Doctoral theses at NTNU, 2021:331

Bjørn Magnus Mathisen

Using similarity learning to enable decision support in aquaculture

Doctoral thesis

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of
Philosophiae Doctor
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Bjørn Magnus Mathisen

Using similarity learning to enable decision support in aquaculture

Thesis for the Degree of Philosophiae Doctor

Trondheim, Oktober 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Department of Computer Science

ISBN 978-82-326-5184-9 (printed ver.)
ISBN 978-82-326-5625-7 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2021:331

Printed by NTNU Grafisk senter

To my daughters Ane and Nora. And to my partner Astrid.

Abstract

Aquaculture (AQ) is an industry that cultivates food in water. This includes many types of seafood such as salmon, trout, and whitefish, as well as shellfish and algae. Farms for seafood production are typically described as sites by the industry. In Norway, the site locations are normally regulated and allocated by the government. Artificial intelligence (AI) and machine learning (ML) has not yet been widely adopted in the industry. AI/ML would potentially be able to support the industry in automation, operation and decision support.

The aquaculture industry is expanding across the globe. This is a result of technological development and the need for more food production to feed a growing population. In 2012, the Norwegian seafood industry was expected to grow five-fold from 2007 until 2050 [1]. According to industry representatives and the government[1,2], this is still the case today. As a result of this expansion, the industry needs to increase the number of production sites. While expanding, the industry needs to keep the environmental impact of such production sites to a minimum. As production sites pollute their immediate surroundings, they should ideally not be in constant production over a long period of time. Additionally, the production sites cannot be too concentrated geographically to minimize the environmental impact and risk of spreading diseases such as sea lice. As a result, the number of available sites is decreasing, and the industry now looks to increasingly more exposed locations for their aquaculture operations. Exposed aquaculture sites are subject to rough conditions and are often inaccessible. Typical aquaculture sites are well sheltered. To ensure the same level of safety, aquaculture sites that are more exposed would require more resources and a more robust physical infrastructure. Also, the level of exposure often leads to more downtime, where personnel is waiting for the weather to clear up to perform their tasks.

The aquaculture industry is a conservative industry and has not progressed far in terms of digitalization and instrumentation compared to many other comparable industries such as oil and gas. The push towards more exposed aquaculture operations is

[1] https://www.nrk.no/trondelag/sjomat-norge-onsker-a-femdoble-sjomatnaeringen-__-vil-koste-500-milliarder-1.14501218

[2] https://www.pwc.no/no/publikasjoner/pwc-seafood-barometer-2017.pdf

now changing this, where increasing the level of automation and remote work would significantly contribute to decreasing the risks to personnel. Such automated operations require the application of digital technologies both for operations and decision-support. This development is supported by the availability of more operational data from the aquaculture industry in recent years. As a result, the connectivity and data availability allows for data-driven services and utilization of ML.

Data-driven models and ML support in the aquaculture industry include both operational use cases and decision support systems (DSSs). Operational use cases for aquaculture include 1) computer vision for situation recognition needed for automatic fish feeding, and 2) robotics that can perform necessary operations such as cage cleaning or extracting fish. As such, operational use cases are use cases where ML models are used in real-time or close to real-time. In contrast, DSSs are typically used as a planning tool. DSSs use data-driven models in the context of supporting decision-making or operational planning. Such systems are designed to help operators by predicting operational properties, such as production, structure movements, or waves.

Most decision-makers, especially from conservative industries, prefer an understandable and explainable DSS. When the DSS explains the recommendation it produces, it increases the trust in that recommendation, and as a result, the usefulness of the DSS. Many machine learning methods and their resulting models are not easy to explain to most users. One way of alleviating this is to use case-based reasoning (CBR)[2]. CBR captures previous experiences or situations in the form of cases that consist of a problem description and the corresponding solution. As part of a DSS, CBR would store previous situations where the DSS was used and the resulting action or solution. In this way, the DSS user can be presented with the previous situation most similar to the current situation and the resulting action for that situation. The input of a DSS can be the current state. In the case of using CBR for planning in a DSS, the CBR input can be a prediction (e.g., a predicted situation for which the CBR can retrieve a solution). Presenting an actual recorded situational experience and resulting action along with the prediction provides an indirect explanation and strengthens the user's confidence in the DSS.

The work described in this thesis investigates the use of machine learning to increase the level of automation in aquaculture operations, focusing on decision support. A general framework for designing a DSS is introduced, from data gathering to the user interface. This framework outlines the steps from sensors readings, preprocessing of the data, combining the data with knowledge and experience from the

users of the DSS, using the data to feed machine learning, knowledge models, and numerical models to then predict a future state which can be used to make informed decisions. In addition, a CBR-based DSS can store previously recorded situations where the DSS was applied (cases). The DSS can then use this repository to retrieve and present the user with the previously recorded cases that are most similar to the predicted state. To do this, the DSS must retrieve the case most relevant (similar) to the one predicted by the DSS or input by the DSS user (query case). Retrieving the most similar case requires the DSS to compute the similarity between the query case and the cases in the repository.

Measuring similarity between cases is a focus of research within machine learning and case-based reasoning. Manual modeling this similarity can be challenging. Building on previous state-of-the-art machine learning methods, we propose a new method for learning such similarity measures from data (similarity learning), which can be used for retrieving cases: Extended Siamese Neural Networks (ESNN). ESNN is a similarity learning (SL) method that outperforms the accuracy and training speed of state-of-the-art methods across domains. Extending the testing of ESNN, we developed a dataset for describing situations in aquaculture operations. We demonstrated that ESNN also outperformed state-of-the-art methods for retrieving the most similar operational situations.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of *Philosophiae Doctor* in Computer Science at the Department of Computer Science, Norwegian University of Science and Technology (NTNU). The research presented here was conducted under the supervision of Professor Agnar Aamodt (until 2020), Associate Professor Kerstin Bach (from 2020) and my co-supervisors Professor Helge Langseth and Gunnar Senneset, and supported by the Norwegian Research Council through the EXPOSED Aquaculture Research Centre (grant number 237790). The PhD project started 15th of September 2015. During the PhD project, several major events happened. I became a father for the second time, and the world was hit by Covid-19. While working on this PhD project, I also had a part-time position (25%) at the SINTEF research institute.

The thesis is a collection of four papers presented in chronological order of writing. The included papers have been published or accepted for publication for scientific conferences or journals. The papers have been reformatted to have consistent formatting within the thesis and deviates visually from the published versions.

Acknowledgements

First and foremost, thanks to Professor Agnar Aamodt, Associate Professor Kerstin Bach and Professor Helge Langseth for guiding this work from its fumbling beginning to the end. Agnar was my supervisor until 2020. After his retirement, Kerstin stepped in as my supervisor in 2020. Helge Langseth has been a co-supervisor who gave more support than required. They have all guided me with wisdom, kindness, and patience. The work has been challenging and seemed insurmountable at times. It could not have been done without the support from them. I would like to thank my MSc advisor Keith Downing for guiding me into the field of AI & ML and creating my deep interest in the field. In addition, I would like to give my thanks for the support I received from EXPOSED Aquaculture Research Centre through my mentor Gunnar Senneset and center director Hans Vanhauwaert Bjelland. My partner Astrid gave me motivation, support and helped with proofreading. I also received much motivation from my children Ane and Nora, and my

friends and family. I also want to thank my parents that raised me to be curious.

I would also like to thank my colleagues who have made my working days in the department a true joy; Håkon, Heri, Birgit, Berit, Ellen, Eliezer, Tarik, Joakim and many more.

Bjørn Magnus Mathisen
Trondheim, October 2021

List of Papers

Paper I

Bjørn Magnus Mathisen, Agnar Aamodt, Kerstin Bach and Helge Langseth "Data driven case base construction for prediction of success of marine operations". In: *A.A. Sanchez-Ruiz and A. Kofod-Petersen (Ed.): Proceedings of ICCBR 2017 Workshops (CAW, CBRDL, PO-CBR), Doctoral Consortium, and Competitions co-located with the 25th International Conference on Case-Based Reasoning (ICCBR 2017)*, Trondheim, Norway, 2017, June 26-28, pp 102-111, CEUR-WS.org, 2017.

Paper II

Bjørn Magnus Mathisen, Agnar Aamodt, Kerstin Bach and Helge Langseth "Learning similarity measures from data". In: *Progress in Artificial Intelligence* (2019), 9(2), 129-143.

Paper III

Bjørn Magnus Mathisen, Kerstin Bach, Espen Meidell, Håkon Måløy and Edvard Schreiner Sjøblom. "FishNet: A unified embedding for salmon recognition". In: *Giuseppe De Giacomo and Alejandro Catalá and Bistra Dilkina and Michela Milano and Senén Barro and Alberto Bugarín and Jérôme Lang (Ed.): ECAI 2020 - 24th European Conference on Artificial Intelligence - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, Santiago de Compostela, Spain, 2020, 29 August-8 September, pp 3001-3008, IOS Press, 2020.

Paper IV

Bjørn Magnus Mathisen, Agnar Aamodt and Kerstin Bach. "Using Extended Siamese Networks in a CBR system to provide decision support in aquaculture operations" *Applied intelligence* (2021), online publication: https://doi.org/10.1007/s10489-021-02251-3.

Contents

Preface vii

List of Papers ix

Contents xi

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Challenges . . . 3
- 1.3 Research goals and research questions . . . 5
- 1.4 Research context . . . 8
- 1.5 Thesis structure . . . 9

2 Background 11
- 2.1 Decision support systems . . . 11
- 2.2 Case-based reasoning . . . 12
- 2.3 Neural networks . . . 16
- 2.4 Similarity learning and metric learning . . . 16

3 Decision support systems in aquaculture 23
- 3.1 Introduction . . . 23
- 3.2 Method . . . 24
- 3.3 Analysis . . . 26
- 3.4 Discussion . . . 32

4 Research results 35
- 4.1 Research contributions . . . 35
- 4.2 List of publications . . . 36
- 4.3 Contributions towards research questions . . . 39
- 4.4 Summary of auxiliary papers . . . 49
- 4.5 Source code . . . 51

5 Conclusion 53
- 5.1 Conclusion . . . 53
- 5.2 Future research directions . . . 55

References . . . 56

Papers 66

Data driven case base construction for prediction of success of marine operations **67**
I.1 Introduction 69
I.2 Related work 71
I.3 Method 72
I.4 Results 76
I.5 Conclusions and future work 77
I.6 Acknowledements 78
References 78

Learning similarity measures from data **81**
II.1 Introduction 83
II.2 A framework for similarity measures 86
II.3 Related work 89
II.4 Method 93
II.5 Experimental evaluation 102
II.6 Conclusions and future work 107
II.7 Acknowledgements 110
References 110

FishNet: A Unified Embedding for Salmon Recognition **115**
III.1 Introduction 117
III.2 Related Work 119
III.3 The FishNet Approach 121
III.4 Dataset and Evaluation 125
III.5 Discussion 132
III.6 Conclusion 134
III.7 Acknowledgements 135
References 136

Using Extended Siamese Networks to Provide Decision Support in Aquaculture Operations **139**
IV.1 Introduction 141
IV.2 Related Work 143
IV.3 Operational situation dataset and Case definition 145
IV.4 Extended Siamese Neural Networks 151
IV.5 Evaluation 152
IV.6 Conclusions and Future Work 157
IV.7 Acknowledgements 158
References 158

Appendices **163**

Defining the initial case-base for a CBR operator support system in digital finishing **165**
A.1 Introduction 167
A.2 Method 174
A.3 Results 177
A.4 Discussion and lessons learned 181
A.5 Conclusion and further work 182
A.6 Acknowledgments 182
References 182

Demonstrating the myCBR Rest API **187**
B.1 Introduction 189
B.2 myCBR Rest API 190
B.3 Experiments and Applications 191
B.4 Conclusion and Outlook 193
References 194

Use Case applying machine-learning techniques for improving operation of the distribution network **197**
C.1 Introduction 199
C.2 Machine Learning Techniques In Fault Handling . 200
C.3 Outage and Fault Management - an example for application of new techniques 202
C.4 Discussion and Limitations 205
C.5 Conclusions and future work 206
C.6 Acknowledgements 207
References 207

Chapter 1

Introduction

1.1 Motivation

The Norwegian aquaculture industry has ambitious growth targets. This is a goal from the industry itself as well as a goal set by the Norwegian government. However, the industry and its operations are already straining on the environment surrounding the aquaculture operation. This includes waste on the seafloor from fish feed and fish excrement. Escaping salmon and sea lice transmitting to wild salmon is a recurring problem. Sea lice are one of the biggest challenges in the aquaculture industry. Additionally, many of these problems increase in severity when the density of aquaculture operations increases. Many of the best locations for aquaculture operations are already overcrowded, and regulators will not let these sites be host to more aquaculture operations. Thus the industry has two options to grow: it can change the way it operates to reduce its environmental impact (e.g., closed cages to reduce pollution, escaping, and sea lice spreading), or the industry can try to utilize unused locations such as exposed locations. Exposed aquaculture sites have been under-utilized because they are more resource-demanding to operate. In addition, such sites have an increased risk for personnel.

Aquaculture is already one of the occupations with the highest work-related risk in Norway [3, 4]. The aquaculture industry can alleviate the work-related risks by increasing the robustness of the aquaculture site structures. While creating more robust structures may increase the safety of the personnel and fish at the site, it is not guaranteed to ensure continuous operations at an exposed location. The operation of an aquaculture location requires the use of different types of boats which perform different types of operations on the location. As a result, most of these operations involve interaction between a moored flexible structure and a boat. Such operations are inherently sensitive to rough conditions such as wind, weather, and waves because these conditions quickly introduce relative movement between the structure and the boat. Most exposed locations are also more remote, with longer travel time from any on-shore site to the aquaculture site. This results in longer personnel-transport in rougher conditions, increasing the number of days where personnel stay on-shore because of bad weather.

Currently, there are multiple efforts to alleviate the above-mentioned obstacles to exposed aquaculture operations. One of the obvious efforts is to increase the robustness of the aquaculture installations. This includes efforts such as Ocean Farm 1[1] by Salmar and "Jostein Albert"[2] by NordLaks. However, as previously mentioned, these do not negate the problems that arise from rough weather affecting the interaction between structures and boats. Reducing the amount of manual work on aquaculture sites overall will have a broader positive impact on risk to personnel on exposed aquaculture installations. Additionally, the digitalization of industry and society increases. This development results in cheaper and better sensors, cameras, and connectivity (e.g., 5G). These developments will provide the industry with more high-quality data from its operations.

Decision-makers within the aquaculture[3] industry faces complex decision problems. "Will the conditions at my aquaculture site be good enough to successfully de-lice the salmon stock?" This is just one example, but answering these types of questions relies on knowledge from different fields of science and heterogeneous data from many different sources. Accurately predicting the success of a delouse operation requires knowledge from many different fields of science. It requires knowledge from meteorology, fish biology, economics, ocean modeling, and more. In addition, the task could require collecting data from previous aquaculture installations (location, time, and amount), weather forecast, ocean model output, fish stock models, and previous logs of fuel consumption. Coming to a decision that is well-grounded in that knowledge is demanding. However, successfully making a well-grounded decision help decision-makers optimize the operation of the business. As stated earlier, aquaculture is a growing industry, and as a result, the industry needs to hire more workers. Digitalization of operations and DSSs can support retaining knowledge from workers that retire and transferring knowledge to newly hired workers.

Increasing the level of automation primarily reduces the amount of manual work needed per production unit(i.e., increased productivity). It also means that more tasks can be done from a remote location, so that personnel can operate without being on-site, increasing the safety of the operations while still enabling growth and productivity.

In recent years, machine learning has matured and is currently used across industries to increase automation and for use in decision support. To further increase the level of automation in the aquacul-

[1] https://www.salmar.no/havbasert-fiskeoppdrett-en-ny-aera/

[2] https://www.nordlaks.no/havfarm/havfarm1

[3] For this study we define "aquaculture" as the industry dealing with cultivation and farming of aquatic biomass.

ture industry, and especially to enable the industry to operate safely in exposed locations, the potential in ML is high and expected to grow.

1.2 Challenges

There are two main challenges with applying machine learning to advance the aquaculture industry in terms of automation; **1)** Lack of relevant and high-quality data and **2)** no strong culture for data-driven analysis.

The low digital maturity and overall lack of digitalization culture are apparent in the low adoption rate of sensors and data gathering in aquaculture site operations. The lack of digitalization also means less adoption of digital tools into the work processes in the industry. Finally, it means less trust in digital tools. The lack of data comes as a result of this lack of digitalization.

Lack of data and trust in digital tools present technology suppliers and researchers with challenges when developing automation solutions for the aquaculture industry. Methods which require high volumes of data may not be suitable in all aspects of the aquaculture industry. Likewise, the aquaculture industry requires machine learning models that are explainable. Deep learning (DL) is based on artificial neural networks (ANN) and is a highly successful method for building ML models of complex phenomenons. It could help the industry solve and automate problems. At the same time, DL models require a large amount of labeled training data that are often not available in the aquaculture industry. Even tasks connected to video data, which seems like a natural fit for deep learning, are laborious [5] as there are little to no labels attached to the data. In other aquaculture applications, there is both little data and little to no labels [6, 7].

In addition to strict requirements for volumes of high-quality data, DL models have lower explainability than many other ML methods because of a high level of complexity. Methods for generating explanations for how DL models work exists, such as LIME [8] and SHAP [9]. These systems depend on the user having some knowledge of how machine learning models work for the user to understand the explanations. These systems may not be a satisfactory solution in industries where users may not have that knowledge.

Building systems that are useful for the aquaculture industry means building systems that create trust in the system and are constructed in the context of data scarcity. CBR is one way of addressing both explainability and data scarcity. CBR explains by example, which is more intuitive to inexperienced users than

explaining models through frameworks such as LIME and SHAP. CBR does not only give a true/false or success/failure prediction to its users but presents the user with several of the most similar previously recorded cases with their recorded solution. If the user agrees with this similarity assessment, the output of a CBR-based DSS is the user's solution to previous cases. Availability of cases in such a system will improve the trust in the system as the system is transparent. A DSS that reads data through sensors and other data sources and provides an aquaculture operator with the most similar previous case would then alleviate some of the challenges listed above.

The similarity measure used in CBR is often hard to model manually. Many CBR systems encode expert knowledge in the similarity measures, but this process is resource-demanding. Learning the similarity measure from recorded data alleviates this. This process, while still requiring labeled data, requires less training than many other machine learning problems [10].

To summarize, a DSS for aquaculture built on machine learning methods needs to be able to learn from data sets that are typically smaller than most used in deep learning. The DSS also needs to be able to explain the output presented to the user. This explanation needs to be intuitive so that it can be understood by a wide range of users.

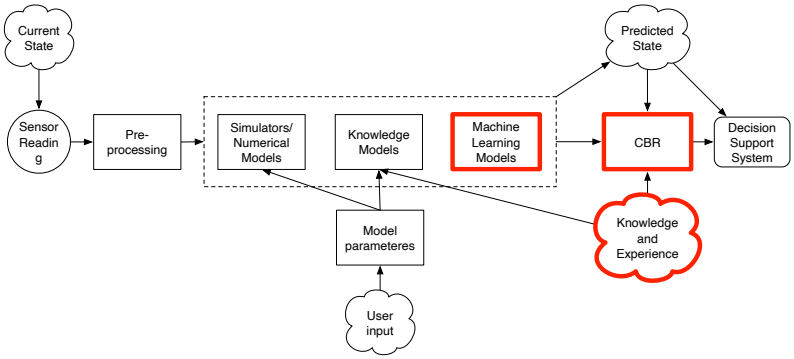


Figure 1.1: The general architecture of a modern decision support system employing sensor readings through preprocessing, and also using an ensemble of different types of models to provide decision support for the user. The highlighted modules of the DSS architecture are the focus of the work in this PhD project.

A general architecture for a DSS is shown in Figure 1.1. This shows the flow of information through a DSS from start to end, from the state of the decision problem to the output presented to the user. The

DSS reads the current state of the decision problem through sensors or user input. This input is then preprocessed before being used as input to the models. This preprocessed data can be used for setting parameters for numerical-, analytical- or machine learning models. These models are then used to predict a future state of the decision problem (e.g., how much movement will the predicted weather induce on the aquaculture installation). DSSs typically support the user in making a decision that will have an impact in the future, e.g., planning a service operation on an aquaculture operation. Contrastively, DSSs are typically not used to support processes that require an immediate or instant reaction. In cases where a system wants to support immediate situations, automation without user input is better suited.

During the design and implementation phase of a DSS, the preprocessed data can also be converted to a dataset for training machine learning models. However, as DSSs are never fully automated by design, user input is always required. DSSs typically help users make decisions about future events, and as such, date and time are needed user input (e.g., the user is planning on cleaning a fish cage in two days). The DSS then uses the models to compute a prediction for the future state (e.g., the DSS predicts local weather conditions for that location two days from now). These predictions can be used by the DSS directly or used as input to a CBR system. This enables a CBR to retrieve previous situations which are similar to the predicted situation ("Future State" in 1.1, e.g., the weather conditions in two days are similar to weather conditions when a cleaning operation failed at the same location). Knowledge and experience are often used for designing an effective DSS for a decision problem. DSS designers will apply this knowledge if designing numerical or analytical models for a DSS. It will also be important for specifying how cases and their solutions are stored in a CBR system.

1.3 Research goals and research questions

This thesis focuses on increasing automation within the aquaculture industry through data-driven models. Increasing the level of automation reduces the amount of manual work needed per production unit, which increases productivity. More automation also means that more tasks can be done remotely. Working remotely reduces the worker's exposure to harsh conditions on the aquaculture site. This will increase the safety of the operations while still enabling growth and productivity.

We investigate how ML can enable DSS to increase the level of automation in the aquaculture industry. Increasing automation will

allow the industry to operate safely in exposed locations.

This section describes the main objectives of this PhD project. First, an overall research goal is presented. This goal answers some of the challenges outlined in Section 1.2. This overall goal is then materialized into four more specific research questions (RQs 1-4). The work presented later in the thesis contributes to answering these research questions and work towards the research goal.

Research goal

Advancing our understanding of how machine learning can help the aquaculture industry expand into exposed areas.

The research goal is achieved through answering four research questions, further split into two groups. The first two research questions (RQ1 and RQ2) address the domain-specific parts of the questions on using DSS and ML in aquaculture. The final research questions (RQ3 and RQ4) are domain-independent. They pertain to how the methods examined through RQ1 and RQ2 can be adapted and extended.

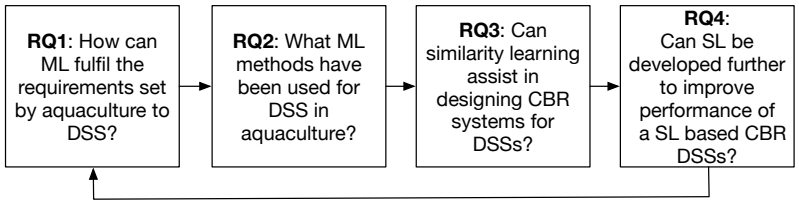


Figure 1.2: The relation betwen the research questions in the PhD thesis. **RQ1** asks how ML can fulfil the requirements set by aquaculture to DSSs. **RQ2** examines which ML methods that have already been applied to DSSs for aquaculture. **RQ3** examines if similarity learning (SL) can assist in designing CBR systems for DSSs.Finally, **RQ4** addresses the if SL methods can be developed further to increase performance of similarity learning based CBR DSSs.

1.3.1 Domain related research questions

To work towards the research goal, one must understand how machine learning can be applied to DSS in aquaculture. The solution has to address the two main challenges listed in Section 1.2, namely low digital maturity that increase the need for explainable models and

low availability of high-quality data in the aquaculture industry. This problem can be formulated as the following research question.

Research question 1

How can we use ML to make useful DSSs for aquaculture that are explainable and work in a data-scarce domain?

Previous work may have addressed some of the challenges listed in Section 1.2. If the goal is to innovate and not replicate previous work, it is important to extend and build upon results from previous work. More concretely, we need to investigate if ML has been applied in DSSs for the aquaculture industry. And, if ML has been applied to DSSs for the aquaculture industry, what types of ML methods and how they were applied. This leads to the next research question.

Research question 2

What types of machine learning methods have been used in decision support systems within the aquaculture application domain?

1.3.2 Method research questions

As mentioned in Section 1.2, the hypothesis is that using CBR and similarity learning as parts of a DSS can result in beneficial tools for industries that have low trust in digital solutions as well as low amounts of gathered data. The following research question is formulated to investigate the applicability of similarity learning to extend the state-of-the-art for CBR as a DSS.

Research question 3

How can similarity learning assist in designing CBR systems for DSSs?

Finally, if similarity learning can be used as a machine learning method to create CBR-based decision support systems, can it be developed further to improve performance in this context? This question is formulated in the final research question.

Research question 4

How can similarity learning methods be developed further to improve performance of a similarity learning based CBR DSSs?

1.4 Research context

The research done in this thesis was done as a part of the SFI[4] EXPOSED center. This center was created to develop knowledge, technology, and innovation to enable more effective and sustainable aquaculture production in exposed aquaculture environments. The PhD project described in this thesis was one of the first PhD projects started in the SFI EXPOSED center. The goal of this PhD project was described as applying Machine Learning to study and design enabling technology, to further the goals of the center.

Some of the data used in the PhD project were gathered through EXPOSED partners (such as Anteo ASA providing data for Paper I). Other data were gathered by SINTEF as the main research partner in the project (such as data for Paper IV). Finally, the data used in Paper III was provided by aquaculture technology companies (Sealab, now part of Cageeye).

The work presented in this thesis can be split into three different phases. At the early stages of SFI EXPOSED, the data gathering part of the project was only starting up. Thus the first phase of this PhD project consisted of experimenting and exploring early data sources. This included the work presented in Paper I where we gathered and preprocessed multiple data sources that were relevant to the aquaculture sites that took part in our use case. After this, we explored how to make ML models adapt to the local conditions. To enable the identification of differences between aquaculture sites, we developed ESNN, which is an extension of Siamese Neural Networks (SNNs). We introduced this novel method in Paper II and showed how it outperforms SNNs in terms of accuracy while matching the training time of SNNs. Finally, ESNN was used to develop a prototype DSS for aquaculture operations. In Paper IV ESNN and two other similarity learning methods were tested and evaluated on aquaculture data.

- **Phase one**: Early data exploration (Paper I and experiments shown in Figure 4.4 and Figure 4.5)

[4] A SFI center is a Norwegian research council center for research-driven innovation

- **Phase two**: Method development (Paper II)
- **Phase three**: Method testing in domain (Paper III and Paper IV)

1.5 Thesis structure

This thesis is composed of two parts. Part one provides the overall motivation, structure, and main results from the thesis work and is divided into five chapters. Chapter 1 presents the context of the thesis research, including the motivation behind developing automation and decision support systems for the aquaculture industry and the main challenges to address as part of the thesis work. Chapter 2 presents the scientific background of the PhD thesis for the reader. Chapter 3 answers RQ2 and presents a systematic mapping of scientific literature that describes state-of-the-art for DSSs in aquaculture. Chapter 4 presents a summary of the results produced in the thesis as a response to the research questions described in Section 1.3. Chapter 5 evaluates and discusses the results in the context of the original motivation and state-of-the-art of the research field and points to interesting and promising future directions of research that spring from the results of this thesis. Part two contains the four main papers published as part of the thesis work and three auxiliary papers.

Chapter 2

Background

This chapter gives an overview of the scientific fields that this thesis builds upon. Topics that are covered in this thesis are DSSs, ML in DSSs, artificial neural networks (ANN), CBR, and similarity learning (SL). First, we describe DSS that encapsulates CBR systems. Then we introduce CBR and how CBR systems can be described and designed. ANNs are then described. ANNs are used to learn similarity functions which are an important part of designing CBR systems. Finally, we present methods for learning similarity functions (similarity learning) and a framework for categorizing similarity learning methods.

2.1 Decision support systems

Decision-makers are faced with complex strategic or operational decisions when managing businesses or organizations. DSSs are designed to enable decision-makers to be well informed when making such decisions. DSSs are built to support decision-makers by integrating information from different sources and present them in ways that enable the user to make more informed decisions. DSSs try to combine domain and technical knowledge and package it in a way that can be of practical use for non-scientists [11].

DSSs originated in the 1960s [12] and have been used in multiple domains such as medicine [13, 14], power grid [15], fisheries [16] and aquaculture [17].

Typically, a DSS reads the current state of the decision problem through user input or sensors, then uses models (learned, analytical or knowledge-based) to predict a future state. It could also use models for classification, e.g. classifying if a salmon in an image has a disease. Decision-making is usually motivated by optimizing some future goal: revenue or plans for future business operations. Thus DSSs usually try, through models, to answer how a decision will impact such a future goal (e.g. will performing a de-licing operation be successful if done in five days)

DSSs can be categorized based on the underlying methods used to produce the predictions that are presented to the user. Alternatively, categorization can be based on the type of problem that the methods are applied to. We divide DSSs into five different categories:

- **Model-driven DSSs** employs numerical or analytical models to produce predictions for the DSS. Using a model-driven DSS, the user inputs parameters that are relevant to the decision problem. These parameters are then used to initialize the models that the DSS is based on, producing and output aiming to help the user make better decisions.

- **Geographical DSSs/Spatial DSSs** are a type of DSS that is structured and designed to assist the user with decision problems that are of a spatial or geographical nature. Such decision problems can be e.g., land allocation planning, resource allocation planning, or urban planning. The underlying methods to produce the output for decision support can be model, knowledge, data, or ml-driven. However, the focus and structure of the DSS will be on how to group the information and DSS output according to geographical or spatial dimensions.

- **Multi-criteria DSSs** defines a group of DSSs that supports the user in decision making according to more than one criteria. Many DSSs support the user in making decisions to optimize one criterium (e.g. revenue or productivity). However, multi-criteria DSSs will support the user with making decisions that are optimal in relation to more than one criteria (e.g. production and environmental impact for an aquaculture production site)

- **Data-driven DSS** is a type of DSS where great emphasis is put into integrating as much operating data as possible into the DSS. This includes manipulating the data or forming the DSS to fit the data.

- **Machine learning DSS** is a type of DSS where the DSS is based on models that are learned from the data pertaining to the decision problem. Where ML-based DSS always uses the data to create models for the DSS, data-driven DSS does not do so through ML methods.

Decision support systems are discussed in more detail in Chapter 3 where a systematic mapping of DSSs in aquaculture is presented.

2.2 Case-based reasoning

Case-based reasoning [18] is a computational method based on a model of human cognition. CBR is founded on the observation that humans often solve novel problems by remembering past experiences that are similar, i.e., the assumption that similar problems have similar

solutions. Presenting the user with the previous problem that led to the solution (output) makes CBR-based systems more transparent and intuitive for users. CBR can be seen as a machine learning method that works in a way that is very explainable to the user. Case-based reasoning is a lazy machine learning method. Eager machine learning methods such as neural networks and decision trees try to find a general model that fits each data point (or batch of such) at training time. In contrast, CBR delays generalization until query time. As a result, lazy learning methods are often preferred in situations with less data, where generalization through all data points is not possible.

CBR stems from, among others, Schank's work on language[19]. This work introduced Memory Organization Packets (MOPs) that organized episodic memory, which could be used to understand new experiences. MOPs can be seen as an early version of cases in CBR.

This work then led to Schank's work on dynamic memory[20] which is a predecessor of CBR. Janet Kolodner developed MOPs further into episodic MOPS (EMOPS) [21] that moved the theory closer to what we know as CBR now.

A new problem input to a CBR system is solved by reusing solutions of a similar problem solved earlier. In CBR, previously recorded problems and their solutions are called cases. Cases are stored in a case base. Cases are designed as two sets of features, with one set describing the problem while the other set is describing the solution. These features of a problem are usually implemented as attribute values. The solution can be described by attribute values, single values, predictions, instructions, or a decision. CBR systems are designed so that the features of a case describes a problem in a way that distinct problems that have distinct solutions can be separated using these features. A CBR system will typically start out with a case base populated by a low number of archetypical cases that cover previously seen problems and their solutions. New problems that the CBR system are queried with that are sufficiently novel will be stored in the case base for future use. The complete CBR process, as seen when queried with a new case, is a cyclic four-step process as shown in Figure 2.1:

1. A new query case is compared with the cases stored in the case base. The most similar cases are then **retrieved**.

2. The most similar cases are then combined with the query case and **reused** to solve the problem posed by the query problem.

3. The solution generated is then tested for success and possibly **revised** and updated accordingly.

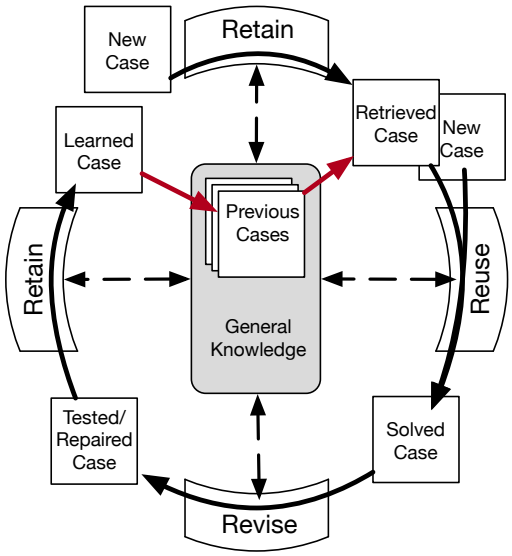


Figure 2.1: The CBR cycle [2] depicting the 4 R's: Retrieve, Reuse, Revise and Retain.

4. Finally, useful new cases and their solution are **retained** in the case base for future use.

Concretely, cases represent previous experiences or instances of a problem coupled with their solutions. Storing input that is associated with a specific solution/classification differentiates CBR from many other machine learning methods. A CBR system can give the user output in the form of a prediction or classification. CBR systems can also attach the previously recorded case that was the reason for that prediction or classification. This enables the CBR system user to view the problem description and solution part of the case simultaneously. Cases are typically as descriptive as possible for that reason and to make similarity calculation as accurate as possible. The solution of a case describes how to solve the problem. This could be a programmatically described solution for automatic problem resolution or a textual description for a user to implement. Implementations of CBR systems can differ in many ways, such as how to calculate the similarity or how many similar cases are retrieved and used. The design and implementation of a CBR system as well as the cases stored in the CBR system is the knowledge of that CBR system.

Richter et al. [22] grouped this CBR system knowledge into four knowledge containers: 1) Vocabulary, 2) Similarity measure, 3) Case

base and, 4) Solution transformation. The vocabulary contains what the CBR system can represent (e.g., a fish farmer cannot use a CBR system with features designed for sheep farming). The similarity measure specifies which cases are similar as a function of the features of the cases. Thus this knowledge creates a mapping of which solutions are most suited for which problem. A case base is a repository containing the problem descriptions and their associated solutions that the system can present to the user. Solution transformation contains knowledge about how to adapt the previously-stored solutions to fit new problems.

A case in CBR is then typically described through a vector $\boldsymbol{x}$ with a set of features n, from which similarity can be measured across different cases.

For cases $\boldsymbol{x}$ and $\boldsymbol{y}$ with n features and a weight vector $\boldsymbol{w}$ the equation for locally weighted similarity is:

$$GlobSim(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}) = \frac{\sum_{i=0}^{n} LocSim_i(x_i, y_i) * w_i}{\sum_{i=0}^{n} w_i}, \tag{2.1}$$

where w_i weights the importance of the i-th feature when calculating the similarity between cases. The weight vector $\boldsymbol{w}$ is typically defined for all cases, but may be defined for each individual similarity calculation. Finally, the function $LocSim_i(\cdot, \cdot)$ calculates the local similarity between each feature and can be defined per feature or be uniform across features.

A visual explanation of the mapping done by the similarity function from the problem to the solution space is shown in Figure 2.2.

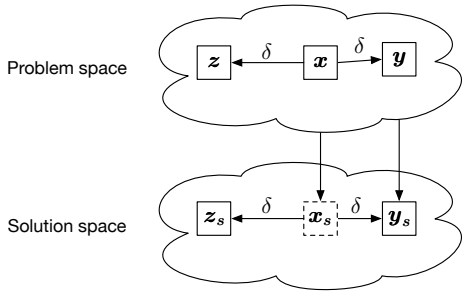


Figure 2.2: The mapping from problem space ($\boldsymbol{x}$,$\boldsymbol{y}$ and $\boldsymbol{z}$) to solution space ($\boldsymbol{x}_s$,$\boldsymbol{y}_s$ and $\boldsymbol{z}_s$) for the query case $\boldsymbol{y}$ and the cases stored in the case base $\boldsymbol{x}$ and $\boldsymbol{z}$. In line with the CBR model that states that similar problems have similar solutions, this example would retrieve $\boldsymbol{x}$ as the most similar case to $\boldsymbol{y}$ as the distance δ between $\boldsymbol{y}$ and $\boldsymbol{x}$ is smaller than the distance between $\boldsymbol{y}$ and $\boldsymbol{z}$.

The similarity function in CBR can be modeled using a knowledge-based, analytical or numerical -model. Similarity functions can also be learned from data (Similarity Learning).

2.3 Neural networks

Artificial neural networks (ANN) have been used in machine learning for a long time and are universal approximators [23, 24]. Typically, neural networks are used as an approximator of a unary function, such as a function that maps inputs to a classification. ANNs are also used for regression or time series prediction. In the case of time series prediction, the input can be a sequential or stacked time series. The corresponding output is the n predicted next values of that time series. ANNs are also used for seq2seq mapping (used for NLP). Figure 2.3 shows an example of an ANN which maps an input vector $\boldsymbol{i}$ of six values to an output vector of $\boldsymbol{o}$ of four values.

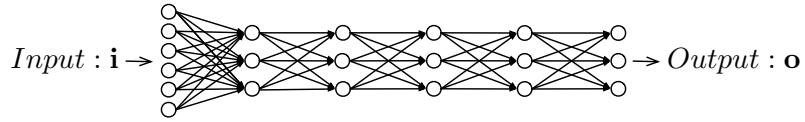


Figure 2.3: Illustration of an ANN with four hidden layers of three neurons, six input neurons and three output neurons. The input vector of six values is denoted as $\boldsymbol{i}$ and the output vector is denoted as $\boldsymbol{o}$.

The architecture of ANNs can be designed to solve different types of tasks. Long-short term memory (LSTM) [25] is an ANN designed to compute its output based on a sequence of data as input. Convolutional neural networks (CNNs) [26, 27] are designed to tackle datapoints with inherent spatial geometry such as pictures.

2.4 Similarity learning and metric learning

Similarity learning (SL) is a type of machine learning where the goal is to learn a relationship between two data points. More specifically, SL learns a binary function that maps two data points to an output that represents some relation between them ($S(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{s}$). This is in contrast to many machine learning methods that try to learn a unary function between one data point and its label ($F(\boldsymbol{x}) = \boldsymbol{l}$). Outside of CBR, SL is often called deep metric learning (DML), Siamese Neural networks (SNN), or triplet networks. Contrastive learning (CL) is highly related to DML but encompasses a larger set of goals (not

only learning the similarity between two data points) and methods. Typically, the relationship that SL aims to learn is the distance (δ) or the similarity ($1 - \delta$) between the two data points.

SL is suited for different sets of problems than unary ML methods. SL is applicable to clustering problems or matching [28]. A general type of problem that unary ML methods are ill-suited for is when the number of possible classes/labels grows too large. A normal way of representing labels at the output of a neural network is one-shot encoding. One such example would be facial recognition or re-identification, as illustrated by our work presented in Paper III. In re-identification tasks, the number of classes/labels is equal to the number of individuals you want to identify. In such cases, the number of labels quickly grows to an unmanageable number for output encodings such as one-hot encoding. Formed as a SL problem, a SL method would try to learn a binary function of the similarity between two pictures of faces. This way, the re-identification system would compare new pictures with already identified pictures to identify unlabeled pictures of faces. This SL architecture is invariant with the number of possible individuals or labels.

