below:

```
# Python pseudo-code for hyperparameter optimization
 1:  from sklearn.model_selection import GridSearchCV

# Create a parameter grid
 2:  param_grid = {"kernel__k1__constant_value":
                        [1e-3, 1e-2, 1e-1, 1, 10],
                   "kernel__k2__length_scale":
                      [1e-2, 1e-1, 1, 10]}

# Perform Grid Search
 3:  grid_search = GridSearchCV(gp, param_grid=param_grid)
 4:  grid_search.fit(X_train, y_train)
```

Summary In essence, the recommendation scalar serves as an effective heuristic. It aims to direct the Bayesian Optimization to regions in the search space that are likely to yield better outcomes, thus improving the efficiency of the optimization process.

6.5 Benchmarking Unit

In this section, the focus is on detailing the implementation steps for benchmarking the developed blackbox optimization method. The primary aim here is to establish a set of procedures and metrics for comparing the performance of the optimization methods used, which include Bayesian Optimization, Random Search, Gridsearch, and Simulated Annealing. This serves to validate the efficiency of the method in determining optimal camera coordinates in the x-direction, particularly in the context of flow regime classification in stirred vessels.

The first step is to specify the performance metrics that will be used for evaluation. Common metrics such as accuracy, precision, and recall could be employed for the CNN model trained at each camera position. Additional metrics tailored for blackbox optimization, like the number of function evaluations or convergence rate, are also considered (Martinez-Cantin, 2019; Shahriari et al., 2016).

Next, the procedure for training the CNN model at each camera position is outlined. This involves specifying the architecture, hyperparameters, and the training and validation datasets. The Bayesian Optimization method is then applied to optimize the model's performance based on the chosen metrics. A standardized data generation method is used for creating the input to each optimization method. This involves setting the initial sampling strategy, which could be either random or based on some heuristic. The optimization methods are then run on this dataset, and their performance is logged for comparison.

The Bayesian Optimization method is implemented by defining the GP Regression model, the acquisition function, and other necessary components (Brochu et al., 2010; Shahriari et al., 2016). The other optimization methods, namely Random Search, Gridsearch, and Simulated Annealing, are implemented in a similar manner, but with their own specific components (Samora et al., 2016).

For each optimization method, the best-found camera coordinates in the x-direction are noted, and the CNN model is trained at these positions. The performance metrics are then computed for each model. A logging system is put in place to record all the necessary data for later analysis. This includes recording the metrics at each iteration, the coordinates selected, and any other relevant information. By detailing these steps, this section provides a guide on how the benchmarking unit is implemented for assessing the blackbox optimization methods in the context of this work. This lays the groundwork for the analysis and evaluation that will follow in the subsequent chapters.

6.6 Benchmarking

Benchmarking is an integral component of this work, aimed at evaluating the effectiveness of the optimization algorithms on the constructed objective function. Various techniques are employed for this purpose, as outlined below.

Brute Force Brute force is the simplest, yet most computationally intensive approach for optimization. The algorithm evaluates the objective function at every possible camera position within the feasible region as explained in the section 2.6.

```
# Pseudo-code for Brute Force
for position in all_possible_positions:
    evaluate_objective_function(position)
```

Random Search The random search algorithm randomly samples points from the feasible region and evaluates the objective function at these points. Although computationally less intensive than brute force, random search offers no guarantee of finding the global optimum, as explained in section 2.6.

```
# Pseudo-code for Random Search
for _ in range(N_iterations):
    random_position = sample_random_position()
    evaluate_objective_function(random_position)
```

Grid Search Grid search discretizes the solution space into a grid and evaluates the objective function at each grid point. This technique is computationally expensive but provides better coverage of the solution space compared to random search, as explained in section 2.6.

```
# Pseudo-code for Grid Search
for position in grid_positions:
    evaluate_objective_function(position)
```

Simulated Annealing Simulated Annealing (SA) aims to find optimal camera positions by iteratively exploring neighboring solutions. The algorithm employs a temperature parameter that guides its search. Two key hyperparameters, maxTemp and minTemp, are optimized to set up an effective cooling schedule.

Initially, hyperparameters maxTemp and minTemp are optimized using methods like grid search or random search (Samora et al., 2016). These set the upper and lower bounds for the temperature.

```
# Pseudo-code for Simulated Annealing
 1:  optimize_hyperparameters()
# Optimize maxTemp and minTemp
 2:  current_temperature = maxTemp
 3:  current_position    = random_initial_position()
 4:  for _ in range(N_iterations):
 5:  |  new_position    = get_neighboring_position(
                                        current_position)
 6:  |  delta           = evaluate_objective_function(
           new_position) - evaluate_objective_function(
                                        current_position)
 7:  |   if delta < 0 or random()
                   < exp(-delta / current_temperature):
 8:    |     current_position     = new_position
# Update temperature
 9:    |     current_temperature *= cooling_rate
```

Here, maxTemp is the starting temperature, and cooling_rate is a factor by which the temperature is multiplied in each iteration. The temperature is thus reduced exponentially until it reaches minTemp.

The Δ variable computes the difference between the new solution and the previous one. A negative Δ signifies that the new solution is better than the previous one. The distinctive feature of Simulated Annealing (SA) is its ability to accept not only better solutions but also worse solutions. In the case of a worse solution ($\Delta > 0$), the algorithm decides randomly whether to accept it, guided by an exponential function. This probabilistic acceptance is governed by the temperature parameter.

In summary, the temperature parameters maxTemp and minTemp control the degree of randomness in the algorithm's decision-making. By fine-tuning these parameters, a balance between exploration and exploitation is achieved, allowing the algorithm to effectively locate optimal solutions. For more information on simulated annealing, refer to section 2.5

Variance Reduction To improve the reliability of the benchmarking results and to mitigate the numerical initial value problem, each algorithm is executed multiple times, and the arithmetic average of the results is computed for each iteration. This reduces the variance of the performance metrics and provides a more accurate depiction of each algorithm's capabilities (Rasmussen & Williams, 2006).

7 Results and Evaluation

This chapter presents empirical findings answering research questions and hypotheses using key graphs: figure 7.3 for accuracy, figure 7.5 for variance, and figure 7.4 for cumulative regret. We evaluate optimization algorithms—grid search, Bayesian optimization with and without explainable A.I. (XAI) features , random search, and simulated annealing—to robustly address the research hypotheses, particularly the effectiveness of XAI in camera positioning within bioreactors.

Initial results focus on the benchmarking performance of the implemented algorithm with incorporated XAI features, through metrics like best-found value, cumulative regret and variance as described in chapter 2.10. This directly aligns with Research Hypothesis 2 on the integration of XAI into optimization algorithms. This thesis also examines the role of saliency maps in flow regime classification within the bioreactor as a use-case study, in line with the research questions (Chapter 1.3).

It is necessary to recognize the limitations of relying on only one bioreactor model for this study. The subsequent discussion chapter will elaborate on the broader implications and limitations, taking into account the empirical data and observations at the core of this research.

7.1 Experiment Data Analysis

The results of the data collection phase are discussed in this section. The data is used to create an optimization landscape for benchmarking the implemented algorithms. Refer to Figure 7.1, which shows data points of the performance for every single CNN trained on the dataset. The figure also displays a polynomial regression graph with a R^2 score of 0.33 that tries to fit the data points. Given the low R^2 score, the behaviour of the curve must be interpreted with caution. The x-axis represents the yaw angle, ranging from -45 to +45 degrees, while the y-axis shows the model's accuracy in predicting the flow regime in the reactor. One might notice that the data points do not exhibit a clear pattern of correlation at first glance, appearing rather 'cloudy'. This is largely due to noise introduced during the data collection phase and explained in 5.4.

To explain the graph, focusing on the curve in Figure 7.1, between -10 and +10 yaw, a dense cloud of data points around the 0.87 mark indicates a plateau in the polynomial. This suggests that frontal camera positioning yields the best performance. Interestingly, there are optima at both extremes of the curve, around -38 and +38 yaws. This is consistent with recent research, which found that flatter camera angles better capture reflections in the reactor's inner window areas. For more in-depth information on this, refer to the thesis by Wecke et al. (2022). Between the middle state and the mid-yaw values, as well as around -20 and +20 yaws, there are noticeable minima in the curve. These anomalies can be attributed to the absence of key areas of interest at these positions. Specifically, the central step stirrer reflection point is missing due to occlusion by the reactor wall, elaborated on in (Wecke et al., 2022) and the window's side reflections are also less prominent. Despite the noisy data, the polynomial trend aligns well with the current literature. Limitations of these observations will be addressed in the discussion section 8.