Another benefit of using SL over unary ML methods is that it can learn easier from small datasets. Solving complex problems with deep neural networks requires large models. Large neural networks need thousands of examples per class to correctly model the relationship between input and output [29]. Similarity learning, on the other hand, uses CL (comparing instances of classes to each other), so for each class, you would have a point of learning per instance for every other class. CL can also use other instances of the same class for training. Using pairs of data points connected to classes is not the only useful setting for contrastive learning. Contrastive learning could also be set up with a pair of series as input where the output would be a similarity of a future value in those two series (regression or time-series prediction). The fact that contrastive learning learns more from a dataset compared to traditional feed-forward networks means that similarity learning is well suited to tasks where we want to learn the relationship between many classes, such as facial re-identification. As such, similarity learning has been applied to tasks where the number of classes is high. This includes tasks such as signature fraud detection re-identification [30, 5], visual tracking [31, 32, 33], matching networks [28].

In practice, such similarity functions are learned from a dataset compromised of pairs or triplets of datapoints [34]. In the case of pairs of data points, if the datapoints' labels are the same, the similarity should be high or the distance low. Typically the weights of such networks are set by using backpropagation [35, 36] (e.g. as seen in [37,

38]). Another way of setting the weights of neural networks is using evolutionary algorithms (EAs) [39, 40]. EAs are also being used to evolve the topology in addition to the weights [41].

Learned similarity functions or metric functions are used in machine learning applications and machine learning methods such as CBR. CBR system designers apply similarity learning to learn a function used for retrieving similar cases. The retrieved cases should be similar to the query case in terms of the problem description.

Similarity learning can be fully or partially based on data or user-feedback through some learning process. However, SL based on user feedback, such as the work done by Stahl et al. [42] is not considered in this SL comparison. This is because, in the context of the goal of this PhD thesis, we want to increase the level of automation, and SL based on user feedback involves less automation.

Given a pair of data points $(\boldsymbol{x}, \boldsymbol{y})$, an embedding function $G(\cdot)$ and $C(\cdot)$ which models the distance between two embeddings, we can define a similarity function $\mathbb{S}$ as:

$$\mathbb{S}(\boldsymbol{x}, \boldsymbol{y}) = C(G(\boldsymbol{x}), G(\boldsymbol{y})), \tag{2.2}$$

where $G(\boldsymbol{x}) = \hat{\boldsymbol{x}}$ and $G(\boldsymbol{y}) = \hat{\boldsymbol{y}}$ represents embedding or information extraction from data points $\boldsymbol{x}$ and $\boldsymbol{y}$, i.e. $G(\cdot)$ highlights the parts of the data points most useful to calculate the similarity between them. $C(G(\boldsymbol{x}), G(\boldsymbol{y})) = C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ models the distance between the two data points based on the embeddings $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$. An illustration of this process can be seen in Figure 2.4. In the example shown in Figure 2.4, the query datapoint is $\boldsymbol{x}$. Assuming a linear distance metric, e.g. $\delta(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2$. Let $\boldsymbol{x}_s$ be the true solution/target of $\boldsymbol{x}$. Notice how $\boldsymbol{x}_s$ is closer to $\boldsymbol{y}_s$ than $\boldsymbol{z}_s$ ($\delta_s(\boldsymbol{x}_s, \boldsymbol{y}_s) < \delta_s(\boldsymbol{x}_s, \boldsymbol{z}_s)$) in the solution space shown in our example in Figure 2.4. However by just looking at the input feature vectors $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ - one can see that $\boldsymbol{z}$ is closer to $\boldsymbol{x}$ than $\boldsymbol{y}$ ($\delta_p(\boldsymbol{x}, \boldsymbol{y}) > \delta_p(\boldsymbol{x}, \boldsymbol{z})$). One way of solving this is to embed the features vectors into an embedding space where the distance between them is closer to the true distance in solution space:

$$\|\delta_e(\boldsymbol{x}, \boldsymbol{y}) - \delta_s(\boldsymbol{x_s}, \boldsymbol{y_s})\| < \|\delta_p(\boldsymbol{x}, \boldsymbol{y}) - \delta_s(\boldsymbol{x_s}, \boldsymbol{y_s})\| \tag{2.3}$$

Similarity functions can satisfy the inequality in Equation 2.3 through modelling the embedding function $G(\cdot)$ or the distance function itself $C(\cdot)$ ($\delta_e(\cdot)$), or both. The main difference between these approaches being that $G(\cdot)$ is a function of one data point, while $C(\cdot)$ is a function of both data points (or their embeddings).

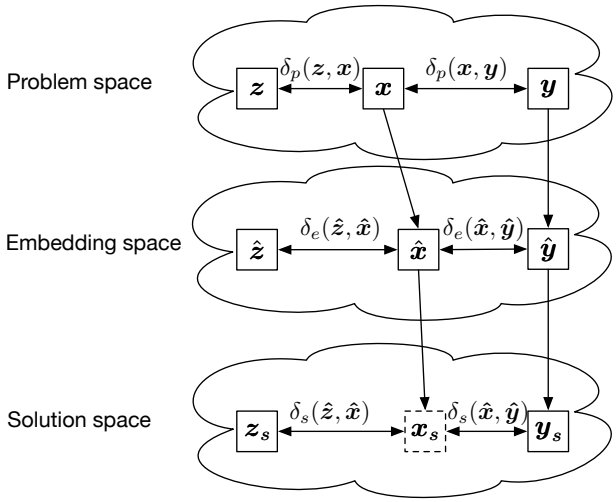


Figure 2.4: This figure illustrates similarity functions using embeddings in the context of the framework described by Equation 2.2.

The functions C and G can be either manually modeled or learned from data. With respect to this, we enumerate all of the different configurations (similarity function types) of Equation 2.2 and describe their main properties and give examples of literature for each type below. Note that we will use $\mathbb{S}(\cdot)$ to annotate the similarity measurement and $C(\cdot)$ for the sub-part of the similarity measurement that calculates the distance between the two outputs of $G(\cdot)$. $\mathbb{S}(\cdot)$ is distinct from $C(\cdot)$ unless $G(\boldsymbol{x}) = \boldsymbol{x}$.

Table 2.1 lists how different types of similarity metrics implement Equation 2.2. Similarity functions can implement $\mathbb{S}$ four different ways. Type 1 is similarity functions where the embedding function $G(\cdot)$ and the distance function $C(\cdot)$ is modeled. In this type of similarity function, the designer models $G(\cdot)$ to extract the most important parts of a data point for calculating similarity. $C(\cdot)$ is then designed to calculate similarity based on the output of $G(\cdot)$. $C(\cdot)$ is designed such that it highlights which differences between the two outputs of $G(\cdot)$ should have the greatest effect on the similarity. The work done by Nikpour et al. [43] shows a type 1 similarity function. Type 1 similarity functions are typically used where the designer already has a model for similarity or where the similarity function is well known. In type 2 similarity functions the embedding function $G(\cdot)$ is modeled while the distance function $C(\cdot)$ is learned from data. Examples of type 2 similarity functions are described in works done by Stahl et al. [44] which learns local similarity measures using an

evolutionary algorithm. Gabel et al. [38] models $G(\boldsymbol{x}) = \boldsymbol{x}$ (identity function), then uses an ANN to model the similarity function based on a concatenation of the two inputs. Type 2 similarity functions are typically applied in cases where the designer of the function does know which parts of the data point are important for similarity calculation but not how to combine the two data points.

		$C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$	
		Modeled	Learned
$G(\cdot)$	Modeled	Type 1: [43]	Type 2: [42, 45, 46, 47]
	Learned	Type 3: [48, 37, 49]	Type 4: [50]

Table 2.1: Different types of similarity measures in the proposed framework.

In type 3 similarity functions the embedding function $G(\cdot)$ is learned while the distance function $C(\cdot)$ is modeled. A popular example of this is the Siamese Neural Network (SNN) [48] where $G(\cdot)$ is modeled as an ANN. SNNs uses each data point as an input to an ANN ($G(\cdot)$) which converts the input vector to an embedded data point. This embedding is then used to calculate similarity through a static function, typically euclidian distance or manhattan distance. Type 3 similarity functions can be applied in cases where the function designer does not know which parts of the data points are important for calculating similarity.

Finally, in type 4 similarity both $G(\cdot)$ and $C(\cdot)$ are learned from data. There are few examples of type 4 similarity functions used in literature. During our literature study, done as part of Paper II, we found none. However, later work by Xiamoeng et al. [51] applied a type 4 similarity function that was very similar to the similarity function presented in Paper II. As shown by Paper II type 4 similarity functions are similar in terms of performance to type 3 similarity functions. However, when calculating measuring performance on datasets that are known to be hard to classify, type 4 similarity measures outperform type 3.

In most examples of how similarity learning is applied above, including the work done in Paper III and Paper IV, the similarity function is being applied on datasets meant for classification. This means the similarity learning objective is to output 100% similarity if two data points belong to the same class. However, this not the only objective of similarity learning. In the context of CBR, the usefulness of a similarity function is often measured on how useful the retrieved solution is. This is also called the utility of the retrieved case.

This aspect of similarity functions is not made distinct in this framework. One such aspect is the utility of the similarity function as discussed in Stahl et al. [42]. The utility is a measure of the usefulness of the classification or solution computed by the similarity function. Typically similarity metrics only use the feature vector (or problem description) to compute the similarity between two data points. The work by Stahl [42] expands upon this to include the utility of the solution. The retrieved case does not solely have the most similar problem but the most useful solution given the query case.

In re-identification tasks, the utility of a similarity function would be akin to not only retrieving the correct individual but also the most useful picture of that individual. E.g., return a nighttime picture of person X if the query picture is a nighttime picture of person X. However, this is built on feedback from users, so if users report that a daytime picture of the same person is more useful even if the query is a nighttime picture, the system is designed to learn that utility function. The utility of a similarity function could also include the adaptability of a retrieved case based on this similarity function, e.g., how easy is it to adapt the retrieved case to be a solution for the query case. Such a similarity function can be learned or modeled, and while it can still be described by Equation 2.2 it is not made distinct in that framework.

Chapter 3

Decision support systems in aquaculture[1]

3.1 Introduction

In this chapter, we present a systematic mapping study done to establish what recent (1980 to 2018) research had been done on the application of DSSs and ML-supported DSSs in aquaculture. This was needed as the goal of this PhD thesis was to apply ML through DSSs to enable aquaculture to expand to more exposed locations. Although the taxonomy and general process of creating, using, and maintaining DSSs is well documented both in case studies and research, the literature provides little information regarding assessments of its effectiveness and implementation in aquaculture. During our literature search, we found very few systematic literature reviews of DSS research. Those existing were exclusively within the domain of clinical medicine and, they targeted effects of DSSs [13, 14] and how to improve such systems [53]. There have been some non-systematic reviews of DSSs in aquaculture, including studies by Bergara-Solana et. al [54] and Leung [55]. Also, closely related to research on DSS in aquaculture is research on spatial DSS, which has been studied in non-systematic surveys [56, 57]

The primary research hypothesis of this systematic literature mapping is that there is little empirical knowledge of the effectiveness of DSS in the aquaculture domain. The secondary research hypothesis is that little research has been done on DSSs using machine learning in aquaculture.

Context: There has been little research into ML-based DSSs for aquaculture. Neither does there exist a DSS that uses real-time information from the aquaculture location.

Objectives: To conduct a mapping study to survey existing research on DSSs in aquaculture in order to identify useful approaches and clarify needs for further research.

[1]This chapter is based on a systematic mapping previously published to arXiv pre-print archive[52]. The mapping has since been updated with literature published up until 2019. The original and updated versions did include literature describing DSSs used in both fisheries and aquaculture. However, only literature related to aquaculture are included in the present exposition.

Method: A systematic mapping study of the available literature following the best-practice methods laid out by previous systematic review practitioners[58].

Results: 12 papers have been identified by topic, system classification, and relevance for the aquaculture domain. The study found that aquaculture DSSs rarely evaluate their system empirically. The study also identified only one study applying ML for a DSS in aquaculture.

Conclusions: The majority of studies on aquaculture decision support systems published over the last 30 years do not use DSSs based on machine learning. We also found that descriptions of data-driven methods for creating the models that the DSSs rely on are scarce in the literature discussing DSS in aquaculture.

3.2 Method

To gather data on the current state of DSS research within the domain of aquaculture, we conducted a systematic literature review, more specifically, a systematic literature mapping. This mapping study has been conducted in compliance with a pre-defined protocol created for this study to reduce the possibility of researcher bias [58]. As a result, the study complies with a well-known and defined method, providing reproducibility and rigor while at the same time acquiring knowledge about the field and answering our research questions.

The review protocol presented in this section is an essential component for providing context and domain classification. A protocol must be developed separately for each mapping study in order to define the main guidelines for conducting systematic mapping studies. Both Kitchenham [58] and Budgen et al. [59] states that the research questions in mapping studies are likely to be broader than in traditional Systematic Literature Reviews (SLRs) to adequately address the wider scope of the study. Kitchenham [58] also states that mapping studies will likely return a large number of studies which in turn will give a much broader coverage then the outcome of the SLR. On the basis of this, the systematic mapping study was selected as the method for achieving a broad resolution on the research questions as opposed to an SLR. Below we specify our review protocol used for this systematic mapping study. This protocol includes research questions, search questions, search engines, inclusion and exclusion criteria.

The following three research questions (RQs) were formulated in order to characterize the field of DSS within the aquaculture domain and, as a result, answer Research Question 2 (RQ2) raised in Section 1.3:

RQ2.1 What decision support systems exist for aquaculture?

RQ2.2 What are the most investigated aquaculture DSS topics, and how have DSS topics changed over time?

RQ2.3 Does DSSs in aquaculture use ML, and do they build ML models using captured and grounded data?

The following sources were used for this study:

- IEEE Explore
- CM Digital Library
- Google Scholar
- Citeseer library
- Springer
- Ei Compendex

These sources were selected because they are among the most important repositories for acquiring data in computer science, and collectively they addressed the main digital libraries deemed appropriate for this study. No researchers were contacted directly in this survey. The results retrieved from executing a search on the sources of literature were either dismissed or accepted into the study selection process, based on inclusion- and exclusion criteria. The inclusion- and exclusion criteria were used to exclude papers that are not relevant to answer the research questions. The study used the following inclusion criteria:

1. Only studies written in English or Norwegian;
2. Studies noting or referencing any of the subjects described in the research questions (e.g. "Decision support system" or "Aquaculture", see search strings below) in their title or abstract;
3. Studies published after 1990[2];
4. Studies that had no restriction on geographical placement (studies tying their results to geographical locations).

[2]And not after 2018 as this is when this study finished.

And the following exclusion criteria:

1. Meta-studies or reviews (we only included primary studies);
2. Duplicate results found in another search engine;
3. Analogous studies reporting similar results, only the most complete study was considered;
4. Inaccessible studies or books;
5. Literature that was not in the form of a pdf file.
6. No studies exclusively describing DSSs aquaculture in lakes or ponds were considered.

No quality assurance or assessment was performed during the search phase in order to achieve maximum coverage. In agreement with our review protocol, search terms were used for identifying relevant papers in the field of aquaculture DSSs. The process for synthesizing the query strings in our review protocol was derived from Kitchenham et al. [60, 58]. The search terms from a candidate set were selected using a trial. The candidate set was populated by deriving terms from the research questions. The Boolean "AND" was used to link keywords from different populations in the search strings. This resulted in the following final search strings:

- "Decision support system" AND "empirical evidence" (Q1)
- "Decision support system" OR "Decision support system in aquaculture" (Q2)

It is important to note that we chose to not include "operator support"/"operational support" that can be considered a weak synonym of DSS. However, operational support does not share the main motivation/stakeholder as DSS and focuses more on operational aspects.

3.3 Analysis

The method applied in order to identify relevant studies can be divided into three discrete steps, where the first stage was applying the search query on the literature sources. All query strings were applied to all of the search engines. This resulted in 1537 papers. After removing duplicates and applying all inclusion and exclusion criteria, this resulted in 12 papers.

The analysis is focusing on the research questions in the mapping study and is primarily confined into two issues: what decision making systems from aquaculture exists, and which empirical results do these systems provide, especially regarding real-time analytics.

3.3.1 Classification of selected studies and results

This study presents findings in the form of a qualitative synthesis, close to what Kitchenham [58] describes as a line of argument synthesis as this study tries to infer as much domain knowledge as possible. Consequential to the research questions and future work is the research regarding aquaculture DSSs, therefore, we start by presenting the number of journals found during the search phase, sorted by publishing year. It should be noted that the numbers presented in Figure 3.1 represent the 12 publications that were selected for this study and subsequently. These 12 publications form the basis for answering the research questions.

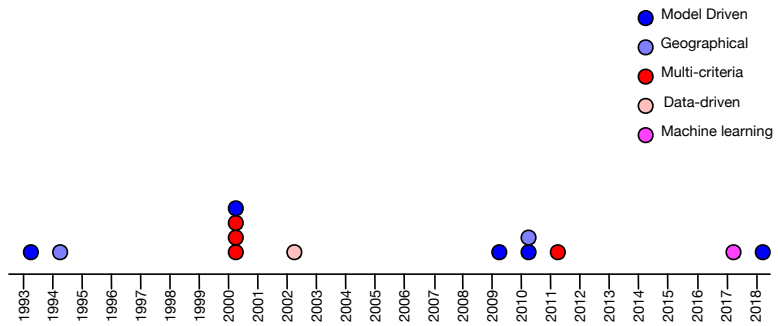


Figure 3.1: Number of publications per year that was gathered as part of this literature review. The search did not result in any papers published before 1993.

While DSSs for aquaculture have a long history going back to the early 1980s, e.g., Scuse et al.[61], the literature surrounding them is sparse. Figure 3.1 shows several periods during the last 25 years in which the systematic mapping study did not identify any relevant literature published with regards to the criteria previously defined. These periods include the years 1995-1999, 2001, 2003-2008, and 2012-2016. These periods of elevated interest are consistent with the result presented in Figure 3 of Arnott et al. [62]. Arnott et al. [62] describes a declining trend in DSS publications. However, the year 2000 seems to be an outlier. This could be a coincidence. Another possibility

is that the year 2000 was at the start of the decline in research of aquaculture DSS.

The declining DSS publishing trend of the last five years is not unique to the aquaculture disciplines as pointed out in [62], noting an overall decline in the number of DSS related publications since the early 1990's. Arnott et al. [62] speculates that the decline in DSS publications might stabilize in the coming years as DSS reaches a more balanced position within the domain of information systems, noting that the declining use of DSS might be due to the adoption of other models like the technology acceptance model. While the publishing trend is currently declining, Figure 3.1 shows that decision support systems within aquaculture is still being researched, but to a lesser degree than in previous years.

The following sections provide a discussion of how each research question was addressed in the mapping study. Results from the mapping study will be presented for each RQ, followed by a discussion of their implications.

3.3.2 (RQ-2.1) What decision support systems for aquaculture exists?

The results of this mapping are presented in Table 3.1.

Table 3.1: The selected studies.

Study	year
Bourke, G., Stagnitti, F., and Mitchell, B. "A decision support system for aquaculture research and management". In: *Aquacultural Engineering* vol. 12, no. 2 (1993), pp. 111–123	1993
Silvert, W. "A decision support system for regulating finfish aquaculture". In: *Ecological modelling* vol. 75 (1994), pp. 609–615	1994
Bolte, J., Nath, S., and Ernst, D. "Development of decision support tools for aquaculture: the POND experience". In: *Aquacultural engineering* vol. 23, no. 1 (2000), pp. 103–119	2000
El-Gayar, O. F. and Leung, P. "ADDSS: a tool for regional aquaculture development". In: *Aquacultural Engineering* vol. 23, no. 1 (2000), pp. 181–202	2000

Table 3.1: (Cont.) The selected studies.

Study	year
Ernst, D. H., Bolte, J. P., and Nath, S. S. "AquaFarm: simulation and decision support for aquaculture facility design and management planning". In: *Aquacultural Engineering* vol. 23, no. 1 (2000), pp. 121–179	2000
Nath, S. S., Bolte, J. P., Ross, L. G., and Aguilar-Manjarrez, J. "Applications of geographical information systems (GIS) for spatial decision support in aquaculture". In: *Aquacultural Engineering* vol. 23, no. 1 (2000), pp. 233–278	2000
Hargrave, B. T. "A traffic light decision system for marine finfish aquaculture siting". In: *Ocean & coastal management* vol. 45, no. 4 (2002), pp. 215–235	2002
Halide, H., Stigebrandt, A., Rehbein, M., and McKinnon, A. "Developing a decision support system for sustainable cage aquaculture". In: *Environmental Modelling & Software* vol. 24, no. 6 (2009), pp. 694–702	2009
Silvert, W. "Decision support for stakeholders". English. In: vol. 1. Ottawa, ON, Canada, 2010, pp. 523–529	2010
Radulescu, C. and Rahoveanu, M. "A multi-criteria evaluation framework for fish farms". English. In: *Studies in Informatics and Control* vol. 20, no. 2 (2011), pp. 181–6	2011
Magno-Tan, M., Alejandrino, A., Dela Cruz, C., Inoc, A., and Coronado, A. "Web-based decision support system for broodstock management of Siganus guttatus (Bloch, 1787) in open fish cage". English. In: *International Journal of Machine Learning and Computing* vol. 7, no. 6 (2017), pp. 208–12	2017
Cobo, Á., Llorente, I., Luna, L., and Luna, M. "A decision support system for fish farming using particle swarm optimization". In: *Computers and Electronics in Agriculture* (2018)	2018

A further classification of the type of DSS used in each paper has been performed and is shown in Table 3.2. As can be seen from the table, the majority of the aquaculture DSS systems are either model-driven, geographical, or multi-criteria. Only one data-driven and ML-based DSS were found in this study.

This indicates that up until 2018, model-driven, geographical and multi-criteria DSS were the most studied DSS types within the aquaculture domain. This ML DSS and is the most fitting and applicable methodologies for the aquaculture DSS domain.

DSS type	Studies
Model-driven DSS	[63, 17, 69, 70, 73]
Geographical DSS	[64, 67, 70]
Multi-criteria DSS	[65, 66, 17, 71]
Data-driven DSS	[68]
Machine learning DSS	[72]

Table 3.2: DSS studies grouped by DSS methodology.

Table 3.2 shows that model-driven systems remain the overall most popular design, but also suggest that the aquaculture domain contain distinct requirements, which are best solved by differing methods. Also one can observe that there is little overlap between the DSS types. The sum of types of DSSs in Table 3.2 is 13 and the number of studies is 12 so only on study actually inhabits more than one type. There is no reason why a DSS using geographical models cannot also be using neural networks as well. The most likely reasons is research method (focusing on measuring effect of one type of DSS rather than multiple) and effort (implementing more than one is more resources demanding)

3.3.3 (RQ-2.2) What are the most investigated aquaculture DSS topics and how they have changed over time.

In addition to classifying the result set based on their type, as seen in Table 3.2, the papers were also classified by their problem domains. The results of this classification are presented in Table 3.3, which shows what papers corresponds to what topic which shows how the popularity of the different topics has changed over time.

Associated DSS topic	Study ID (SID)
Research	[63]
Scheduling and planning	[65], [67]
Sustainability	[69]
Management decisions	[64], [65], [66], [17], [69], [70], [71], [72], [73]

Table 3.3: DSS systems grouped by topics.

Decision support systems for aquaculture are complex, as can be seen in e.g. Ernst et al. [17]. Discovering all parameters influencing decisions that can be mapped to real-world scenarios is difficult. This study shows that most aquaculture DSS only takes a small subset of criteria into consideration, as shown in Table 3.3. While there are some exceptions to the case where papers cover multiple topics, e.g. [69], it is worth noting that some of the topics covered are often closely related, as is the case with sustainability and management decisions. As such, it is natural that papers will often go into related topics.

The study could not find any trends regarding the popularity of different topics, and how these topics changed over time, a result most likely due to the small data-set.

From Table 3.3 we may infer that management decisions are the most researched form of DSS topics during the most recent years. Scheduling and planning are also related to management decisions, as both concern productivity. Thus it is not surprising to see both of the topics in the literature, as productivity and efficiency is a universal concern of all business, including aquaculture.

Lastly, the analysis of the studies shown in Table 3.3 shows that only one study [69] addresses more than one aquaculture DSS topic. Similarly, Table 3.2 shows only two studies [17, 70] that studies more than one DSS method. This supports the assumption that DSSs are complex, e.g., by way of involving models from different fields of science. This complexity makes a resource threshold for implementing and studying combinations of DSS methods or combinations of DSS topics given the same method. Another possibility can also be that it is much easier to do one analysis at a time, both in terms of experimental method and presentation to the reader.

3.3.3.1 (RQ-2.3) Does DSSs in aquaculture use ML and do they build ML models using captured and grounded data?

An important quality for a DSS is how well it represents the true situation of the decision problem. This quality can be threatened by using out-of-date models or out-of-date input data (e.g., simulation or model parameters). We searched the retrieved literature for research on applying machine learning in DSSs for aquaculture.

In this context, the study defines machine learning as creating models of phenomena based on data. This is in opposition to more static AI methods such as rule-based expert systems (e.g., see the traffic light system of Hargrave in [68]), which is not considered machine learning in this work. Our search did only find one study applying machine learning as part of the DSS design. Magno-Tan et al. [72] used machine learning to build prediction models that were used in a DSS to support aquaculture operations. The DSS uses different prediction modules based on ANNs to predict features about the welfare of the fish including, dissolved oxygen, water temperature, and salinity. The predicted values are then used to produce graphs, reports, prediction of fish-kill and recommended tasks/actions to the DSS user.

There are several possible reasons for the scarcity of ML usage in DSS in aquaculture. One of which is the lack of digitalization within the field. Until recently (2018), very few commercial aquaculture operations gathered data for usage in DSS or ML. In aquaculture, data should be available through operational requirements (e.g. video monitoring, production logging, operational logging, positional logging of ships). It is an obvious disadvantage for DSS scientists not to draw on these resources for model creation and prediction. This should be an area of focus in future DSS research within this domain.

3.4 Discussion

Noticeably throughout the study, only one [68] of the papers found contained rigorous empirical evaluation. However, most contained an overview of the system and an outline of their methodology. Some studies used validation data sets. Hargrave et al. [68] validated their results against experts and actual decisions. Some studies claim to present validation of their results but providing such a short description that the reader is left uncertain with regards to the validation process.

Although the resulting set of papers does not provide a unified view of DSS practice, it offers a broad picture and experience of the problem domain. The resulting set of papers could provide those

looking to create new DSS within the aquaculture domain with some helpful insights regarding what methodologies are best suited in the design of the DSS depending on the target domain, and an indication of the possibilities within DSSs.

It can be noted that only one of the resulting papers used machine learning for creating models as the basis for a decision support system. Additionally, none used real-time information as a basis for the decision support system. The results of the mapping study provide the reader with useful and relevant sources of information that could be helpful in the design or research on DSSs in aquaculture.

Ideally, in aquaculture, a DSS should aid management at a high level of abstraction on the basis of large quantities of data. The DSS should build predictive models based on gathered data. A DSS then uses predictions from these models to aid management in increasing the probability of an optimal outcome. However, designing a system that uses constant up-to-date data for running simulations/applying these data to models poses difficulties. First and foremost such the system is required to keep continuous connections to the various data sources to manage the data; furthermore, the system is responsible for completing its analysis within a given deadline. These responsibilities often conflict as no downtime can be expected to complete the deadline responsibilities of the system, and updating data can cause analytics processes to be invalidated during its execution.

Most DSSs for aquaculture do not need to be real-time systems to aid management, and the increased value to the customer of such a real-time system is often given low priority because of the cost of implementation.

The research questions require an assessment of the selected data sources to determine whether we identified all relevant publications and whether our initial classification of the problem domain is denoted correctly for analysis. As there did not exist any previous systematic mapping study on the search terms, some publications in the result-set could be missing due to narrow search terms.

How well our study answered the research questions is influenced by the choices made as part of the review protocol. This includes the search criteria, the scope of the search, and the search terms used that are again limited by the search engine's capabilities. Our primary method for avoiding these pitfalls has been to employ our third-party reviewers to perform a random test on the search queries to find papers we missed in our systematic mapping study. The third-party reviewer did not find any significant articles missed in this study; therefore, we argue that we did manage to provide the most relevant documents published in scientific journals and computer science literature. However, studies that were not peer-reviewed

have not been considered for this study. Thus, it is possible that we might have missed relevant studies. Still, given this limitation, we presume the study adequately addresses the principal research questions. There exists no former mapping study on this particular domain that the authors could find. Because the topics addressed in this study are selective, there exists little evidence in either direction that we have omitted a major topic that provides a substantial empirical evaluation that is in direct relevance to this paper.

We did find some studies which fell outside our criteria but are still interesting. However, we found that extending the criteria in any way to include these studies resulted in too large of a data corpus to be handled within the time frame of our project. We will include them for reference for the readers. These include big data in aquaculture [74], short-term prediction of marine sensors data [75], decision support for feeding systems [76], expert systems based on CBR for diagnosis of fish disease [77] and operational support in fish farming through CBR [78]. More recent works also involve ML for computer vision[79].

Finally, this study identified that there is a need for investigation on how to build DSSs for aquaculture that is based on ML models and how they perform. This thesis will try to address this by investigating how ML can fulfill the requirements set by the aquaculture industry to DSSs.

Chapter 4

Research results

This chapter describes the results from the PhD project using the research questions set down in Section 1.3. The PhD project has set a research goal to "Develop and evaluate new technology for the aquaculture industry using machine learning" and listed four research questions to help make the research goal more concrete. Answering these four research questions will contribute to the research goal. Below we briefly summarize the contributions of this PhD project. Then the contributions to each research question are summarized.

4.1 Research contributions

The research done throughout this thesis has six main contributions.

C1 A systematic literature mapping study of previous studies of DSSs in aquaculture (Chapter 3). The mapping study helped establish the current state of research on DSSs in the aquaculture domain and identified open areas.

C2 A framework to combine important data and align across different datasets (Paper I and Paper III). This contribution helped map out data sources for enriching domain-specific datasets with data from public data sources such as the FROST API[1] and useful models such as NORA10 [80, 81].

C3 A framework to analyze current methods for learning similarity measures (Paper II). This framework describes a similarity function formally, then enumerates the different configurations such a function can have. These configurations are then exemplified through different similarity functions in machine learning research. The framework also describes features of the different configurations.

C4 An extension of current methods for calculating similarity or distance between two data points; Extended Siamese Neural Network (ESNN Paper II). As a result of the framework listed above, an ESNN was developed as an extension of SNNs, with a

[1] https://frost.met.no

second layer of learning compared to SNNs. It was shown that this enables ESNN to learn similarity better on some data sets.

C5 A demonstration of the usefulness of similarity learning in the aquaculture domain through the application of SNNs. (Paper III). The study of similarity learning in the aquaculture domain showed that similarity learning applied as re-identification could be used on salmon.

C6 A demonstration of how ESNNs outperforms SNNs for similarity learning (Paper IV) in the aquaculture domain. We demonstrate that ESNN outperforms SNNs for similarity learning used as a basis for a CBR system that is used as a DSS for aquaculture operation planning.

4.2 List of publications

This section gives an overview of the publications included in this thesis. This selection represents a subset of the publications produced in this project, some additional relevant publications are summarized in Section 4.4.

The first paper addresses how to apply machine learning and CBR to build a decision support system for the aquaculture industry. We collected data from aquaculture operations and combined the operational data with weather data and ocean models to create a dataset of operational situations. This dataset is then used as a basis to try to predict the failure of an aquaculture operation.

Paper I (Mathisen 2016)

Title:
Data driven case base construction for prediction of success of marine operations

Published at conference:
Case-Based Reasoning and Deep Learning Workshop - CBRDL-2017

Following up on the work in Paper I, a function that combined the level of exposure of an aquaculture site and the weather at the time of an operation was missing. This led to applying SNNs to embed and combine such features into an embedding space (see Figure 2.4). The next paper presents an Extended SNN (ESNN) architecture to model which differences of these two embeddings are most important to accurately calculate the similarity between two data points, in

addition to the regular SNNs that extract which features (and non-linear combinations of features) of the two data points are important for calculating their similarity. To illustrate the performance of ESNN, we presented results from testing the performance ESNNs on 14 different datasets.

Paper II (Mathisen 2019)

Title:
Learning similarity measures from data
Publised in journal:
Progress in Artificial Intelligence

The third paper presents work on applying facial recognition research on salmon. A dataset with pictures of individual salmon was created from a video stream of salmon swimming in an aquaculture cage. Bounding boxes of salmon heads were extracted from each of the video frames using YOLOv3 [82] and clustered according to individuals using DBSCAN [83] combined with a custom distance metric. This dataset was then used for training of a fishnet, inspired by FaceNet[30]. Fishnet was tested with three different neural network architectures for computing embeddings (VGG-16, Mobilenet v2 and Inception ResNet v2) and trained using triplets. Fishnet achieved a 96,4% true positive rate (TPR) at 1% false positive rate (FPR).

Paper III (Mathisen 2020)

Title:
FishNet: A Unified Embedding for Salmon Recognition
Publised at conference:
Twenty-fourth European Conference on Artificial Intelligence (ECAI), tenth Conference of Prestigious Applications of Artificial Intelligence

The last paper combines the data collected in Paper I with the ESNN method developed in Paper II to develop the foundations of a DSS on a CBR system using ESNN as a similarity function. The paper demonstrates how ESNN enables decision support for aquaculture operations. We train an ESNN on operational data captured from aquaculture sites to calculate similarity according to whether or not an operation was successful. This ESNN was then used as a similarity function to retrieve the most similar previous operations in response to a query. This design was evaluated quantitatively and qualitatively.

Paper IV (Mathisen 2021)

Title:
Using Extended Siamese Networks to Provide Decision Support in Aquaculture Operations
Publised in journal:
Applied Intelligence

In Figure 4.1 we show how the publications listed above relate to each other. Chapter 3 is used as the basis to establish how DSS systems are constructed and used in aquaculture. This systematic literature mapping also showed that ML is not yet widely adopted in aquaculture. In Paper I, we mapped different data sources together to create a dataset to use as a basis for a case base. Furthermore, the investigations of how to combine the exposure level of each site with the weather at the time and location of each operation led to the method developed in Paper II. The knowledge about SNNs and how they work was applied in the development of fishnet in Paper III. Finally, in Paper IV, the data sources and framework for combining them presented in Paper I was reused to create a dataset for operational situations in aquaculture. Results from all previous papers were used when applying ESNN to prototype a CBR-based DSS for operational situations in aquaculture.

Figure 4.2 shows how the different research questions relate to the contributions produced by this PhD project. Research question 1 "How can machine learning fulfill the requirements set by aquaculture to decision support systems?" relates to the systematic literature mapping done in Chapter 3 (C1), the framework to combine data sources from aquaculture sites (C2) and the framework to analyze similarity functions (C3). Research question 2, "What types of machine learning methods have been used in decision support systems within the aquaculture application domain?" is answered by the systematic literature mapping (C1). Research question 3 "Can similarity learning be used as a machine learning method to enable CBR for decision support systems?" is answered by the framework for comparing similarity functions (C3), ESNN (C4), Fishnet (C5) and demonstrating that ESNN outperforms SNNs (C6). Research question 4, "Can similarity learning methods be developed further to improve performance?" is answered by demonstrating that ESNNs outperform SNNs (C5).

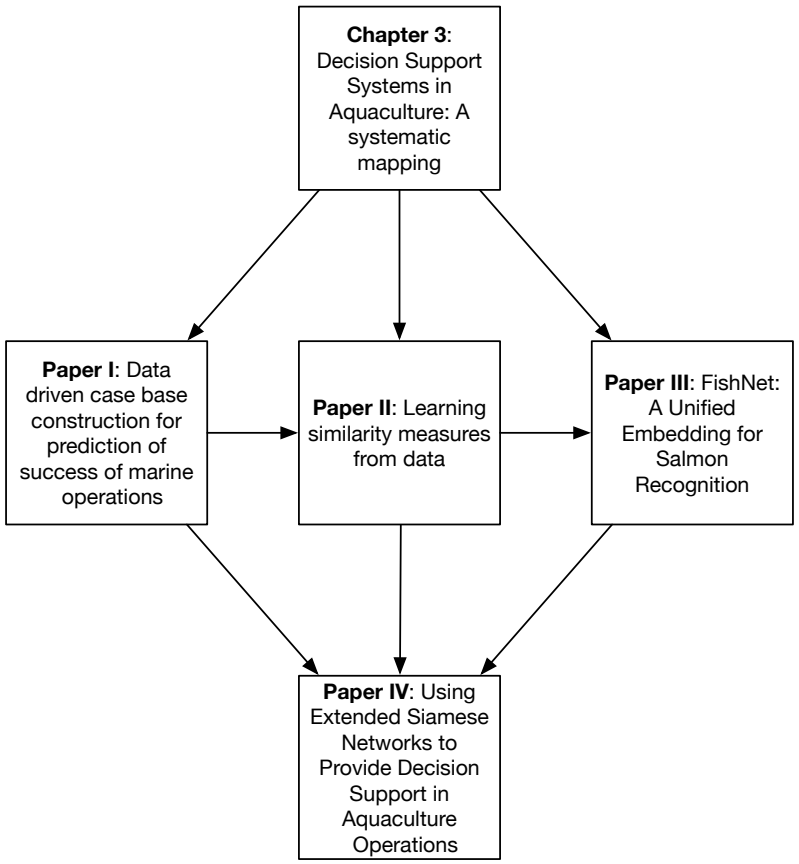


Figure 4.1: The relation between the research outputs of the PhD thesis. Chapter 3 was used as input for all publications. Methods and data from Paper I was used in Paper IV and inspired Paper II. Knowledge about SNNs from Paper II was used for Paper III. Paper IV used knowledge and methods developed in all previous publications.

4.3 Contributions towards research questions

The following section summarizes the contributions made towards each research question.

RQ1: How can we use ML to make useful DSSs for aquaculture that are explainable and work in a data-scarce domain? The systematic mapping presented in Chapter 3 could only find one study that used ML as part of a DSS for aquaculture. The study by Magno et al. [72] applied ML to predict parameters

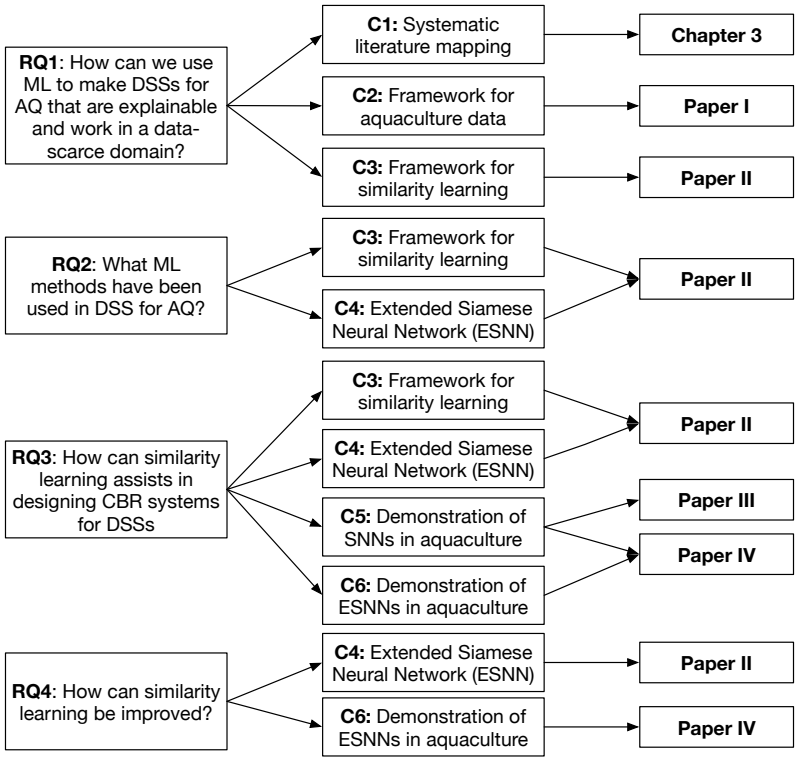


Figure 4.2: The four research questions and how they are answered by the six different contributions presented in this thesis. The contributions are then linked to the four papers and the literature mapping in chapter 3 that constitutes the contributions.

related to the welfare of the fish (e.g. salinity, temperature, and oxygen levels), and shows one way of applying ML in the context of DSS systems. In general, a DSS can use the current state of the data related to the DSS as input. This input would then be used by a ML model to predict features that are important in the DSS context (such as oxygen levels for fish). The aquaculture industry is lagging behind other industries in terms of the technical level of digitalization, such as data gathering, data quality, and data labeling. At the same time, the industry is also lagging behind on non-technical level of digitalization. General trust and thus reliance on IT systems are low. This means that any ML practitioner needs to apply methods that can work with small datasets - while also creating trust in the user through intuitive explanations. Frameworks like LIME [8] and

SHAP [9] may not provide an explanation that provides the necessary intuition for users that are not ML experts. More specifically, the user of the system needs to be able to interpret the output of the model.

CBR uses previous instances or situations to learn and can thus expand on methods such as LIME and SHAP by explaining by example. The users of a DSS get a recommendation "Next week's operation should be performed in the same way as the operation you did three weeks ago because it is the most similar operation in terms of operational parameters and external factors" and also displays the previous operation. This leaves the user to reflect upon why these two operations are similar according to the system. Specifically, the user would need to understand the similarity using their domain expertise, not the details of the underlying machine learning method. This could be expanded to present the user with not only the most similar case but the n most similar cases. The user could also report errors and enable the CBR-based DSS to learn over time.

Paper I demonstrates how to augment and enrich data gathered from aquaculture. Aquaculture is, still, a data-poor application domain in comparison to other domains such as oil or energy. In this paper, we demonstrated how to augment the aquaculture data with relevant data sources such as ocean and weather models. Finally, we demonstrated how dependent modeling operations aquaculture sites are on local conditions and augment the site data with data describing the local conditions of each site.

In Paper IV we showed that a DSS based on CBR could be designed and implemented using ESNN as a similarity learning tool. In this work, we showed that ESNN could learn a similarity function from a relatively small dataset ($\tilde{8}00$ data points). We also showed that this similarity function performed well in terms of retrieving relevant cases. In addition, ESNN created functions that resulted in a more smooth transition in the calculated similarity between irrelevant cases and the most relevant case. This enables the user to not only two sets of cases, one containing irrelevant cases and one containing relevant cases. Thus a user of a DSS that employs a similarity function learned by ESNN could present a user with a set of very similar cases in addition to the most similar case.

RQ2: What types of machine learning methods have been used in decision support systems within the aquaculture application domain?

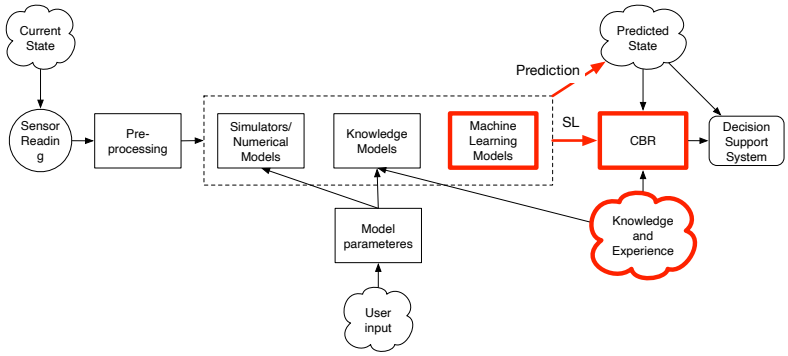


Figure 4.3: This figure shows the general DSS architecture introduced in Figure 1.1 modified to highlight the two different ways ML can be applied in this DSS architecture. ML models are mainly used for predictive modeling in DSSs. These models produce a predicted state that can interpreted directly by the user (like water temp or salinity for aquaculture). Parts of the same predicted state could also be used to query a CBR system to see if any predicted problematic state was similar to any recorded states. The other way of applying ML in DSS is to train models to calculate this similarity between recorded cases/situations (shown as SL in the figure).

The systematic mapping study in Chapter 3 only found one study on DSS that applies ML in the aquaculture industry. Very little research on DSS in aquaculture has been done, and there is a lack of empirical evaluation of these systems. Another concern raised by the study indicates that there exist few to no well-documented multi-criteria DSS systems for aquaculture. Our findings suggest that as the digitalization and data capture increase in the aquaculture industry, there is a great potential for applying machine learning to support DSSs in the industry. This will help ground the DSSs in real data, and when deployed, in real-time data feeding into the systems. One could even design systems that continuously re-trains the models with on-site data to increase overall test performance.

We still don't see large-scale adoption of ML for DSSs in the aquaculture industry because of the lack of data needed to create the ML systems.

When digitalization increases, we will see more use of ML in DSSs for aquaculture.However, successful use of ML in aquaculture is not only dependent on how much data the aquaculture industry gathers, but also on the quality of that data and how well the practitioners annotate and describe it.

ML models can contribute to DSSs in aquaculture in many ways,

such as directly optimizing control or predicting the effect of actions (through reinforcement learning). However, in Figure 4.3 we illustrate two ways ML can be used in a DSS. The first way is prediction (illustrated by "Prediction" in Figure 4.3). Through predicting a state that describes the problem that concerns the DSS the user can use the DSS to plan. This state is typically parameters describing important aspects of the problem being addressed by the DSS. For aquaculture, this could typically be salinity, water temperature, and water current. This future state could also be used to query a CBR system to retrieve previous past experiences that are similar to this predicted state. The second way is to use SL to model similarity between two situations (illustrated by "SL" in the figure). Typically the SL model would use a predicted state to retrieve a set of previously recorded situations that are most similar. This set of previously recorded situations would then support the DSS user in choosing the best decision.

The study [72] we did find in the literature mapping falls within the former category. In this work, they used ML to predict parameters that were important to the system described by the DSS.In this case, the crucial parameters were oxygen levels, salinity, and water temperature. These parameters are all important in terms of aquaculture production, and thus a good prediction of the parameters can be a substantial addition to the information presented to the DSS user. This is also the case with other important parameters of aquaculture production, such as the physical or mechanical situation of an aquaculture site. As motivated in Section 1.2 this is even more important in DSSs that are used for EXPOSED aquaculture operations. Thus predicting environmental factors such as wind, waves and currents can also be useful for planning operations on aquaculture installations.

In an early stage of this PhD project, we implemented and evaluated models to predict the effects of these environmental factors such as movement in installations. Figure 4.4 shows an experiment where data gathered from a buoy close to an exposed aquaculture location Figure 4.5 shows the accuracy of two predictive ANN models. The first model, "4" uses 4 steps of history as input to predict a future timestep. The second model, "6" uses 6 timesteps as input. The figure shows how the accuracy of predictions decreases when the model is trained and tested to predict further into the future.

Predicting the effects of environmental factors instead of directly predicting the environmental factors have the added benefit that such a model would inherently have to take into account the local conditions. Wind of the same strength and from the same direction would not have the same effect on different aquaculture installations, as the local conditions are different.

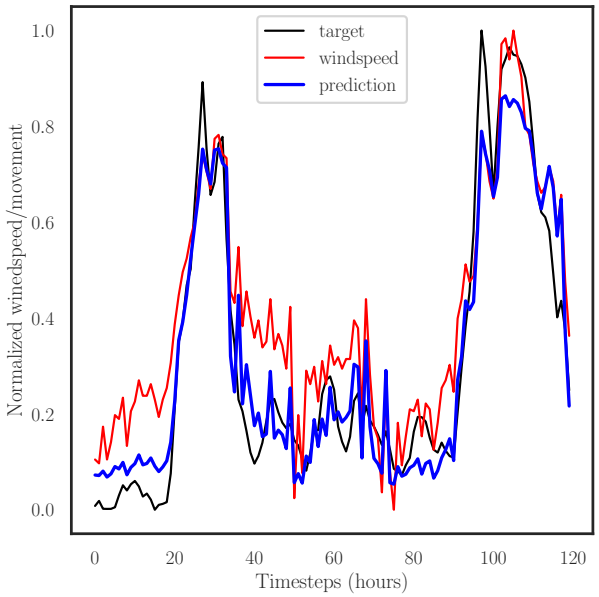


Figure 4.4: A graph showing a neural network predicting (blue line) the variation on the x-axis of an accelerometer mounted on the aquaculture cage (black line). This variation of the accelerometer values would be indicative of how much the cage is moving and how hard it would be to stand on the cage platform. All values are shown over 120 hours. All values on the graph are normalized from 0 to 1 (with original wind speed ranging from 0.82 m/s to 15.32 m/s and target/prediction of change in acceleration ranging from 0.003 $\frac{m}{s^3}$ to 0.042 $\frac{m}{s^3}$)

As discussed earlier in this section and illustrated in Figure 4.3, ML models can also be used to learn similarity (SL). This can be used to retrieve recorded situations that are similar to the predicted or current situations. This will be detailed in the next section.

RQ3: How can similarity learning assists in designing CBR systems for DSSs? Similarity learning makes the creation of CBR systems much less dependent on domain experts, as the similarity function does not need to be created manually. Modeling the similarity function using data that describes the problem being solved by the DSS reduces the design decisions down to what features need to be part of that data. DL models can automate this by learning an accurate representation [29] of the data. This enables an SNN to

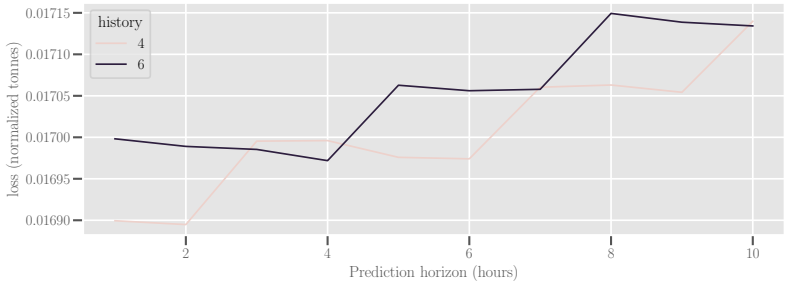


Figure 4.5: A graph showing a neural network predicting the anchor load (loss is normalized with 0 being equivalent to 21.454 kg and 1 being equivalent to 469.48 kg) of an aquaculture cage where the horizontal axis is the amount of hours forward in time for the prediction. The graph shows two different neural networks, one which takes into account four hours of history to predict, and one who uses six hours as input data.

determine which features of the data are most important in terms of measuring similarity.

The work done in Paper IV demonstrates an application of ESNN in the aquaculture domain. There we retrieve past operational situations, characterized by the location and the weather effect (effect of the given weather at a specific location). The system presents the user with a list of previous operational situations that is sorted by how similar they are with the planned or current situation. That way, the user is informed of the way prior users handled previous situations. Figure 4.6 shows the validation loss of ESNN and two reference similarity learning methods (*chopra*[37] and *gabel*[38]). ESNN outperforms the two reference similarity methods. ESNN also learns the model quickly, almost as quickly as a standard SNN (*chopra*). In many cases, SNNs have a head start in terms of accuracy as SNNs compute the final similarity resulting from the two embeddings using a static distance metric. In comparison, ESNN has to learn the function for the similarity computation based on the two embeddings.

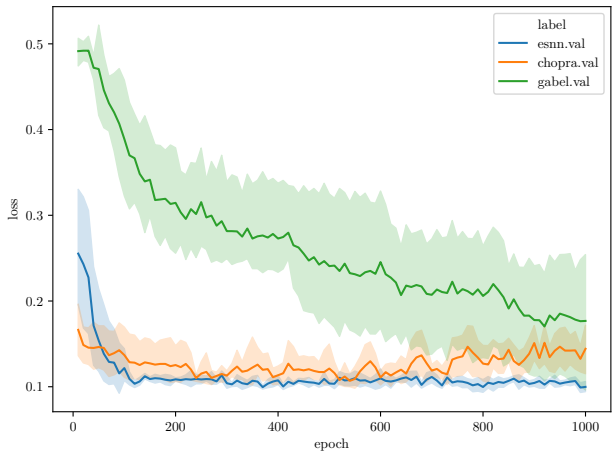


Figure 4.6: Validation results across 1000 epochs for the different similarity functions for comparing different aquacuture operational situations. *chopra.val* is the validation performance of *chopra*[37] and *gabel.val* is the valdiation performane of *gabel*[38]. Finally, *esnn.val* is the validation performance of ESNN. ESNN converges almost as quick as SNNs (*chopra.val*) and then quickly outperforms *chopra.val*. *gabel* converges much slower, as it has to learn the ordering of the datapoints ($S(\boldsymbol{x}, \boldsymbol{y}) = S(\boldsymbol{y}, \boldsymbol{x})$ is not a part of the architecture design as in *chopra* or ESNN). However, *gabel.val* does converge to the same accuracy as *esnn.val* given enough epochs.

In the Fishnet architecture shown in Paper III we apply SL in a more direct way. Here the ML model is used as a similarity function to measure the likeness of salmon individuals within one aquaculture cage. Re-identifying individual salmon enables a DSS to build statistics for growth and disease for individual salmon. Individual statistics and prediction of growth and disease are very beneficial for optimization of production. An example of this can be seen in Figure 4.7 where three different fish (per row) are shown with two different pictures of each fish (columns). The similarity of the images of salmon is shown between the salmon pictures, and all of the pictures of the same salmon have higher similarity than to other salmon. In this work, we trained a SNN based on Facenet[30] to re-identify salmon. We trained the SNN architecture using a dataset of labeled salmon heads. This dataset contained 15000 images of 715 individual salmon. We applied five augmentations to each of the salmon heads and ended up with 225000 images. A validation test performed was done on a test-set of 6000 images from 29 different salmon. This test showed

that the fishnet architecture has good performance in re-identifying salmon with a 96.4% true positive rate given a 1% false positive rate using the Inception ResNet v2 as an embedding network.

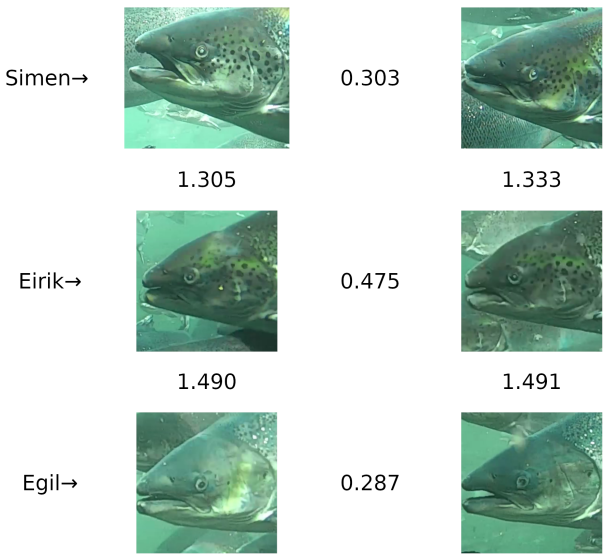


Figure 4.7: An illustration of the distances between six images from salmon with three different identities. Each row contains two images of the same salmon: "Simen" at the top, "Eirik" in the middle and "Egil" at the bottom. The average distance between the same salmon is 0.36 while comparisons between different salmon average at 1.40.

RQ4: How can similarity learning methods be developed further to improve performance?

This research question asks how we can augment the design of SNNs to increase performance.

Siamese Neural Networks were introduced by Bromley et.al [48] to measure the similarity of two signatures to detect signature fraud. At the time, SNNs were not adopted by CBR systems to learn similarity functions.

SNNs and other similarity functions enable users to calculate the similarity between any pairs of objects. It can be used for different applications, including fraud detection [48], face verification [37], gesture classification [84] and as part of matching architectures [28].

The SNN architecture can also be expanded to train on triplets of data points instead of pairs like in the work done by Lefebvre et al. [85] and Hoffer et al. [34].

SNNs learn to embed data points into an embedding space where two data points that are labeled as similar are closer than in non-embedding space. The SNNs extract the parts of the data points that are most important for calculating the similarity. After doing this embedding on two data points, SNNs use a static distance metric like L2 or manhattan distance to calculate the distance in embedding space. Through the work described in Paper II, we designed an extension to the SNN architecture called extended siamese neural networks (ESNN). In addition to extracting the most important parts of a data point and using that as a basis for embedding, ESNN has an additional step in similarity calculation. Instead of calculating the similarity between the two embedding points via a static function, ESNN uses a neural network to calculate the similarity. To do this, ESNN calculates the distance between the two embeddings and the absolute value of the resulting vector. This vector is then used as input to a neural network that calculates the similarity. The ESNN architecture is end-to-end trainable as the error of the similarity calculation is the basis for backpropagation back through the similarity network and the embedding network.

This means that an SNN can learn a function to extract and embed the most important parts for each data point in terms of similarity. ESNN can use that same embedding to also compute which differences of the two embeddings are most important to compute similarity. This enables ESNN to learn ways to compute similarity based on both datapoint, not only indirectly through a static function and the computed loss, but also directly. In our experiments, this enables the ESNN to compute accurate similarity on datasets that are harder to separate (e.g., the mammography dataset from UCI ML repository[86]) compared to SNNs. This can be seen in Figure 4.8 which shows the training loss for ESNN, a SNN (*chopra*), and *gabel* (a SL method by developed Gabel [38] introduced in Section 2.4). Similar accuracy can also be achieved by the *gabel* method for learning similarity. This is because the *gabel* method concatenates both data points into a single data vector before doing similarity computation. As a result, *gabel* sees both data points while learning similarity in the same way ESNN does. This does, however, introduce more complexity in training as the ordering of the two data points affects the output (*gabel* is not invariant to input pair ordering), in contrast to SNNs and ESNN. Figure 4.9 shows training performance for a *chopra* and *gabel* and ESNN. The figure shows that while the performance increases slowly for the *gabel* method, the SNN and ESNN methods reduces

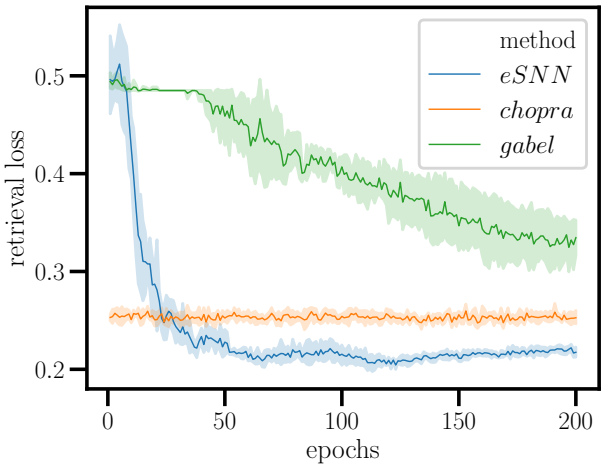


Figure 4.8: Validation retrieval loss during training on the Mammographic mass UCI ML dataset [86].

the loss during training much faster. Finally, Figure 4.10 shows the validation loss during training on the balance UCI ML dataset[86] as a function of how many data points are seen by the training method. ESNN and SNNs have to evaluate fewer data points because both of these architectures are invariant to the ordering of the two input data points ($\mathbb{S}(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{S}(\boldsymbol{y}, \boldsymbol{x})$). For the architecture used by *gabel* this is not the case, and this model has to train on more data points to learn the same similarity function. To summarize, we can see that the experiments done in Paper II shows that architectures based on SNNs (invariant to input datapoint ordering) are faster to train. On the other hand, the same experiments show that architecture that can perform learning on the part of the architecture that sees both data points (as is the case with *gabel* and ESNN) outperforms architectures that only see single datapoints while learning (SNNs) on hard problems such as seen in Figure 4.8. Our experiments show that ESNN is a SL method that combines fast training with the ability to achieve high accuracy on hard problems.

4.4 Summary of auxiliary papers

In addition to the main publications listed in the previous section, the PhD work done in the project has also contributed to other publications. These publications are not directly connected to the

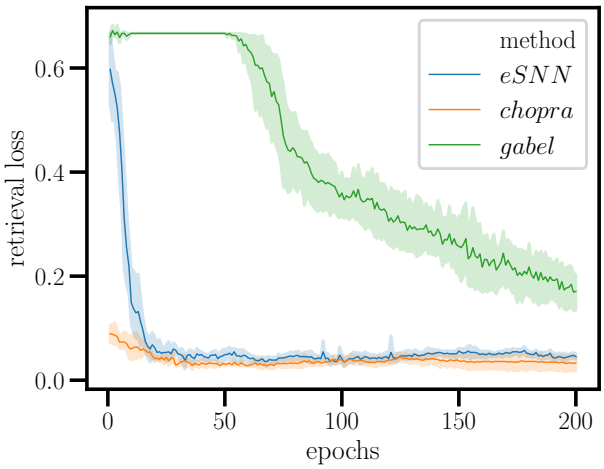


Figure 4.9: Validation retrieval loss during training on the Iris UCI ML dataset [86].

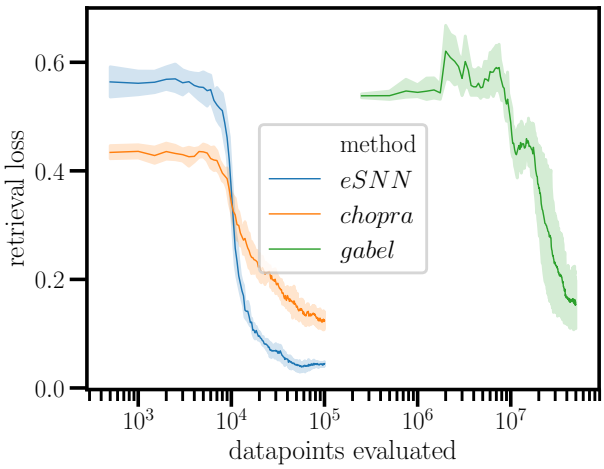


Figure 4.10: Validation retrieval loss during training on the balance UCI ML dataset [86], which illustrates the difference in amount of evaluations needed to achieve acceptable performance.

research topic. The work described in these auxiliary publications and the main publications of the PhD thesis share methods in that they study CBR and ML as a central part of a decision support system. These auxiliary papers are as such part of the same methodological theme as the main papers described earlier. These publications added indirectly to the contributions of this thesis.

Appendix A Leendert Wilhelmus Marinus Wienhofen and Bjørn Magnus Mathisen. "Defining the initial case-base for a CBR operator support system in digital finishing". In: *Goel, Ashok; Díaz-Agudo, M Belén; Roth-Berghofer, Thomas (Ed.): Case-Based Reasoning Research and Development - 24th International Conference, ICCBR 2016*, Atlanta, GA, USA, October 31 - November 2, Proceedings, pp. 430444, Springer, 2016.
In this paper, we described the initial design and prototype implementation of a CBR-based operator support system.

Appendix B Kerstin Bach and Bjørn Magnus Mathisen and Amal Jaiswal.**Demonstrating the myCBR Rest API**. In *Kapetanakis, Stelios; Borck, Hayley (Ed.): Workshops Proceedings for the Twenty-seventh International Conference on Case-Based Reasoning co-located with the Twenty-seventh International Conference on Case-Based Reasoning (ICCBR 2019)*, Otzenhausen, Germany, September 8-12, 2019, pp. 144155, CEUR-WS.org, 2019.
A paper detailing the expansion of myCBR with a REST interface and demonstrating how this increases usefulness for practitioners, researchers, CBR students, and teachers.

Appendix C Jørn Foros and Maren Istad and Andrei Z. Morch and Bjørn Magnus Mathisen. **Use Case applying machine-learning techniques for improving operation of the distribution network**. In: *25th International Conference on Electricity Distribution (CIRED 2019)*, Madrid, Spain, June 3-6, 2019, paper 2114, ISSN 2032-9644, AIM, 2019.
This paper describes a use case that illustrates how machine learning and CBR can be used as a decision support system for power distribution networks.

4.5 Source code

In addition to scientific progress and publications, the PhD work has produced significant source code to run experiments. The main contributions are summarized in the following.

- Extended siamese neural network and surrounding test suite, implemented in Keras 2 and tensorflow 2 - Used in paper Paper II
  https://github.com/ntnu-ai-lab/esnn
- New version of Extended siamese neural network and surrounding test suite - Used in paper Paper IV.
  https://github.com/ntnu-ai-lab/esnn-aqcbr
- As part of the work paper Appendix B - Expansion of features in the myCBR CBR implementation https://github.com/ntnu-ai-lab/mycbr-sdk and co-creation of the REST API of myCBR https://github.com/ntnu-ai-lab/mycbr-rest
- As part of the work done to replicate the similarity learning method from in paper Paper II https://github.com/ntnu-ai-lab/RProp

Chapter 5

Conclusion

This chapter summarizes the main research contributions of this PhD project as they relate to the research questions defined in Section 1.3. Furthermore, future research directions that build on the work conducted throughout this thesis are included.

5.1 Conclusion

This thesis has presented the work done in the PhD project. Through the EXPOSED SFI center, this PhD project set the goal of developing enabling digital technologies for the aquaculture industry. More specifically, to provide tools to increase the level of automation in the aquaculture industry. This would enable the industry to expand into more exposed locations without the increased risk or loss of production. Data-driven methods and machine learning models are especially promising for providing such automation. At the same time, the aquaculture industry has two fundamental challenges that limit the adoption of new technologies, namely low digital maturity and lack of high-quality data.

In this thesis, we have investigated if machine learning can contribute towards the goal of more automated and optimized aquaculture operations by enabling data-driven DSSs. The thesis explored developing ML models for DSS based on both traditional data collection (buoy data seen in Figure 4.4) and more novel data usage as illustrated by Fishnet in Paper III. Chapter 3 presented a systematic literature mapping where we discovered that there was very little research on using machine learning for DSSs in aquaculture. This shows there is a need for doing more research on this topic.

We argue that the aquaculture industry requires an intuitively explainable DSS to get true adoption in the aquaculture industry due to the low digital maturity and lack of trust in digital tools. CBR represents such a solution. A CBR system can retrieve previously recorded situations that are similar to the current or predicted situation and present them to the user. This could include the recorded solution to that situation.

However, designing CBR systems can be demanding as encoding knowledge/expertise is non-trivial in most cases. This is especially true for the similarity function. The similarity function computes the

similarity between the input and the previously recorded situations. Designing this function typically requires deep domain expertise. In our work, we investigated how to learn a similarity function from gathered data. Through this investigation, we showed that similarity learning could help make DSSs for aquaculture be less dependent on design and domain expertise. This way, DSS designers can easier adapt DSSs to the local conditions of each aquaculture site without redesigning the DSS.

Basing DSS design for aquaculture on machine learning methods enables the design to stay constant across aquaculture sites. The DSS users and designers would only need to retrain the ML models used by the DSS. In Paper IV we also showed that it is possible to improve current methods for similarity learning to fit in such a solution.

In Paper II we extended the Siamese neural network architecture. SNNs extract the most important parts of each data point into embeddings and learn which parts of each data point are essential for similarity. In designing ESNN, we extended this to learn the essential parts of the differences between those two embeddings.

We demonstrated that learning to model the similarity of two embeddings, not only embedding the two data points, are important to achieve good performance on datasets that have been shown hard to classify.

The aquaculture industry is important for Norway, and moving more aquaculture sites to exposed locations demand more automation and optimized operations. The work presented in this thesis have both shown promising results in terms of which methods to use for CBR systems for DSS in aquaculture, as well as providing important guidance into which data to collect in order to improve DSS in aquaculture further.

This thesis introduced four research questions in Section 1.3. In Section 4.3 we demonstrated how the work done throughout this PhD project has answered these research questions. Through this work, we illustrated how SL could be used to create DSSs for a data-scarce domain such as aquaculture. We did not evaluate our experimental system (AQCBR from Paper IV) in actual deployment. However, our experiments based on real data illustrated the viability. We also introduced a new SL method which we named extended siamese neural networks (ESNN). Our experiments showed that ESNN has the accuracy of a SL method like *gabel* while retaining the training speed of SNNs. We did not extend testing of ESNN to larger datasets (except for MNIST), nor to sequence similarity learning. However, in accordance with our experiments and the architecture, we see no reason that the performance should not be similar relative to SNNs for larger or sequential datasets.

5.2 Future research directions

This PhD project was done in an applied context. As a result, many of the directions of future investigations are within testing the technologies developed in this PhD project in operational systems rather than simulated environments or testing based on historical data. However, there are also methodological results from the PhD project. As a result, there are future directions of investigation that center around the ESNN method and how it performs in other tasks that are suitable for SL/DML/CL methods.

How CBR-based DSSs can help data-scarce industries such as aquaculture: Data gathering is increasing in aquaculture, but high-quality labeled data is still far between. Applying CBR as part of a DSS for supporting aquaculture industry operations could have an impact on how the industry can use ML-based technologies. However, the effect of how presenting the most relevant recorded case when facing planning tasks on actual operations is still unknown. To evaluate this, we would need to test systems such as AQCBR presented in Paper IV in actual operations, then evaluate its effect on operations through metrics and questionnaires.

How SL could benefit DSS for aquaculture via salmon re-identification: As shown by Paper III SL could also be applied more directly in DSS for aquaculture by identifying individuals. This would enable the operator of an aquaculture site to make decisions based on individual-level data. How to best make use of re-identification in such a use case would need to be studied more and evaluated in cooperation with industry stakeholders.

In what type of applications is the added model capacity of Extended Siamese Neural Networks (ESNN) needed: The work presented in Paper II and IV shows that ESNN does outperform other similarity learning methods on some types of tasks. However, SL can be used for many other types of tasks than retrieval/classification of unlabeled data points. SL can also be used to cluster unlabeled data, e.g., using a pre-trained ESNN for clustering a related but unlabeled dataset. This could have benefits above pure unsupervised methods. ESNN could also be used as a custom metric in combination with methods such as DBSCAN (as done in Paper III, but with a pre-trained ESNN instead of a static custom metric).

References

[1] Olafsen, T., Winther, U., Olsen, Y., and Skjermo, J. "Value created from productive oceans in 2050". In: *SINTEF Fisheries and Aquaculture* (2012), p. 83.

[2] Aamodt, A. and Plaza, E. "Case-based reasoning: Foundational issues, methodological variations, and system approaches". In: *AI communications* vol. 7, no. 1 (1994), pp. 39–59.

[3] Holen, S. M., Utne, I. B., Holmen, I. M., and Aasjord, H. "Occupational safety in aquaculture–Part 1: Injuries in Norway". In: *Marine Policy* vol. 96 (2018), pp. 184–192.

[4] Holen, S. M., Utne, I. B., Holmen, I. M., and Aasjord, H. "Occupational safety in aquaculture–Part 2: Fatalities in Norway 1982–2015". In: *Marine Policy* vol. 96 (2018), pp. 193–199.

[5] Mathisen, B. M., Bach, K., Meidell, E., Måløy, H., and Sjøblom, E. S. "FishNet: A unified embedding for salmon recognition". In: *Proceedings of the Twenty-fourth European Conference on Artificial Intelligence*. 2020, pp. 3001–3008.

[6] Mathisen, B. M., Aamodt, A., and Langseth, H. "Data driven case base construction for prediction of success of marine operations". In: *Proceedings of ICCBR 2017 Workshops (CAW, CBRDL, PO-CBR), Doctoral Consortium, and Competitions co-located with the 25th International Conference on Case-Based Reasoning (ICCBR 2017)*. 2017, pp. 102–111.

[7] Mathisen, B. M., Bach, K., and Aamodt, A. "Using extended siamese networks in a CBR system to provide decision support in aquaculture operations". In: *Applied Intelligence* (2021).

[8] Ribeiro, M. T., Singh, S., and Guestrin, C. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 2016, pp. 1135–1144.

[9] Lundberg, S. and Lee, S.-I. "A unified approach to interpreting model predictions". In: *arXiv preprint arXiv:1705.07874* (2017).

[10] Krizhevsky, A., Sutskever, I., and Hinton, G. E. "Imagenet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. Vol. 25. 2012, pp. 1097–1105.

[11] Lannan, J. "Users Guide to PONDCLASS: Guidelines for Fertilizing Aquaculture Ponds". In: *Pond Dynamics/Aquaculture CRSP, Oregon State University, Corvallis, Oregon* (1993).

[12] Power, D. J. "Decision support systems: a historical overview". In: *Handbook on Decision Support Systems 1*. 2008, pp. 121–140.

[13] Garg, A. X., Adhikari, N. K., McDonald, H., Rosas-Arellano, M. P., Devereaux, P., Beyene, J., Sam, J., and Haynes, R. B. "Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: a systematic review". In: *Jama* vol. 293, no. 10 (2005), pp. 1223–1238.

[14] Hunt, D. L., Haynes, R. B., Hanna, S. E., and Smith, K. "Effects of computer-based clinical decision support systems on physician performance and patient outcomes: a systematic review". In: *Jama* vol. 280, no. 15 (1998), pp. 1339–1346.

[15] Golshani, A., Sun, W., Zhou, Q., Zheng, Q. P., and Tong, J. "Two-stage adaptive restoration decision support system for a self-healing power grid". In: *IEEE Transactions on Industrial Informatics* vol. 13, no. 6 (2017), pp. 2802–2812.

[16] Mardle, S. and Pascoe, S. "A review of applications of multiple-criteria decision-making techniques to fisheries". In: *Marine Resource Economics* (1999), pp. 41–63.

[17] Ernst, D. H., Bolte, J. P., and Nath, S. S. "AquaFarm: simulation and decision support for aquaculture facility design and management planning". In: *Aquacultural Engineering* vol. 23, no. 1 (2000), pp. 121–179.

[18] Kolodner, J. L. "An introduction to case-based reasoning". In: *Artificial intelligence review* vol. 6, no. 1 (1992), pp. 3–34.

[19] Schank, R. C. "Language and memory". In: *Cognitive science* vol. 4, no. 3 (1980), pp. 243–284.

[20] Schank, R. C. *Dynamic memory: A theory of reminding and learning in computers and people*. 1983.

[21] Kolodner, J. L. "Reconstructive memory: A computer model". In: *Cognitive science* vol. 7, no. 4 (1983), pp. 281–328.

[22] Richter, M. M. “Knowledge containers”. In: *Readings in Case-Based Reasoning* vol. Morgan Kaufmann Publishers (2003).

[23] Cybenko, G. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* vol. 2, no. 4 (1989), pp. 303–314.

[24] Zhou, D.-X. “Universality of deep convolutional neural networks”. In: *Applied and Computational Harmonic Analysis* vol. 48, no. 2 (2020), pp. 787–794.

[25] Hochreiter, S. and Schmidhuber, J. “Long short-term memory”. In: *Neural computation* vol. 9, no. 8 (1997), pp. 1735–1780.

[26] Fukushima, K. “Neural network model for a mechanism of pattern recognition unaffected by shift in position-Neocognitron”. In: *IEICE Technical Report, A* vol. 62, no. 10 (1979), pp. 658–665.

[27] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* vol. 1, no. 4 (1989), pp. 541–551.

[28] Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. “Matching networks for one shot learning”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 3630–3638.

[29] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*. Vol. 1. 2. 2016.

[30] Schroff, F., Kalenichenko, D., and Philbin, J. “Facenet: A unified embedding for face recognition and clustering”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[31] Valmadre, J., Bertinetto, L., Henriques, J., Vedaldi, A., and Torr, P. H. “End-to-end representation learning for correlation filter based tracking”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2805–2813.

[32] Gao, P., Zhang, Q., Wang, F., Xiao, L., Fujita, H., and Zhang, Y. “Learning reinforced attentional representation for end-to-end visual tracking”. In: *Information Sciences* vol. 517 (2020), pp. 52–67.

[33] Gao, P., Yuan, R., Wang, F., Xiao, L., Fujita, H., and Zhang, Y. “Siamese attentional keypoint network for high performance visual tracking”. In: *Knowledge-Based Systems* vol. 193 (2020), p. 105448.

[34] Hoffer, E. and Ailon, N. “Deep metric learning using triplet network”. In: *International Workshop on Similarity-Based Pattern Recognition*. Springer. 2015, pp. 84–92.

[35] Kelley, H. J. “Gradient theory of optimal flight paths”. In: *Ars Journal* vol. 30, no. 10 (1960), pp. 947–954.

[36] Linnainmaa, S. “The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors”. In: *Master’s Thesis (in Finnish), Univ. Helsinki* (1970), pp. 6–7.

[37] Chopra, S., Hadsell, R., and LeCun, Y. “Learning a similarity metric discriminatively, with application to face verification”. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 539–546.

[38] Gabel, T. and Godehardt, E. “Top-down induction of similarity measures using similarity clouds”. In: *Case-Based Reasoning Research and Development*. Ed. by Hüllermeier, E. and Minor, M. Cham, 2015, pp. 149–164.

[39] D. J. Montana, L. D. D. “Training feedforward networks using genetic algorithms”. In: *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*. 1989, pp. 762–767.

[40] Kitano, H. “Designing neural networks using genetic algorithms with graph generation system”. In: *Complex Systems* vol. 4, no. 4 (1990), pp. 461–476.

[41] Stanley, K. O. “Efficient evolution of neural networks through complexification”. PhD thesis. The University of Texas at Austin, 2004.

[42] Stahl, A. “Learning feature weights from case order feedback”. In: *Proceedings of the 4th International Conference on Case-Based Reasoning (ICCBR 2001)*. Springer. Vancouver, 2001, pp. 502–516.

[43] Nikpour, H., Aamodt, A., and Bach, K. “Bayesian-supported retrieval in BNCreek: A knowledge-intensive case-based reasoning system”. In: *International Conference on Case-Based Reasoning*. Springer. 2018, pp. 323–338.

[44] Stahl, A. and Gabel, T. “Using evolution programs to learn local similarity measures”. In: *Proceedings of the 5th International Conference on Case-Based Reasoning (ICCBR 2003)*. Trondheim, 2003, pp. 537–551.

[45] Langseth, H., Aamodt, A., and Winnem, O. M. "Learning retrieval knowledge from data". In: *Sixteenth International Joint Conference on Artificial Intelligence, Workshop ML-5: Automating the Construction of Case-Based Reasoners. Stockholm*. Citeseer. 1999, pp. 77–82.

[46] Reategui, E. B., Campbell, J. A., and Leao, B. F. "Combining a neural network with case-based reasoning in a diagnostic system". In: *Artificial Intelligence in Medicine* vol. 9, no. 1 (1997), pp. 5–27.

[47] Abdel-Aziz, A., Strickert, M., and Hüllermeier, E. "Learning solution similarity in preference-based CBR". In: *Proceedings of the 22nd International Conference on Case-Based Reasoning (ICCBR 2014)*. Springer. 2014, pp. 17–31.

[48] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. "Signature verification using a" siamese" time delay neural network". In: *Advances in neural information processing systems*. 1994, pp. 737–744.

[49] Maggini, M., Melacci, S., and Sarti, L. "Learning from pairwise constraints by similarity neural networks". In: *Neural Networks* vol. 26 (2012), pp. 141–158.

[50] Mathisen, B. M., Aamodt, A., Bach, K., and Langseth, H. "Learning similarity measures from data". In: *Progress in Artificial Intelligence* (Oct. 2019), pp. 129–143.

[51] Ye, X., Leake, D., Huibregtse, W., and Dalkilic, M. "Applying class-to-class siamese networks to explain classifications with supportive and contrastive cases". In: *International Conference on Case-Based Reasoning*. Springer. 2020, pp. 245–260.

[52] Mathisen, B. M., Haro, P., Hanssen, B., Björk, S., and Walderhaug, S. "Decision support systems in fisheries and aquaculture: A systematic review". In: *arXiv preprint arXiv:1611.08374* (2016).

[53] Kawamoto, K., Houlihan, C. A., Balas, E. A., and Lobach, D. F. "Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success". In: *Bmj* vol. 330, no. 7494 (2005), p. 765.

[54] Vergara-Solana, F., Araneda, M. E., and Ponce-Diaz, G. "Opportunities for strengthening aquaculture industry through multicriteria decision-making". In: *REVIEWS IN AQUACULTURE* vol. 11, no. 1 (Feb. 2019), 105–118.

[55] Leung, P. “Multiple-criteria decision-making (MCDM) applications in fishery management”. English. In: *International Journal of Environmental Technology and Management* vol. 6, no. 1-2 (2006). multiple-criteria decision-making;fishery management;analytic hierarchy process;decision support system; pp. 96–110.