The noisy nature of the data can be primarily attributed to the sampling density of the flow regime map as shown in 5.3. Moreover, increasing the camera's distance from the reactor to prevent lens damage during robot movement may introduce additional noise into the data. Firstly, reducing the camera's distance from the reactor is not an option, as this would impede the free movement of the robot. Additionally, employing a zoom mechanism would not resolve the issue, as occlusion on the reactor window would still persist. Secondly, it would be possible to reduce this noise by configuring the reactor in various states. However, doing so would drastically increase the time requirements for data collection. This is primarily due to the nonlinear relationship between the time dedicated to the experimental phase and the number of different states per regime. Broadly speaking, expanding the number of reactive states would extend the total experimental timeframe from days to weeks.

In more concrete terms, while the image sampling rate is high - meaning it is not time-consuming to collect more images in a single state - the changeover time between states is the bottleneck. Increasing the number of images in the same reactor state will not increase the information yield sufficiently, in comparison to changing the reactor state. This was also shown in the work from Kröger et al.; Wecke et al. (2022, 2022). Transitioning the reactor from one state to another takes about 20 seconds, and this needs to be done for every jaw angle and each flow regime. Given these constraints, the design choice was made to accept the noise levels in the data, as further refinement would be out of scope for this study.

Despite the inherent noise, the data serves two critical roles in this work. Firstly, it demonstrates that features extracted using explainable AI techniques can be successfully integrated into optimization algorithms. Secondly, the use of a polynomial model serves as a proof-of-concept for the research hypothesis, effectively addressing the problem of optimal camera positioning in bioreactors for this specific setup. For further details on limitations and broader implications, the reader is referred to the subsequent discussion chapter 8.

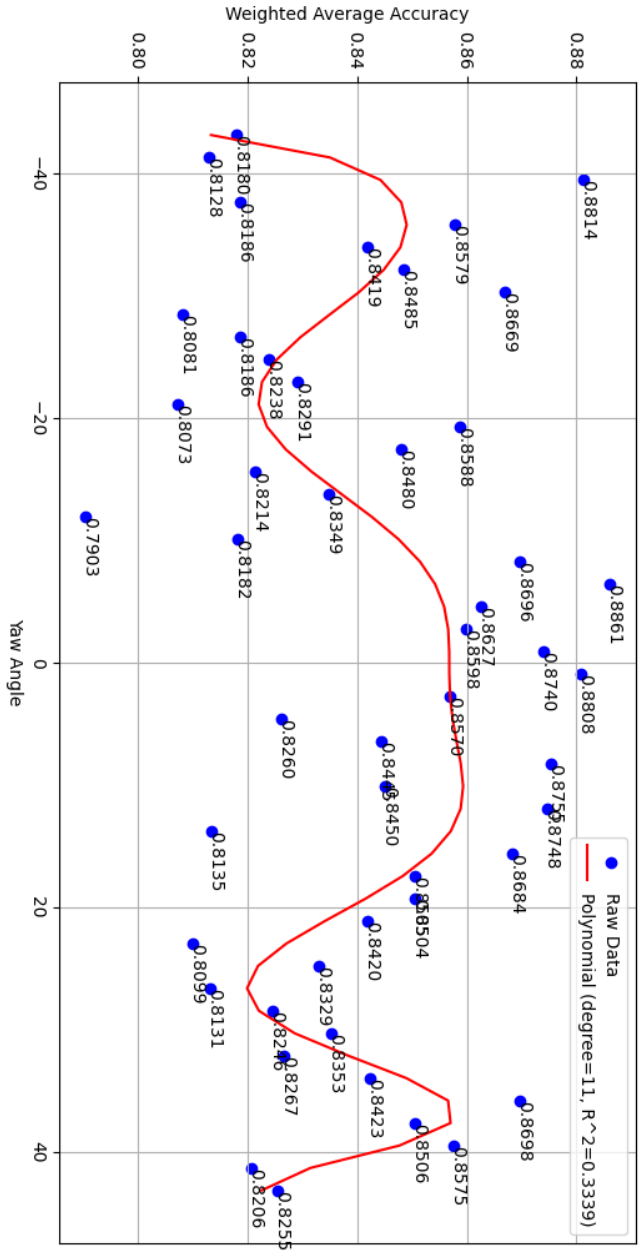


Figure 7.1: Polynomial Regression with R^2 Score of 0.3339

7.2 Recommendation Scalar

The Recommendation Scalar results shown in figure 7.2 were computed from the experimental data (chapter 7.1). For a deeper understanding of the computation of the Weighted Average Grad-Chem++ Recommendation Scalar, it is advised to refer to the Data Collection section. The figure in question (7.2) indicates that the camera's position around the bioreactor window, represented by the yaw position on the x-axis, significantly influences the Recommendation Scalar shown on the y-axis, which ranges from -200 to +150.

The curve closely resembles a sine wave. At a yaw position of 0, the Recommendation Scalar is approximately 1, indicating no need for a positional adjustment. However, moving the camera to negative yaw degrees results in a positive scalar value, which suggests a camera repositioning to the right. Conversely, at yaw positions ranging from +20 to +40 degrees, the Recommendation Scalar dips to its minimum value of -180, suggesting a move to the left. When reaching extreme angles, such as -40 or +40 degrees, the scalar value starts to lessen, signifying that the camera view is recentering. This recentering at extreme angles indicates that new features become visible due to reflections on the bioreactor window, a finding supported by (Wecke et al., 2022). The coherence between the Recommendation Scalar and the accuracy mapping, seen in Figure 7.3, serves as validation for the use of Explainable AI methods in optimizing camera positions, corroborating the research hypothesis outlined in Section 1.3.

Furthermore, the observed data suggests that optimal camera positioning in neural networks is not straightforward. Positions that may seem suboptimal or non-intuitive to the human eye, specifically at extreme yaw angles of -40 and +40 degrees, may actually offer a different, yet effective perspective for capturing features. These positions might not outperform the central position but present viable alternatives, highlighting the complexity of the camera positioning task.

The correlation between the scalar and accuracy levels at these extreme yaw angles further establishes the robustness of the employed methods. This synergistic mapping between the scalar and accuracy is visually represented in Figure 7.3.