[56] Crossland, M. D., Wynne, B. E., and Perkins, W. C. “Spatial decision support systems: An overview of technology and a test of efficacy”. In: *Decision support systems* vol. 14, no. 3 (1995), pp. 219–235.

[57] Malczewski, J. “GIS-based multicriteria decision analysis: a survey of the literature”. In: *International Journal of Geographical Information Science* vol. 20, no. 7 (2006), pp. 703–726.

[58] Kitchenham, B. A. and Charters, S. *Guidelines for performing systematic literature reviews in software engineering*. Tech. rep. Keele University, 2007.

[59] Budgen, D., Turner, M., Brereton, P., and Kitchenham, B. “Using mapping studies in software engineering”. In: *Proceedings of PPIG*. Vol. 8. Lancaster University. 2008, pp. 195–204.

[60] Kitchenham, B. A., Mendes, E., and Travassos, G. H. “Cross versus within-company cost estimation studies: A systematic review”. In: *Software Engineering, IEEE Transactions on* vol. 33, no. 5 (2007), pp. 316–329.

[61] Scuse, D. H. and Arnason, A. N. “Information manipulation in biological decision-support systems”. In: *Proceedings of the Sixteenth Hawaii International Conference on System Sciences, 1983*. Vol. 1. Western Periodicals Company. 1983, p. 377.

[62] Arnott, D. and Pervan, G. “A critical analysis of decision support systems research revisited: the rise of design science”. In: *Journal of Information Technology* vol. 29, no. 4 (2014), pp. 269–293.

[63] Bourke, G., Stagnitti, F., and Mitchell, B. “A decision support system for aquaculture research and management”. In: *Aquacultural Engineering* vol. 12, no. 2 (1993), pp. 111–123.

[64] Silvert, W. “A decision support system for regulating finfish aquaculture”. In: *Ecological modelling* vol. 75 (1994), pp. 609–615.

[65] Bolte, J., Nath, S., and Ernst, D. “Development of decision support tools for aquaculture: the POND experience”. In: *Aquacultural engineering* vol. 23, no. 1 (2000), pp. 103–119.

[66] El-Gayar, O. F. and Leung, P. "ADDSS: a tool for regional aquaculture development". In: *Aquacultural Engineering* vol. 23, no. 1 (2000), pp. 181–202.

[67] Nath, S. S., Bolte, J. P., Ross, L. G., and Aguilar-Manjarrez, J. "Applications of geographical information systems (GIS) for spatial decision support in aquaculture". In: *Aquacultural Engineering* vol. 23, no. 1 (2000), pp. 233–278.

[68] Hargrave, B. T. "A traffic light decision system for marine finfish aquaculture siting". In: *Ocean & coastal management* vol. 45, no. 4 (2002), pp. 215–235.

[69] Halide, H., Stigebrandt, A., Rehbein, M., and McKinnon, A. "Developing a decision support system for sustainable cage aquaculture". In: *Environmental Modelling & Software* vol. 24, no. 6 (2009), pp. 694–702.

[70] Silvert, W. "Decision support for stakeholders". English. In: vol. 1. Ottawa, ON, Canada, 2010, pp. 523–529.

[71] Radulescu, C. and Rahoveanu, M. "A multi-criteria evaluation framework for fish farms". English. In: *Studies in Informatics and Control* vol. 20, no. 2 (2011), pp. 181–6.

[72] Magno-Tan, M., Alejandrino, A., Dela Cruz, C., Inoc, A., and Coronado, A. "Web-based decision support system for broodstock management of Siganus guttatus (Bloch, 1787) in open fish cage". English. In: *International Journal of Machine Learning and Computing* vol. 7, no. 6 (2017), pp. 208–12.

[73] Cobo, Á., Llorente, I., Luna, L., and Luna, M. "A decision support system for fish farming using particle swarm optimization". In: *Computers and Electronics in Agriculture* (2018).

[74] Duan, Q., Liu, Y., Zhang, L., and Li, D. "State-of-the-art review for application of big data technology in aquaculture". Chinese. In: *Nongye Jixie Xuebao/Transactions of the Chinese Society for Agricultural Machinery* vol. 49, no. 6 (2018). Analysis techniques;Aquaculture industry;Automatic decision;Big data platforms;Big data technologies;Development trends;Intelligent analysis;State-of-the art reviews; pp. 1–16.

[75] O'Mara, A. and Shahriar, M. S. "Short-term prediction of marine sensor data with fuzzy clustering". English. In: *International Journal of Pattern Recognition and Artificial Intelligence* vol. 29, no. 3 (2015). Conductivity data;Environmental Monitoring;Fuzzy pattern;Marine sensors;Novel techniques;Short term;Short term prediction;Water quality variables;

[76] Zhou, C., Lin, K., Xu, D., Chen, L., Guo, Q., Sun, C., and Yang, X. "Near infrared computer vision and neuro-fuzzy model-based feeding decision system for fish in aquaculture". English. In: *Computers and Electronics in Agriculture* vol. 146 (2018). Adaptive network based fuzzy inference system;Automatic adjustment;Delau-nay triangulations;Feeding behavior;Implementation process;Near- infrared images;Theoretical foundations;Water quality parameters; pp. 114–124.

[77] Yuan, H., Mao, Z., and Zhao, B. "Research of vannamei expert system based on CBR and Grey AHP". In: *2010 International Conference on Intelligent Computation Technology and Automation*. Vol. 2. May 2010, pp. 1065–1068.

[78] Tidemann, A., Bjørnson, F. O., and Aamodt, A. "Operational support in fish farming through case-based reasoning". In: *Advanced Research in Applied Artificial Intelligence*. 2012, pp. 104–113.

[79] Måløy, H., Aamodt, A., and Misimi, E. "A spatio-temporal recurrent network for salmon feeding action recognition from underwater videos in aquaculture". In: *Computers and Electronics in Agriculture* vol. 167 (2019), p. 105087.

[80] Breivik, Ø., Reistad, M., and Haakenstad, H. "A high-resolution hindcast study for the North Sea, the Norwegian Sea and the Barents Sea". In: *10th International Workshop on Wave Hindcasting and Forecasting*. 2007.

[81] Reistad, M., Breivik, Ø., Haakenstad, H., Aarnes, O. J., Furevik, B. R., and Bidlot, J.-R. "A high-resolution hindcast of wind and waves for the North Sea, the Norwegian Sea, and the Barents Sea". In: *Journal of Geophysical Research: Oceans* vol. 116, no. C5 (2011).

[82] Redmon, J. and Farhadi, A. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[83] Xu, X., Ester, M., Kriegel, H.-P., and Sander, J. "A distribution-based clustering algorithm for mining in large spatial databases". In: *Proceedings 14th International Conference on Data Engineering*. Feb. 1998, pp. 324–331.

[84] Berlemont, S., Lefebvre, G., Duffner, S., and Garcia, C. "Siamese neural network based similarity metric for inertial gesture classification and rejection". In: *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*. Vol. 1. IEEE. 2015, pp. 1–6.

[85] Lefebvre, G. and Garcia, C. “Learning a bag of features based nonlinear metric for facial similarity”. In: *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*. IEEE. 2013, pp. 238–243.

[86] Dheeru, D. and Karra Taniskidou, E. *UCI Machine Learning Repository*. 2017.

Papers

I

Data driven case base construction for prediction of success of marine operations

Bjørn Magnus Mathisen, Agnar Aamodt, Helge Langseth

Abstract

It is a common situation to have lots of recorded data that you want to use for improving a process in your organization or make use of this data to provide new services or products. Starting with one primary data set we describe a system that enhances this data set to a level such that it can be used by a deep learning system. This deep learning system then creates a model based on this data set, trying to predict operational windows for marine operations. Using this model the system extracts cases for use in a CBR-system aimed at providing operational support. This paper describes the partial implementation and results of this system.

I.1 Introduction

Critical operations are often meticulously planned and subject to many parameters that decide if and how these operations are performed. Some of these parameters are called operational time windows, which in marine environments often are connected to external factors such as weather.

This paper uses machine learning to predict favorable operational time windows or warn of unfavorable operational windows, so that critical operations can be planned with better accuracy, e.g. when the operation should ideally take place. One way of doing this is to look at historical data of previously executed operations. By combining data on successful and unsuccessful operations with the relevant context of that operation, we create a data set that can be used to find indicators for success or failure in advance. Which context that is relevant is dependent on the nature of operational window; wind and fog are important contexts for aviation, while waves and current are important for marine operations but not aviation.

This paper focuses on marine operations, and we analyze event data captured from boats moving in and out of zones connected to aquaculture installations. Next, we calculate the duration of these events and connect them to the relevant context and the associated success or failure classification.

The data used in this analysis is gathered as part of the EXPOSED project[1]. This project aims to develop enabling- and applied technologies for exposed aquaculture operations. The work we describe aims to improve planning of operations on aquaculture installations on exposed locations.

[1] http://exposedaquaculture.no/en/

The data is a subset of boats moving across geofences attached to aquaculture installations. This system consists of two zones around every aquaculture installation in Norway: One outer zone 400 meters from the outer points of the structures holding the fish themselves (not including the control building/fishfeed silos). The inner zone is 20 meters from the structure. These limits are in adherence to government regulations that no boat should fish within the outer zone and no boat should move within the inner zone unless the boat is there to operate on the installation.

An example of geofencing zones are shown in Fig. 1 below.

Figure I.1: The Green line show the outer geofence zone, the red line shows the inner geofence zone.

An event is created each time a boat crosses any of the geofence zones, marking the time. Table 1 below shows an example of a typical event.

Event ID	Location-ID	Vessel Name	Time	LZ	ET
81766	12966	Vessel A	2014-09-02 21:39:32	1	1
81767	12966	Vessel A	2014-09-02 21:40:11	1	2

Table I.1: This table shows an example of two events with corresponding location id, vessel name, time, LocationZone (LZ) and EventType (ET). These two events show a vessel entering (ET=1) and leaving (ET=2) the outer zone (LocationZone=1) of location 12966.

In data gathered in the EXPOSED project, the aquaculture industry reports on several possible problems with fish feed carriers interacting with aquaculture installations: Approaching the feed barges, often placed in shallow waters; Knowing which barge container to fill with what feed; Planning according to weather and route to enable the installation crew to attend the operation; And the fact that impact and currents from the boat can damage the installation.

As our data only gives us the time spent in two different proximities to the aquaculture installation there will be limits to which types of operational problems we can detect, and it will be very hard to discern between different causes (other than bad weather which is very general) of any detected problem.

The architecture of the full decision support system for EXPOSED is illustrated in Fig. I.2. In this paper we only present results from parts of the system. Future work will integrate these results with the other modules (e.g. knowledge models) to complete the system to a state where it can be verified in the field.

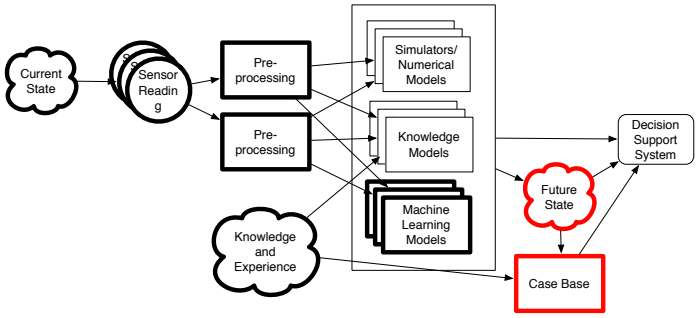


Figure I.2: The architecture of the planned systems. The parts implemented are highlighted, the case base and the future state is highlighted in red as being the current target for development.

Our main hypothesis is that given enough contextual weather data a deep neural network should be able to predict the length of a maritime operation at a aquaculture installation, enabling us to predict favorable operational windows. The main contribution of this paper is to show the reader the process of gathering, collating, filtering of data and subjecting this data to an analysis.

This paper is structured as follows; Section 2 introduces related work and our work in the light of this previous work. Section 3 describes the methods used in our work as well as the data sources used. Section 4 shows the result of our experiments, while section 5 presents the conclusion along with a discussion of the results.

I.2 Related work

In this work we aim to extract cases from a time series of events, CBR research has been done on several aspects of automatic case-authoring.

In CBR there has been a lot of focus on how to measure competence and utility of a case-base [87, 88]. In [89], they do this via reversing deletion policies constructed in [90] that try to improve case base utility without degrading competence.

Several works [91, 92, 93] use NLP to extract cases from structured and unstructured ([94, 95, 96]) text.

More specifically connected to the task of extracting cases from time series is the work done by Bach et.al. [97] where they employ clustering of time-series events in time and space, in combination with other detection methods. Funk et. al [98] uses different models of how predictive (or discriminatory) different time-series patterns are to different medical diagnosis of stress. For more insight into work done in time-series analysis connected to CBR research we suggest chapter 3.3 in [99]

The work presented in this paper shares the approach of Bach et al. [97] in that we try to extract the useful data points from the time series via clustering and filtering. Our work differs from the previous work in that we have very few verified cases apriori or during learning. In other words, the time-series is in all practical sense unlabeled for our use. We will try to apply common knowledge about how long an operation usually takes to perform. Then we can extract failed operations from the even time series to create cases that exemplify failed operations.

I.3 Method

To enable the deep learning system to correctly model and predict the time spent at an installation, we need to provide it with as much context data as possible for each of the event data points. In addition, we need the data to be as noise free as possible, thus we want to filter away operations that naturally have a high degree of variation in time spent at the location. We address these two requirements by combining the primary data set with other data sets, to enable us to provide filtering and context. An illustration of this process can be seen in Fig. I.3 . Below we describe each of the data sets.

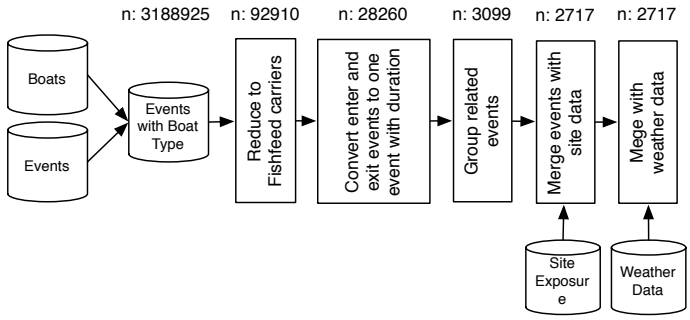


Figure I.3: This figure illustrates how the different data sources are combined and filtered to provide the deep learning system as much context as possible.

Boat data set As mentioned in the introduction we do not want to analyze all the traffic data of all of the boats. To verify that our method is usable in at least one instance, we want to look at a specific type of boat that has stable characteristics when it comes to the parameters (e.g. time and stability of time) of the operations it executes on the installation. We chose fishfeed boats in this case, as they only do one type of operation. That way we do not need to deduce the type of operation from the event data (one less hidden variable). In addition, this operation should be stable in the time it takes to execute it. To filter the data accordingly we need to combine the event data set with a data source that describes the boats. We can then easily extract the fishfeed boats.

NORA10 data set NORA10 [80, 81] is a data set that describes output of a precise weather model (hind-cast), that is validated by measurements. It has a higher resolution (10km) than most other models (e.g. the much used ERA[2] model with 80km resolution) as it is re-sampled for this specific region around Norway. We sample this model for each of the installations and at each time of each event (in the case of long events we use the median time of the event). We sample every datatype that we think will have an impact on the time spent on an operation: wind speed, wave direction, wind direction, significant swell wave height and significant wave height.

Exposure data set SINTEF EXPOSED has produced a data set [100] that describes the degree of exposure for a large number of the installations that are used in the event data set. This data set provides a level of exposure for 360 degrees around the installation (from 0 to max, where max is no land in sight). We combine our weather data with this (described above), thus we combine the wind direction of the wind with how exposed the location is in the direction of the wind using a filter that combines exposure level from +/- 10 degrees around the direction of the wind.

I.3.1 Extracting time spent in zones.

The data set needs to contain the time spent in the zones around the aquaculture installations. The raw data only contains events of entering and exiting the zones. To extract this we sequentially find each exit from a zone then search backwards for the entry to that zone by the same boat, then compute the time spent in that zone.

I.3.2 Grouping events close in time

After converting all discrete events into events with a duration, we still ended up with a lot of extremely short events. This is most probably caused by boats trying to stay close to the installation but the dynamic positioning system moves them in and out of the inner or outer zones. To counter this fact we grouped all events with the same boat at the same location within 1 hour into one event. However, after this grouping there is still 63% (or 244) of the events within the first 10 minute window. These are events

[2] http://www.ecmwf.int/en/research/climate-reanalysis/era-interim

within a zone that is less than ten minutes in duration and without another event in the same location within one hour of the original event. There are three possible explanations for these strange events: 1. The boat is passing through the location, and not returning for at least one hour. Or otherwise briefly enters and exists the zone, without this fact having any effect on the operation. 2. The boat tries to perform an operation at the location but has to abort and leaves within ten minutes. 3. The event was not registered correctly when the data was gathered. The most probable cause for most of these events are boats that travel through the zone heading for another location. This hypothesis can be tested by removing outer zone events from the distribution. As the inner zone is small, very few of these big fishfeed carrier boats would drive through the inner zone of an aquaculture installation when heading somewhere else. We can still see 244 events that are of duration 10 minutes or less within the inner zone of an aquaculture installation. Figure I.4 looks at the 1 minute distribution within the first 10 minutes to try to find the causes for the high number of short stay events. And once again we can see that many of the events are very short, with very few events lasting more than 3 minutes. This further supports our first hypothesis.

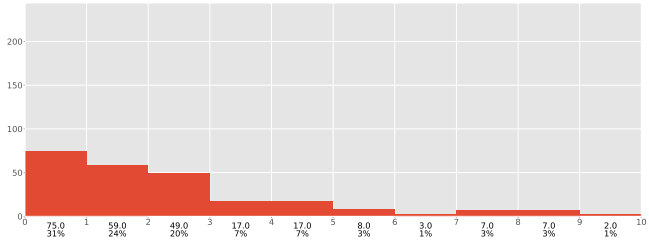


Figure I.4: Distribution of events over length of stays in all inner zone after grouping all events within a 1 hour time window. Zoomed into the first 10 minutes.

One problem with our approach so far is that some events are very far apart in time as well as having different zone types. One example being one boat having a 0 second stay in the inner zone of location 31437 at 18:23 the 28th of November, however the boat entered the outer zone of the same site at 17:04 the same day, and exited zone 1 of that location at 18:24. We can then conclude that the boat spent approximately 1 hour and 20 minutes at the location in the outer zone, then very briefly entered the inner zone before leaving the location. Again supporting the first hypothesis. From this we can see that including inner zone in analyzing fishfeed carrier operations adds very little information to our analysis as the fishfeed carriers do not enter the inner zone when transferring fishfeed. As a consequence we discard the inner zone data. We are still left with 2401 events with a duration shorter than 10 minutes. Fig I.5 shows the distribution of these events length in stay. We can see that most of these are shorter than 5

minutes, and most probably does not represent actual maritime operations (or failed tries), but rather traveling through the zone. Thus we discard events shorter than 10 minutes, giving us the final distribution shown in Fig. I.6.

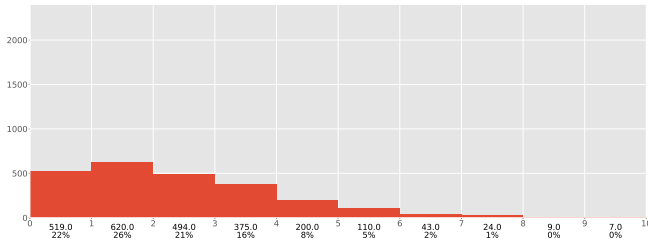


Figure I.5: Distribution of events over length of stays in all outer zone after grouping all events within a 1 hour time window. Zoomed into the first 10 minutes.

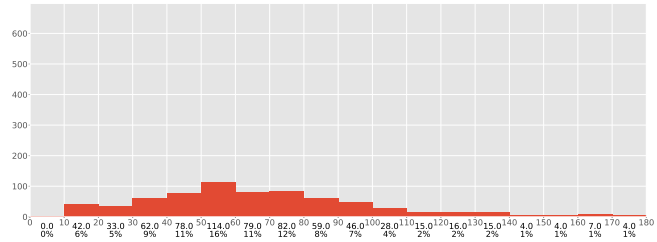


Figure I.6: Distribution of events over length of stays in all outer zone after grouping all events within a 1 hour time window. With all stays smaller than 10 minutes removed.

I.3.3 Predicting the operational time using Deep Learning

To extract cases that exemplify instances where the weather conditions stops a fishfeed operation from being successfull, we are currently building a deep learning model aimed at predicting the time spent at the installation, with the given weather and level of exposure at the time and location. The input to the model is: draft and length of the boat, wind speed[3], distance between the model grid point and actual site coordinate, wave direction[3], wind direction[3], maximum level of exposure at location, significant swell wave height[3], month, hour, wind effect (wind speed combined with exposure

[3]Measured at the closest grid point in NORA10

levels in the wind direction +/- 10 degrees) and significant wave height. The output of the model is the amount of time spent on the installation.

The regression was implemented using python. We used sklearn for preprocessing and scaling (MinMax scaling) of input data (including regression target). The Keras library for deep learning was used for the regression itself, with a input layer of $inputcolumns + 1 = 14$ nodes. We used 3 hidden layers with 13 nodes each and a output layer of 1 node. All nodes used the ReLU activation function.

I.4 Results

The current results show that there is little information in the gathered data (through the NORA10 model and exposure levels) that account for the variance shown in the time spent at the locations. The neural network models presented in the previous Section I.3.3 gets very low accuracy (0.11%, which means the predictor is very slightly better than just outputting the average) in terms of predicting how long a fish feed boat stays at a aquaculture installation. Figure I.7 shows the length of all of the events in the chronologically in blue and the predicted length in orange. The "Time Spent" axis is normalized values of the time spent in near a installation where $y = 1.0$ represents the longest stay recorded in the training data. There are obvious differences between predicted and true values; predicted values consistently returns too high values, and fails to predict short stays. A cross validated ($cv = 5$) hyper parameter grid search was performed and showed no better performance at 10 hidden layers with 56 nodes in each hidden layer.

After we received the disappointing results we created scatter plots of two weather variables in relation to the length of stay at the installations. Typically most would assume there would be a pattern of some correlation between the weather and the length of stay. However Figure I.8 shows that neither wind (I.8a) or waves (I.8b) reveals any obvious correlation patterns against time spent at installations.

In addition we did a principal component analysis of the data, to discover if there where any clear principal components that could contain the variance in the data. The components returned: $C = (0.127, 0.117, 0.109, 0.099, 0.091, 0.039, 0.034, 0.028$
$, 0.020, 0.011, 0.008, 0.004, 0.002, 0.000)$ Where the sum of components $sum(C) = 0.6967$ indicating that the total of the components could account for little of the variance. Finally we tried a standard method for non-linear regression as a base-line result to measure the DNN against. We tried Epsilon-Support Vector Regression (SVR) which scored with a coefficient of determination $R^2 = -0.83$ which is worse than constantly predicting the mean of the target (which would give $R^2 = 0.0$). This final result shows in the context of the other results listed above us that the data set may not contain the features needed to predict the length of the stay at a installation.

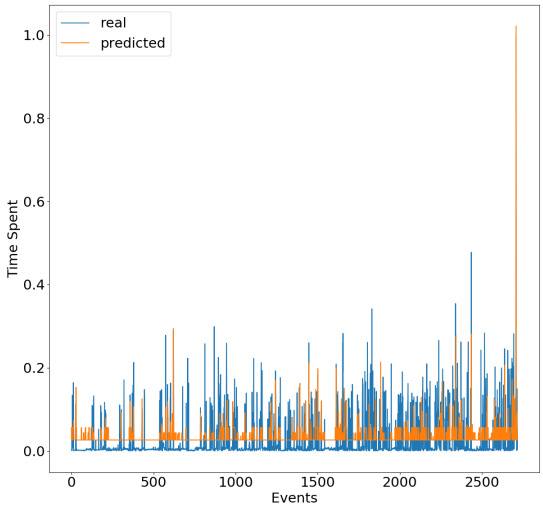
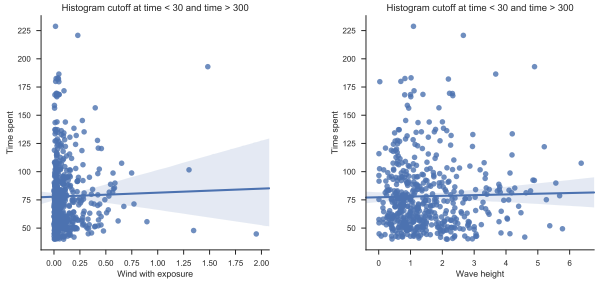


Figure I.7: This shows the DNN model try to predict the amount of time spent at a installation in orange, and the actual time spent in blue. The X-axis is simply the record number, where the record are ordered along the time axis.



(a) Wind correlation (b) Wave correlation

Figure I.8: Scatter plot illustrating correlation between the weather and the time spent at the installation

I.5 Conclusions and future work

We started the work with a hypothesis that whether or not a fishfeed boat operation (loading of fishfeed from boat to barge) succeeded depended on the weather, and that such a failure could be detected from the length of

time the fishfeed boat stayed at the aquaculture installation. Our analysis did not find any deterministic correlation between the weather and location data and the length of the stay at the installation. There can be many reasons for this, we will try to list some of the reasons we think are probable;

The first possibility is that despite our efforts to remove noise from the data, the data still contains noise. This includes the three factors listed in the introduction section and other possibilities we have not considered.

Second, given the size of the boats and their stability, they can operate during harsh conditions. In addition these boats are expensive in operation, and even more expensive if they fail to deliver feed at the appointed time, possibly starving the fish at the installation. Thus these boats are already subject to careful operational planning. It may therefore be that there is none to very few failed fishfeed operations in the data captured. An additional consequence is that the time spent during operations has very low variance.

Extending this work would start with confirming these possible explanations for the lack of correlation found in our data. We would also like to gather further data, extending the number of events beyond the current 2700. This would enable us to train and test our models with more rigor and less uncertainty.

I.6 Acknowledements

None of the work done in this paper would have been possible without the support of the EXPOSED project. Special thanks to ANTEO (http://anteo.no/) for providing data to this experiment and working with us to make use of this data.

References

[80] Breivik, Ø., Reistad, M., and Haakenstad, H. "A high-resolution hindcast study for the North Sea, the Norwegian Sea and the Barents Sea". In: *10th International Workshop on Wave Hindcasting and Forecasting*. 2007.

[81] Reistad, M., Breivik, Ø., Haakenstad, H., Aarnes, O. J., Furevik, B. R., and Bidlot, J.-R. "A high-resolution hindcast of wind and waves for the North Sea, the Norwegian Sea, and the Barents Sea". In: *Journal of Geophysical Research: Oceans* vol. 116, no. C5 (2011).

[87] Smyth, B. and McKenna, E. "Modelling the competence of case-bases". In: *Proceedings of european Workshop on Advances in Case-Based Reasoning*. 1998, pp. 208–220.

[88] Smyth, B. and McKenna, E. “Building compact competent case-bases”. In: *Proceedings of the third International Conference on Case-Based Reasoning*. 1999, pp. 329–342.

[89] Zhu, J. and Yang, Q. “Remembering to add: competence-preserving case-addition policies for case-base maintenance”. In: *IJCAI*. Vol. 99. 1999, pp. 234–241.

[90] Smyth, B. and Keane, M. “Remembering to forget: A competence-preserving deletion policy for cbr”. In: *Proceedings IJCAI-95*. 1995.

[91] Yang, C., Farley, B., and Orchard, B. “Automated case creation and management for diagnostic CBR systems”. In: *Applied Intelligence* vol. 28, no. 1 (Feb. 2007), pp. 17–28.

[92] Yang, Q. and Cheng, H. “Case mining from large databases”. In: *Lecture Notes in Computer Science* (), pp. 691–702.

[93] Zaluski, M., Japkowicz, N., and Matwin, S. “Case authoring from text and historical experiences”. In: *Lecture Notes in Computer Science* (2003), pp. 222–236.

[94] Bach, K., Althoff, K.-D., Newo, R., and Stahl, A. “A case-based reasoning approach for providing machine diagnosis from service reports”. In: *Case-Based Reasoning Research and Development*. Case-Based Reasoning Research and Development. 2011, pp. 363–377.

[95] Dufour-Lussier, V., Ber, F. L., Lieber, J., and Nauer, E. “Automatic case acquisition from texts for process-oriented case-based reasoning”. In: *Information Systems* vol. 40, no. nil (2014), pp. 153–167.

[96] Farley, B. “From free-text repair action messages to automated case generation”. In: *Proceedings of AAAI 1999 Spring Symposium: AI in Equipment Maintenance Service & Support, Technical Reprot SS-99-02, Menlo Park, CA, AAAI Press*. 1999, pp. 109–118.

[97] Bach, K., Gundersen, O. E., Knappskog, C., and Öztürk, P. “Automatic case capturing for problematic drilling situations”. In: *International Conference on Case-Based Reasoning*. Springer. 2014, pp. 48–62.

[98] Funk, P. and Xiong, N. “Case-based reasoning and knowledge discovery in medical applications with time series”. In: *Computational Intelligence* vol. 22, no. 3-4 (Aug. 2006), pp. 238–253.

[99] Gundersen, O. E. “Enhancing the situation awareness of decision makers by applying case-based reasoning on streaming data”. PhD thesis. NTNU, 2014.

[100] Lader, P., Kristiansen, D., Alver, M., Bjelland, H. V., and Myrhaug, D. “Classification of aquaculture locations in norway with respect to wind wave exposure”. In: *Proceedings of the ASME 2017 36th International Conference on Ocean, Offshore and Arctic Engineering OMAE2017*. 2017.

Learning similarity measures from data

Bjørn Magnus Mathisen, Agnar Aamodt, Kerstin Bach, Helge Langseth

II

II

Abstract

Defining similarity measures is a requirement for some machine learning methods. One such method is case-based reasoning (CBR) where the similarity measure is used to retrieve the stored case or set of cases most similar to the query case. Describing a similarity measure analytically is challenging, even for domain experts working with CBR experts. However, data sets are typically gathered as part of constructing a CBR or machine learning system. These datasets are assumed to contain the features that correctly identify the solution from the problem features, thus they may also contain the knowledge to construct or learn such a similarity measure. The main motivation for this work is to automate the construction of similarity measures using machine learning. Additionally, we would like to do this while keeping training time as low as possible. Working towards this, our objective is to investigate how to apply machine learning to effectively learn a similarity measure. Such a learned similarity measure could be used for CBR systems, but also for clustering data in semi-supervised learning, or one-shot learning tasks. Recent work has advanced towards this goal, relies on either very long training times or manually modeling parts of the similarity measure. We created a framework to help us analyze current methods for learning similarity measures. This analysis resulted in two novel similarity measure designs. One design using a pre-trained classifier as basis for a similarity measure. The second design uses as little modeling as possible while learning the similarity measure from data and keeping training time low. Both similarity measures were evaluated on 14 different datasets. The evaluation shows that using a classifier as basis for a similarity measure gives state of the art performance. Finally the evaluation shows that our fully data-driven similarity measure design outperforms state of the art methods while keeping training time low.

II.1 Introduction

Many artificial intelligence and machine learning (ML) methods, such as k-nearest neighbors (k-NN) rely on a similarity (or distance) measure [49] between data points. In Case-based reasoning (CBR) a simple k-NN or a more complex similarity function is used to retrieve the stored cases that are most similar to the current query case. The similarity measure used in CBR systems for this purpose is typically built as a weighted Euclidean similarity measure (or as a weight matrix for discrete and symbolic variables). Such a similarity measure is

designed with assistance of domain experts by adjusting the weights for each attribute of the cases to represent how important they are (one example can be seen in [101], or generally described in chapter 4 of [102])

In many situations the design of such a function is non-trivial. Domain experts with an understanding of CBR or machine learning are not easily available. However, before or during most CBR projects, data is gathered that relates to the problem being solved by the CBR system. This data is used to construct cases for populating the case base. If the data is labeled according to the solution/class, it can be used to learn a similarity measure that is relevant to the task being solved by the system. Exploring efficient methods of learning similarity measures and improving on them is the main motivation of this work.

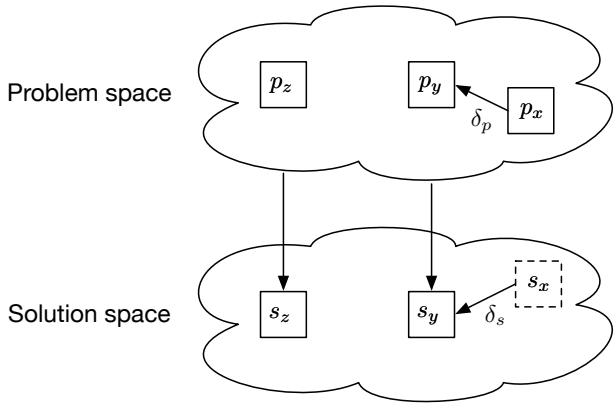


Figure II.9: Illustration of problem and solution spaces [103]. p_y and p_z are two problem descriptions with features describing a problem each of which has a corresponding (s_y and s_z) solution in solution space. δ_p illustrates the distance between a new problem p_x and a stored problem p_y. Correspondingly δ_s is the distance between the solution s_y and the solution s_x which is the (unknown) ideal solution to p_x. A fundamental assumption in CBR is that if the similarity between p_x and p_y is high then the similarity between the unknown solution s_x to p_y is high ($\delta_p \approx \delta_s$): Similar problems have similar solutions.

In the CBR literature, similarity measurement is often described in terms of problem- and solution spaces. Problem space is where the features of a problem describe the problem; this is often called feature space in non-CBR ML literature. Solution space, also referred to as

target space, is populated by points describing solutions to points in the problem space. The function that maps a point from the problem space to its corresponding point in the solution space is typically the goal of supervised machine learning. This is illustrated in Figure II.9.

A similarity measure in the problem space represents an approximation of the similarity between two cases or data points in the solution space (i.e. whether these two cases have similar or dissimilar solutions). Such a similarity measure would be of great help in situations where lots of labeled data is available, but domain knowledge is not available, or when the modeling of such a similarity measure is too complex.

Learned similarity measures can also be used in other settings, such as clustering. Another relevant method type is semi-supervised learning in which the labeled part of a dataset is used to cluster or label the unlabeled part.

How to automatically learn similarity measures has been an active area of research in CBR. For instance, Gabel et al. [38] train a similarity measure by creating a dataset of collated pairs of data points and their respective similarities. This dataset is then used to train a neural network to represent the similarity measure. In this method the network needs to extract the most important features in terms of similarity for both data points, then combine these features to output a similarity measure. Recent work (e.g. Martin et al. [104]) has used Siamese neural networks (SNN) [48] to learn a similarity measure in CBR. SNNs have the advantage of sharing weights between two parts of the network, in this case the two parts that extract the useful information from the two data points being compared. All of these methods for learning similarity measures have in common that they are trained to compare two data points and return a similarity measurement. Our work of automatically learning similarity measures is also related to the work done by Hüllermeier et al. on preference-based CBR [105, 106]. In this work the authors learn a preference of similarity between cases/data points, which represents a more continuous space between solutions than a typical similarity measure in CBR. This type of approach to similarity measures is similar to learning similarity measures by using machine learning models, in that both can always return a list of possible solutions sorted by their similarity.

In this work we have developed a framework to show the main differences between various types of similarity measures. Using this framework, we highlight the differences between existing approaches in Section II.3. This analysis also reveals areas that have not received much attention in the research community so far. Based on this we developed two novel designs for using machine learning to learn

similarity measures from data. Both of the two designs are continuous in their representation of the estimated solution space.

The novelty of our work is three-fold: First showing that using a classifier as a basis for a similarity measure gives adequate performance. Then we demonstrate similarity measure designed to use as little modeling as possible, while keeping training time low, outperforms state of the art methods. Finally to analyze the state of the art and compare it to our new similarity measure design we introduce a simple mathematical framework. We show how this is a useful tool for analyzing and categorizing similarity measures.

The remainder of this paper describes our method in more detail. Section II.2 describes the novel framework for similarity measurement learning, and Section II.3 then summarizes previous relevant work in relation to this framework. In Section II.4 we describe suggestions of new similarity measures, and how we design the experimental evaluation. Subsequently, in Section II.5 we show the results of this evaluation. Finally, in Section II.6 we interpret and discuss the evaluation results and give some conclusions. We present some of the limitations of our work as well as possible future paths of research.

II.2 A framework for similarity measures

We suggest a framework for analyzing different functions for similarity with $\mathbb{S}$ as a similarity measure applied to pairs of data points $(\boldsymbol{x}, \boldsymbol{y})$;

$$\mathbb{S}(\boldsymbol{x}, \boldsymbol{y}) = C(G(\boldsymbol{x}), G(\boldsymbol{y})), \tag{II.1}$$

where $G(\boldsymbol{x}) = \hat{\boldsymbol{x}}$ and $G(\boldsymbol{y}) = \hat{\boldsymbol{y}}$ represents embedding or information extraction from data points x and y , i.e. $G(\cdot)$ highlights the parts of the data points most useful to calculate the similarity between them. An illustration of this process can be seen in Figure II.10.

$C(G(\boldsymbol{x}), G(\boldsymbol{y})) = C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ models the distance between the two data points based on the embeddings $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$. The functions C and G can be either manually modeled or learned from data. With respect to this we will enumerate all of the different configurations of Equation II.1 and describe their main properties and how they have been implemented in state of the art research. Note that we will use $\mathbb{S}(\cdot)$ to annotate the similarity measurement and $C(\cdot)$ for the sub-part of the similarity measurement that calculates the distance between the two outputs of $G(\cdot)$. $\mathbb{S}(\cdot)$ is distinct from $C(\cdot)$ unless $G(x) = I(x) = x$.

In the following we characterize the different types of similarity measures:

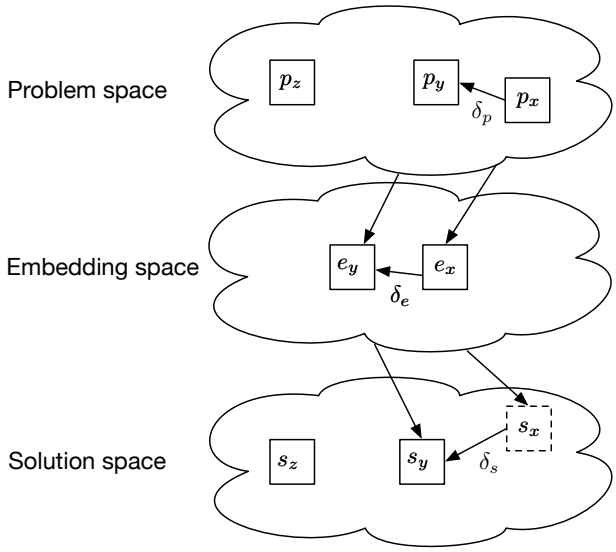


Figure II.10: Illustrating how $G(\cdot)$ from Equation II.1 adds another space, the embedding space, between the problem and the solution space [103] (see Figure II.9). $C(\cdot)$ then combines the two embeddings of p_y and p_x (e_y and e_x respectively) and calculates the similarity δ_e between them. The main assumption is that distance in embedding space (δ_e) is close to the distance in solution space (δ_s) ; if the embedded points e_x and e_y are similar, then the true (unknown) solution s_x is similar to solution s_y. The main contribution of $G(\cdot)$ is to create a embedding space such that the distance in embeddings space (δ_e) is a better estimate of the distance in solution space (δ_s) than the distance in problem space (δ_p).

Type 1 A typical similarity measure in CBR systems would model $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and $G(\cdot)$ from domain knowledge. Such a similarity measure is typically modeled by experts with the relevant domain knowledge together with CBR experts, who know how to encode this domain knowledge into the similarity measures.

For example when modeling the similarity measure of cars for sale, where the goal is to model the similarity of cars in terms of their final selling price. In this example, domain experts may model the embedding function $G(\cdot)$ so that the amount of miles driven has a greater importance than the color of the car. $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ could be modeled such that difference in miles driven is less important than difference the number of repairs done on the car. More details and examples can be found in [107].