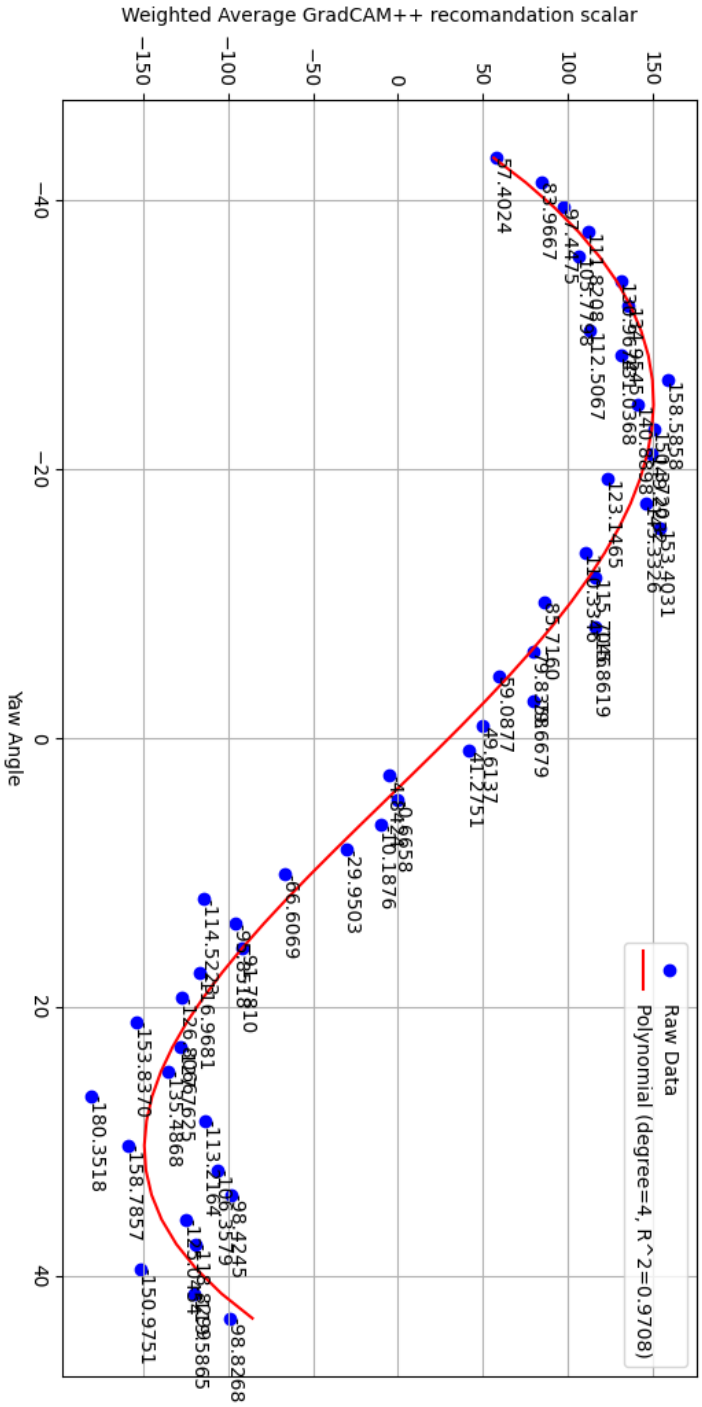


Figure 7.2: Polynomial Regression with R^2 Score of 0.3339

7.3 Benchmarking Results and Quantitative Analysis

Quantitative analysis in this work employs a multifaceted approach to dissect the experimental data thoroughly. This comprehensive evaluation aims to aid in model selection, fine-tuning, and overall understanding of the performance metrics. The findings from this quantitative analysis serve as a cornerstone for understanding the dataset's properties, the algorithms' effectiveness, and their suitability for the problem tackled in this work.

7.3.1 Accuracy Results from the Benchmarking Process

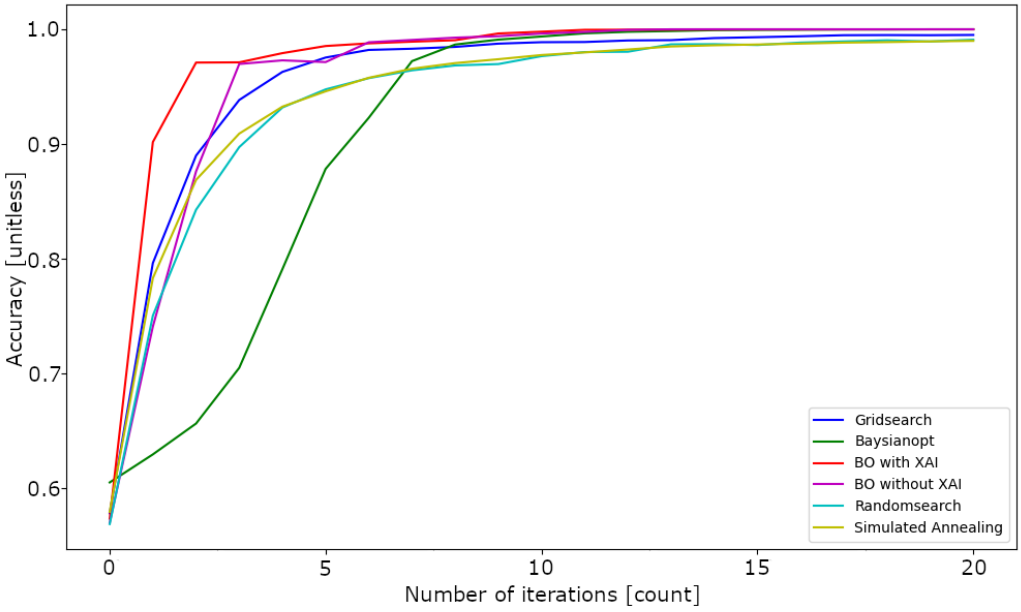


Figure 7.3: Final Results for Accuracy in Optimization Benchmarking

Refer to figure 7.3 to see the accuracy performance of various optimization algorithms over multiple iterations. The figure was obtained through 200 test runs to minimize the impact of random initialization on the results. Each graph represents the optimization performance for positioning the camera, with accuracy scaled to a normalized value of 1, indicating the reach of a global maxima rather than a 100% accuracy rate.

Among the algorithms, BO with XAI features demonstrates superior performance, underscoring its efficacy for integrating external knowledge. This aligns closely with the research hypothesis concerning the benefits of integrating features extracted via Explainable AI methods.

In contrast, the library-based version of Bayesian Optimization underperforms, even lagging behind Random Search. The limitation in hyperparameter tuning options, specifically the inability to adjust the kernel size, likely contributes to its lesser effectiveness. Simulated Annealing, albeit hyperparameter-tuned, shows only a marginal improvement over Random Search, particularly in early iterations. This is attributed to the fine-tuning of temperature parameters, which offers a slight edge. Grid Search outperforms Random Search due to the data's significant plateau region, where uniform sampling has a higher likelihood of yielding favourable results.

The efficiency of the self-implemented Bayesian Optimization, both with and without XAI, can be attributed to the probabilistic model being skewed toward the data's centre, aided by the tunable hyperparameters.

Additional Specific Observations:

- Self-implemented Bayesian Optimization shows excellent robustness, as indicated by the consistently low variance across iterations.
- The efficacy of the XAI recommendation scalar becomes more pronounced as the camera moves within the -45 to +45 degree range, further boosting the performance of Bayesian Optimization with XAI features.

The insights derived from Figure 7.3 answer multiple research questions and provide a robust comparative analysis for evaluating the utility of various optimization methods in this context.

7.3.2 Cumulative Results Interpretation

Refer to figure 7.4 for the cumulative results over all experiments. The curve may indicate total regret or cumulative loss. A flattening curve suggests that the optimization process is converging, and fewer regretful decisions are being made. Specifically, BO methods, particularly when combined with Explainable AI, demonstrate faster convergence, reaching an optimal configuration within just 10 iterations.

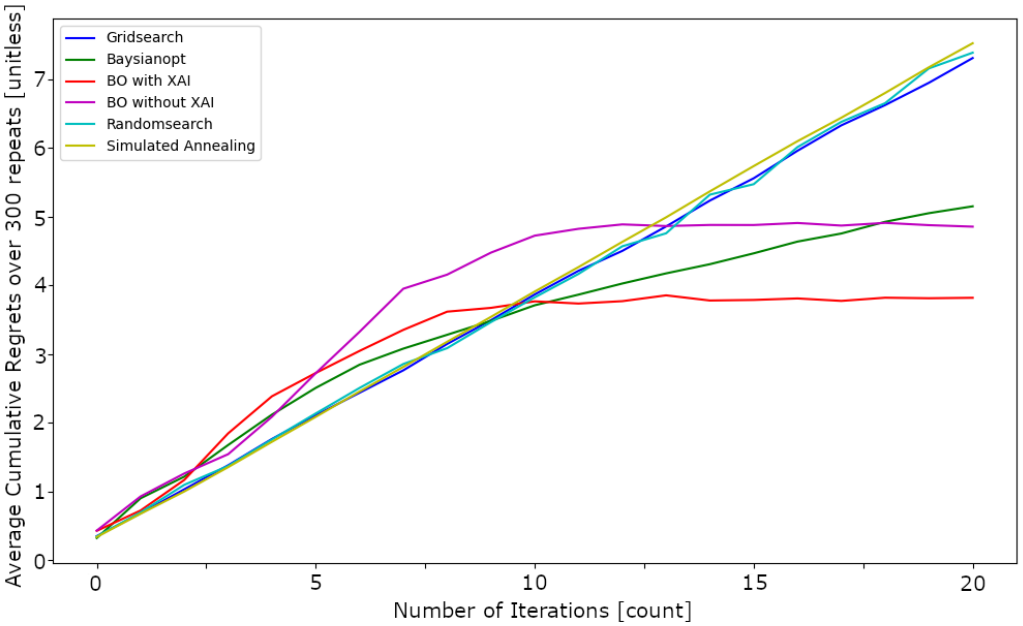


Figure 7.4: Cumulative Results for the Experiments

This rapid convergence not only confirms the superior performance of BO with XAI but also provides evidence supporting the research hypothesis that integrating XAI features can be effectively implemented for camera positioning in bioreactors. In contrast, Grid Search and Random Search show slower convergence rates, exploring the parameter space for extended periods. This is particularly noteworthy when considering complex landscapes, indicating that BO methods might be more efficient in such scenarios.

Additional observations shed light on the behaviour of individual algorithms:

- A variance plateau between the 1st and 3rd iterations for the library-based BO suggests that this version may get stuck in local optima or face challenges in efficient search space navigation.
- The consistently low variance for BO with XAI across different iterations underscores the algorithm's robustness against various starting conditions and noise in the dataset. This makes it a reliable choice for the optimization tasks at hand.

- Despite having tuned hyperparameters, Simulated Annealing performs similarly to Grid Search and Random Search in terms of cumulative regret. This may imply that the search strategy used by Simulated Annealing is not particularly well-suited for this complex optimization problem.

A steep curve in Figure 7.4 might necessitate a reconsideration of the chosen algorithm or hyperparameters to improve performance.

7.3.3 Analysis of Variability

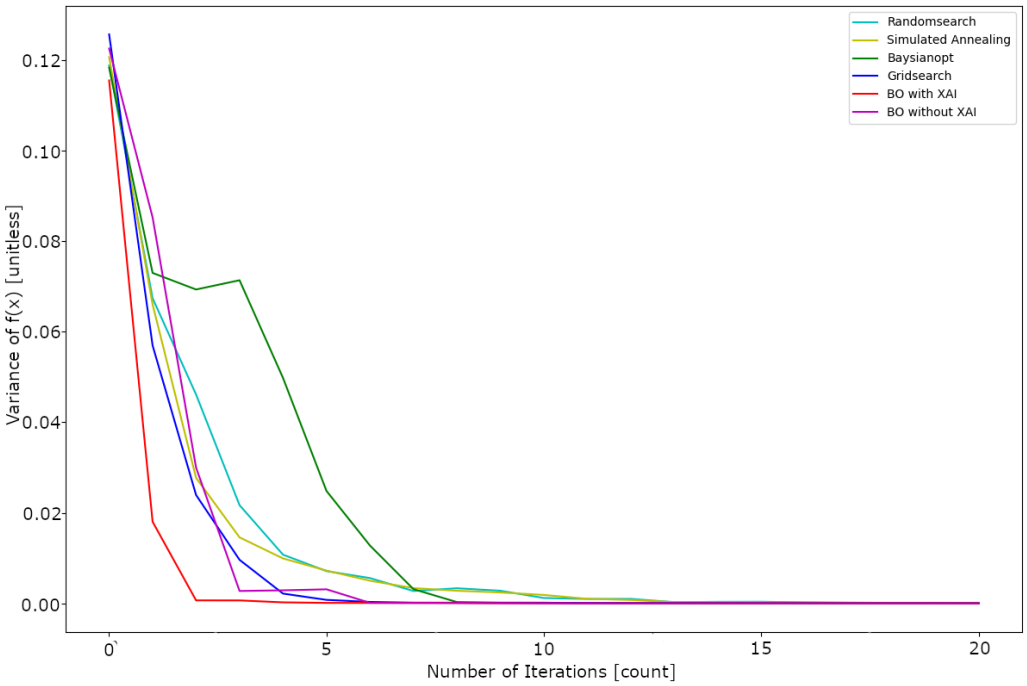


Figure 7.5: Variability in Experimental Results

Refer to figure 7.5 for insights into the variability of experimental results. High variability could indicate sensitivity to initial conditions or random seeds, resulting in inconsistent performance. In contrast, low variability implies model robustness regardless of different starting conditions.

Specifically, BO with XAI shows the least variability across multiple runs, echoing its robust performance observed in previous sections. This aligns well with the research hypothesis that integrating features from Explainable AI can offer reliable solutions in camera position optimization for bioreactors. The low variability in BO with XAI suggests that this method can handle different starting conditions and noise in the dataset effectively, making it a particularly reliable choice for real-world applications.

The library-based BO, however, exhibits high variability. This could be due to non-configurable hyperparameters, which might not be well-suited for this specific problem. This aligns with previous observations of the algorithm getting stuck in local optima or facing difficulties in search space navigation. Interestingly, the variance is calculated from the delta in accuracy across 300 runs, giving us high confidence in these observations.

Regarding production environments, consistency in algorithmic performance is crucial. Fluctuations in performance can lead to erratic behaviour, making the system unreliable for continuous operation. The low variability in BO with XAI's performance is thus a highly desirable feature, potentially making it more suitable for deployment in a production setting where robust and consistent performance is required.

7.4 Answering the Research Questions

RQ1: From these algorithms, which is most suitable for external knowledge incorporation?

The Bayesian optimization shows the most promise for incorporating external knowledge due to the ease of posterior probability manipulation, as explained in the State-of-the-art section 2.4 (Martinez-Cantin, 2019). This is underlined by Figure 7.3, where this algorithm demonstrates the best performance under the compared solution.

RQ2: How can saliency maps be used to create a heuristic for an informed search algorithm?

Saliency maps serve as an effective heuristic for informed search algorithms by highlighting areas of interest in the camera's field by computing the weighted arithmetic average of the pixel intensities. The model accuracy chart in Figure 7.3 indicates that using these features leads to better performance, thereby confirming the second research hypothesis.

RQ3: Which other novel optimization algorithms are most suited for a quantitative comparison?

Brute force algorithms like Grid search and Random search serve as baseline benchmarks, which is a common practice, as recommended in reference literature 2.2. Also, more prominent search algorithms like Simulated Annealing are used for comparison. Furthermore, a Python library Bayesian optimization serves as a reference for the self-implemented Bayesian optimization to rule out implementation mistakes to a certain degree. Furthermore, the self-implemented Bayesian optimization is compared with the external information and without, therefore comparing just the influence of the XAI feature on the optimization task (Martinez-Cantin, 2019).

RQ4: What quantitative metrics can be used for comparison?

The most effective metrics for comparison include the 'Best Found Value', 'Cumulative Regret', and 'Variance Across Multiple Runs'. The Best Found Value serves as an essential metric because it represents the optimal solution discovered by the algorithm, as shown in Figure 7.3. On the other hand, Cumulative Regret, illustrated in Figure 7.4, provides a more comprehensive view by depicting how each evaluation deviates from the global optimum over time. Additionally, variance between multiple algorithm runs, as indicated in Figure 7.5, gauges the reliability of the algorithm. A solution with a high variance may be less desirable than one with more consistent, albeit slightly less optimal, results, especially in production settings.

RQ5: How does the algorithm compare to other optimization methods in terms of these metrics?

The custom algorithm outshines competing methods in terms of accuracy, as substantiated by Figure 7.5. Additionally, the low variability observed in Figure 7.5 implies that the algorithm exhibits robustness. As for the Cumulative Regret, curve displayed in Figure 7.4, it begins with a more explorative phase that eventually flattens. This could be interpreted in two ways: either the algorithm has become trapped in a local optimum, or it has thoroughly explored the search space. Upon reviewing the Gaussian regression model, it becomes apparent that the latter scenario is the case. The Bayesian optimization algorithm has approximated the underlying function to such a degree that no further improvements are expected.

RQ6: What are the limitations of using saliency maps for camera position optimization in the context of bioreactors?

The use of saliency maps in this context has several limitations. Primarily, the saliency maps only capture information in a local neighborhood around the object of interest, limiting the broader context that could be important for optimal camera positioning. Additionally, the reliability of the saliency maps is contingent on the accuracy of the CNN model and the correctness of the labels. If the model makes incorrect assumptions or classifications, the saliency map would also reflect these inaccuracies, making it challenging for a human to differentiate true information from false positives.