		$C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$	
		Modeled	Learned
$G(\cdot)$	Modeled	Type 1	Type 2
	Learned	Type 3	Type 4

Table II.2: Table showing different types of similarity measures in our proposed framework.

Type 2 This type represents similarity measures that models $G(\cdot)$ and learns the function $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$. In this context $G(\cdot)$ can be viewed as an embedding function. Since $G(\cdot)$ is not learned from the data it is not interesting to analyze it as part of learning the similarity measure, as processing the data through $G(\cdot)$ could be done in batch before applying the data to $\mathbb{S}(\boldsymbol{x}, \boldsymbol{y})$. Learning $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ needs to be done with a dataset consisting of triplets of the data points $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$, and s being the true similarity between $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$.

A special case of Type 2 is when $G(\cdot)$ is set to be the identity function $I(\boldsymbol{x}) = G(\boldsymbol{x}) = \boldsymbol{x}$, while $C(\boldsymbol{x}, \boldsymbol{y})$ is learned from data. Examples of this type are presented for example in Gabel et al. [38] where the similarity measure always looks at the two inputs together, never separately.

Type 3 In this type of similarity measure the embedding/feature extraction $G(\cdot)$ is learned and $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ is modeled. Typically the embedding function learned by $G(\cdot)$ resembles the function that is the goal during supervised machine learning. Within the similarity measurement $\hat{\boldsymbol{x}} = G(x)$ is used as an embedding vector for calculating similarity, when in classification $\hat{\boldsymbol{x}}$ would be the softmax vector output. Using a pre-trained classification model as a starting point for $G(\boldsymbol{x}) = \hat{\boldsymbol{x}}$ as input to e.g. $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = \|\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}\|_1$ should give good results for similarity measurements if that model had high precision for classification within the same dataset.

However it is not given that the best embedding vector for calculating similarity is the same as the embedding vector produced by a $G(x)$ trained to do classification.

Type 4 This measure is designed so that both $G(\cdot)$ and $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ are learned.

We will design, implement and evaluate similarity measures based on Type 1, Type 3, Type 2 and Type 4 in Section II.4. These results

will be shown in Sections II.5.

To allow $\mathbb{S}$ as a similarity measurement for clustering e.g. k-nearest neighbors, a similarity measure should fulfill the following requirements:

Symmetry $\mathbb{S}(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{S}(\boldsymbol{y}, \boldsymbol{x})$ The similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$ should be the same as the similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$.

Non-negative $\mathbb{S}(\boldsymbol{x}, \boldsymbol{y}) \geq 0 | \forall \boldsymbol{x}, \boldsymbol{y}$ The similarity between to datapoints can not be negative.

Identity $\mathbb{S}(\boldsymbol{x}, \boldsymbol{y}) = 1 \iff \boldsymbol{x} = \boldsymbol{y}$ The similarity between two datapoints should be 1 iff $\boldsymbol{x}$ is equal to $\boldsymbol{y}$.

Some of these requirements are not satisfied by all types of similarity measures, i.e. symmetry is not a direct design consequence of Type 2 but of Type 3 if $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ is symmetric. Even if symmetry is not present in all similarity measures [108] it is important for reducing training time, as the training set size goes from $N(N-1)$ to $N(\frac{N}{2} - 1)$. Symmetry also enables the similarity measure to be used for clustering.

In the next section, we will relate current state of the art to the framework in context of the different types.

II.3 Related work

To exemplify the framework presented in the previous section we will relate previous work to the framework and the types of similarity measurements that derive from the framework. This will also enable us to see possibilities for improvement and further research.

As stated in Section II.1 our motivation is to automate the construction of similarity measures. Additionally, we would like to do this while keeping training time as low as possible. Thus we will not focus on Type 1 similarity measures as this type uses no learning. Both Type 2 and Type 4 require a different type of training dataset than a typical supervised machine learning dataset, as $C(\boldsymbol{x}, \boldsymbol{y})$ is typically dependent on the order of the data points (see Section II.4). Thus given our initial motivation, Type 3 similarity measures seems to be the most promising type of similarity measure to focus on. However, it is worth investigating similarity measures of Type 4, to see if the added benefit of learning $C(\boldsymbol{x}, \boldsymbol{y})$ outweighs the added training time. Or if it is possible to make it symmetric (as defined in the previous section) so that the training time could become similar to Type 3.

Thus we will focus on summarizing related work from Type 3 similarity measures, but also add relevant work from Type 1, Type 2 and Type 4 for reference.

Type 1 is a type of similarity measure which is manually constructed. A general overview and examples of this type of similarity measure can be found in [107]. Nikpour et al. [43] presents an alternative method which includes enrichment of the cases/data points via Bayesian networks.

Type 2

In Type 2 similarity measures only the binary $C(\boldsymbol{x}, \boldsymbol{y})$ operator of the similarity measure $\mathbb{S}(\boldsymbol{x}, \boldsymbol{y})$ is learned, while $G(\cdot)$ is either modeled or left as the identity function ($G(\boldsymbol{x}) = I(\boldsymbol{x}) = \boldsymbol{x}$). Stahl et al. have done a lot of work on learning Type 2 similarity measures from data or user feedback. In all of their work they formulate $C(\boldsymbol{x}, \boldsymbol{y}) = \sum \boldsymbol{w}_i * sim_i(\boldsymbol{x}_i, \boldsymbol{y}_i)$ where for each feature i, sim_i is the local similarity measure and $\boldsymbol{w}_i$ is the weight of that feature. In [42] Stahl et al. describe a method for learning the feature weights.

In [44] Stahl et al. introduce learning local similarity measures through an evolutionary algorithm (EA). First they learn attribute weights ($\boldsymbol{w}_i$) by adopting the method previously described in [42]. Then they use an EA to learn the local similarity measures for each feature ($sim_i(\boldsymbol{x}, \boldsymbol{y})$). In [109] Stahl and Gabel present work were they learn weights of a modeled similarity measure, and the local similarity for each attribute through an ANN. Reategui et al. [46] learn and represent parts of the similarity functions ($C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$) through ANN. Langseth et al. [45] learn similarity knowledge ($C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$) from data using Bayesian networks, which still partially relies on modeling the Bayesian networks with domain knowledge.

Abdel-Aziz et al. [47] use the distribution of case attribute values to inform a polynomial local similarity function, which is better than guessing when domain knowledge is missing. So this method extracts statistical properties from the dataset to parametrize $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$.

Gabel and Godehardt [38] use a neural network to learn a similarity measure. Their work is done in the context of Case-based Reasoning (CBR) which uses the measure to retrieve similar cases. They concatenate the two data points into one input vector. Thus in the context of our framework $G(\cdot)$ is modeled as a identify function $I(x) = x$ and $C(\boldsymbol{x}, \boldsymbol{y})$ is learned.

Maggini et al. [49] uses SIMNNs which they also see as a special case of the Symmetry Networks [110] (SNs). In SIMMNs $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and $G(\cdot)$ are both a function of both $\boldsymbol{x}$ and $\boldsymbol{y}$ data points and there is thus no distinct $G(\cdot)$. They also have a specialized structure imposed on their network to make sure the learned function is symmetric. SIMNN is in essence an extended version of a Siamese neural network, but

without a distinct distance layer usually present in SNN architectures. They focus on the specific properties of the network architecture and the application of such networks in semi-supervised settings such as k-means clustering. The pair of data points ($\boldsymbol{x}$ and $\boldsymbol{y}$) are being compared two times, the first time at the first hidden layer, then at the output layer. Since there are no learnable parameters before this comparison all the learning is done in $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and $G(\boldsymbol{x})$ is the activation function of the input layer.

Type 3

One way of looking at a similarity measure is as an inverse distance measure, as similarity is the semantic opposite of distance. There has been much work on learning distance measures. Most of this work can be categorized as a Type 3 similarity measure as the learning tasks only aims to learn the embedding function $G(\cdot)$ then combine the output of this function with a static $C(\cdot)$ (e.g. a $L2$ norm function). The most well known instance of a Type 3 learned distance measure is Siamese neural networks (SNNs), it is highly related to the Type 2 similarity measure by Maggini et al.'s Similarity neural networks (SIMNN) [49].

The main characteristic of SNNs is sharing the weights between the two identical neural networks. The data points we want to measure the similarity for are then input to these networks. This frees the learning algorithm of learning two sets of weights for the same task. This was first used in [48] (using $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = cos(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and $G(\cdot)$ being learned from data) to measure similarity between signatures. Similar architectures are also discussed in [110].

Chopra et al. [37] uses a SNN for face verification, and pose the problem as an energy based model. The output of the SNN are combined through a $L1$ norm (absolute-value norm $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = \|\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}\|$) to calculate the similarity. They emphasize that using a $L2$ norm (Euclidean distance) as part of the loss function would make the gradient too small for effective gradient descent (i.e. create plateaus in the loss function). This work is closely related to Hadsell et al. [111], where they explain the contrastive loss function used for training the SNN (also used in [37, 104]) by analogy of a spring system.

Related to this Vinyals et al. [28] uses a similar type of setup for matching an input data point to a support set. It is framed as a discriminative task, where they use two neural networks to parametrize an attention mechanism. They use these two networks to embed the two data points into a feature space where the similarity between them are measured. However, in contrast to SNNs and SIMNNs, their two networks for embedding the data points are not identical, as one network is tailored to embed a data point from the support set, but also given the rest of the support set. Thus the

embedding of the support set data point is also a function of the rest of the support set. With $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ being modeled as a cosine softmax, this is similar to the examples of Type 3 similarity measures mentioned previously (e.g. [48, 84]). However a major difference is that signal extraction functions are not equal: $\mathbb{S}(\boldsymbol{x}, \boldsymbol{y}) = C(f(\boldsymbol{x}), g(\boldsymbol{x}))$ with $f(\boldsymbol{x}) \neq g(\boldsymbol{x})$ (only stating that $f(\cdot)$ may potentially equal $g(\cdot)$). Since $f(\cdot)$ and $g(\cdot)$ are not sharing weights between them, the architecture is variant (or asymmetric) to the ordering of input pairs. Thus the architecture needs up to twice as much training to achieve the same performance as a SNN.

In much of the same fashion as Chopra et al. did in [37], Berlemot et al. [84] uses SNNs combined with an energy based model to build a similarity measure between different gestures made with smart phones. However they adapt the error estimation from using only separate positive and negative pairs to a training subset including; a reference sample, a positive sample and a negative sample for every other class. They train $G(\cdot)$ while keeping a static $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = cos(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$. This training method of using triplets for training SNNs was also described by Lefebvre et al. [85]. A similar approach can be seen in Hoffer et al. [34], however they do not use a set of negative examples per reference point for each class as Berlemont et al do. Instead they use triples of $(\boldsymbol{x}, \boldsymbol{x}^+, \boldsymbol{x}^-)$, $\boldsymbol{x}$ being the reference point, $\boldsymbol{x}^+$ being the same class and $\boldsymbol{x}^-$ being a different class.

Koch et al. [112] uses a Convolutional Siamese Network (CSN), with $G(\cdot)$ implemented as a CNN and $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ implemented as $L1(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$. This is done in a semi-supervised fashion for one-shot learning within image recognition. They learn this CSN for determining if two pictures from the Omniglot [113] dataset is within the same class. The model can then be used to classify a data point representing an unseen class by comparing it to a repository of class representatives (Support Set).

CSNs are also used in the context of CBR by Martin et al. [104] to represent a similarity measure in a CBR system. The CSN is trained with pairs of cases and the output is their similarity. During training they have to label pairs of cases as 'genuine' (both cases belong to the same class) or 'impostor' (the cases belong to different classes). This requires that the user has a clear boundary for the classes. In relation to our framework this similarity measure learns $G(\cdot)$, while $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ is static. With $G(\cdot)$ implemented as a convolutional neural network, and $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ implemented as Euclidean distance ($L2$ norm).

In general using SNNs for constructing similarity measures have a major advantage as you can easily adopt pre-trained models for $G(\cdot)$ to embedding/preprocess the data points. For example to train a model for comparing two images one could use ResNet [114] for

$G(\cdot)$ then use the $L1$ norm as $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$. This would be a very similar approach to the similarity measure used by Koch et al. [112] with $\mathbb{S}(\boldsymbol{x}, \boldsymbol{y}) = \|(G(\boldsymbol{x}), G(\boldsymbol{y}))\|_1$, the main difference being that $G(\cdot)$ is designed for bigger pictures.

There are only very few examples of Type 4 similarity measures in the literature. In Zagoruyko and Komodakis's work [115] they investigate different types of architectures for learning image comparison using convolutional neural networks. In all of the architectures they evaluate $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ is learned, but in some of these architectures $G(\cdot)$ is not symmetric, i.e. $\mathbb{S}(\boldsymbol{x}, \boldsymbol{y}) = C(G(\boldsymbol{x}), H(\boldsymbol{y}))$ where $G(\boldsymbol{x}) \neq H(\boldsymbol{x})$. Arandjelović and Zisserman's work [116] use a very similar method to many Type 3 similarity measures for calculating similarity. However their input data is always pairs of two different data types and is as such different from most of the other relevant work leaving $G(\cdot)$ unsymmetrical just as in Zgoruyko et al. [115] and Vinyals et al. [28]. In contrast to the Type 3 similarity measures including [28], Arandjelović et al. also learns $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$, which they call a fusion layer.

All similarity measure of Type 3 we found in the literature use a loss function that includes feedback from the binary operator part of $\mathbb{S}$ $(C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}))$. In the case of SNNs even if $C(\boldsymbol{x}, \boldsymbol{y})$ is non-symmetric $(C(\boldsymbol{x}, \boldsymbol{y}) \neq C(\boldsymbol{y}, \boldsymbol{x}))$ the loss for each network would be equal as they are equal and share weights. That means that ordering of the two data points being compared during training has no effect, i.e. the training effect of $(\boldsymbol{x}, \boldsymbol{y})$ is equal to that of $(\boldsymbol{y}, \boldsymbol{x})$. This means a lot of saved time during training, as the training dataset could be halved without any negative effect on performance.

However the implementation of $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ would then decide how much training one would need to adapt a pre-trained model from classifying single data points to measuring similarity between them. One could view the process of starting with a pre trained model for the dataset, then training the model with loss coming from $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ as adapting the model from classification to similarity measurement.

One way of creating a Type 3 similarity measure using a minimal amount of training would be to pre-train a network on classifying individual data points. Then apply that network as $G(\cdot)$ that feeds into a $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = \|\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}\|$ in a similarity measurement. Evaluation of such a similarity measurement has not been reported in literature, and such a similarity will be explored in the next section.

II.4 Method

The framework presented in Section II.2 and the subsequent analysis of previous relevant work presented in Section II.3 shows that there are

unexplored opportunities within research on similarity measurements.

Given the initial motivation we seek methods that work well in domains where domain knowledge is very resource demanding. This requires that as much as possible of the similarity measure $\mathbb{S}(\boldsymbol{x}, \boldsymbol{y}) = C(G(\hat{\boldsymbol{x}}), G(\hat{\boldsymbol{y}}))$ is learned from data rather than modeled from domain knowledge. There are some exceptions to this, such as applying general binary operations, such as norms (e.g. $L1$ or $L2$ norm), on the two data points ($\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$) preprocessed by $G(\cdot)$. In these cases there is little domain expertise involved in designing $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ other than intuition that the similarity of two data points is closely related to the norm between $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$.

The most promising type of similarity measures from this perspective are Type 3 and Type 4 where $G(\cdot)$ is learned in Type 3 and both $C(\boldsymbol{x}, \boldsymbol{y})$ and $G(\cdot)$ are learned in Type 4. However, to test any new design we need to have reference methods to compare against. For reference, we chose to implement one Type 1 similarity measure, two similarity measures of Type 2 (including Gabel et. al's) similarity measures and Chopra et. al's Type 3 similarity measure. The Type 1 similarity measure uses a similarity measure that weights each feature uniformly. The Type 2 is identical to the Type 1 similarity measure except that it uses a local similarity function for each feature which is parametrized by statistical properties of the values of that feature in the dataset.

One unexplored direction of creating similarity measures is creating a SNN similarity measure (Type 3) through training $G(\cdot)$ as a classifier on the dataset later being used for measuring similarity. Then using that trained $G(\cdot)$ to construct a SNN similarity measure. This is in contrast to the usual way of training SNNs (as seen in e.g [37, 48]) where the loss function is a function of pairs of data points, not single data points. The motivation for exploring this type of design is that it shows the similarity measuring performance of using networks pre-trained on classifying data points directly as part of a SNN similarity measure. This will be detailed in Subsection II.4.2.

Finally, we will explore Type 4 similarity measures which have seen little attention in research so far. To make our design as symmetric as possible we will use the same design as SNNs for $G(\cdot)$ and introduce novel design to also make $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ symmetric. That way our design is fully symmetric (invariant to ordering of the input pair) and thus training becomes much more efficient. All of the details of this design will be shown in Subsection II.4.3. Both of our proposed similarity methods implement $G(\cdot)$ as neural networks. The Type 4 measurement design implements $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ as a combination of a static binary function and neural network.

II.4.1 Reference similarity measures

As a reference for our own similarity measure we implemented several reference similarity measures of Type 1, Type 2 and Type 3. First we implemented a standard uniformly weighted global similarity ($t_{1,1}$) measure which can be defined as:

$$t_{1,1}(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{S}(\boldsymbol{x}, \boldsymbol{y}) = C(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i}^{M} \boldsymbol{w}_i \cdot sim_i(\boldsymbol{x}_i, \boldsymbol{y}_i), \tag{II.2}$$

where $sim_i(\boldsymbol{x}_i, \boldsymbol{y}_i)$ denotes the local similarity of the i-th of M attributes. In $t_{1,1}$ all weights and local similarity measures are uniformly distributed, and not parametrized by the data.

We extended this with a Type 2 similarity measure $t_{2,1}$, which is based on the work from Abdel-Aziz et al. [47], where the local similarity measures are parametrized by the data from the corresponding features.

Furthermore we implemented a Type 2 similarity measure *gabel* as described by Gabel et al. [38]. The architecture of *gabel* can be seen in Figure II.11.

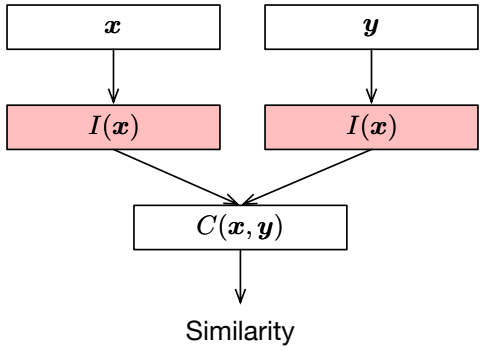


Figure II.11: Architecture of a ANN similarity measure as used in Gabel [38] (Type 2), where $G(\cdot)$ is set to be the identity function $G(\boldsymbol{x}) = I(\boldsymbol{x}) = \boldsymbol{x}$.

Lastly we implemented the Type 3 similarity measure *chopra* described by Chopra et al. We did not implement the extension done to the contrastive loss function as seen in [84, 85] as the change in the training dataset would be too big. This change would make comparisons between the methods harder to justify. Also none of these works showed any comparisons with previous SNNs in terms of any increased performance in relation to regular contrastive loss.

II.4.2 Type 3 similarity measure

In this subsection we will detail how we model the Type 3 similarity measure $t_{3,1}$ which uses an embedding function $G(\cdot)$ trained as a classifier. This embedding function maps the input point, $\boldsymbol{x}$, to an embedding space (see Figure II.10) which dimensions represents the probabilities of $\boldsymbol{x}$ belonging to a class. We then model the similarity measure between two points as the a static function ($C(\cdot)$ between their two respective embeddings.

For this we choose the $L2$ norm. So replacing $C(\cdot)$ for $L2$ in Equation II.1: $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = \|\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}}\|_2$, we can redefine Equation II.1 to be:

$$\mathbb{S}(\boldsymbol{x}, \boldsymbol{y}) = t_{3,1}(\boldsymbol{x}, \boldsymbol{y}) = 1.0 - \|G(\boldsymbol{x}) - G(\boldsymbol{y})\|_2 \tag{II.3}$$

where $G(\cdot)$ outputs the modeled solution as a n dimensional vector (the feature vector output from the network to the softmax function for n classes) for the case based on the problem attributes of data point x. This means that if the $G(\boldsymbol{x})$ evaluates the two cases as very similar in terms of classification $G(\boldsymbol{x}) \approx G(\boldsymbol{y})$ and $\|G(\boldsymbol{x}) - G(\boldsymbol{y})\| \approx 0$ then $\mathbb{S}(\boldsymbol{x}, \boldsymbol{y}) \approx 1.0$. This architecture is also illustrated in Figure II.12

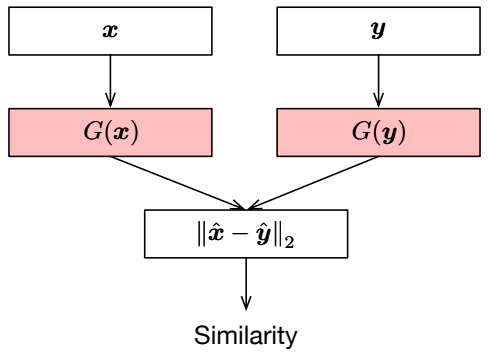


Figure II.12: Architecture of the $t_{3,1}$ similarity measure where $G(\cdot)$ is trained to output softmax vectors for classification and the similarity is calculated as a modeled $L2$ norm between these two vectors (Type 3).

Following the model for the $t_{3,1}$ similarity measure we define the loss estimate as log-loss between $G(\boldsymbol{x}) = \hat{\boldsymbol{x}}$ and $\boldsymbol{t}$, where $\boldsymbol{t}$ is the is true classification softmax vector, $\hat{\boldsymbol{x}}$ is the class probability vector output from $G(\boldsymbol{x})$. Notice that the error estimate of $t_{3,1}$ does not depend on the output of $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$.

A data-set of size N would then be defined as:

$$\boldsymbol{T} = \Big[(\boldsymbol{x}^1, \boldsymbol{t}^1) \dots (\boldsymbol{x}^N, \boldsymbol{t}^N)\Big], \tag{II.4}$$

where $\boldsymbol{x}^N$ is the problem part of the N-th data point and $\boldsymbol{t}^N$ is the solution/target part of the N-th data point.

If the relation between the problem part of the data point ($\boldsymbol{x}$) and the solution part of the data point ($\boldsymbol{t}$) is complex, the network architecture needs to be able to represent the complexity and any generalizations of patterns in that complexity.

II.4.3 Type 4 similarity measure

As previously explained, Type 4 similarity measures are currently the most unexplored type of similarity measure. It is also the type of similarity measure that requires the least amount of modeling. In principle Type 4 similarity measures learns two things: $G(\cdot)$ learns a useful embedding, where the most useful parts of $\boldsymbol{x}$ and $\boldsymbol{y}$ is encoded into $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$. $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ learns how to combine those embeddings to calculate the similarity of the original $\boldsymbol{x}$ and $\boldsymbol{y}$.

In Type 4 similarity measures both $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and $G(\cdot)$ are learned. In our Type 4 similarity method we will use an ANN to represent both $G(\cdot)$ and $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$. This has the advantage that the learning on $\mathbb{S}(\boldsymbol{x}, \boldsymbol{y})$ is an end to end process. The loss computed after $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ can be used to compute gradients for both $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and $G(\cdot)$. $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ will learn the binary combination best suited to calculate the similarity of the two embeddings, while $G(\cdot)$ will learn to embed the two data points optimally for calculating their similarity in $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$. In principle any ML method could be used to learn $G(\cdot)$ and $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$, but not all ML methods lend themselves naturally to back-propagating the error signal from $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ through $G(\cdot)$ and back to the input.

We define our Type 4 similarity method, Extended Siamese Neural Network ($eSNN$) as shown in figure II.13.

Given that this similarity method outputs similarity and the loss function is a function of the input, we get a new general loss function for similarity, defined per data-point as follows:

$$L_s(\boldsymbol{x}, \boldsymbol{y}, s) = |s - C(G(\boldsymbol{x}), G(\boldsymbol{y}))|, \tag{II.5}$$

where $\boldsymbol{s}$ is the true similarity of case $\boldsymbol{x}$ and $\boldsymbol{y}$. Since this loss function is dependent on pairs of data points and the true similarity between them, we need to create a new dataset based on the original dataset. This new dataset consists of triplets of two different data points from the original dataset and the true similarity of these two data points:

$$\boldsymbol{T} = \Big[(\boldsymbol{x}^1, \boldsymbol{y}^1, s^1) \dots (\boldsymbol{x}^N, \boldsymbol{y}^N, s^N)\Big], \tag{II.6}$$

where $\boldsymbol{s}^N$ is 1 if $\boldsymbol{x}^N$ and $\boldsymbol{y}^N$ belong to the same class and 0 otherwise.

It is worth to mention that this dataset is of size $N(N-1)$ for the similarity measure to train on all possible combinations of the N data points. Certain similarity measure architectures (e.g. *gabel* from Gabel et al.[38] or Zagoruyko et al.'s similarity measures [115]) needs to train on a dataset containing all possible combinations of data points (of size $N(N-1)$) as training on the triplet $(\boldsymbol{x}, \boldsymbol{y}, s)$ does not guarantee that the model learns that $\mathbb{S}(\boldsymbol{y}, \boldsymbol{x}) = s$. Thus the training dataset must also include the triplet $(\boldsymbol{y}, \boldsymbol{x}, s)$. However this may be largely avoided by using architectures (such as those seen in SNNs and SNs) that exploit symmetry and weight sharing. To achieve this we modeled $C(\boldsymbol{x}, \boldsymbol{y})$ as a ANN where the first lay.er is an absolute difference operator on two vectors: $\boldsymbol{z} = ABS(\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}})$. where $\boldsymbol{z}$ is the element-wise absolute difference between $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{x}}$. The rest of $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}})$ is hidden layers of ANN that operate on $\boldsymbol{z}$. This way $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}})$ becomes invariant to the ordering of inputs to $S(\boldsymbol{x}, \boldsymbol{y})$. Consequently the model only needs to train on order-invariant unique pairs of data points, reducing training set size from $N(N-1)$ to $N(\frac{N}{2} - 1)$. The resulting architecture of $eSNN$ can be seen in II.13.

In Subsection II.4.2 we argue why $G(\cdot)$ trained to correctly classify its input is a good embedding function for calculating similarity. As a result we added two loss signals to $G(\cdot)$ during training. These loss signals are calculated from the difference between the embedding of the data point produced by $G(\cdot)$ and the correct soft-max classification vector.

This also introduced an opportunity for exploring the relative importance of the embedding function $G(\cdot)$ and the binary similarity function $C(\cdot)$ in terms of the performance of the similarity measure. This could be done by weighting the three different loss signals ($\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$ and similarity as shown in Figure II.13) during training and measuring the effect of that weighting on the performance. We define our weighted loss function as such:

$$\begin{aligned} L(\alpha, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}) = \frac{(1-\alpha)}{2} \cdot (L_c(\boldsymbol{x}, \boldsymbol{t}_x) + L_c(\boldsymbol{y}, \boldsymbol{t}_y)) \\ + \alpha \cdot L_s(\boldsymbol{x}, \boldsymbol{y}, s), \end{aligned} \tag{II.7}$$

where $L_s(\cdot)$ is defined in Equation II.5, $\boldsymbol{t}_x$ is the true label of data point $\boldsymbol{x}$, $\boldsymbol{t}_y$ is the true label of data point $\boldsymbol{y}$ and $L_c(\boldsymbol{v}_1, \boldsymbol{v}_2)$ is the categorical cross entropy loss between two softmax vectors. We use

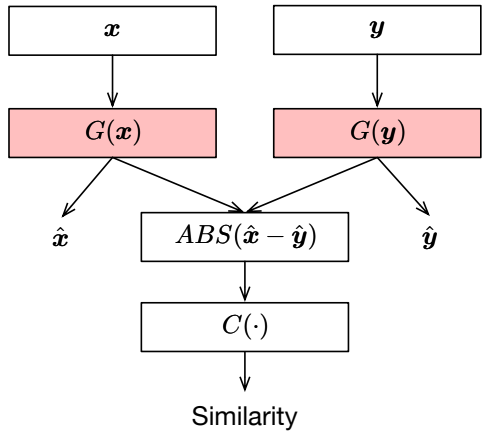


Figure II.13: Architecture of a $eSNN$ where we combine the symmetry of SNNs with the ability to learn $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$. $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ is expanded in this picture to highlight the $ABS(\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}})$ operation done as the first operation of $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ to keep C invariant to the ordering of inputs. It also illustrates the two additional loss signals to $G(\cdot)$ which helps train the similarity measure.

this formula and tested with different 100 different values of α in the range $[0, 1]$ to find the weighting scheme best for performance. The results can be seen in Figure II.14.

Figure II.14 seems to indicate that $\alpha = 0.15$ is ideal for this dataset. We have used $\alpha = 0.15$ throughout the experiments reported in Section II.5.

II.4.4 Network parameters

For all similarity measures tested using ANN and all datasets except MNIST, $G(\cdot)$ and $C(\cdot)$ where implemented with two hidden layers of 13 nodes. This was done to replicate the network parameters used by Gabel et al. to ensure we had comparable results. For the MNIST dataset test both *chopra* and $eSNN$ used three hidden layers of 128 nodes for $G(\cdot)$, and the same for $C(\cdot)$

Other than the network architecture we also wanted to choose which optimizer to use for learning the ANN model. We wanted to chose the RProp [117] to be more comparable with the results from Gabel et al. which also used RProp optimizer. Our tests seen in Figure II.15 shows that RProp outperforms all other optimizer tested (ADAM and RMSProp). This is consistent with the results reported by Florescu and Igel [118]. This should hold true until the the run-

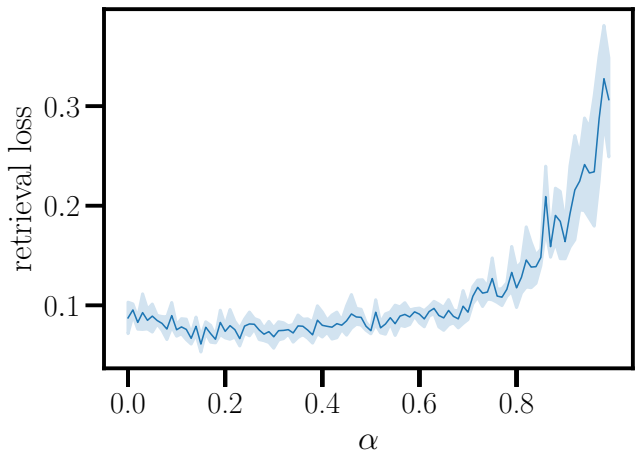


Figure II.14: Showing results from weighting the three different output in terms of signal strength to loss measured on the UCI dataset balance scale [86] (5-fold cross validation and repeated 5 times). This measurement was done using training data of size $N(\frac{N}{2}-1)$. The effect of α is much less impactful on the validation result after 200 or more epochs of training when training on $N(N-1)$ datasets. However choosing the correct α using $N(\frac{N}{2}-1)$ datasets does impact the speed of training for $eSNN$ when training on $N(N-1)$ datasets.

time performance of RProp degrades with dataset size, as RProp uses full batch sizes.

II.4.5 Evaluation protocol and implementation

The different similarity measures presented earlier in this section requires different training data sets. The reference Type 1 similarity measures ($t_{1,1}$) requires no training. While $t_{2,1}$ and $t_{3,1}$ does not require a similarity training consisting of triplets as described in Equations II.6. All other similarity measures evaluated was trained using identical training datasets. As a result, all similarity measures were trained on a dataset consisting of all possible combinations of data points (as explained in II.4.3) as this is required by the *gabel* similarity measure. However, results highlighting the differences in training performance when using the different training datasets can be seen in Figure II.21.

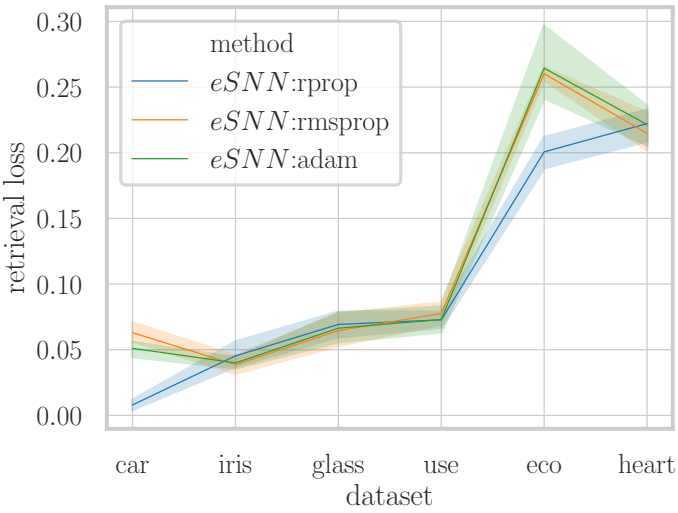


Figure II.15: Testing how the RProp algorithm performs in comparison with ADAM and RMSProp. Our proposed architecture performs best using the RProp algorithm (5-fold cross validation and repeated 5 times).

The results reported in the next section are all 5-fold stratified cross validation repeated 5 times for robustness. The performance reported is an evaluation of each similarity measurement using the part of the dataset (validation partition) that was not used for training. Using the similarity measure being evaluated, we computed the similarity between every data point in the validation partition (V) and every data point in the training partition (T). For each validation data point ($x_v \in V$) we find the data point in the training set T with the highest similarity ($x_t = \arg\max_{x_i \in T}(\mathbb{S}(x_v, x_i))$). If x_t has the same class as x_v from the validation partition, we scored it as 1.0, if not, we scored it as 0.0.

The implementation was done in Keras [4] with Tensorflow as backend. The methods was measured on 14 different datasets available from the UCI machine learning repository [86]. Results was recorded after 200 epochs and 2000 epochs (the latter number to be consistent with Gabel et al. [38]) to reveal how fast the different methods were achieving their performance.

[4] Code available at NTNU OpenAI lab github page: https://github.com/ntnu-ai-lab

II.5 Experimental evaluation

To calculate the performance of our similarity measure we chose to use the same method of evaluation as Gabel et al. [38] to make the similarity metrics more easily comparable. In addition this evaluation method of using publicly available datasets from the UCI machine learning repository [86] make the results easy to reproduce. We selected a subset of the original 19 datasets, choosing not to use regression datasets, resulting in a set of 14 classification datasets. The datasets' numerical features were all normalized, categorical features were replaced by a one-hot vector.

The validation losses from evaluating the similarity measures on the 14 datasets are shown in Figures II.16 and II.17. Figure II.16 shows the results after training for 200 epochs, while Figure II.17 shows the results after 2000 epochs. This has been done to illustrate how the differences between the similarity measures develop during training. In addition the 200 and 2000 epoch runs are independent runs (i.e. Figure II.17 is not the same models as seen in Figure II.16 1800 epochs later)

The numbers that are the basis of these figures are also reported in Table II.3 for 200 epochs and Table II.4 for 2000 epochs. The tables are highlighted to show the best result per dataset. In some cases the differences between two methods for one dataset was smaller than the standard deviation thus highlighting more than one result.

Finally, to illustrate that $eSNN$ scales to larger datasets we report results from the MNIST dataset in Figure II.18. The MNIST results are not validation results, as calculating the similarity between all the data points in the test set and the training set (as per the evaluation protocol described in Section II.4.5) was too resource demanding.

Table II.3 shows the validation losses of the different similarity measures on the different datasets. Our proposed Type 4 similarity measure $eSNN$ has 11% less validation loss than the second best (Type 3) similarity measure *chopra* (Chopra et al. [37]). The other Type 3 similarity measures follow with $t_{3,1}$ having 51% higher loss and *gabel* (Gabel et al. [38]) with 52% more loss. The Type 1 similarity measure had 61% more loss but managed to be the best similarity measure for the glass dataset. At last Type 2 similarity measure had 69% higher loss than $eSNN$ on average.

The results when training for 2000 epochs are quite different from those at 200 epochs, as seen by how much closer the other similarity measures are in Figure II.17 than in Figure II.16. $eSNN$ still outperforms all other similarity measures on average, but the second best similarity measure $t_{3,1}$ is much closer with just 6.9% higher loss. *gabel* is 11.8% worse, *chopra* is 14.7% worse, $t_{1,1}$ is 61.2% worse

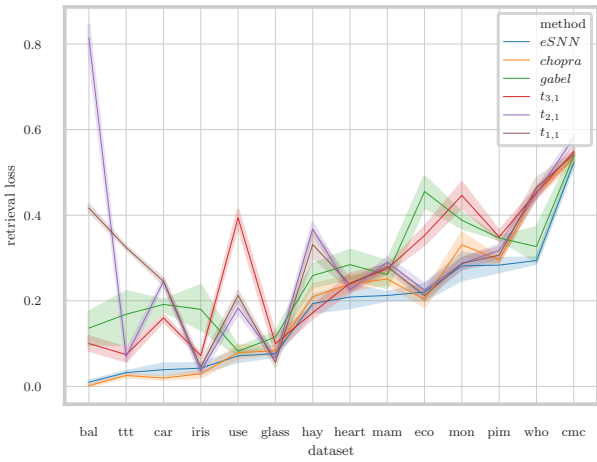


Figure II.16: Performance of $eSNN$ in comparison to reference similarity measures and state of the art similarity methods over all test datasets trained over 200 epochs.

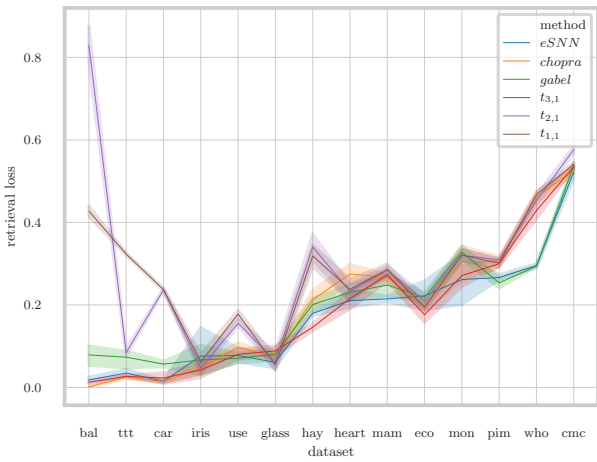


Figure II.17: Performance of $eSNN$ in comparison to reference similarity measures and state of the art similarity methods over all test datasets trained over 2000 epochs.

	$eSNN$	*chopra*	*gabel*	$t_{3,1}$	$t_{1,1}$	$t_{2,1}$
bal	0.01	**0.00**	0.14	0.10	0.42	0.81
car	0.04	**0.02**	0.19	0.16	0.25	0.25
cmc	**0.52**	0.53	0.54	0.55	0.54	0.58
eco	0.22	**0.20**	0.46	0.35	0.21	0.22
glass	0.08	0.08	0.12	0.10	**0.06**	0.07
hay	0.19	0.21	0.26	**0.17**	0.33	0.37
heart	**0.21**	0.24	0.28	0.24	0.24	0.23
iris	0.04	**0.03**	0.18	0.07	0.05	0.04
mam	**0.21**	0.25	0.26	0.27	0.28	0.29
mon	**0.28**	0.33	0.39	0.45	0.29	0.29
pim	**0.28**	0.30	0.35	0.35	0.31	0.32
ttt	**0.03**	0.03	0.17	0.07	0.32	0.07
use	**0.07**	0.08	0.08	0.39	0.21	0.18
who	**0.29**	0.45	0.33	0.45	0.46	0.45
Sum	**2.47**	2.75	3.75	3.72	3.97	4.17
Average	**0.18**	0.20	0.27	0.27	0.28	0.30

Table II.3: Validation retrieval loss after 200 epochs of training, in comparison to state of the art methods. $eSNN$ has the smallest loss in 8 of 14 datasets. The best result for each dataset is highlighted in bold.

and finally $t_{2,1}$ is 69% worse than $eSNN$.

The gap between $eSNN$ and the state of the art is considerable at 200 epochs. This gap shrinks from 11% at 200 epochs to 6.9% at 2000 epochs, which is still a considerable difference.