7.5 Summary

In summary, our findings demonstrate that features extracted from Explainable AI, particularly saliency maps, effectively evaluate the quality of camera positions in bioreactor applications, thereby confirming our first research hypothesis. Additionally, the integration of these features into optimization algorithms has been successfully implemented as a proof-of-concept for the problem of optimal camera positioning within bioreactors, substantiating our second research hypothesis. The case study shows that saliency maps can serve as a heuristic device for informed search algorithms, offering a robust solution for camera position optimization. Quantitative metrics such as accuracy and the Recommendation Scalar, as shown in Figure 7.3 and Figure 7.2, provide a reliable basis for algorithm comparison. Finally, the quantitative analysis also confirmed the choice of Bayesian optimization with XAI with a tuned hyperparameter set to the classification task, the most suitable black-box optimization algorithm among the compared algorithms, for addressing the problem of optimal camera positioning in the given case-study.

8 Discussion

This chapter will discuss the results and limitations, providing a more nuanced evaluation of the experiment data collected for flow regime classification and the benchmarking of the Explainable A.I. featured Bayesian optimization algorithm (XAI BO). The focus will be on a critical examination of the work's limitations, while also discussing possible solutions under the current constraints of the model. Additionally, the findings will be discussed in a practically oriented manner to gauge their utility in real-world applications.

8.1 Critical Examination of Limitations

The limitations of this study mainly stem from its sole focus on a single bioreactor and the lack of flowchart sampling. Although polynomial regression techniques were used to mitigate noise and to facilitate benchmarking, the results remain constrained in terms of their generalizability. Moreover, the lack of noise handling on behalf of the optimization algorithms questions the robustness of the findings in more dynamic, real-world settings.

Experiment Data: The numerous limitations of the experimental data warrant a cautious interpretation of the results. The first limitation pertains to the low R^2 score of the polynomial regression shown in figure 7.1. Information on the design decisions regarding data regression can be found in Section 5.4. This suggests that the fitted curve may not fully capture the true function, which is essential when evaluating the optimization landscape for camera positions in bioreactors.

Another limitation is spawned by the high noise level in the data, as discussed in Section 7.1. The noise could be a result of several factors, such as the variability in flow regimes and the camera's distance from the bioreactor, affecting the overall-performance metrics like accuracy and variance shown in Figure 7.3 and Figure 7.5, respectively. While efforts were made to account for these sources of noise, their existence adds a layer of complexity to the interpretation of the results and could potentially impact the reliability of the saliency maps used for informed search algorithms, thus affecting the validation of the second research hypothesis.

Lastly, the dataset only considers a single bioreactor model. While this constraint allows for a focused study, it also limits the generalizability of the findings. Although this research offers valuable insights into the role of Explainable AI in optimizing camera positions for bioreactors, the results should not be directly extrapolated to other bioreactor models or setups without additional validation.

Benchmarking results: When examining the limitations of the benchmarking results, it is vital to emphasize a number of issues. First, while the Bayesian Optimization with XAI features shows superior performance in terms of accuracy and convergence, as highlighted by Figures 7.3 and 7.4, the algorithm's performance might be sensitive to the specific dataset and problem domain of camera positioning in bioreactors. Therefore, caution should be exercised when generalizing these results to other domains or datasets.

Second, the focus on a limited set of metrics - best found value, cumulative regret, and variance - provides an invaluable but constrained view of the algorithm's performance. While these metrics offer key insights into the algorithm's effectiveness and robustness, they may not cover all facets of real-world deployment requirements, such as computational time and resource consumption.

Third, the choice of 200 test runs for obtaining Figure 7.3 can be described as a trade-off between computational time and the robustness of the results. While the sample size is large enough to minimize the impact of random initialization, it may not be comprehensive enough to capture the complete behavior of the optimization algorithms in question, especially for algorithms like Simulated Annealing, which shows only marginal improvements over Random Search.

Lastly, the high variance observed in the library-based Bayesian Optimization, as seen in Figure 7.5, raises questions about its suitability for this specific problem. Upon further investigation, this could also be an artifact of the limitation in hyperparameter tuning options and may not necessarily reflect inherent flaws in the Bayesian Optimization algorithm itself.

In summary, while the empirical results strongly support the research hypotheses and answer the research questions, these limitations suggest that further investigations are warranted, especially in terms of their broader applicability and the current understanding of these optimization methods.

8.2 Discussion of Solutions to Limitations

One way to address these limitations is by extending the dataset to include multiple bioreactors and by incorporating flowchart sampling. Bayesian optimization, known for its capacity to handle noise, could be slightly adjusted to accommodate this variability. These amendments would bolster the robustness and generalizability of the algorithm, especially in a production environment.

Benchmarking Our custom Bayesian Optimization with Explainable AI (BO with XAI) algorithm demonstrated superior performance when compared to other tested methods, confirming Research Hypothesis 2. Specifically, the algorithm excelled in terms of accuracy, as proven with by Figure 7.3. Saliency maps, a feature of Explainable AI, served as an effective heuristic guide that improved the algorithm's speed and reliability. These results corroborate the suitability of our BO with XAI approach for the complex task of optimizing camera positions in bioreactors.

The algorithm also showed rapid convergence within just 10 iterations, as depicted in Figure 7.4. This rapid convergence is especially critical in practical settings where computational resources are often limited. Meanwhile, the algorithm's low variance across iterations, highlighted in Figure 7.5, adds another layer of reliability, indicating that the algorithm is not just effective but also consistently dependable under varying conditions. Furthermore, the library-based version of Bayesian Optimization had limitations, particularly concerning hyperparameter tuning options, which rendered it less suitable for the optimization task at hand. This points to the unique advantages of the custom implementation over existing library-based solutions.

Metrics such as the R^2 score, convergence rate, and result variability were employed to rigorously compare our method's performance against other algorithms like Simulated Annealing and Random Search. These alternatives, while noteworthy, exhibited slower convergence rates and higher variance, which makes them less reliable for the optimization problem under investigation.

This thesis also adapts our model to mitigate the noise introduced by the specific conditions of our bioreactor by employing polynomial regression. This indicates that while the model is optimized for an academic setting, further work is necessary for its adaptation to broader industrial applications. By considering these empirical observations and metrics, this thesis offers a nuanced discussion of the limitations and strengths of the various optimization algorithms examined in this research. The empirical evidence strongly supports the effectiveness and reliability of our custom Bayesian Optimization with XAI for real-world applications.

Data collection: Addressing the limitations related to data collection is crucial for enhancing the robustness and generalizability of this study's findings. One straightforward approach to overcome the single bioreactor constraint would be to extend the dataset to include multiple bioreactors of different sizes and configurations. This would not only make the results more generalizable, but also introduce an additional layer of complexity that mimics real-world conditions better.

Moreover, the issue of high noise levels could be tackled by means of a more rigorous and comprehensive data collection methodology. Employing a higher resolution camera or using multiple cameras at different positions could provide a clearer and more accurate view of the bioreactor's internal conditions, thus reducing noise. Techniques such as signal processing or advanced filtering methods could further mitigate the impact of noise, enhancing the reliability of the model.

The lack of flowchart sampling also represents a gap in the current methodology. Introducing a flowchart sampling approach would allow for a more dynamic understanding of how different flow regimes affect the optimization problem. This would be especially useful for creating a more adaptable model that can handle variations in flow conditions, thus making the optimization process more robust.

Finally, collecting more data points would also contribute to a more reliable evaluation of the optimization algorithms, specifically when benchmarking their performance. A larger dataset would provide a better understanding of the algorithms' behavior over a more extended range of conditions, thereby offering more rigorous insights into their strengths and weaknesses. This could be particularly beneficial for algorithms that showed only marginal improvements over others, as it would provide a more nuanced understanding of where they might be more applicable.

By systematically addressing these data collection-related limitations, the study can significantly improve the validity of its findings, contributing to both academic and practical knowledge in the optimization of camera positions for bioreactors.

8.3 Practice-Oriented Discussion of Findings

In this section, this thesis aims to contextualize the research findings in practical, real-world scenarios, specifically focusing on their potential utility in optimizing camera positions in bioreactors for industrial and research purposes.

Firstly, the Bayesian Optimization algorithm enhanced with Explainable AI (XAI) features, notably saliency maps, proved highly effective in our specific use-case. The algorithm's rapid convergence, as shown in Figure 7.4, makes it a compelling choice for scenarios where timely decisions are crucial. This can be particularly beneficial in industrial settings where quick optimization can lead to significant cost savings and enhanced process efficiency.

Secondly, the robustness of the Bayesian Optimization with XAI, evidenced by its low variance in Figure 7.5, adds an extra layer of reliability. In a real-world setting where conditions can vary, the need for a robust algorithm cannot be overstated. Companies and research institutions may find this feature particularly appealing, as it lessens the chances of optimization failure due to variations in starting conditions or environmental noise. While the study provides a foundational understanding of the capabilities of Bayesian Optimization with XAI for this application, further research and tests would be required to validate its generalizability across different types of bioreactors or even other types of optimization problems.

In addition, although the algorithm performed exceptionally well in terms of the metrics used in this study, its computational requirements should be assessed before deployment in real-world scenarios. Depending on the complexity and size of the problem, specialized hardware might be needed to run the algorithm efficiently.

To sum up, the research findings present a compelling argument for the potential utility of Bayesian Optimization with Explainable AI in practical, real-world applications. The demonstrated robustness, accuracy, and speed of the algorithm make it a promising candidate for optimization tasks in industrial bioreactors. However, the limitations noted earlier should serve as a roadmap for future research and should be further developed to adapt and validate this approach for a broader range of applications.

9 Summary and Outlook

This chapter aims to wrap up the key findings and contributions of this thesis, while also providing an outlook on future research directions. The focus will be on summarizing the answers to the examined research questions, highlighting the implications of these findings, and suggesting ways to extend this research further.

The core of this thesis was centered around optimizing camera positions in bioreactors using Bayesian Optimization and Explainable AI (XAI), particularly saliency maps. The custom Bayesian Optimization with XAI algorithm demonstrated superior performance, as evidenced by multiple metrics such as accuracy, cumulative regret, and variance across multiple runs. The algorithm was not only effective but also consistently reliable, showing a low degree of variability and rapid convergence.

The empirical results confirmed both research hypotheses: first, that features extracted through XAI can effectively evaluate the CNN performance in different camera positions in bioreactor applications; second, that integrating these features into optimization algorithms leads to better performance in terms of finding an optimal camera position for CNN classification compared to non-informed optimization algorithms for this specific optimization problem in flow regime classification. However, the limitations observed in the dataset and benchmarking metrics suggest areas for further investigation.

Moving forward, there are several promising avenues for future research. Extending the dataset to include multiple bioreactor models could provide broader applicability and robustness to the algorithm. Further improvements might include a more comprehensive set of evaluation metrics and a detailed investigation into the algorithm's computational requirements.

The adaptability of Bayesian Optimization to handle noise could also be explored more in future works, especially when adding more variability to the dataset, such as different flow regimes and additional bioreactor models. Moreover, the role of XAI features in the optimization algorithm could be fine-tuned to achieve even better results in terms of speed and reliability. Lastly, the methods and findings could potentially be extended to other optimization tasks beyond camera positioning in bioreactors, opening the door for more versatile applications.

Overall, while the present study offers valuable insights and a strong foundation, it is a stepping stone for more advanced research in this interdisciplinary field of optimization and machine learning.

References

Anderson, R. L. (1953). Recent Advances in Finding Best Operating Conditions. *Journal of the American Statistical Association*, *48*(264). https://doi.org/10.1080/01621459.1953.10501200 (cit. on p. 22)

Audet, C., & Kokkolaras, M. (2016). Blackbox and derivative-free optimization: theory, algorithms and applications. https://doi.org/10.1007/s11081-016-9307-4. (Cit. on p. 52)

Bau, D., Zhou, B., Khosla, A., Oliva, A., & Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-January*. https://doi.org/10.1109/CVPR.2017.354 (cit. on p. 24)

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13* (cit. on pp. 14, 15, 23).

Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., & Cox, D. D. (2013). Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms (cit. on p. 15).

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A.-L., Deng, D., & Lindauer, M. (2021). Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges. http://arxiv.org/abs/2107.05847 (cit. on p. 22)

Biswas, A. (2016). Interpretation of gravity and magnetic anomaly over thin sheet-type structure using very fast simulated annealing global optimization technique. *Modeling Earth Systems and Environment*, *2*(1). https://doi.org/https://doi.org/10.1007/s40808-016-0082-1 (cit. on p. 21)

Brochu, E., Cora, V. M., & de Freitas, N. (2010). Bayesian Optimization for a Better Dessert. *Bayesian Optimization for a Better Dessert*. http://dx.doi.org/10.1007/978-3-319-31204-0_1 (cit. on pp. 14, 15, 17, 46, 76)

Chattopadhay, A., Sarkar, A., Howlader, P., & Balasubramanian, V. N. (2018). Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018*, *2018-January*. https://doi.org/10.1109/WACV.2018.00097 (cit. on pp. 24, 26, 27)

Das, A., Agrawal, H., Zitnick, L., Parikh, D., & Batra, D. (2017). Human Attention in Visual Question Answering: Do Humans and Deep Networks Look at the Same Regions? *Computer Vision and Image Understanding*, *163*. https://doi.org/10.1016/j.cviu.2017.10.001 (cit. on p. 30)

Devan, P. A. M., Hussin, F. A., Ibrahim, R. B., Bingi, K., Nagarajapandian, M., & Assaad, M. (2022). An arithmetic-trigonometric optimization algorithm with application for control of real-time pressure process plant. *Sensors (Basel, Switzerland)*, *22*(2). https://doi.org/https://doi.org/10.3390/s22020617 (cit. on p. 21)

Dhillon, A., & Verma, G. K. (2020). Convolutional neural network: a review of models, methodologies and applications to object detection. https://doi.org/10.1007/s13748-019-00203-0. (Cit. on p. 25)

Eggensperger, K., Feurer, M., Falkner, S., & Hutter, F. (2018). A Conceptual Comparison of Bayesian Optimization and Hyperband (cit. on pp. 34, 35, 52, 61).

Falkner, S., Klein, A., & Hutter, F. (2018). BOHB: Robust and Efficient Hyperparameter Optimization at Scale (cit. on pp. 14, 15).

Gaspar, T., & Oliveira, P. (2011). Single pan and tilt camera indoor positioning and tracking system. *European Journal of Control*, *17*(4). https://doi.org/10.3166/ejc.17.414-428 (cit. on p. 21)

Islam, A., Hossan, M. T., & Jang, Y. M. (2018). Convolutional neural network scheme–based optical camera communication system for intelli-

gent internet of vehicles. *International Journal of Distributed Sensor Networks*, *14*(4). https://doi.org/10.1177/1550147718770153 (cit. on p. 4)

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, *13*(4). https://doi.org/10.1023/A:1008306431147 (cit. on pp. 13, 16, 18)

Khaydarov, V., Heinze, S., Graube, M., Knüpfer, A., Knespel, M., Merkelbach, S., & Urbas, L. (2020). From stirring to mixing: Artificial intelligence in the process industry. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, *2020-September*. https://doi.org/10.1109/ETFA46521.2020.9212018 (cit. on pp. 3, 4, 23, 28)

Klein, A., Falkner, S., Mansur, N., & Hutter, F. (2017). RoBO: A Flexible and Robust Bayesian Optimization Framework in Python. *Conference on Neural Information Processing Systems (NIPS): Bayesian Optimization Workshop*, (Nips) (cit. on p. 52).

Kröger, C. (2019). *KI-basierte Erkennung von Strömungsregimen im Bioreaktor* (tech. rep.). (Cit. on pp. 28–31, 53, 54, 58, 59).

Kröger, C., Khaydarov, V., & Urbas, L. (2022). Data-driven, Image-based Flow Regime Classification for Stirred Aerated Tanks. In *Computer aided chemical engineering*. https://doi.org/10.1016/B978-0-323-95879-0.50228-9. (Cit. on pp. 3, 23, 53, 57, 58, 81)

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, *1*(4). https://doi.org/10.1162/neco.1989.1.4.541 (cit. on pp. 4, 24, 59)

Lee Kim, C., & Kim. (2020). The Effect of Image Resolution on Deep Learning in Radiography (cit. on p. 3).