To illustrate the difference in terms of training efficiency between different types similarity measure, we show the validation loss for *gabel*, *chopra* and $eSNN$ during training. Specifically, for each epoch we test the loss of each similarity measure by the same method as described in subsection II.4.5. Figure II.19 and Figure II.20 shows validation loss during training of $eSNN$, *chopra* and *gabel* on the UCI Iris and Mammographic mass datasets [86] respectively. This exemplifies the training performance of these methods in relation to the Iris and Mammographic mass dataset results reported in the tables above. One can also note that in training for the Mammographics dataset as seen in Fig. II.19 *chopra* never achieves the same performance as $eSNN$. In contrast, while training on the Iris dataset (as seen in Fig. II.20), which is a less complex dataset than the Mammographic dataset, *chopra* achieves the same performance as $eSNN$.

Figure II.21 shows the validation loss during training when *chopra*

	eSNN	*chopra*	*gabel*	$t_{3,1}$	$t_{1,1}$	$t_{2,1}$
bal	0.02	**0.00**	0.08	0.01	0.43	0.83
car	**0.01**	**0.01**	0.06	0.02	0.24	0.24
cmc	**0.52**	0.53	0.54	0.53	0.54	0.58
eco	0.22	0.20	0.22	**0.18**	0.19	0.21
glass	0.06	0.07	0.08	0.09	**0.05**	0.06
hay	0.18	0.21	0.20	**0.15**	0.32	0.34
heart	**0.21**	0.27	0.23	0.22	0.24	0.23
iris	0.08	0.05	0.07	**0.04**	0.06	0.05
mam	**0.21**	0.27	0.25	0.27	0.29	0.28
mon	**0.26**	0.30	0.33	0.27	0.32	0.32
pim	0.27	0.31	**0.25**	0.30	0.30	0.31
ttt	**0.03**	**0.03**	0.07	**0.03**	0.32	0.08
use	0.08	0.10	**0.07**	0.08	0.18	0.16
who	0.30	0.46	**0.29**	0.43	0.47	0.45
Sum	**2.45**	2.81	2.74	2.62	3.95	4.14
Average	**0.18**	0.20	0.20	0.19	0.28	0.30

Table II.4: Validation retrieval loss after 2000 epochs of training, in comparison to state of the art methods. *eSNN* has the smallest validation retrieval loss in 6 of 14 datasets in addition to the lowest average loss. The best result for each dataset is highlighted in bold.

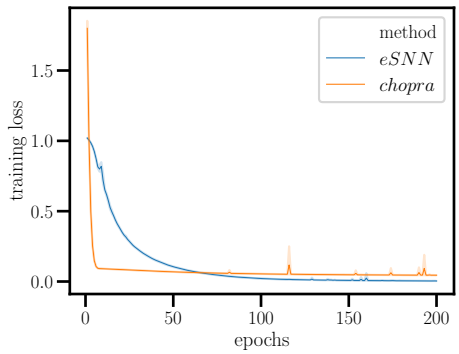


Figure II.18: Training loss (not validation retrieval loss) during training on the MNIST dataset for *chopra* and *eSNN*. *gabel* could not be evaluated as training on a $N(N-1)$ sized dataset for MNIST is too resource demanding.

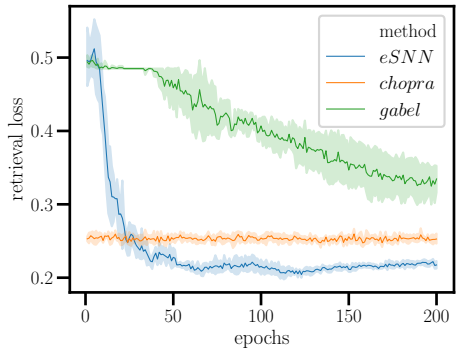


Figure II.19: Validation retrieval loss during training on the Mammographic mass UCI ML dataset [86]. The Figure shows that the mammograph dataset is a dataset that needs learning outside of embedding via $G(\cdot)$. *chopra* starts out good as $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}})$ is already designed as the $L1$ norm. However $eSNN$ and *gabel* catches up when it learns an equivalent and better $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}})$ function.

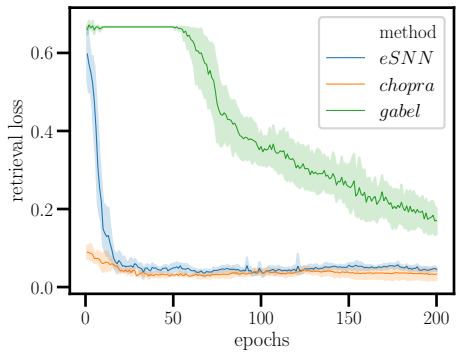


Figure II.20: Validation retrieval loss during training on the Iris UCI ML dataset [86] . Since *chopra* starts out with very low validation loss. It seems probable that the static $L1$ norm $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}})$ used by *chopra* is close to optimal for correctly identifying if the two data points belong to the same class or not. The performance increase done by *chopra* is a slight optimization of $G(\cdot)$. The performance increase done during training by *gabel* and $eSNN$ is mainly by learning a $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}})$ equivalent in function to that used by *chopra*, and secondary a slight optimization of $G(\cdot)$. $eSNN$ catches up to *chopra* in performance after around 20 epochs, however gabel takes longer (5% validation loss at 2000 epochs) as shown in Table II.4

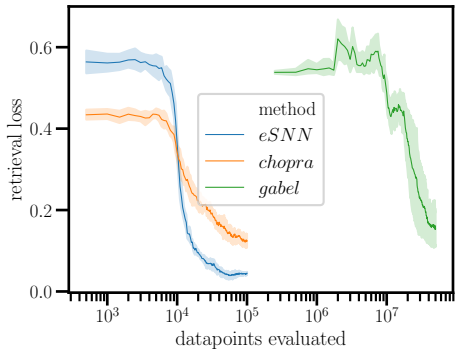


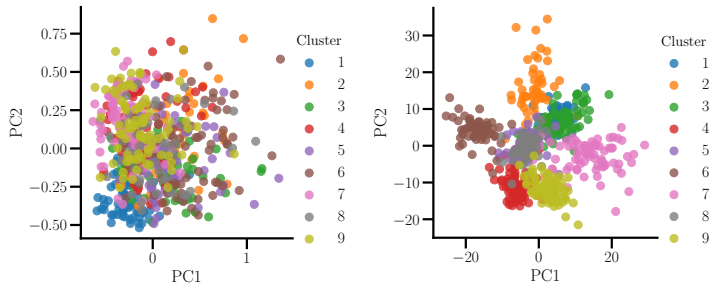
Figure II.21: Validation retrieval loss during training on the balance dataset, which illustrates the difference in amount of evaluations needed to achieve acceptable performance. Chopra achieves good performance very quickly, but is outperformed by $eSNN$ soon. Both have very good performance before having evaluated less (N) data points than used by one epoch needed by gabel ($N(N-1)$)

and $eSNN$ are using a training dataset of size N and *gabel* is using a training dataset of size $N(N-1)$. This figure illustrates how much fewer evaluations a SNN similarity measure like *chopra* or symmetric Type 4 similarity measure such as $eSNN$ needs than a similarity measurement that is not invariant to input ordering, while still having excellent relative performance.

Finally in Figure II.22 and II.23 we show how $eSNN$ can be used for semi-supervised clustering. The figures show PCA and T-SNE clustering of embeddings produced untrained and trained $eSNN$ networks respectively from the MNIST dataset. The embeddings are the vector output of $G(\cdot)$ for each of the data points in the test set. The embeddings shown are computed from a test set that is not used for training. The figures show that $eSNN$ learns a way to correctly cluster data points that it has not used for training.

II.6 Conclusions and future work

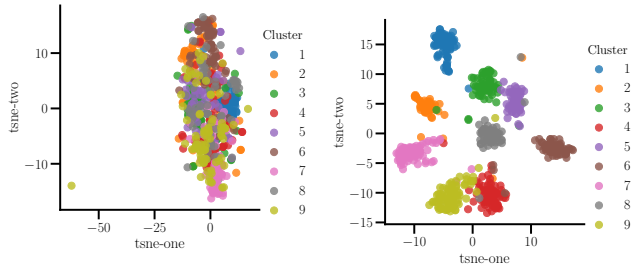
Section II.5 shows that all of the learned similarity measures outperformed the classical similarity measure $t_{1,1}$ and also $t_{2,1}$ where the local (per feature) similarity measures were adapted to the statistical properties of the features [47]. In practice one should weight the importance of each feature according to how important it is in terms of similarity measurement. In many situations the number of possible



(a) PCA clustering on the MNIST dataset before training

(b) PCA clustering on the MNIST dataset after training

Figure II.22: PCA clustering showing the two first principal components ($PCA1$ and $PCA2$) of the embeddings produced by $eSNN$ from MNIST input before (II.22a) and after (II.22b) training.



(a) T-SNE clustering of the MNIST dataset before training

(b) T-SNE clustering of the MNIST dataset after training

Figure II.23: T-SNE clustering of embeddings produced by $eSNN$ from MNIST input before (II.23a) and after (II.23b) training.

attributes to include in such a function can be overwhelming, and modeling them in the way we did in $t_{1,1}$ and $t_{3,1}$ also overlooks possible co-variations between the attributes. Both of these problems can be addressed using the proposed method to model the similarity using machine learning on a dataset that maps from case problem attributes to case solution attributes.

However one should be careful to note that all of the learned similarity measure are built on the assumption that similar data points have similar target values ($\delta_s \approx \delta_e \approx \delta_p$ in Figure II.10). If this assumption does not hold, learning the similarity measure might be much more difficult.

We have also presented a framework for how to analyze and group different types of similarity measures. We have used this framework to analyze previous work and highlight different strengths and weaknesses of the different types of similarity measures. This also highlighted unexplored types of similarity measures, such as Type 4 similarity measures.

As a result we designed and evaluated a Type 3 similarity measure $t_{3,1}$ based on a classifier. The evaluations showed that using a classifier as a basis for a similarity measure achieves comparable results to state of the art methods, while using much less training evaluations to achieve that performance.

We then combined strengths from Type 4 and Type 3 similarity measures into a new Type 4 similarity measure, called Extended Siamese Neural Networks ($eSNN$), which:

- Learns an embedding of the data points using $G(\cdot)$ in the same way as Type 3 similarity measures, but using shared weights in the same way as SNNs to make the operation symmetrical.
- Learns $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$, thus enabling extended performance in relation to SNN and other Type 3 similarity measurements.
- Restricts $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ to make it invariant to input ordering, and thus obtaining end to end symmetry through the similarity measure.

Keeping $eSNN$ symmetrical end-to-end enables the user of this similarity measure to train on much smaller datasets than required by other types of similarity measures. Type 3 measures based on SNNs also have this advantage, but our results show that the ability to learn $C(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ is important for performance in many of the 14 datasets we tested. Our results showed that $eSNN$ outperformed state of the art methods on average over the 14 datasets by a large margin. We also demonstrated that $eSNN$ achieved this performance much faster given the same dataset than current state of the art. In addition, the symmetry of $eSNN$ enables it to train on datasets that are orders of magnitude smaller. Our case-study of clustering embeddings produced from $eSNN$ show that the $eSNN$ model can be used for semi-supervised clustering.

Finally we demonstrated that the training of this similarity measure scales to large datasets like MNIST. Our main motivation for this work was to automate the construction of similarity measures while keeping training time as low as possible. We have shown that $eSNN$ is a step towards this. Our evaluation shows that it can learn similarity measures across a wide variety of datasets. We also show

that it scales well in comparison to similar methods and scales to datasets of some size such as MNIST.

The applications for $eSNN$ as a similarity measure are not only as a similarity measure in a CBR system. It can also be used for semi-supervised clustering: training $eSNN$ on labeled data, then use the trained $eSNN$ for clustering unlabeled data. In much the same fashion it could be used for semi-supervised clustering, using $eSNN$ as a matching network in the same fashion as the distance measure is applied in Vinyals et al. [28].

In continuation of this work we would like to explore what is actually encoded by learned similarity measures. This could be done by varying the different features of a query data point $\boldsymbol{q}$ in $\mathbb{S}(\boldsymbol{x}, \boldsymbol{q})$ and discovering when that data point would change from one class to another (when the class of the closest other data point changes) - this would form a multi-dimensional boundary for each class. This boundary could be explored to determine what the similarity measure actually encoded during the learning phase.

Another interesting avenue of research would be to apply recurrent neural networks to embed time series into embedding space (see Figure II.10) to enable the similarity measure to calculate similarity between time series which is currently a non-trivial problem.

The architecture of similarity measures still require more investigation, e.g. is the optimal embedding from $G(\cdot)$ different from the softmax classification vector used in normal supervised learning? If so it is worth investigating why it is different.

II.7 Acknowledgements

We would like to thank the EXPOSED project and NTNU Open AI Lab for the support to do this work. Thanks also to Gunnar Senneset and Hans Vanhauwaert Bjelland for their great support during our work.

References

[28] Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. "Matching networks for one shot learning". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3630–3638.

[34] Hoffer, E. and Ailon, N. "Deep metric learning using triplet network". In: *International Workshop on Similarity-Based Pattern Recognition*. Springer. 2015, pp. 84–92.

[37] Chopra, S., Hadsell, R., and LeCun, Y. “Learning a similarity metric discriminatively, with application to face verification”. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 539–546.

[38] Gabel, T. and Godehardt, E. “Top-down induction of similarity measures using similarity clouds”. In: *Case-Based Reasoning Research and Development*. Ed. by Hüllermeier, E. and Minor, M. Cham, 2015, pp. 149–164.

[42] Stahl, A. “Learning feature weights from case order feedback”. In: *Proceedings of the 4th International Conference on Case-Based Reasoning (ICCBR 2001)*. Springer. Vancouver, 2001, pp. 502–516.

[43] Nikpour, H., Aamodt, A., and Bach, K. “Bayesian-supported retrieval in BNCreek: A knowledge-intensive case-based reasoning system”. In: *International Conference on Case-Based Reasoning*. Springer. 2018, pp. 323–338.

[44] Stahl, A. and Gabel, T. “Using evolution programs to learn local similarity measures”. In: *Proceedings of the 5th International Conference on Case-Based Reasoning (ICCBR 2003)*. Trondheim, 2003, pp. 537–551.

[45] Langseth, H., Aamodt, A., and Winnem, O. M. “Learning retrieval knowledge from data”. In: *Sixteenth International Joint Conference on Artificial Intelligence, Workshop ML-5: Automating the Construction of Case-Based Reasoners. Stockholm*. Citeseer. 1999, pp. 77–82.

[46] Reategui, E. B., Campbell, J. A., and Leao, B. F. “Combining a neural network with case-based reasoning in a diagnostic system”. In: *Artificial Intelligence in Medicine* vol. 9, no. 1 (1997), pp. 5–27.

[47] Abdel-Aziz, A., Strickert, M., and Hüllermeier, E. “Learning solution similarity in preference-based CBR”. In: *Proceedings of the 22nd International Conference on Case-Based Reasoning (ICCBR 2014)*. Springer. 2014, pp. 17–31.

[48] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. “Signature verification using a" siamese" time delay neural network”. In: *Advances in neural information processing systems*. 1994, pp. 737–744.

[49] Maggini, M., Melacci, S., and Sarti, L. “Learning from pairwise constraints by similarity neural networks”. In: *Neural Networks* vol. 26 (2012), pp. 141–158.

[84] Berlemont, S., Lefebvre, G., Duffner, S., and Garcia, C. "Siamese neural network based similarity metric for inertial gesture classification and rejection". In: *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*. Vol. 1. IEEE. 2015, pp. 1–6.

[85] Lefebvre, G. and Garcia, C. "Learning a bag of features based nonlinear metric for facial similarity". In: *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*. IEEE. 2013, pp. 238–243.

[86] Dheeru, D. and Karra Taniskidou, E. *UCI Machine Learning Repository*. 2017.

[101] Wienhofen, L. W. M. and Mathisen, B. M. "Defining the initial case-base for a cbr operator support system in digital finishing". In: *Case-Based Reasoning Research and Development: 24th International Conference, ICCBR 2016, Atlanta, GA, USA, October 31 - November 2, 2016, Proceedings*. Ed. by Goel, A., Díaz-Agudo, M. B., and Roth-Berghofer, T. Cham, 2016, pp. 430–444.

[102] Bergmann, R. *Experience management: foundations, development methodology, and internet-based applications*. 2002.

[103] Leake, D. B. *Case-Based Reasoning: Experiences, lessons and future directions*. 1996.

[104] Martin, K., Wiratunga, N., Sani, S., Massie, S., and Clos, J. "A convolutional siamese network for developing similarity knowledge in the SelfBACK dataset". In: *Proceedings of the 25th International Conference on Case-Based Reasoning Workshops (CBRDL 2017)*. CEUR Workshop Proceedings. Trondheim, 2017, pp. 85–94.

[105] Hüllermeier, E. and Schlegel, P. "Preference-based CBR: First steps toward a methodological framework". In: *International Conference on Case-Based Reasoning*. Springer. 2011, pp. 77–91.

[106] Hüllermeier, E. and Cheng, W. "Preference-based CBR: General ideas and basic principles". In: *IJCAI*. 2013, pp. 3012–3016.

[107] Cunningham, P. "A taxonomy of similarity mechanisms for case-based reasoning". In: *IEEE Transactions on Knowledge and Data Engineering* vol. 21, no. 11 (2009), pp. 1532–1543.

[108] Tversky, A. "Features of similarity." In: *Psychological review* vol. 84, no. 4 (1977), p. 327.

[109] Stahl, A. and Gabel, T. "Optimizing similarity assessment in case-based reasoning". In: *Proceedings of the National Conference on Artificial Intelligence*. Vol. 21. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. 2006, p. 1667.

[110] Shawe-Taylor, J. "Symmetries and discriminability in feedforward network architectures". In: *IEEE Transactions on Neural Networks* vol. 4, no. 5 (1993), pp. 816–826.

[111] Hadsell, R., Chopra, S., and LeCun, Y. "Dimensionality reduction by learning an invariant mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. IEEE. 2006, pp. 1735–1742.

[112] Koch, G., Zemel, R., and Salakhutdinov, R. "Siamese neural networks for one-shot image recognition". In: *ICML Deep Learning Workshop*. Vol. 2. 2015.

[113] Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. "Human-level concept learning through probabilistic program induction". In: *Science* vol. 350, no. 6266 (2015), pp. 1332–1338.

[114] He, K., Zhang, X., Ren, S., and Sun, J. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[115] Zagoruyko, S. and Komodakis, N. "Learning to compare image patches via convolutional neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 4353–4361.

[116] Arandjelovic, R. and Zisserman, A. "Look, listen and learn". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2017, pp. 609–617.

[117] Riedmiller, M. and Braun, H. "A direct adaptive method for faster backpropagation learning: The RPROP algorithm". In: *Neural Networks, 1993., IEEE International Conference on*. IEEE. 1993, pp. 586–591.

[118] Florescu, C. and Igel, C. "Resilient backpropagation (Rprop) for batch-learning in TensorFlow". In: *ICLR 2018 workshop permission* (2018), To appear in.

FishNet: A Unified Embedding for Salmon Recognition

Bjørn Magnus Mathisen, Kerstin Bach, Espen Meidell, Håkon Måløy, Edvard Schreiner Sjøblom

III

This work was supported by the Research Council of Norway through the EXPOSED project(grant number 302002390) and the Norwegian Open AI Lab

III

Abstract

Identifying individual salmon can be very beneficial for the aquaculture industry as it enables monitoring and analyzing fish behavior and welfare. For aquaculture researchers identifying individual salmon is imperative to their research. The current methods of individual salmon tagging and tracking rely on physical interaction with the fish. This process is inefficient and can cause physical harm and stress for the salmon. In this paper we propose FishNet, based on a deep learning technique that has been successfully used for identifying humans, to identify salmon.We create a dataset of labeled fish images and then test the performance of the FishNet architecture. Our experiments show that this architecture learns a useful representation based on images of salmon heads. Further, we show that good performance can be achieved with relatively small neural network models: FishNet achieves a false positive rate of **1%** and a true positive rate of **96%**.

III.1 Introduction

The Atlantic salmon farming industry in Norway has experienced a massive growth in the past four decades. The industry has gone from producing 4.300 tonnes of salmon in 1980, to almost 1.240.000 tonnes in 2017 [119]. In 2017, the total economical results from salmon production was calculated to be over 61 billion Norwegian kroner (NOK) [119]. This makes salmon farming one of the most profitable industries in Norway, and it is considered as one of the most important industries in a post oil Norway [120]. However, the industry is still largely driven by manual labor. For example, the total number of lice in a breeding cage is indicative of fish welfare and an important metric for deciding whether delousing measures should be initiated. Today's method for lice counting relies on manually inspecting individual fish and then estimating the total number of lice in the cage from these numbers. Other measurements such as disease and weight measurements also use similar methods, based on a few individual fish measurements. These methods are highly reliant on the fish inspected to be representative for the total population within the cage. However, salmon is a schooling fish and organize themselves according to hierarchical structures [121, 122]. This means that different types of individuals will be present at different layers of the school. As the sampling methods used in the industry relies on small samples the methods are prone to selecting the same type of individuals for inspection every time. This could result in skewed estimations and lead to wrong operations being performed. As these operations are

often both costly and harmful for the fish, large economic gains can be made from more precise estimates.

To improve measurement quality, a method of ensuring that different individuals are measured every time is needed. Previous attempts at solving this problem have included a variety of techniques. However, the techniques have almost exclusively relied on physical engagement with the salmon. The techniques include surgical implantation of tags and external mutilation, such as fin-clipping, freeze brands, tattoos, visible implant tags, and external tag identifiers attached by metal wire, plastic, or string [123]. This is a problem both from an animal welfare and product quality perspective. Bacterial growth and unpleasant sensory properties has shown to increase more quickly in salmon experiencing stress in their lifetime prior to being slaughtered. This results in reduced shelf life of the finished product [124]. A computer vision method for uniquely identifying individuals would solve this problem by minimizing the impacts from invasive techniques.

In this paper, we introduce an approach for accurately identifying individual salmon from images, using a deep neural network called FishNet. By accurately identifying individual salmon, we can ensure that no salmon is measured multiple times, thereby guaranteeing a more accurate estimation of the total population. Our approach is based on FaceNet [30] and DeepFace [125] which have been proven to work well in the field of face verification in humans. These networks are able to verify the identity of people in images with human levels of accuracy. They have also been shown to be robust to noise in terms of changing lighting conditions. By training a similar architecture on images of fish rather than humans, we enable accurate identity predictions without physical interaction.

Being able to track salmon at an individual level could enable tracking a single individual throughout its lifespan, from salmon spawn to finished product, linking salmon fillets to the life-story of the individual. Other opportunities include monitoring individual weight development, treating salmon only when the need arise and delousing only the individuals that suffer from lice, thereby preventing unnecessary harm to healthy salmon. Individual salmon tracking could also enable new research areas that require monitoring of individuals over time such as feeding behavior, detection of diseases and social behavior. FishNet can facilitate such research through offering a non-invasive and efficient approach to identifying salmon.

The rest of this paper is structured in the following way. In Section III.2 we outline the current state-of-the-art within the problem area of individual recognition of salmon. And as a results of the method chosen to solve the problem we also outline the current state-of-the-

art of using machine learning to identify individuals from pictures. Following this, we present our approach to the problem of individually recognizing salmon in Section III.3. The dataset used for evaluation and the evaluation of our proposed solution are presented in III.4. We present a discussion of our results in Section III.5. Finally Section III.6 presents our conclusions and thoughts on future directions of research for this work.

III.2 Related Work

Since the problem we address is inter-disciplinary, related work is two-fold: one area of research covers the detection and identification of fish and salmon in particular while the other one focuses on the classification of images. In this section well discuss the relevant work representing the state-of-the-art.

There has only been very limited work conducted to identify unique fish/salmon without engaging directly with the fish. Earlier attempts of uniquely identifying salmons have relied on insertion of RF-ID chips or other physical marking systems [126]. This is approach is only feasible in research settings and should be minimized as it potentially injures the fish. In real-world deployments with hundreds of cages and millions of fish a more scalable approach is desirable. Throughout the recent years the field of automatically identifying salmon has grown as the fish-farming industry collaborate more and more with data-driven approaches. Especially in Norway, projects such as the Exposed Aquaculture Operations Center for research based innovation[5] or the Seafood Innovation Cluster[6] emphasize on applying Internet of Things, Big Data and Artificial Intelligence methods.

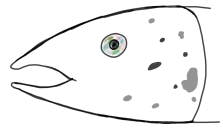


Figure III.24: Melanin patterns on a salmon head.

SINTEF SalmID [126] is a study that investigated the possibility of recognizing individual salmon based on the assumption that each individual has an unique pattern. They found that there was done little work on this area regarding Atlantic salmon, but point at other work using the melanophore pattern of different animals to uniquely

[5] https://exposedaquaculture.no/en/
[6] http://www.seafoodinnovation.no/

identify them. The constellation of such melanin patterns on the head of the fish can be utilized for identification. In the SalmID approach the recognition part is based on manually selected features of the salmon rather than learned representations.

Additional work utilizing melanophore patterns has been presented by Hammerset [127] who apply deep neural networks to discover the location of salmon heads and eyes. In this work a simple blob detection algorithm is used to discover the melanophore spots. The locations of the spots and the eye are then translated into a polar representation which is saved in a database with the identity of the salmon. On the test set with images from 333 individuals the algorithm recognized 40.4% (5922 of 14652 images) of the images as belonging to an individual salmon, of these 40.4% the algorithm correctly identified the individual with an accuracy of 99.7% (5902 of 5922). Thus the total test accuracy was 40.2% (5902 of the total 14652 images classified as the correct individual)

Identifying individuals among humans has been an active research field for a long time. Earlier work has been based on eigen value analysis of data matrices such as EigenFaces [128] and its successors in FisherFaces [129] and Laplacianfaces [130].

More recent work on individual recognition is based on deep learning approaches such as the model presented in the DeepFace paper [125] in which they are making the images of faces more uniform (frontalization). These frontalizations are fed into a convolutional layer followed by a max pooling layer and another convolutional layer. According to the authors, these three layers mainly extract low level features and make the network robust to local translations. The last convolutional layer is followed by three locally connected layers. This is done because the different regions of an aligned image have different characteristics, so the spatial invariance assumption of convolution does not hold. An example of this is that the eyes of a person will always be above the nose. The final two layers of the network they use are fully connected. These layers are able to identify relations between features in different locations in the feature maps. The first fully connected layer is used as the face representation vector, and the output of the second one is fed into a softmax which produces a class distribution for an input image. To verify whether two images are of the same person, the authors propose three approaches: (1) an unsupervised method in which the similarity of two images is simply defined as the inner product of the two representation vectors, (2) a weighted χ^2 distance in which the weight parameters are learned using a linear support vector machine and (3) a siamese network, in which the network (except the top layer used for softmax classification) is duplicated. One image is fed into each part of the network and the

absolute difference between the feature vectors is computed. A fully connected layer is added and the network is trained to predict a single logistic unit (whether the images are of the same person). Training is only enabled for the new layers, and they are trained using standard cross entropy loss. All three methods yielded good results compared to the state-of-the-art at the time. The siamese network approach required a lot more training data to avoid overfitting compared to the other approaches.

A related approach has been presented as FaceNet [30] which describes a system that learns and optimizes a vector representation directly, rather than extracting the representation from a bottleneck layer (like DeepFace). FaceNet learns a 128-dimensional feature vector (embedding) that represents a face. Unlike the DeepFace approach there is no 2D or 3D alignment done on the images. FaceNet is a variant of a Siamese Neural Network (SNN) originally proposed by Bromley [48]. In contrast with the original SNNs FaceNet uses triplet loss to train the network. The network is presented with three images (the anchor image, the positive image (same person, but different image and the negative image (image of any other person). According to the authors, it is important to select triplets that are hard for the model to correctly discriminate, to ensure that the network converges as quickly as possible during training. The triplets are chosen from within each mini-batch, and all anchor-positive pairs are used in combination with negative examples. The authors describe several different deep neural network architectures, where the major differences between them are the number of trainable parameters. The number of parameters in the networks range from about 4 million to 140 million. When evaluating the networks the L_2-distance between two images is compared. If the distance is above a certain threshold they are classified as different. According to the authors they are able to reduce the error reported by the DeepFace paper by a factor of seven. The smaller inception networks perform nearly as good as the very deep networks.

III.3 The FishNet Approach

To recognize individual salmons we adapt the FaceNet [30] architecture and training method. FaceNet is a type of Siamese neural network[48, 50] which has two datapoints as input, and the output is the distance between them. This can also be extended to work on e.g. triplets of data points, outputting more than one distance. FaceNet is trained on a dataset consisting of triplets consisting of a anchor data point, a positive data point and a negative data point. The anchor

data point with a given label, the positive data point is a different data point with the same label, in contrast the negative data point has a different label. Figure III.25 illustrates this with three example images of salmon, two of which are from the same individual salmon, while the third image is of another individual salmon.

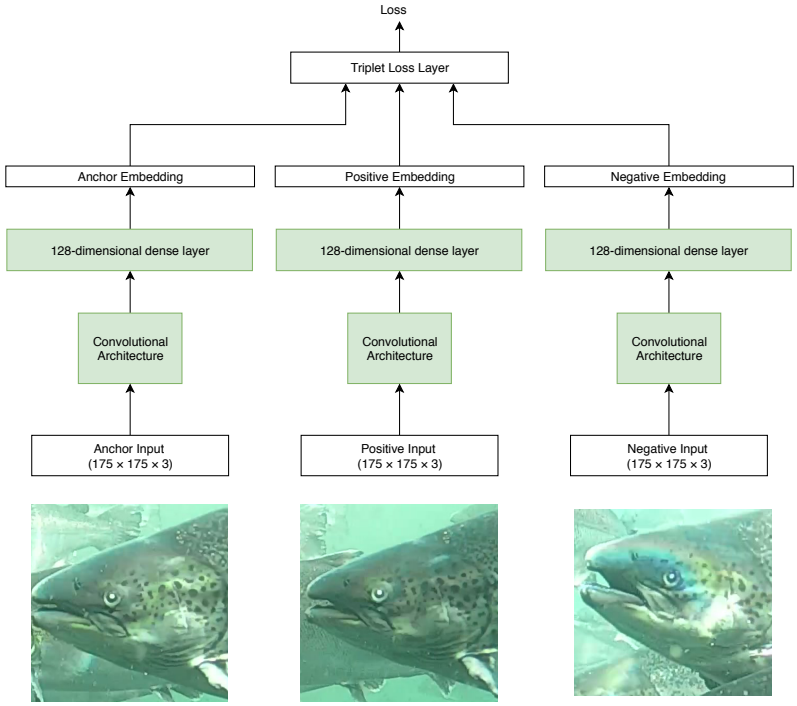


Figure III.25: Generic architecture with triplet loss. Parts of the network with shared weights are colored green. the input size is the size of the images (175x175x3) and the output is the 128-length embedding vector. The differences between the model architectures tried in our experiments is how the convolutional architecture is modeled, and the size of that convolutional model. This figure shows an example of salmon heads, with the anchor input to the left, positive input (same individual as the anchor input) in the center and finally the negative input (different indivdual than the anchor input).

The goal of training FaceNet is to minimize the distance between the anchor and positive data point, while maximizing the distance to the negative data point. This training process is illustrated in Figure III.26.

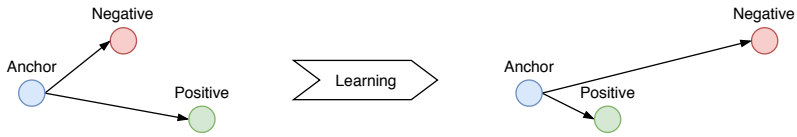


Figure III.26: Triplet loss minimizes the distance between images of the same salmon and maximizes the distance to images of other salmon (adapted from [30]).

To compute the loss during the training, a custom triplet loss layer was used. Equation III.8 defines how the loss L is computed for a minibatch of size m.

$$L = \sum_{i}^{m} \left[\|\hat{x}_i^a - \hat{x}_i^p\|_2^2 - \|\hat{x}_i^a - \hat{x}_i^n\|_2^2 + \alpha \right]_+ \tag{III.8}$$

Here $\hat{x} = f(x)$ is the embedding of image x, x^a is the anchor data point, x^p is the positive data point, x^n is the negative datapoint and α is a parameter that encourages better learning.

This is identical to how the triplet loss is defined in the FaceNet [30]. The loss computes the distance between the anchor and the positive, and the anchor and the negative. The goal is to have the positive distance be smaller than the negative distance. The difference between the positive and negative distance are summed. To encourage larger distances the margin α is added to the loss function. To avoid negative loss, the loss is set to the maximum of the loss of the triplet and 0.

Careful triplet selection is important [30] for the training process of the network. The training goal of the algorithm is to ensure that the embeddings of two images (anchor and positive) of the same salmon are closer to each other than any images of other salmon (negatives) by a margin α. In our experiments, the value for α was set as 0.2, the same as used in the FaceNet paper.

To ensure effective training, it is important to select triplets that violate this constraint. To do this, the method computes the embeddings for images during training, and then select samples only among the triplet that violate this training samples. For efficiency purposes, this is done within each batch. First, a random set of salmon images are sampled from the training dataset. Then the images are fed through the network to generate embeddings. Finally, the embeddings are used to select triplets where the difference between the negative and positive embeddings are within α. Algorithm 1 describes this

process.

Algorithm 1: Triplet selection

```
Input: embedding vectors
Input: number of fish
Input: number of embeddings per fish
Input: α
Data: triplets = []
foreach fish do
    for anchor in embeddings of current fish do
        negative distances = L2-distances from anchor to
         embeddings of other fish
        for positive in remaining embeddings of current fish do
            compute distance between anchor and positive
            negatives = find all negative embeddings where
             negative_dist − positive_dist < α
            select a random negative from negatives and
             append (anchor, positive, negative) to triplets
        end
    end
end
shuffle triplets
return list of triplets
```

Using Algorithm 1 to select the triplet used for training, we ensure that training is performed on triplets the network can learn from. Using triplets that already satisfy the constraint of α would not contribute to further training, and only slow down the process. Calculating the hardest triplets for the entire dataset every epoch would be computationally very slow. Additionally, if we were to select the hardest triplets every time it could cause poor training. This is because selection of hardest triplets would be dominated by for example mislabeled or low quality images.

III.3.1 Neural Network Architectures

During our experiments, we trained different neural network architectures to produce embeddings. All the networks shared a general architecture of a convolutional neural network where the top layer (classification layer) was replaced by a 128-dimensional dense layer to represent the embedding of the input image. Figure III.25 shows an illustration of this architecture, which is used to compute the em-

Network Architecture	# Parameters	Pretrained with
FishNet1 (Inception ResNet v2)	55M	ImageNet
FishNet2 (MobileNet v2)	2.4M	ImageNet
FishNet3 (VGG-16)	15M	ImageNet

Table III.5: The different neural network architecture models used in the experiments. From a large model (FishNet1 based on Inception ResNet v2) to a model that is 20 times smaller (FishNet2 based on MobileNet v2) that can be deployed on a mobile device.

bedding for one image. Table III.5 we show the different types of architectures we evaluated as part of this work. This was done to investigate the effect of using different convolutional architectures and model sizes to produce embeddings. The corresponding results to the architectures listed in this table is listed in Table III.6.

To train the network using triplet loss, the network needs to use more than one image at once. To achieve this, the convolutional and embedding parts need to be replicated once for each image. Note that the weights are shared between the instances. The output from the embedding layers is fed into a custom layer that computes the triplet loss, which in turn is used to train the model. Figure III.25 illustrates the model used for training. Table III.5 shows the neural network architectures used in the experiments.

All models were initialized with the convolutional weights pretrained on the ImageNet dataset [131]. The assumption being that features learned for image classification may be a useful starting point for learning how to distinguish salmon from each other, thereby reducing the amount of training data needed to train the models.

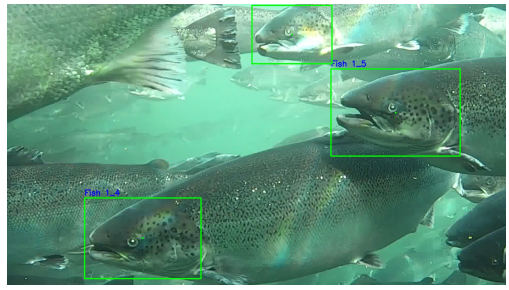
III.4 Dataset and Evaluation

As far as we know, there is currently no data set of labeled fish to use for training and evaluating methods for identity recognition. Thus, to evaluate the FishNet method we needed to create a dataset of labeled pictures of salmon heads. To do this we aquired a video clip of salmon swimming from SeaLab[7].

The original data was a video stream of salmon swimming across the view of the camera. The video was filmed at 30 FPS (frames per second) meaning we had 30 images per second of video. Figure III.27a shows a frame captured from this video. Salmon heads in the images were marked manually with a bounding-box tool. After manually labeling approximately 500 bounding-boxes as salmon heads

[7] https://www.sealab.no/

(a) An example of a frame from the original video.



(b) An example of bounding-boxes of salmon heads detected by the YOLOv3 model trained to detect salmon heads.

Figure III.27: Overview of different stages the dataset creation.

the bounding-boxes were used to train a YOLOv3 ([82]) network to recognize salmon heads. This YOLOv3 model was then used to create bounding-boxes on every salmon head in all video frames, as seen in Figure III.27b. Figure III.25 and Figure III.37 shows examples of the resulting cropped bounding-boxes of two salmon heads. These bounding boxes is then extracted as a $175x175$ image. These images are then clustered to achieve clusters of images for each individual salmon. Equation III.9 describes the distance function used in the clustering algorithm. If two bounding-boxes are in the same frame, the distance is set to an arbitrarily high value. If the bounding-boxes are not in the same frame, the intersection over union is measured to check how closely the bounding-boxes overlap. Then a temporal distance is added by computing a weighted distance of the frame numbers. This is done to ensure that overlapping bounding-boxes in frames next to each other receive a low distance value. These distance metrics are combined into one single distance (Equation III.9) metric which is used by DBSCAN[83] to cluster the images. This process produces clusters of images of the same individual salmon. This approach works

fairly well except in cases where a salmon disappears behind a different salmon and then reappears again. In those cases it is frequently misidentified as a new salmon. This problem was solved by manually reviewing the labels, and replacing the labels for misidentified salmon.

$$D(b1, b2) = \begin{cases} \infty & : \delta_f = 0 \\ \frac{1-IOU(b1,b2)+}{2} & : \text{otherwise} \end{cases} \tag{III.9}$$

Here $\delta_f = b1_{frame} - b2_{frame}$ and

$$IOU(b1, b2) = \frac{\text{Intersection Area}}{\text{Union Area}}$$

After the salmon heads are extracted they were clustered and finally labeled. This resulted in a dataset of 15000 images of 715 individual salmon. The images were then augmented by tilting the image, moving the image vertically and shifting the brightness of the image. Examples of these augmentations can be seen in Figure III.28. Five augmented images were created for each original image, resulting in a data set containing a total of 225 000 images divided over 715 individuals. The data set was then divided into test and training sets, with 90% of the images being used for training and the remaining 10% being used for testing.

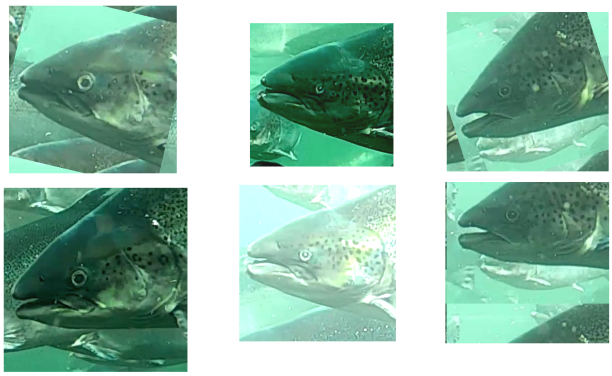


Figure III.28: Augmenting images during the dataset creation. The top right and top left images show tilting augmentation. Centre bottom is color shifting (making the image brighter).