León, J., Escobar, J. J., Ortiz, A., Ortega, J., González, J., Martín-Smith, P., Gan, J. Q., & Damas, M. (2020). Deep learning for EEG-based motor imagery classification: Accuracy-cost trade-off. *PLoS ONE*, *15*(6). https://doi.org/10.1371/journal.pone.0234178 (cit. on p. 22)

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, *18* (cit. on pp. 14, 15).

Markus Esser. (2022). Multimodale datengetriebene Klassifikation vonStrömungsregimes im Bioreaktor. (Cit. on pp. 3, 23, 28, 53, 55, 59).

Martinez-Cantin, R. (2019). Funneled Bayesian optimization for design, tuning and control of autonomous systems. *IEEE Transactions on Cybernetics*, *49*(4), 1489–1500. https://doi.org/10.1109/TCYB.2018.2805695 (cit. on pp. 4, 13, 75, 89, 90)

Ozaki, Y., Suzuki, Y., Hawai, T., Saito, K., Onishi, M., & Ono, K. (2020). Automated crystal structure analysis based on blackbox optimisation. *npj Computational Materials*, *6*(1). https://doi.org/10.1038/s41524-020-0330-9 (cit. on p. 4)

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press. http://dx.doi.org/10.7551/mitpress/3206.001.0001. (Cit. on pp. 17, 78)

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164. https://doi.org/10.1007/s10664-008-9102-8 (cit. on pp. 9, 10, 12, 13)

Samora, I., Franca, M. J., Schleiss, A. J., & Ramos, H. M. (2016). Simulated annealing in optimization of energy production in a water supply network. *Water Resources Management*, *30*(4), 1533–1547. https://doi.org/10.1007/s11269-016-1238-5 (cit. on pp. 14, 15, 52, 76, 77)

Sartorius stedim biotech AG. (n.d.). *BIOSTAT ® D-DCU Your "Fast Lane" to Production* (tech. rep.). https://de.scribd.com/document/424283972/Broch-Biostat-D-DCU-SBI1512-e. (Cit. on p. 54)

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-October*. https://doi.org/10.1109/ICCV.2017.74 (cit. on pp. 23–25, 46)

Seshadri, D., & Jois. (2019). A systematic comparison of deep learning architectures in an autonomous vehicle (cit. on p. 3).

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & De Freitas, N. (2016). Taking the human out of the loop: A review of Bayesian optimization. https://doi.org/10.1109/JPROC.2015.2494218. (Cit. on pp. 13–17, 33, 35, 46, 52, 65, 75, 76)

Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. *2nd International Conference on Learning Representations, ICLR 2014 - Workshop Track Proceedings* (cit. on p. 3).

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in Neural Information Processing Systems*, 2951–2959. http://dx.doi.org/10.5555/2999325.2999464 (cit. on pp. 4, 19, 46)

Torre, E. R. D. R., & Tanimoto, M. (1996). Optimal camera placement for total coverage (cit. on p. 3).

Troniewski, L., & Ulbrich, R. (1984). *THE ANALYSIS OF FLOW REGIME MAPS OF TWO-PHASE GAS-LIQUID FLOW IN PIPES* (tech. rep.). (Cit. on p. 28).

Wang, H., Hu, Z., Sun, Y., Su, Q., & Xia, X. (2018). Modified backtracking search optimization algorithm inspired by simulated annealing for constrained engineering optimization problems. *Computational Intelligence and Neuroscience*, *2018*. https://doi.org/https://doi.org/10.1155/2018/9167414 (cit. on p. 21)

Wecke, L., Khaydarov, V., & Urbas, L. (2022). *Study on Effect of Camera Position in Tasks of Flow Regime Classification in Bioreactors* (tech. rep.). Tu Dresden. (Cit. on pp. 3–5, 23, 28, 30, 53, 55, 59, 80, 81, 83).

Yang, K., Duan, Q., Wang, Y., Zhang, T., Yang, Y., & Huang, R. (2020). Transiently chaotic simulated annealing based on intrinsic nonlinearity of memristors for efficient solution of optimization problems. *Science Advances*, *6*(33). https://doi.org/https://doi.org/10.1126/sciadv.aba9901 (cit. on p. 21)

Yu, V. F., Susanto, H., Jodiawan, P., Ho, T. W., Lin, S. W., & Huang, Y. T. (2022). A Simulated Annealing Algorithm for the Vehicle Routing Problem With Parcel Lockers. *IEEE Access*, *10*, 20764–20782. https://doi.org/10.1109/ACCESS.2022.3152062 (cit. on p. 21)

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8689 LNCS*(PART 1). https://doi.org/10.1007/978-3-319-10590-1{_}53 (cit. on pp. 25, 59)

Zhang, S., & Wang, X. (2013). Human detection and object tracking based on Histograms of Oriented Gradients. *Proceedings - International Conference on Natural Computation*, 1349–1353. https://doi.org/10.1109/ICNC.2013.6818189 (cit. on p. 59)

Zhou, L., Ma, C., Shi, X., Zhang, D., Li, W., & Wu, L. (2021). Salience-CAM: Visual Explanations from Convolutional Neural Networks via Salience Score. *Proceedings of the International Joint Conference on Neural Networks*, *2021-July*. https://doi.org/10.1109/IJCNN52387.2021.9534419 (cit. on pp. 3, 25)

Zhou, Z., Li, X., & Zare, R. N. (2017). Optimizing Chemical Reactions with Deep Reinforcement Learning. *ACS Central Science*, *3*(12). https://doi.org/10.1021/acscentsci.7b00492 (cit. on p. 38)