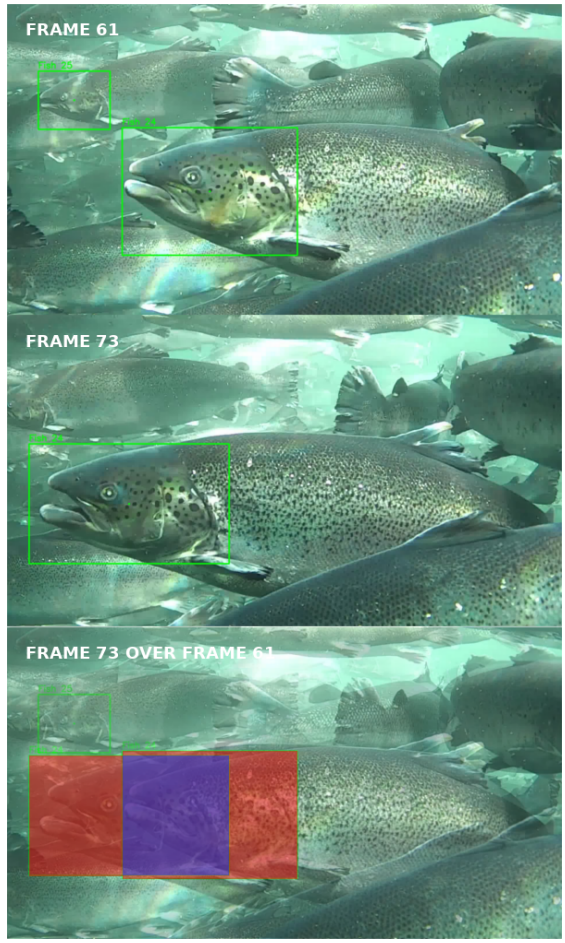


Figure III.29: Illustration of IOU. The top image is frame 61 in the video, the middle is from frame 73, and the bottom image shows the two images stacked on top of each other. Despite being 12 frames apart, the IOU is still quite high. The red and blue area is the union between the bounding boxes, and the blue area alone is the intersection.

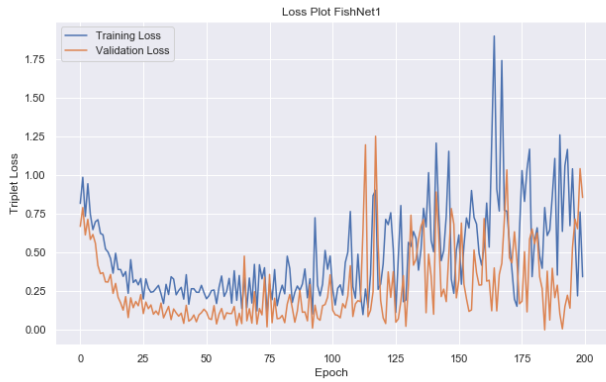


Figure III.30: The loss curves during training for FishNet1.

III.4.1 Evaluation

The experimental setup consisted of a single computer containing an AMD Ryzen Threadripper 2920x 12-core CPU, two GeForce RTX 2080Ti GPUs and 128 GB of RAM. The models were implemented using Tensorflow [132]. Figures III.30, III.31, and III.32 show the loss curves during the training of the three models presented in section III.3.1. One notable observation in the loss curves for FishNet1 is that both the training and validation loss start to fluctuate and increase greatly towards the middle and end of training. This occurs due to the nature of the triplet selection algorithm used during the training phase. The algorithms only uses triplets that fail the triplet constraint test described in section III.3. This means that if the model learns to separate salmon well, there are fewer triplets available for training as the training progresses. By examining the training logs we can see that this in fact happens. Figure III.33 shows show many of the sampled triplets the network was able to use for training.

As seen in Figure III.30, when the models become increasingly good at recognizing individuals the loss starts to fluctuate. This is most likely due to the fact that the training process is down to a very small set of triplets that is very hard to discriminate. When the training process seeks for a model that can discriminate these last triplets, the loss value of the rest of the dataset increases.

The goal of the face verification task is to easily be able to separate the embeddings generated by different identities in the euclidean space. Figure III.34 and Figure III.35 illustrates how the embeddings are distributed in the space before and after training. The points in the plots are of 6000 images from 29 different salmon from the test

Figure III.31: The loss curves during training for FishNet2.

Figure III.32: The loss curves during training for FishNet3.

set. The models used are FishNet1 before and after 200 epochs of training. As we can see from the t-SNE-reduced plots the grouping of embeddings from salmon of the same identity is far better after training. This indicates that the model is able to learn some mapping from the images to embeddings.

To compute metrics such as true positive rate, false positive rate, accuracy etc., a similarity threshold needs to be set. To compare the models we can examine what the true positive rate (the sensitivity)

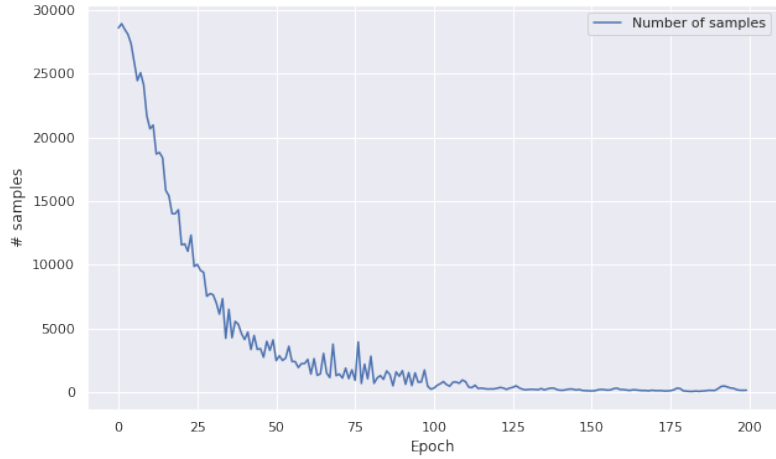


Figure III.33: Number of triplets available for training each epoch for FishNet1. Towards the end of the training only about 100 samples were available for training.

Network Architecture	AUC	TPR @ FPR = 10e-3
FishNet1 (Inception ResNet v2)	0.9977	0.964
FishNet2 (MobileNet v2)	0.9974	0.961
FishNet3 (VGG-16)	0.9919	0.870

Table III.6: The area under the curve and true positive rate (measured when the false positive rate is 10e-3) of the models.

of the system is at a set false positive rate. We have compared the models where the false positive rate is 0.01, that is, where 1% of the negative samples are misclassified as positive. As we can see in Table III.6 FishNet1 and FishNet2 perform approximately equally with a true positive rate of about 96%. FishNet3 performs significantly worse with a true positive rate of 87%.

Figure III.36 shows the ROC curve for the three models we tested. By comparing the area under the curve we can compare the performance of the models across all thresholds. As we can see FishNet1 and FishNet2 perform better than FishNet3, with FishNet1 being the best of the models tested in our experiments. It is interesting to note that the improved results of FishNet1 come at quite a high computational cost compared with FishNet2, a network designed to be able to run on mobile devices. Lastly Figure III.37 shows an example of a visual evaluation of 2 images of 3 different salmon individuals ("Simen", "Eirik" and "Egil"). This shows us that the calculated

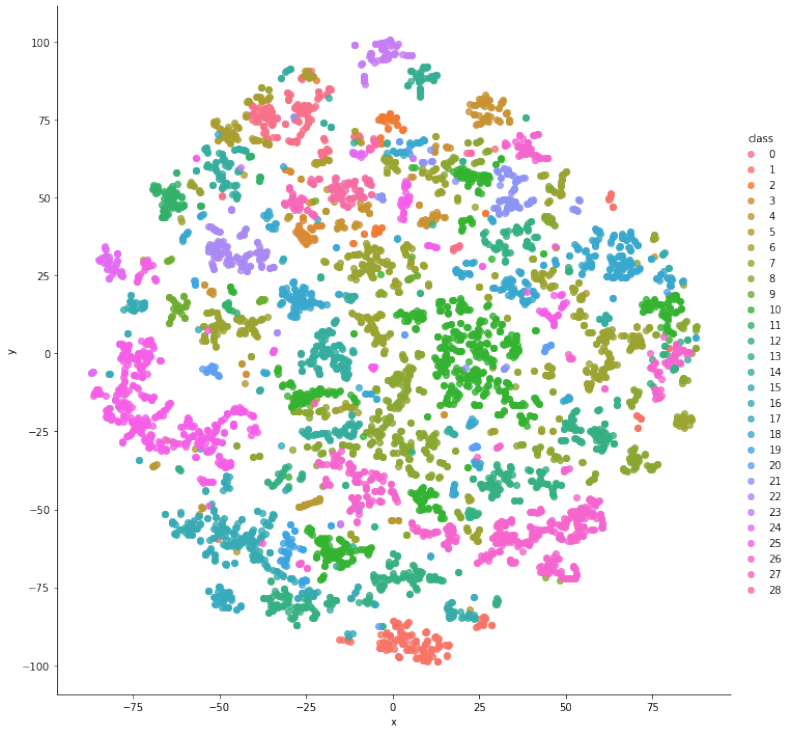


Figure III.34: T-SNE clustering of embeddings produced by a untrained FishNet1 model. The slight clustering visible in the figure is an effect of the inherent clustering done by T-SNE.

distances between different individuals are at least three (on average 3.88) times bigger than the distances between two images of the same individual in this example.

III.5 Discussion

The results shown in the Section III.4 demonstrate that machine learning methods successfully applied for identifying humans from pictures can also be used to identify individual salmon. However it should be noted that this was done with frames extracted from a single video captured over a short period of time. Thus the different frames representing the same individuals in the data set created for this work are very similar. This is somewhat amended by the augmentation done to the frames as described in Section III.4. However, the results from evaluating the method on a data set with these augmentations

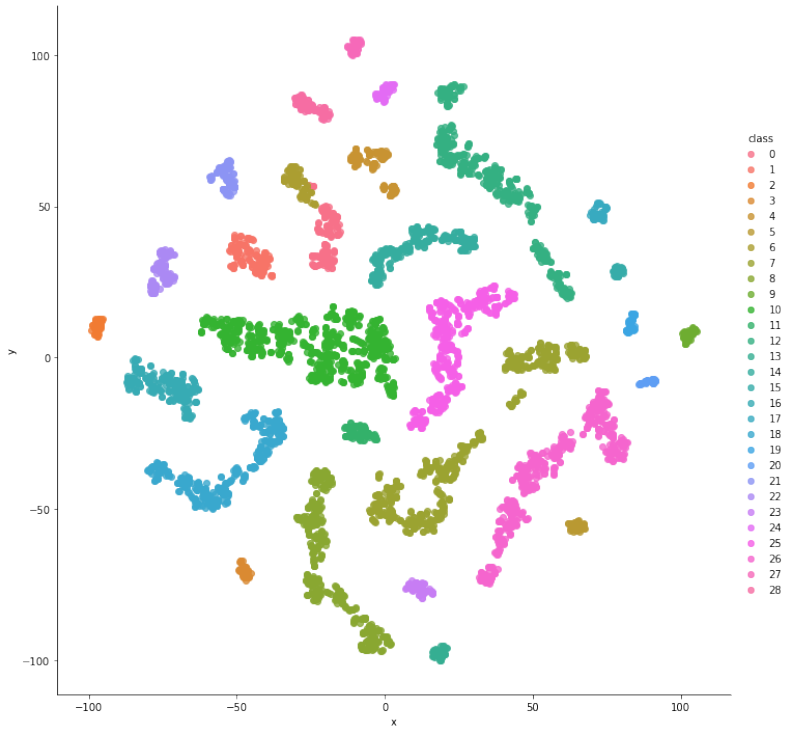


Figure III.35: T-SNE clustering of embeddings produced by a trained FishNet1 model. This is clearly a better clustering than shown in Figure III.34, illustrating that the embedding process extracts useful signals for identifying individuals.

does not enable us to conclude that the method works under all conditions, or over longer periods of time. The video used in this work is a video with very favorable conditions, both in terms of light and water clarity. Training FishNet in more challenging conditions might reduce the performance of the architecture. Thus adversarial regularization using both artificial noise and adversarial examples could be beneficial or even necessary for the architecture to handle such conditions, as this has been shown to increase robustness of deep architectures [133]. The fish may also be damaged mechanically or contract diseases which changes the way individual fish looks over time. This could drastically affect the performance of FishNet on such individuals.

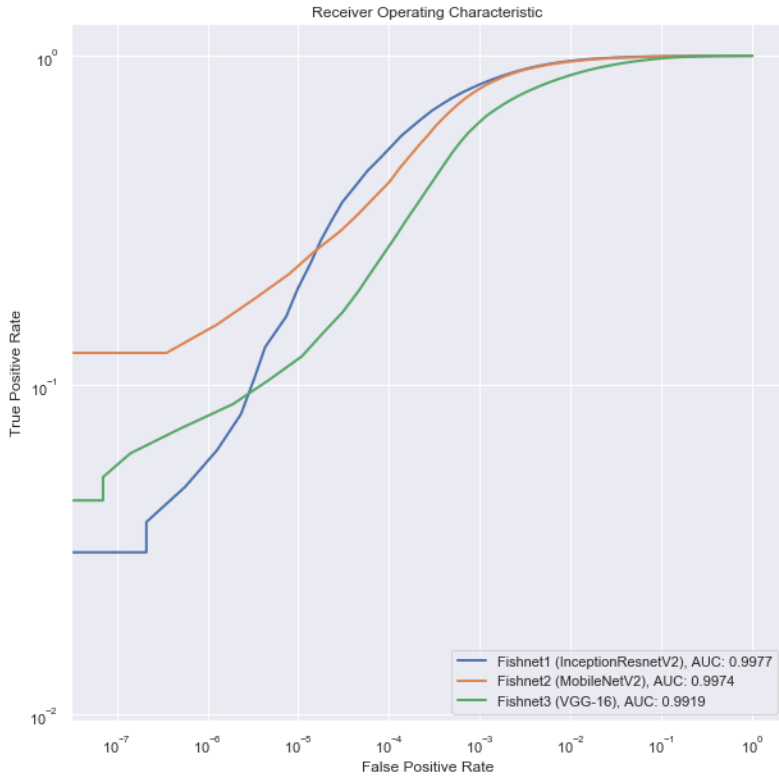


Figure III.36: The ROC curves of FishNet1 (blue), FishNet2 (orange), and FishNet3 (green). The true positive rate and false positive rate is computed across similarity thresholds in the range [0.0, 2.0] in increments of 0.2. The model with the largest area under the curve has the best overall performance (FishNet1, with InceptionResnetV2). Note that the axes in the plot are in logarithmic scale.

III.6 Conclusion

In this paper we presented FishNet, a novel approach for individual fish recognition using a convolutional deep neural network as part of a Siamese neural network architecture based on FaceNet [30]. We trained this model using images of salmon to make the model identify individual salmon. FishNet achieves a false positive rate of 1% and a true positive rate of 96%.

As future work we would like to investigate the model's ability to recognize individuals from spawn to grown fish. We would also like

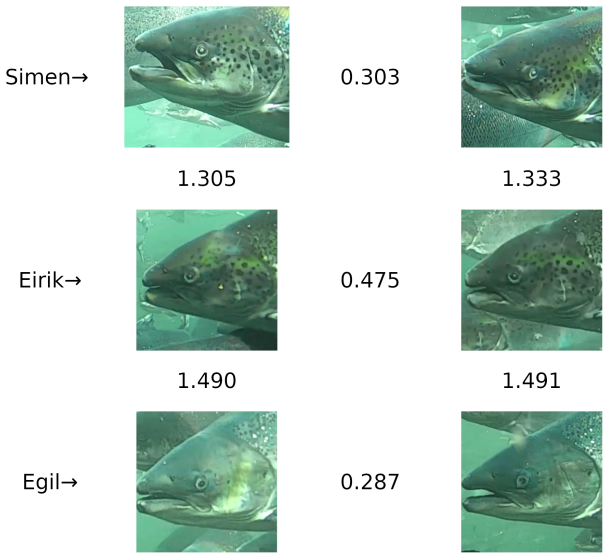


Figure III.37: An illustration of the distances between six images from salmon with three different identities. Each row contains two images of the same salmon: "Simen" at the top, "Eirik" in the middle and "Egil" at the bottom. The average distance between the same salmon is 0.36 while comparisons between different salmon average at 1.40.

to test if we can increase performance by employing other variants of Siamese neural networks such as eSNN[50]. Finally, we would like to investigate what the architecture is actually looking at when recognizing individuals.

III.7 Acknowledgements

This work is an extension of the MSc Thesis "FishNet: A Unified Embedding for Salmon Recognition"[8] by Espen Meidell and Edvard Schreiner Sjøblom. This research has been funded by the Research Council Norway, EXPOSED Aquaculture Research Center (grant number 237790) and the Norwegian Open AI Lab. In addition the

[8] http://hdl.handle.net/11250/2628800

data that formed the basis for the data set was provided by Sealab AS[9]

References

[30] Schroff, F., Kalenichenko, D., and Philbin, J. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[48] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. "Signature verification using a" siamese" time delay neural network". In: *Advances in neural information processing systems*. 1994, pp. 737–744.

[50] Mathisen, B. M., Aamodt, A., Bach, K., and Langseth, H. "Learning similarity measures from data". In: *Progress in Artificial Intelligence* (Oct. 2019), pp. 129–143.

[82] Redmon, J. and Farhadi, A. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[83] Xu, X., Ester, M., Kriegel, H.-P., and Sander, J. "A distribution-based clustering algorithm for mining in large spatial databases". In: *Proceedings 14th International Conference on Data Engineering*. Feb. 1998, pp. 324–331.

[119] SSB. *Akvakultur - årlig, endelige tall - SSB, retrived from https://www.ssb.no/fiskeoppdrett 23.05.2019*. 2018.

[120] Richardsen, R., Stoud Myhre, M., Bull-Berg, H., and T. Grindvoll, I. L. "Nasjonal betydning av sjømatnæringen". In: *Publikasjoner fra CRIStin - SINTEF Ocean* (2018).

[121] Hvas, M., Folkedal, O., Solstorm, D., Vågseth, T., Fosse, J., Gansel, L., and Oppedal, F. "Assessing swimming capacity and schooling behaviour in farmed Atlantic salmon Salmo salar with experimental push-cages". In: *Aquaculture* vol. 473 (Mar. 2017).

[122] Cubitt, K. F., Winberg, S., Huntingford, F. A., Kadri, S., Crampton, V. O., and Øverli, Ø. "Social hierarchies, growth and brain serotonin metabolism in Atlantic salmon (Salmo salar) kept under commercial rearing conditions". In: *Physiology & Behavior* vol. 94, no. 4 (2008), pp. 529–535.

[9] https://www.sealab.no

[123] Merz, J. E., Skvorc, P., Sogard, S. M., Watry, C., Blankenship, S. M., and Nieuwenhuyse, E. E. V. "Onset of melanophore patterns in the head region of Chinook salmon: A natural marker for the reidentification of individual fish". In: *North American Journal of Fisheries Management* vol. 32, no. 4 (2012), pp. 806–816.

[124] Ådland Hansen, A., Rødbotten, M., Eie, T., Lea, P., Rudi, K., and Mørkøre, T. "The effect of crowding stress on bacterial growth and sensory properties of chilled Atlantic salmon fillets". In: *Journal of food science* vol. 77, no. 1 (2012), S84–S90.

[125] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. "Deepface: Closing the gap to human-level performance in face verification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1701–1708.

[126] Eilertsen, A. B. "Identifikasjon av lakseindivider Biometri fase 1 (SalmID)". In: *Publikasjoner fra CRIStin - SINTEF Ocean* (2017).

[127] Hammerset, I. "Biometric recognition and individual tracking of salmon in large-scale sea cages." eng. MA thesis. Norway: Norwegian University of Science and Technology, 2018.

[128] Turk, M. A. and Pentland, A. P. "Face recognition using eigenfaces". In: *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 1991, pp. 586–591.

[129] Belhumeur, P. N., Hespanha, J. P., and Kriegman, D. J. "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection". In: *IEEE Transactions on pattern analysis and machine intelligence* vol. 19, no. 7 (1997), pp. 711–720.

[130] He, X., Yan, S., Hu, Y., Niyogi, P., and Zhang, H.-J. "Face recognition using laplacianfaces". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 3 (2005), pp. 328–340.

[131] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. "ImageNet: A large-scale hierarchical image database". In: *CVPR09*. 2009.

[132] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris

Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-scale machine learning on heterogeneous systems*. Software available from tensorflow.org. 2015.

[133] Sun, S., Guo, P., Xie, L., and Hwang, M. "Adversarial regularization for attention based end-to-end robust speech recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* vol. 27, no. 11 (Nov. 2019), pp. 1826–1838.

Using Extended Siamese Networks to Provide Decision Support in Aquaculture Operations

Bjørn Magnus Mathisen, Kerstin Bach, Agnar Aamodt

Abstract

Aquaculture as an industry is expanding quickly. As a result, new aquaculture sites are being established at more exposed locations, previously deemed unfit because the are more difficult and resource demanding to safely operate. To help the industry deal with these challenges, we have developed a decision support system to make better plans and decisions to facilitate operating these sites in an optimal manner. We propose a case-based reasoning system called aquaculture case-based Reasoning (AQCBR), that is able to predict the success of an aquaculture operation on a specific site, based on previously applied and recorded cases. In particular, AQCBR is trained to learn a similarity function between recorded operational situations/cases and use the most similar case to provide explanation-by-example for it's predictions. The novelty of AQCBR is that it uses extended Siamese neural networks to learn the similarity between cases. Our extensive experimental evaluation shows that extended Siamese neural networks outperform state-of-the-art methods for similarity learning in this task, demonstrating the effectiveness and the feasibility of our approach.

IV.1 Introduction

Aquaculture is currently a growing industry in Norway, expected to grow five-fold by year 2050 [1]. As a result the industry needs to expand to new locations that are increasingly more exposed to harsh weather conditions, as they are located further away from the coast. As the aquaculture industry is set to grow it also faces scrutiny for its environmental impact through lice growth on salmon and downfall of waste to the seabed underneath the locations. Traditionally, the industry has selected locations that are sheltered in fjords or behind islands as more exposure makes the operation of the location more expensive and dangerous. However, as a result of the environmental impact from these aquaculture locations, government agencies do not allow locations to be too closely grouped geographically. Thus new locations for aquaculture operations will be more exposed to weather. The aquaculture industry is already the second most dangerous profession in Norway with regard to work related accidents[3, 4]. An important task in the future is to develop new technology for the aquaculture industry that helps alleviate the increased risk resulting from more exposed locations. Hence it becomes a priority to reduce the amount of manual work and raise the level of automation. Decision

support systems (DSS) can help managers to plan operations on the sites better, and thus minimize the amount of manual work.

Decision support systems for the aquaculture industry have been developed for many years, and utilize a wide range of different machine learning and artificial intelligence techniques [52]. Aquaculture is an industry that traditionally relies upon experience built up over years. As a result the aquaculture industry does not have a tradition for establishing methods and frameworks to create a more formal frame for the industry practices.

Case-Based reasoning (CBR) is a machine learning method where the learning is done via storing cases that describe previously encountered problems and their solutions. A new unseen problem can then be solved by retrieving the stored case which most closely resembles the new problem. The whole process can be described via the CBR cycle [2]: Retrieve the most similar case, Reuse this case to solve the new problem, revise the retrieved case in case the solution had to be adapted to work for the new problem, and finally retain/store the revised case. All parts of this cycle can be adapted when designing a CBR system to solve a problem. However a natural point of focus is the retrieval phase, where the CBR system must calculate the similarity or distance between the problem case and all the stored cases. This is usually done via similarity functions, which can be modeled by domain experts or learned from data. Introducing a DSS system using the CBR methodology for the aquaculture industry can benefit from being able to capture and reuse past concrete experiences for predictions or recommendations produced by the system. Applying DSS in this fashion builds on the tradition in industry of drawing upon previous experiences.

In this work, we propose AQCBR, which is a DSS that supports aquaculture site operators in planning operations for their locations. The procedures in the aquaculture industry is traditionally based on a lot of experience and intuition. For a DSS to be effective in such a work culture, any prediction made by the DSS should also be explained to the user. Although not the main focus of our work, Case-Based Reasoning (CBR) is an established method for explainable systems [134, 135]. AQCBR provides this by using previously recorded cases as a basis for predictions to the user. In our work, we created a case base from a dataset that was collected previously in our project for studying exposed aquaculture operations named EXPOSED[10]. This case base was populated when developing AQCBR. We then adapted several similarity learning methods to AQCBR to learn a representation of the similarity between the operations. We have

[10] https://exposedaquaculture.no/

evaluated the different similarity learning methods quantitatively focusing on AQCBR's usefulness for classification, then qualitatively using similarity matrices. Our work focuses on evaluating the similarity learning methods, and for retrieval we use a standard linear retrieval method that compares all stored cases against the query case.

The main contribution of this paper is to adapt apply Extended Siamese Neural networks [50] (ESNN) to learn the similarity between cases of aquaculture operations that are stored in AQCBR. ESNN is then evaluated in the context of the problem of comparing aquaculture operations and shown to outperform other similarity state of the art learning/distance metric learning methods.

The paper is organised as follows. In Section IV.2 we outline previous work related to machine learning applied to DSS in aquaculture. Section IV.3 describes the different data sources, how the data was gathered, how the data is interpreted, how we combine the different data sources into a case and how the cases are grouped into a case-base. In Section IV.4 we describe the method used for learning the similarity function as well as the other methods used as reference. Section IV.5 presents the results of evaluating the methods on the target task. Finally we discuss and interpret the results from the experiments in Section IV.6.

To make this work reproducible, the code for the experiments described in this paper is available at https://github.com/ntnu-ai-lab/esnn-aqcbr.

IV.2 Related Work

CBR has been applied to DSS systems in aquaculture before, such as the work done by Tidemann et al. [78] on operational support in fish farming, in addition to previous work on CBR for prediction of success of marine operations [6]. As part of the work presented in this paper we will focus on learning a similarity function to correctly retrieve the most similar case for supporting the aquaculture operator in planning operations.

Learning similarity functions from data reduces the work of the developer and domain expert for modeling the similarity function manually when designing a CBR system. Methods for learning similarity measure has been a topic of research for the CBR community for many years [136, 137] and has also been of hightened focus recently [38, 138, 139]. Different types of methods for learning similarity have been used for many tasks, such as deep metric learning for human activity recognition [140] and Extended Siamese Neural Networks for 14 different domains [50]. Dieterle et al.'s work [141] where the features of cases

are weighted by an ANN is another example of a learned similarity function.

Siamese neural networks are a subset of a class of machine learning techniques grouped under the term Deep Metric Learners (DML). DML are optimized to learn an embedding function for datapoints and then calculate the similarity or distance between two such embeddings. One of the first examples of DML was Siamese Neural Networks (SNN), that are trained on pairs of cases. The first example of this was Bromley et al. [48] where a SNN was used to compare signatures. This usage of DML follows a pattern where DML are applied to problems where the number of classes is large such as identification of human activity [140], signature [48] or persons [30]. Typically SNN learns on pairs of datapoints and the loss function is typically calculated from whether the two most similar cases fall within the same class. Recent developments within DML has expanded this to triplet-networks [34] where the DML are trained in triplets of an anchor datapoint, a positive datapoint (same class as the anchor datapoint) and a negative datapoint (different class than the anchor case). Matching networks [28] are also a subclass of DML where DML are trained on representatives from clusters in the dataset. Some methods of Similarity Learning fall outside the DML class of methods, such as the work done by Gabel et al. [38] which learns the similarity of two data points by training on concatenations of each pair of dataset.

Another application of Siamese neural networks is target tracking, i.e. tracking objects across video frames [142, 31, 33, 32] where the siamese architecture is used to compute the distance/correlation of two image patches. In this type of application of SNN the siamese networks are typically convolutional neural networks that extract information from parts of images. Some of the SNNs also employ long-short term memory modules to capture patterns over time between and within frames [33, 32]. The two embeddings computed by the SNNs in tracking problems are combined using a correlation operation. The output of the correlation operation can then be used to estimate the distance between the two data points. Typically SNNs are fully symmetric end-to-end with regards to the two inputs, in contrast some of the SNN methods for visual tracking [31, 32, 33] apply an operation to only one of the signals before they are combined to calculate the distance/correlation.

In previous work we have developed a method we call Extended Siamese Neural Networks (eSNN) as described in [50]. eSNN is an extension of a SNN that has been shown to have greater capacity than SNN for learning to differentiate between classes/categories in a metric learning task. In addition to learning embeddings in the way SNN does, eSNN also learns how to use the differences between

two such embeddings to calculate the distance. This makes eSNN a Type 4 similarity function [50] which has shown the best performance in terms of similarity learning on datasets which are hard to classify. We use eSNN to create a DSS based on a CBR system (AQCBR) to predict failures in operations on exposed aquaculture locations.

IV.3 Operational situation dataset and Case definition

As part of the EXPOSED project operational data was gathered at three different aquaculture locations. Each of the locations was exposed to weather and harsh environments at a level well above average in this industry.

IV.3.1 Reports

The operators on the sites were tasked with recording if a set of possible operations was possible to perform that day or if the weather or environment would be too challenging to safely execute these operations.

The operations considered each day were:

- Go out to the site (all of the personnel typically do not live/sleep on the location).
- Do the daily inspection, go out on the seacage structure and inspect the structure itself, as well as the fish, to ensure good fish welfare.
- Use of a crane on boat in relation to the location structures. Typically strong winds or waves makes operating a crane from a boat very difficult as the length of the crane amplifies the movement generated by the waves on the boat.
- Use winch from a boat to operate on the location (e.g. pull up different parts of the underwater structure)
- Operate a wellboat on location. Wellboats are used to collect fully grown fish for slaughter or deliver spawn to the fishfarm cages.
- Do de-licing operations on location. Typically done by wellboats using either chemical, temperature or mechanical means of de-licing.

- Deliver fishfood and freshwater via feedboat equipped without dynamic positioning.
- Deliver fishfood and freshwater via feedboat equipped with dynamic positioning.

In addition, if any operation was deemed too difficult because of weather or environment, each of the reports had to specify whether it was wind, waves, currents or a combination of them that was a hindrance for that operation. Table IV.7 shows example reports from four days over two weeks.

In total, 708 reports were recorded from 05.12.2016 to 30.12.2018.

IV.3.2 Weather reports

The Norwegian Meteorological Institute provides historical records of weather data through its API[11]. This API provides recorded weather data from the closest weather station to a given point in Norway. Thus we could collect wind speed and wind direction at the location and time for each report. However, the different weather stations and their sensors do fail from time to time, so for some days the closest operational weather station may be further away from the location of the aquaculture operation than others. As a result we calculate the distance from the weather station to the location for each report as a feature in this dataset.

IV.3.3 Exposure level and wind effect

The EXPOSED project has produced a dataset [143] that describes the degree of exposure for most of the aquaculture installations in Norway. The data set provides a level of exposure for 360 degrees around the installation. Exposure level is quantified in the range from 0 to 1, where 0 represents where the installation is shielded by landmass close to the installation and where 1 represents no land within 40km. This dataset provides the exposure level in the direction of the wind at any point in time.

It is intuitive to incorporate the exposure level data into the cases so that a learned similarity function can compare levels of exposure between sites when computing similarity between operational situations. Including all 360 data points per site for every report would be counter productive for several reasons. Firstly, the exposure data does not change over time for each site. Secondly, only a small portion of the exposure data in the direction of the wind on a particular

[11] https://frost.met.no

Parameter/Week number	Week 39	**Week 40**
Date	30.09.2017	02.10.2017
Go out to site wind	0	0
Go out to site wave	0	0
Go out to site current	0	0
Daily inspection wind	0	**1**
Daily inspection wave	0	**1**
Daily inspection current	0	0
Use of crane on boat wind	0	**1**
Use of crane on boat wave	0	**1**
Use of crane on boat current	0	0
Use of nokke on boat wind	0	0
Use of nokke on boat wave	0	0
Use of nokke on boat current	0	0
Usage of wellboat wind	0	**1**
Usage of wellboat wave	0	**1**
Usage of wellboat current	0	0
Usage of service boat wind	0	**1**
Usage of service boat wave	0	**1**
Usage of service boat current	0	0
Delicing wind	0	**1**
Delicing wave	0	**1**
Delicing current	0	0
Usage of feedboat w/anchor wind	0	0
Usage of feedboat w/anchor wave	0	0
Usage of feedboat w/anchor current	0	0
Usage of feedboat w/DP wind	0	0
Usage of feedboat w/DP wave	0	0
Usage of feedboat w/DP current	0	0

Table IV.7: Example of two cases from different aquaculture sites from two days over the period of two calendar week. The features of the cases are binary where "1" (also highlighted with bold in the table) indicates a failure of that operation for that time and location. Conversely "0" indicates that the operation was successful. The features of the cases has values for each operation type, indicating whether it was wind, waves or currents respectively that was the reason for failure. The example from week 39 shows a case where all operations were successful. The second example case from week 40 shows a failure to do daily inspection, use a crane on a boat, use wellboat and service boat as well as de-licing. All of these failures in the case from week 40 was due to wind and waves, not because of currents. We can also see that usage of feedboat was still possible, this is because these boats are bigger and their operations less sensitive to weather.

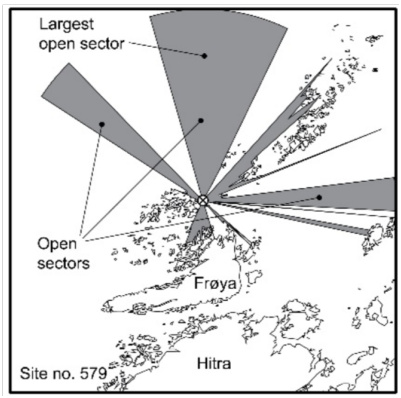


Figure IV.38: Level of exposure for one of the sites [143] that provided data for our work. One can see that the site is exposed to wind and waves from the North (marked as an open sector).

day has an effect on the operations for that day (being exposed in the direction of no wind has little effect). Our solution is to combine the exposure level with the wind direction at the location and time of the report. This way the learned similarity function can take into account the exposure level in the same direction as the wind direction at that time. This is implemented as a lookup function that returns the exposure level for a given wind direction. To make the function more smooth in terms of wind effect we add a Gaussian filter. This will let neighboring exposure levels have an effect on the calculated wind effect. This lookup function gf is defined as follows:

$$gf(wd, el, wis) = G(wis + 1) \bullet el(wd, wis), \tag{IV.10}$$

where $G(\cdot)$ returns a Gaussian filter of size wis as a vector, and $el(wd, wis)$ returns the exposure level in the wind direction wd as well as wis exposure levels adjacent to the wind direction wd. This enables the model to take into account the level of exposure adjacent to the wind direction, and not only the single degree of direction from the wind. In our model we set this window wis to 10 thus accounting for the exposure level ± 5 degrees of the wind direction. This vector of wind exposure levels around the wind direction can then be combined with the wind speed to give us the wind effect. This is defined as we:

$$we(w, wd, el, wis) = gf(wd, el, wis) \cdot w, \tag{IV.11}$$

where w is the wind speed at the site at that current time and all other function parameters are as defined in Equation IV.10.

IV.3.4 Case Definition and Case Base Population

For all eight different types of operations listed in subsection IV.3.1 there can be four different outcomes: No failure or failure because of wind, waves or current. This results in $4 * 8 = 32$ classes, which is too many classes to learn to separate from 708 data points. However, from the perspective of a DSS user in the setting of aquaculture operation planning, a general prediction of operational failure is useful. Thus, grouping the failure types and causes together reduces the resolution, but retains most of the utility of AQCBR as a DSS. After grouping all the failures, we can evaluate AQCBR's ability to predict failures related to weather. Given that these operational failures occur seldom, the dataset is unbalanced, with 88% of all cases not reporting any failures. Given a failed operation, it is highly likely that more wind from the same direction would also be a failure. This makes it simple to generate realistic failure cases from the existing failure cases to expand the training dataset. To generate a realistic case we pick a random failed operation and add a small random value to the wind speed. This is done while making sure the data point is not noise (i.e. has low wind speed, see Figure IV.39). Figure IV.39 shows a pairplot of a subset of the case features, with the cases colored according to class (failure/success). The pairplot shows that most failure cases occur during high wind speeds, but that some occur during low wind speeds. The failure cases of the latter type is not considered during re-balancing of the dataset.

Given this, we can now define the case base for AQCBR. Formally, let the farms data be $d = x_1, x_2, \ldots, x_n$ where x_i is one report containing an operation's success or failure(sf). Further let $el = el_1, \ldots, el_n$ be the dataset of exposure levels where el_i corresponds to the exposure level at the location of report x_i. And let $w = w_1, \ldots, w_n$ be the dataset of weather reports collected for these sites where w_i corresponds to the report x_i. These weather reports contain wind speed (ws), wind direction (wd) and distance to weather station (di). Thus the case can be represented as:

$$C_i(x_i, el_i, w_i) = w_i(ws, wd, di), we(w_i(ws), w_i(wd), el_i, wis), x_i(sf), \tag{IV.12}$$

where $we(\cdot)$ is defined by Equation IV.11 and wis is the window size of the weather effect (how much of the exposure levels to either side of the wind direction should be taken into account). Cases bases are then split up into test (query) and train parts via stratified cross-validation for evaluation of AQCBR (see Section IV.5). Example cases following this definition can be seen in Table IV.8.

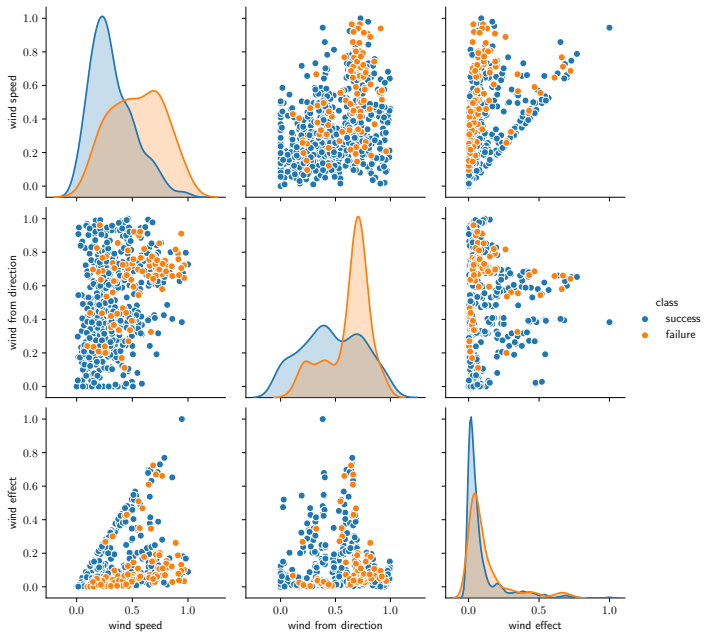


Figure IV.39: Pair plot which shows the correlation between three of the features of the cases of the EXPOSED dataset. The coloring of the data points shows if operation was successful (blue) or failures (orange). All values are normalized (e.g. wind speed=1 is the maximum wind speed in the dataset). We can see that the wind speed feature has two clusters according to failure or success. However, there are also a number of failure cases with low wind speed.

Ex	WS	WD	Di	WE	S/F
A	0.78 (15.6)	0.725 (261)	0.14 (8844)	0.08	failure
B	0.22 (4.5)	0.741 (267)	0.14 (8844)	0.026	success

Table IV.8: This table shows two example cases from the recorded data used for the training and testing done in this work. Each example (Ex) case is described with associated wind speed (WS), wind direction (WD), distance (Di) between site and weather station, the wind effect (WE) and if the case represents a success or a failure (S/F). Example A shows a failed operation with high wind speed (15.6 m/s) from 261 degrees recorded at a weather station 8844 meters from the aquaculture location. Example B shows an operation which did not fail, with much lower wind speeds (4.5 m/s) from the same wind direction reported from the same weather station.