Appendix

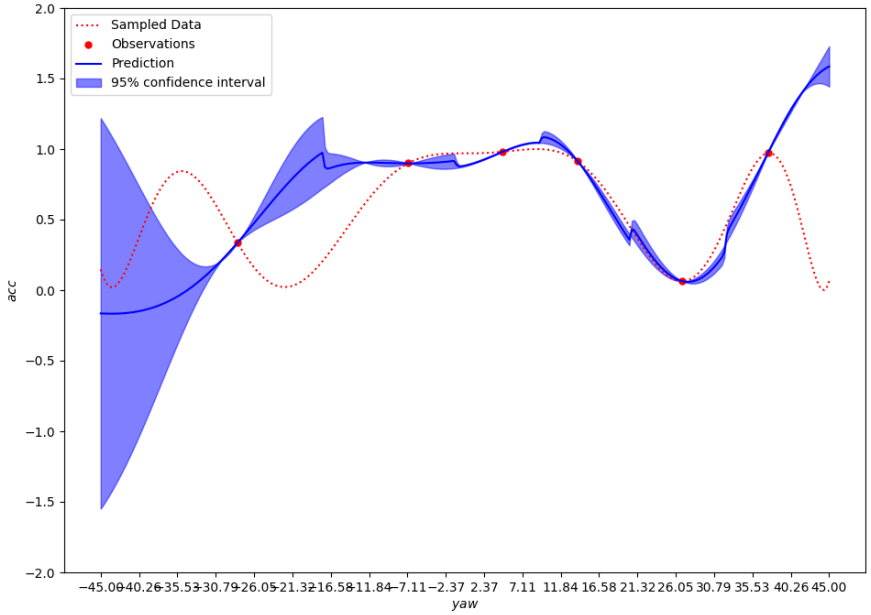


Figure 1: GP model after 6th iteration of the optimisation process

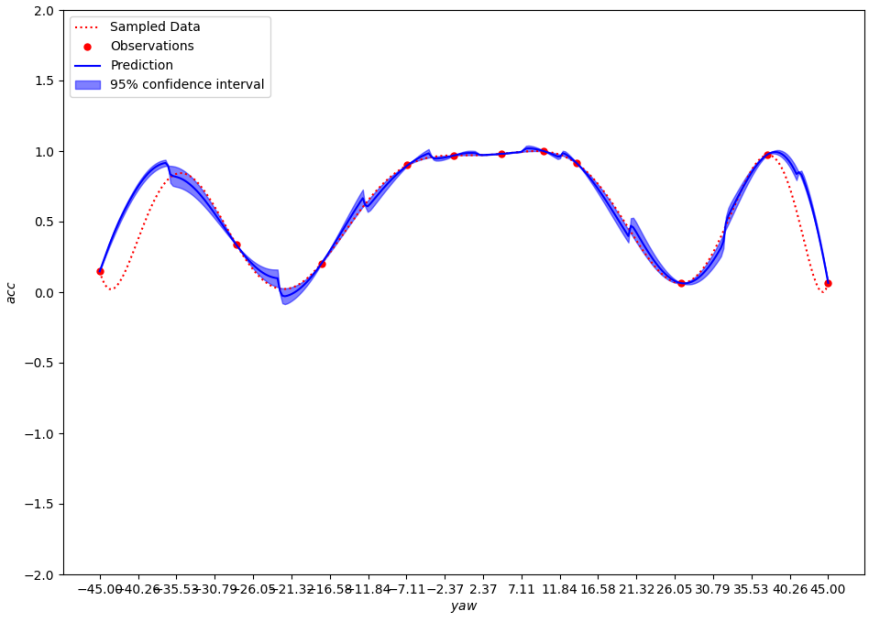


Figure 2: GP model after 11th iteration of the optimisation process

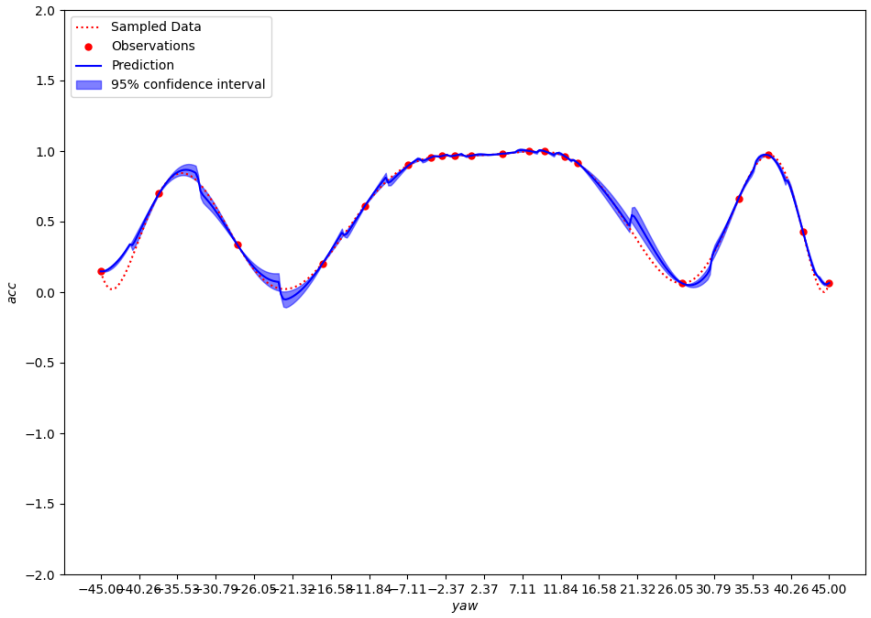


Figure 3: GP model after 20th iteration of the optimisation process

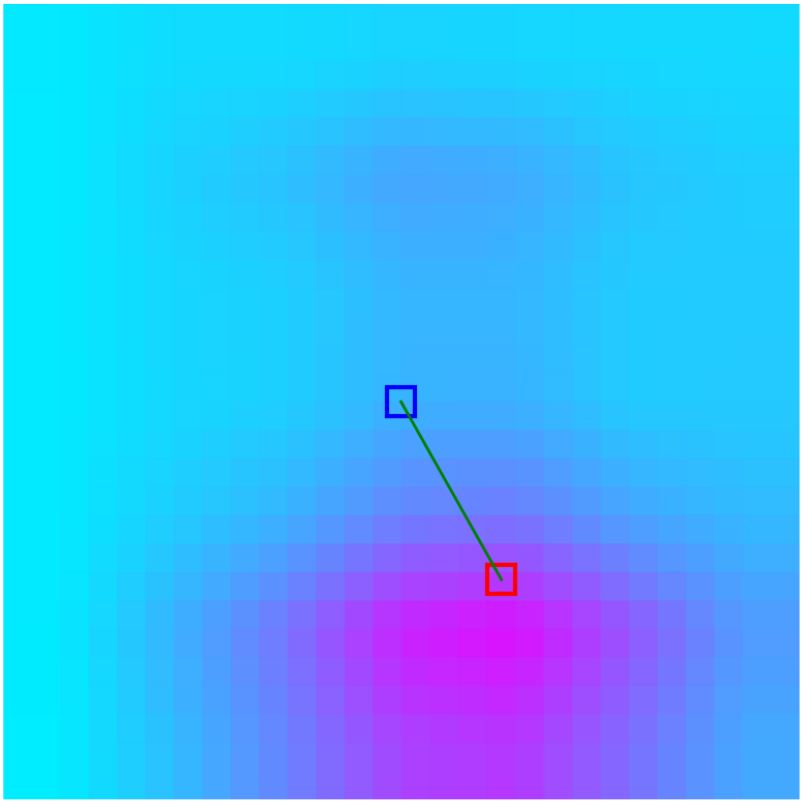


Figure 4: Another Averaged GradCAM++ image with recommendation scalar

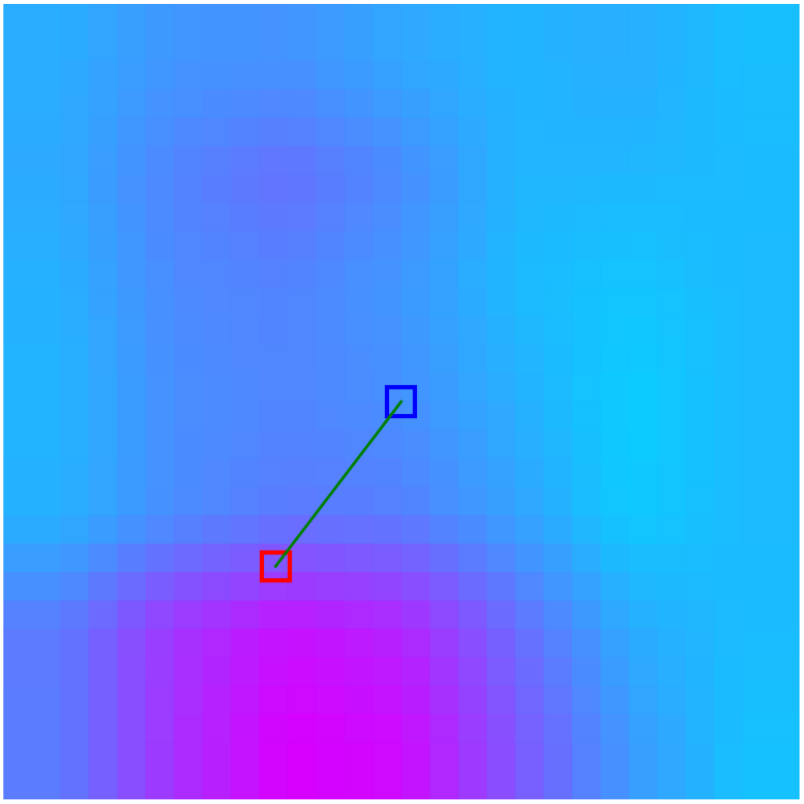


Figure 5: Another Averaged GradCAM++ image with recommendation scalar

Selbstständigkeitserklärung

Hiermit versichere ich, Leonard-Riccardo Hans Wecke, geboren am 29.06.1997 in Cottbus, dass ich die vorliegende Diplomarbeit zum Thema

Anwendung von Saliency-Maps zur Optimierung der Kamerapositionierung in Deep Learning Anwendungen

ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts habe ich Unterstützungsleistungen von folgenden Personen erhalten:

Dr. rer. nat. Valentin Khaydarov, Dipl.-Ing. Jonathan Mädler

Weitere Personen waren an der geistigen Herstellung der vorliegenden Diplomarbeit nicht beteiligt. Mir ist bekannt, dass die Nichteinhaltung dieser Erklärung zum nachträglichen Entzug des Diplomabschlusses (Masterabschlusses) führen kann.

Dresden, den 09.10.2023 .....

Unterschrift