IV.4 Extended Siamese Neural Networks

For AQCBR to perform well it needs to retrieve the most appropriate case from the case base when presented with a query case. Thus after populating the case base (Section IV.3.4) we need to define a similarity function that captures the connection between weather data and exposure level at the site, and if an aquaculture operation could be successful given those circumstances. This could be done in different ways, including manual/analytical modeling. However, in this paper we are focusing on automatic learning of this similarity function using similarity learning. This is because differences between localities, and how these differences change the way weather affects operations, are hard to model manually. Our approach is to learn this connection through induction by creating a machine learning model based on collected data.

Below we briefly describe our method for similarity learning in AQCBR as well as the reference methods used for comparison.

Extended Siamese Neural Networks (ESNN) has been shown to have good performance compared to other methods [50] and was chosen as the primary method of similarity learning for AQCBR. We refer to the implementation of ESNN in this work as *esnn*. For reference we also implemented the similarity learning from Chopra et al.[37] (implementation referred to as *chopra*) and Gabel et al. [38] (implementation referred to as *gabel*). *chopra* is a type 3 similarity function that learns to create useful embeddings and then calculate the L2ă(Euclidian) distance between pairs of embeddings. Figure IV.40 shows the general architecture of the Extended Siamese Neural Network (ESNN).

As seen in Figure IV.40, *esnn* has three outputs, with two outputs used for calculating loss ($\vec{\hat{x}}$ and $\vec{\hat{y}}$). The third output is the distance between the two data points. As seen in Figure IV.40 *esnn* is comprised of an embedding function $G(\cdot)$ and a binary function $C(\cdot,\cdot)$ that uses the two embeddings ($\vec{\hat{x}}$ and $\vec{\hat{y}}$) to compute the similarity between the two input datapoints ($\vec{x}$ and $\vec{y}$). More specifically, $C(\vec{\hat{x}},\vec{\hat{y}}) = C(ABS(\vec{\hat{x}}-\vec{\hat{y}}))$. Let $F(\vec{\hat{x}})$ be the features of the case and $S(\vec{\hat{x}})$ the solution (or target) of case $\vec{x}$. For a pair of two cases $(\vec{x},\vec{y})$ the loss function of ESNN can then be defined as:

$$L(\alpha,\vec{x},\vec{y},s) = \frac{(1-\alpha)}{2}\cdot(L_c(G(\vec{x}),S(\vec{x})) + L_c(G(\vec{y}),S(\vec{y}))) + \alpha\cdot L_s(\vec{x},\vec{y},s), \tag{IV.13}$$

where α is a parameter to weight the importance of the three different outputs of the loss function and s is the true similarity. $L_c(p,q)$ is

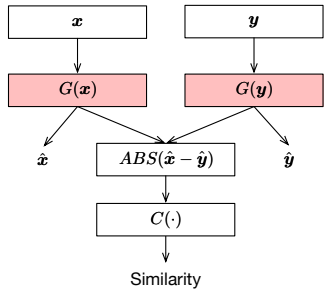


Figure IV.40: Architecture of the ESNN method that provides a similarity function. This shows how the two datapoints $\vec{x}$ and $\vec{y}$ are embedded through $G(\cdot)$ (implemented by a neural network) to produce embeddings $\vec{\hat{x}}$ and $\vec{\hat{y}}$. A vector absolute difference between the two embeddings are then calculated and used as input to a second neural network. In addition the two embeddings are used to calculate loss against the classification of each of the data points.

the categorical cross entropy loss between p and q. Finally, L_s is the similarity loss, the difference between the models predicted similarity and the true similarity, this is defined as:

$$L_s(\vec{x}, \vec{y}, s) = |s - C(G(\vec{x}), G(\vec{y})| \tag{IV.14}$$

The loss function for *chopra* is $L_s(\vec{x}, \vec{y}, s)$ with $C(G(\vec{x}), G(\vec{y})) = |G(\vec{x}) - G(\vec{y})|$, as $C(\vec{\hat{x}}, \vec{\hat{y}})$ is modeled as the L2-distance between the embeddings $\vec{\hat{x}}$ and $\vec{\hat{y}}$. *gabel* uses the same loss $L_s(\vec{x}, \vec{y}, s)$, but as *gabel* does not learn embeddings, $G(\cdot)$ becomes the identity function $I(\cdot)$ while $C(I(\vec{x}), I(\vec{y}))$ is learned as a neural network model.

Early experiments showed that for the case base defined in the previous section, a high α produces the best results as a similarity function for AQCBR. As a result, an α equal to 1 was chosen, as can be seen from Equation IV.13 this results in; $L(\alpha = 1, \vec{x}, \vec{y}, s) = L_s(\vec{x}, \vec{y})$. Thus in our experiments all loss functions are identical in how they measure model performance.

IV.5 Evaluation

The results shown in Figures IV.41 and IV.42 were generated with five-fold stratified cross-validation and repeated five times resulting in a mean and standard deviation for each epoch. The embedding part $G(\cdot)$ of the similarity functions *esnn* and *chopra* was implemented as a fully connected ANN with three layers of size 40, 6 and 3. *esnn*

had additional two fully connected layers of size 4 and 2 to learn the binary function $C(\cdot, \cdot)$. The results in Figure IV.41 and IV.42 show that the *esnn* similarity assessment performs better than *chopra* and *gabel*. A retrieval validation was run every 10th epoch, where every data point from the test set of that fold is used as a query case. The training set is used as the case base. The validation performance was then calculated based on whether the most similar case had the same solution as the query case. Figures IV.41 and IV.42 show the training and validation performance across 1000 epochs. The training performance shows that *esnn* outperforms the reference methods *chopra* and *gabel* in training speed and accuracy, as well as having slightly better validation accuracy. In addition, one can observe that *chopra* achieves performance very quickly, while *gabel* achieves performance much more slowly.

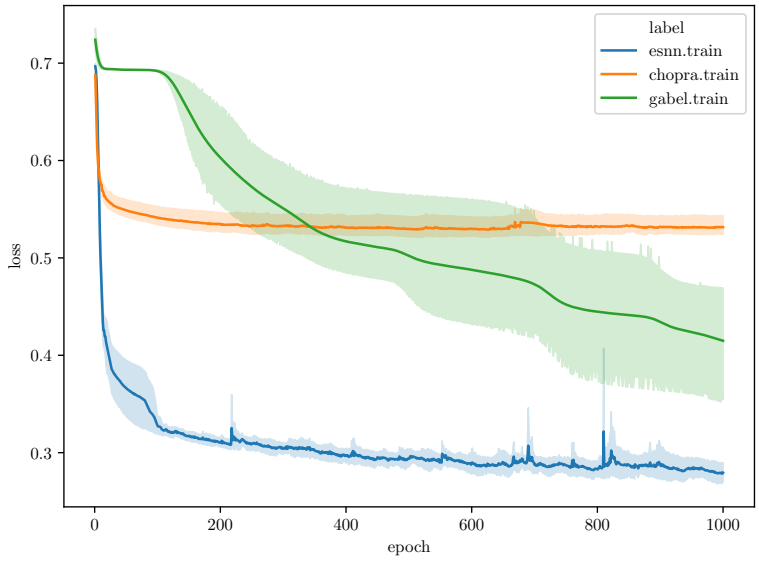


Figure IV.41: Training results across 1000 epochs for the different similarity functions. The experiment was done with five fold cross-validation and then repeated five times for validity. Like the results reported in [50] we can see that *esnn* and *chopra* achieves high training accuracy early on while *gabel* needs more training time as a result of its architecture. We can also observe that *chopra* thresholds in performance at around 0.55 loss while *esnn* and *gabel* improves it's loss beyond this threshold.

After 1000 epochs of training the retrieval performance (measured

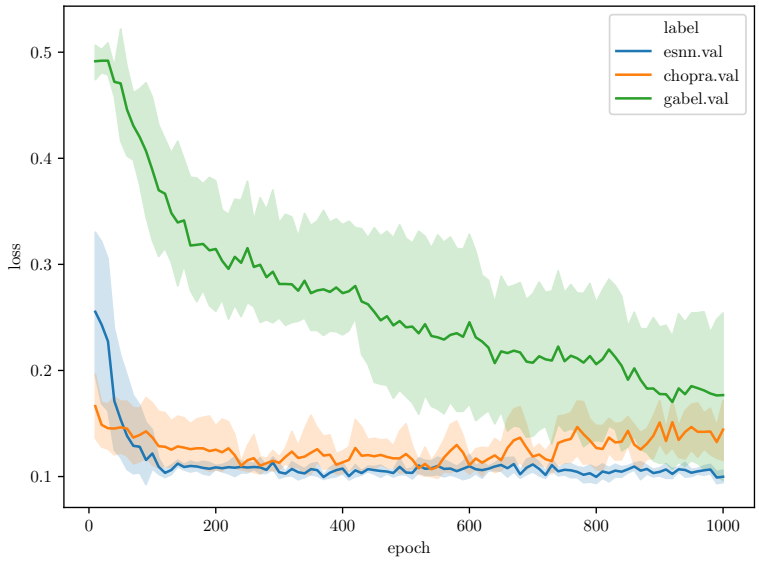


Figure IV.42: Validation results across 1000 epochs for the different similarity functions. Data was recorded along with the data shown in Figure IV.41, thus with 5 fold cross validation and repeated five times. Validation loss *val* was calculated every 10 epoch. This graph shows that the difference between *esnn* and *chopra* in terms of the validation accuracy is much smaller than in the training loss as shown in Figure IV.41. As with the training performance *gabel* catches up after some epochs, while *chopra* seems to overfit after 600 epochs.

as described in last paragraph) is 90% ($\pm 0,7\%$) for *esnn*, $85,57\%$ ($\pm 3,4\%$) for *chopra* and $82,32\%$ ($\pm 8,7\%$) for *gabel*.

To illustrate the qualitative results of the retrieval for each of the methods we generated similarity matrices. Figure IV.43, IV.44 and IV.45 show the similarity matrices and retrieval results for *esnn*, *gabel* and *chopra* respectively. These figures were generated by sampling ten random cases, then calculating the similarity between them for each method. The second to last row of each figure shows the class (failure or success) of the cases in each column. The cell values are the similarity, where 1 is high similarity and 0 is low similarity. The colors of the cells also illustrate the degrees of similarity with high similarity (1) being dark blue and low similarity (0) being a light color. The last row shows the name of the most similar case (except itself) of the case for that column, the color of the cell indicates if this was a correctly retrieved case.

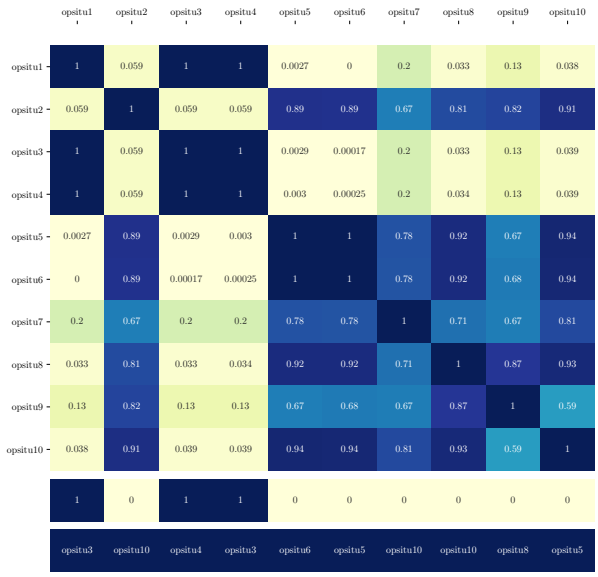


Figure IV.43: Similarity matrix for *esnn*. One can see that the matrix is symmetric, which is a result of the architecture of the ESNN method being based on Siamese Neural Networks(SNN). The retrieved cases are all correctly retrieved. However, *esnn* still seems to output some similarity across classes (e.g. the failure opsitu7 is 0.2 similar to the success cases opsitu1, 2 and 3.) This is in contrast to the similarity matrix of *chopra*.

Figure IV.43 and IV.45 show that both *esnn* and *chopra* perform well as a similarity measure in AQCBR, retrieving the correct cases most often. Figure IV.44 shows that *gabel* performs slightly worse in practice. The figure also shows that *gabel* is not a symmetric similarity measure: the diagonal is not a static value and similarities change when the ordering of cases change $gabel(opsitu2, opsitu3) \neq gabel(opsitu3, opsitu2)$. One can also observe that *chopra* has a less smooth similarity matrix, compared to *esnn* and *gabel*. *chopra* seems to clearly mark that a query case is part of a class (success or failure), with near zero similarity with cases in the case base with a different class from the query class. This is not the case with *esnn* and *gabel* which outputs possibilities of similarities between a query case with

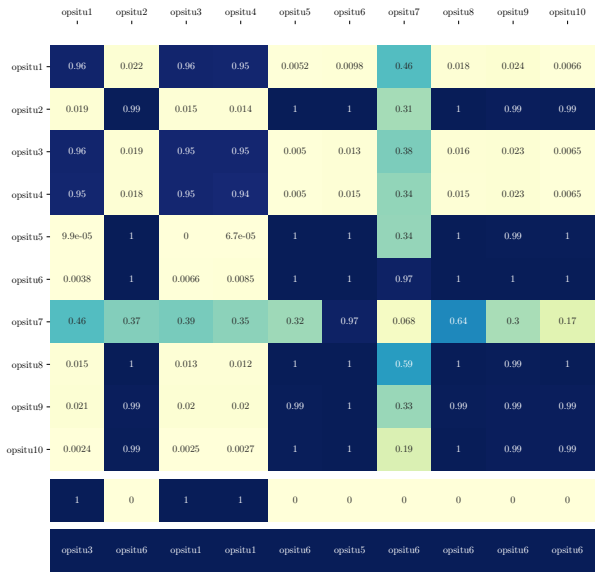


Figure IV.44: Similarity matrix for *gabel*. This matrix shows signs that this similarity learning method is not based on SNN as it is not quite symmetric and the diagonal is not 1. In agreement with *esnn*, this method measures the failure case opsitu7 to be close to the success cases opsitu1, 2 and 3. In contrast to *esnn* *gabel* measures opsitu 5 (failure) to be less close to opsitu7 (failure) than opsitu3 (success), which is an inaccurate measurement.

one class and other cases with a different class. This is likely a result of how *chopra* implements its binary function $C(\cdot, \cdot)$ as a static L2 distance function. As a result *chorpa* approximates a threshold function after training. This suggests that similarity learning methods that learn the binary part of similarity functions (denoted as $C(\cdot, \cdot)$ in Section IV.4) can provide more insight into similarities between cases of different classes than similarity functions that does not learn the binary part. However, in the context of AQCBR, *esnn* and *gabel* could help users find insight into which parts of failed operations are similar to successful operations.

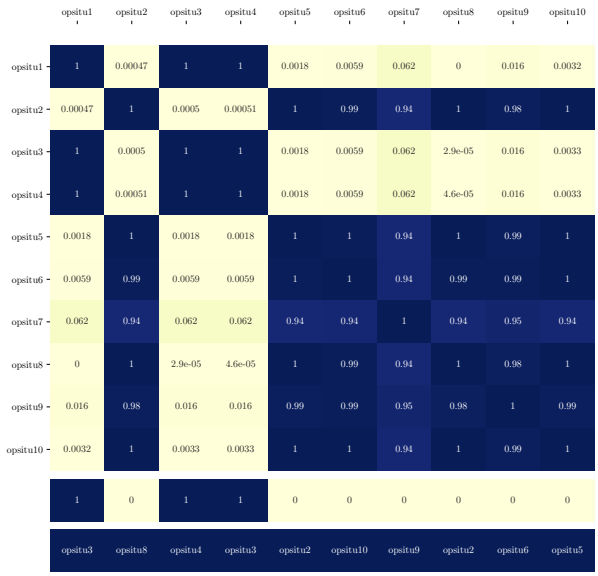


Figure IV.45: Similarity matrix for *chopra*. This similarity matrix is very binary in comparison with the ones from *gabel* and *esnn*, only finding similarity within the same class.

IV.6 Conclusions and Future Work

In this work we have shown the need for decision support tools to help support the aquaculture industry increase the level of automation and planning. To this end, we have presented AQCBR, a DSS based on CBR which uses the novel Extended Siamese neural networks method for learning similarities.

Our results presented in the last section show that the similarity learning method *esnn* outperforms *gabel* and *chopra*. This is consistent with previously reported results from Mathisen et al. [50]. In addition, the similarity matrices and retrieval results in Figures IV.43 and IV.45 show that AQCBR performs well in terms of retrieving previous cases with the same outcome as the query cases. These figures also confirm that all three methods of similarity learning perform well, even with *esnn* giving the best overall performance. We also showed that *esnn* performs well with relatively little data (a total

of 708 data points) given that the learning is reasonable (see Section IV.3.4).

In this paper we have shown that AQCBR can serve as a decision support system for aquaculture operators, as it does not only differentiate feasible from unfeasible operations, it also comes with the explanation capability given by the CBR system providing example cases.

An extension to this work would be to not group all the different types of operations into one as described in Section IV.3.1. However, this would require more data points to allow *esnn* to correctly separate more categories of operational successes and failures. As waves typically build over days, especially if the location is very exposed, including a time series of weather in relation to the location could improve accuracy even more.

IV.7 Acknowledgements

We would like to thank the EXPOSED SFI project (Research Council of Norway - grant number 237790) and Norwegian Open AI Lab for the support to do this work. Thanks also to Helge Langseth, Gunnar Senneset and Hans Vanhauwaert Bjelland for their great support during our work.

References

[1] Olafsen, T., Winther, U., Olsen, Y., and Skjermo, J. "Value created from productive oceans in 2050". In: *SINTEF Fisheries and Aquaculture* (2012), p. 83.

[2] Aamodt, A. and Plaza, E. "Case-based reasoning: Foundational issues, methodological variations, and system approaches". In: *AI communications* vol. 7, no. 1 (1994), pp. 39–59.

[3] Holen, S. M., Utne, I. B., Holmen, I. M., and Aasjord, H. "Occupational safety in aquaculture–Part 1: Injuries in Norway". In: *Marine Policy* vol. 96 (2018), pp. 184–192.

[4] Holen, S. M., Utne, I. B., Holmen, I. M., and Aasjord, H. "Occupational safety in aquaculture–Part 2: Fatalities in Norway 1982–2015". In: *Marine Policy* vol. 96 (2018), pp. 193–199.

[6] Mathisen, B. M., Aamodt, A., and Langseth, H. "Data driven case base construction for prediction of success of marine operations". In: *Proceedings of ICCBR 2017 Workshops (CAW, CBRDL, PO-CBR), Doctoral Consortium, and Competitions co-located with the 25th International Conference on Case-Based Reasoning (ICCBR 2017)*. 2017, pp. 102–111.

[28] Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. "Matching networks for one shot learning". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3630–3638.

[30] Schroff, F., Kalenichenko, D., and Philbin, J. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[31] Valmadre, J., Bertinetto, L., Henriques, J., Vedaldi, A., and Torr, P. H. "End-to-end representation learning for correlation filter based tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2805–2813.

[32] Gao, P., Zhang, Q., Wang, F., Xiao, L., Fujita, H., and Zhang, Y. "Learning reinforced attentional representation for end-to-end visual tracking". In: *Information Sciences* vol. 517 (2020), pp. 52–67.

[33] Gao, P., Yuan, R., Wang, F., Xiao, L., Fujita, H., and Zhang, Y. "Siamese attentional keypoint network for high performance visual tracking". In: *Knowledge-Based Systems* vol. 193 (2020), p. 105448.

[34] Hoffer, E. and Ailon, N. "Deep metric learning using triplet network". In: *International Workshop on Similarity-Based Pattern Recognition*. Springer. 2015, pp. 84–92.

[37] Chopra, S., Hadsell, R., and LeCun, Y. "Learning a similarity metric discriminatively, with application to face verification". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 539–546.

[38] Gabel, T. and Godehardt, E. "Top-down induction of similarity measures using similarity clouds". In: *Case-Based Reasoning Research and Development*. Ed. by Hüllermeier, E. and Minor, M. Cham, 2015, pp. 149–164.

[48] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. "Signature verification using a" siamese" time delay neural network". In: *Advances in neural information processing systems*. 1994, pp. 737–744.

[50] Mathisen, B. M., Aamodt, A., Bach, K., and Langseth, H. "Learning similarity measures from data". In: *Progress in Artificial Intelligence* (Oct. 2019), pp. 129–143.

[52] Mathisen, B. M., Haro, P., Hanssen, B., Björk, S., and Walderhaug, S. "Decision support systems in fisheries and aquaculture: A systematic review". In: *arXiv preprint arXiv:1611.08374* (2016).

[78] Tidemann, A., Bjørnson, F. O., and Aamodt, A. "Operational support in fish farming through case-based reasoning". In: *Advanced Research in Applied Artificial Intelligence*. 2012, pp. 104–113.

[134] Sørmo, F., Cassens, J., and Aamodt, A. "Explanation in case-based reasoning–perspectives and goals". In: *Artificial Intelligence Review* vol. 24, no. 2 (2005), pp. 109–143.

[135] Keane, M. T. and Kenny, E. M. "How case-based reasoning explains neural networks: A theoretical analysis of XAI using post-hoc explanation-by-example from a survey of ANN-CBR twin-systems". In: *Case-Based Reasoning Research and Development*. Ed. by Bach, K. and Marling, C. Cham, 2019, pp. 155–171.

[136] Aha, D. W. "Case-based learning algorithms". In: *Proceedings of the 1991 DARPA Case-Based Reasoning Workshop*. Vol. 1. 1991, pp. 147–158.

[137] Stahl, A. "Learning similarity measures: A formal view based on a generalized CBR model". In: *International Conference on Case-Based Reasoning*. Springer. 2005, pp. 507–521.

[138] Hoffmann, M., Malburg, L., Klein, P., and Bergmann, R. "Using siamese graph neural networks for similarity-based retrieval in process-oriented case-based reasoning". In: *Case-Based Reasoning Research and Development: 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8-12, 2020, Proceedings*. Accepted for publication. 2020.

[139] Ye, X., Leake, D., Huibregtse, W., and Dalkilic, M. "Applying Class-to-Class Siamese Networks to Explain Classifications with Supportive and Contrastive Cases". In: *Case-Based Reasoning Research and Development: 28th International Confer-*

ence, ICCBR 2020, Salamanca, Spain, June 8-12, 2020, Proceedings. Accepted for publication. 2020.

[140] Martin, K., Wijekoon, A., and Wiratunga, N. "Human activity recognition with deep metric learners." In: CEUR Workshop Proceedings. 2019.

[141] Dieterle, S. and Bergmann, R. "A hybrid CBR-ANN approach to the appraisal of internet domain names". In: *International Conference on Case-Based Reasoning*. Springer. 2014, pp. 95–109.

[142] Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., and Torr, P. H. "Fully-convolutional siamese networks for object tracking". In: *European conference on computer vision*. Springer. 2016, pp. 850–865.

[143] Lader, P., Kristiansen, D., Alver, M., Bjelland, H. V., and Myrhaug, D. "Classification of aquaculture locations in norway with respect to wind wave exposure". In: *Proceedings of the ASME 2017 36th International Conference on Ocean, Offshore and Arctic Engineering OMAE2017*. 2017.

Appendices

V

Defining the initial case-base for a CBR operator support system in digital finishing

Leendert Wienhofen, Bjørn Magnus Mathisen

Abstract

Case-based reasoning (CBR) literature defines the process of defining a case-base as a hard and time-demanding task though the same literature does not report in detail on how to build your initial case base. The main contribution of this paper is the description of the methods that we used in order to build the initial case-base including the steps taken in order to make sure that the quality of the initial case set is appropriate. We first present the domain and argue why CBR is an appropriate solution for our application. Then we detail how we created the case base and show how the cases are validated.

A.1 Introduction

Case-based reasoning (CBR) literature defines the process of defining a case-base as a hard and time-demanding task though do not report in detail on how to actually build your initial case base. Öztürk and Tideman say in their 2014 review paper [144]: "Initial population of a case base is a daunting task in classical CBR because it is manually crafted by knowledge engineers who make use of domain experts or written material to extract the case content. ... We believe case grounding problem is the reason why CBR has not seen wide-spread adoption in the industry - because manual extraction of cases from reports and records is costly and time consuming". In this context, we present a knowledge acquisition process that was applied to create an initial set of cases while constructing a CBR system in an industrial setting. We explain the domain in which we applied CBR and argue why it is an appropriate solution for our application. This is followed by a description of a methodological approach for building an initial case base. Revision and validation of the case base and the similarity features are presented in the discussion section.

The main contribution of this paper is the description of the methods that we used in order to build an initial case-base for our CBR system in an industrial domain, including the steps taken in order to make sure that the quality of the initial case set is appropriate.

A.1.1 Background

The case-based decision support system described in this paper is part of a project that is trying to increase the speed of digital conversion. Digital conversion is the process of cutting or milling various types of materials into shapes, based on a digital design. The speed is to be

increased concerning the actual cutting speed as well as the time to shift between different jobs.

This paper will focus on the latter and the main objective, as set forward in the project proposal, is to decrease the time an operator uses between jobs by 80%.

Graphics)

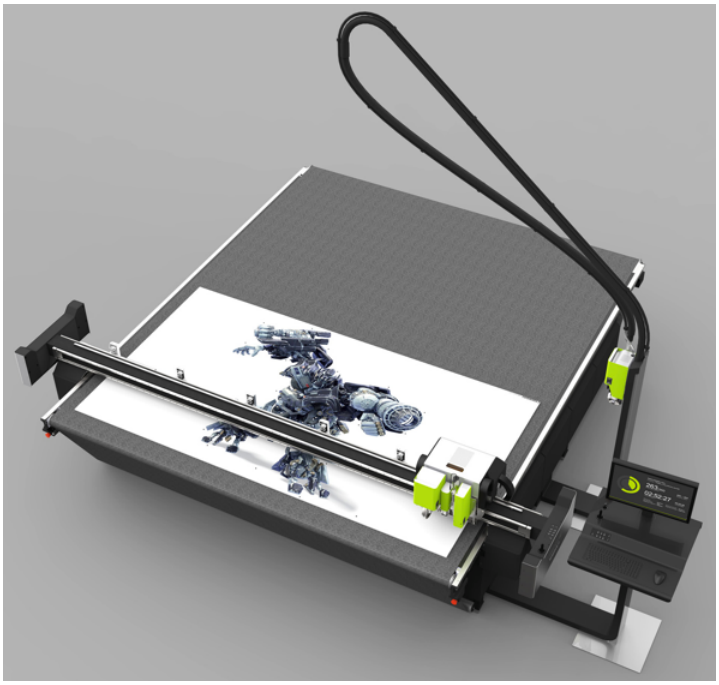


Figure A.46: A digital conversion table from Esko Graphics (Copyright Esko

Digital conversion machines (such as shown in Fig. A.46), also referred to as cutting tables, offer a plethora of different settings and the intervention is suggested to be an intelligent operator user interface to the conversion machine, based on case-based (CBR) and rule-based reasoning (RBR). By automating parts of the process relating to load shifts, the job for the operator will be easier and faster, with a lower margin for errors compared to the current situation.

The finished system should facilitate and automate learning from past experiences (meaning cutting/milling jobs with settings specific for a design and material) within a specific company. Future work will enable the system to share data between deployments of the system,

so that even competing companies can share their experiences without sharing their competitive advantage.

A cutting table is a further development of a flatbed pen plotter, where the pen can be substituted by knives and millings bits, and the drawing paper by other types of material. Operations on the X and Y-axis (given a certain depth and pressure on the Z-axis) vary per material type. The optimal speed and acceleration for a given actuator depend not only on the material type, but also on the vendor (as quality can vary from vendor to vendor), the wear and tear of the actuator as well as the complexity of the design to be cut/milled out from the material, just to name a few. Therefore, these cutting tables have a myriad of settings and require an experienced operator in order to get the best results. Most of the knowledge required for configuring the machine correctly is currently implicit, and knowledge transfer is typically done on a face-to-face basis between operators. Generally speaking, we can say that inexperienced operators do not dare to use the full potential of the table in fear of damaging the materials on which the cutting or milling operation is to be carried out on.

By making domain knowledge explicit in the form of a domain model with instances, an inexperienced user can find similar cases and re-use the settings. In our approach, we take this one step further by applying case-based reasoning, which automatically selects the most applicable case and related setting for the user so that the full potential of the machine can be used.

A.1.2 Case-based reasoning as an enabler for experience transfer

Based on interviews and observations at companies using digital conversion tables, we conclude that experience is typically not stored in a structured manner and knowledge transfer happens in an informal way between co-workers. Operators of these machines typically learn by doing, and because of this the full potential of a machine is not always reached, especially when operated by inexperienced users. Users report that they are afraid of breaking something when they apply parameters they are unsure of.

In some cases a note with settings is taped on the operator console, though these contain proven safe settings for a typical material and tool combination. Another company uses a whiteboard for settings, though it is rarely updated and personnel indicate that they actually do not use the settings that are noted there and rather trust their own feelings concerning the settings. There is no structured means of storing experiences among the companies that have been observed during the case study.

As the working situation is based on a known desired outcome, case based reasoning is an appropriate manner of addressing the problem at hand.

We intend to create a knowledge base where the digital finishing machine retains the settings, material type and other relevant parameters.

A.1.3 Distributed case-based reasoning

In the digital finishing industry companies use many different material types, some on a more regular basis than others. As this is a very experience-based process, chances are that the proper expertise is not available in all companies. By providing access to case bases created in other (competing) companies, one can draw from the experience.

A.1.4 Related work

It has been shown [145] that CBR is well suited as a means of decision support for operators in a manufacturing setting. CBR is a form of AI where the decision making support is based on a known outcome. It takes a case (which is the product to be made) as input and tries to find the most similar case in a case base. This means that cases with a similar profile are suggested.

Competing companies can help each other increase their efficacy by sharing case bases can be achieved via distributed case-based reasoning. Distributed CBR has been around for a while and is well described in among others [146, 147, 148]. However, it seems to be limited to non-competing companies, making knowledge sharing a clear-cut benefit. In order to avoid potential problem with patented designs, we decouple the geometry information from the design, reducing it to an indication of the complexity of the design based on a float where 0.0 is the least complex and 1.0 the most complex. From the technical point of view there is no real difference in the implementation.

Our CBR system will implement explanatory features enabling the operator to choose to either apply the suggested settings or retain the self-chosen setting based on the suggested settings and the corresponding explanation. Based on the interviews, we can state that it is important that the CBR system does not actually make a decision, rather suggests a decision based on the most similar case. This way the system supports the operator in his decision. The explanation helps the operator to understand why a certain proposal has been made by the system and therewith enables to operator to make an

informed decision. The fundamental issues of explanations in CBR are described well in [149].

Aamodt and Plaza [2] have formalized Case-based reasoning for purposes of computer reasoning as a four-step process: Retrieve, reuse, revise and retain.

In order to retrieve a case, one needs to identify features, collect descriptors, interpret problem and infer descriptors. Prior to being able to do that, one needs to have a case base. It is of importance that the right features are extracted from a case as it will be the fundament for further reasoning.Case acquisition is often manually intensive. According to [95], a manually intensive approach for storing experiences of individuals has been widely used in many CBR applications. The general approach –as case bases are very domain specific- is to talk to a domain expert and extract which parameters are of most value and use that as a starting point. Getting the full picture, however, requires talking to more than one expert and an iterative approach in order to make sure that the right parameters are used for the case base. In the following sections we present such an approach.

The case quality needs to be safeguarded as the case base must contain a representative set of problem solution pairs from the domain at the initial stage of the CBR system. At the same time we need to ensure that the case-base yields high quality results. Little attention has been given to case-quality in the available literature, and therefore the CBR expands without inspecting itselfă[91]. We want to address the quality problem by making sure that both the initial case information as well as the cases to be learned will be initiated and checked by humans.

If the case template is wrong, the result will be wrong. There is a need to understand how the case template is defined, in practice. Next we will see who has addressed this central issue and what they can tell us about how to do address it.

Öztürk and Tideman's [144] statement "We believe case grounding problem is the reason why CBR has not seen wide-spread adoption in the industry - because manual extraction of cases from reports and records is costly and time consuming" is one of the main reasons why we report our approach related to knowledge acquisition. We do agree that it is a time consuming effort, though, when consulting existing literature for knowledge acquisition for CBR to learn how to extract and categorize the relevant information, we did not find any clear guidelines or methodological descriptions for case grounding. This might be an additional reason to why CBR has not seen a widespread adoption in the industry.

The recent trend is to (semi)-automate the case acquisition process

[95, 91, 150, 151, 152]. The approach sketched by [95] is based on initiating the case base with random values, though still based on a formalized data-sheet template for case representation. However, there is no mentioning on how the template was established (the assumption is that domain experts have been asked). They state: "Case engineering is among the most complicated and costly tasks in implementing a case-based reasoning system".

The cases that are part of the case-base are supposed to yield solutions to the problems with minimal adaptation or human input. This is desirable as otherwise the major usefulness of a CBR system to reuse existing knowledge would be substantially harmed [153]. This implies that the case base must support this type of knowledge.

Richter [22] describes knowledge containers as keepers of case information. The first requirement is that the case base should only contain cases (p, s) where the utility of s is maximal or at least very good for the problem p. This is knowledge contained in the individual cases.

The case acquisition process itself, meaning the initiation of a case base, is not described though 4 different sources are mentioned in Richter's invited talk at ICCBR in 1995 [154]: domain knowledge, cases, similarity knowledge and adaptation knowledge.

The domain knowledge is what fills the template which can be used for matching cases. According to, template retrieval is similar to SQL queries in databases, where all cases fitting a template of parameters are retrieved. The main merit of using of template retrieval is that the faster retrieval and high currency by prevents irrelevant case from being considered in similarity matching.

Aamodt [155] described a framework for modeling the knowledge contents of CBR systems based on Richters knowledge containers. The model suggests decomposition in three perspectives. The power of using three perspectives (tasks, methods, and models) for knowledge level modeling lies in the interaction between the perspectives, and the constraints they impose on each other. However, there is no description on how to initiate the case base.

Cordier et al [156] state that when there is a lack of domain knowledge, the system may infer a solution that is correct with respect to the knowledge base but not with the real world: making the results invalid in the real world. The FRAKAS system [157] is an approach for interactive domain knowledge acquisition. Learning takes place during the use of the system and aims at acquiring domain or adaptation knowledge. The evaluation of the adapted solution may highlight that it does not meet the requirements of the target problem. In this situation, a reasoning failure occurs and is processed by a learning process. The expert is involved in the process of

identifying inconsistent parts of the solution which helps to augment the knowledge base. The expert is involved in a simple manner to point out faulty knowledge and he/she may provide a textual explanation of the identified error to support complementary off-line knowledge acquisition. The approach defined here is interesting with respect to further population of a knowledge base, and a similar approach can be used both to fill the knowledge base once a basic case set has been established as well as a part of the regular learning curve (one of the 4 R's).

As in the CBR literature little is mentioned on how to populate the initial case-bases, we turn to the cognitive science domain where the fundamental concept is that "thinking can best be understood in terms of representational structures in the mind and computational procedures that operate on those structures."[12] Cognitive science in turn is related to the knowledge management and knowledge engineering field where extracting information from experts in order to create the foundation for among others expert systems has matured over the past decades. Watson [158] does describe how to apply knowledge management for CBR, however, it lacks detail on the establishing of the case base. Cognitive science is also mentioned in [159] and regarding representation of knowledge they state the following: "more generic issues of knowledge representation are seldom addressed". Followed by "The case base plays a special role because the cases can be entered without understanding them. The main point is that knowledge can be shifted between containers (their content is not invariant), which can be modeled using a learning process. In addition, the shifting can be done manually without the support of a learning method".

Our guiding motivating hypothesis is that an operator support system based on case-based reasoning can help speed up the cutting/milling process while maintaining satisfactory quality results.

As the intention is to create an operator support system using CBR, we need a formal representation of the cases. By creating a domain model, we separate domain knowledge from the operational knowledge, enable the reuse of domain knowledge and make domain assumptions explicit. Once the domain model is in place, we can also populate the case base with relevant cases. Finding out what a relevant case is and what needs to be represented in the domain model go hand in hand. Our second hypothesis is that a user-centered iterative approach is a good method to create a good formal representation as a basis for the operator support system.

[12] Thagard, Paul, Cognitive Science, The Stanford Encyclopedia of Philosophy (Fall 2008 Edition), Edward N. Zalta (ed.).

A.2 Method

While many publications (i.e. [160, 161, 162, 163, 164]) do describe the knowledge acquisition approach for their domain, most do it on a relatively technical level. We have applied several methods for knowledge acquisition and the focus has been on a user-centered iterative process. In the subsections below we give a brief explanation of these methods and highlight our experience with these forms of knowledge acquisition.

A.2.1 Research method

To systematically guide our research in this project we used the design science research method is used according to [165], as depicted in Fig. A.47. The research environment consists of machine supplier experts, as well as machine operators. The research is driven by the need to use the machines in an optimal manner, with the assumed outcome a more optimal operation and therewith cost reductions. The knowledge base is based on the existing literature on CBR and knowledge acquisition as well our own findings.

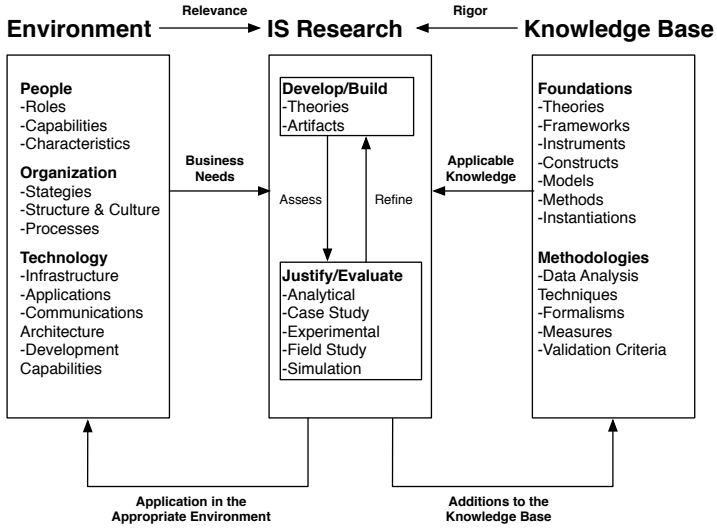


Figure A.47: Conceptual framework for IS research [165]

- What is the effect of introducing an expert system based on CBR on the effectiveness of operators?

- What is the effect of introducing a distributed expert system based on CBR?

User-centered design is conducted prior to the development of complex systems to ensure deep understanding of user and stakeholder roles. The aim is to ensure that system designed support the daily work of end users and the role of stakeholders [166, 167]

We have applied user-centered design in all activities in the iterative process of assessing and refining our artifacts, adding it to our knowledge base. The activities are carried out in close cooperation with real stakeholders by means of various methods for data collection, as described in the sections below.

A.2.2 Data collection

In our study we focus on a single manufacturer of digital conversion tables. The study is based on design science research and evaluation research and has been implemented at 3 different locations that represent a typical customer of this manufacturer.

The intervention is the introduction of a distributed case-based decision support system to support operators to make the right decisions quicker and therewith both reduce the number of errors and speed up the full process of job shifting.

The artifact to be created for this intervention is a research challenge itself as populating a knowledge base is a non-trivial task. A step wise approach for populating a CBR knowledge base will be developed and the effect will be tested.

Some of the needed information can be retrieved from logs, though this is a non-validated information source.

The study is divided in two parts: data collection participants and intervention participants. The data collection methods are described in the sub-sections below.

In both cases the population is recruited the manufacturer of digital conversion tables- and the inclusion criterion is that the participant is currently a customer operating digital conversion machines. We focus on the data collection part in this paper.

For the data collection, we focus around the following questions

- Which information to extract from the operators?
- What is/are the bottleneck(s) in the load shift?
- Which factors impact the time used?
- What is the mean time?

- Does the knowledge of an operator impact the operation? And in what way?
- How much information are companies willing to share with competitors?
- How and when to present suggestions from the expert system to operators?

In the sub-sections below we first present which methods we have used for the data collection and in section 3 we provide the results of the activities.

A.2.2.1 Observation

The first data gathering activity was based on observations. The intention was to form a structure for later interviews and the first subject was asked to explain (while preparing and operating the cutting table) what he was doing and why he was doing it this way. The observer did not interfere with the process.

A.2.2.2 Semi-structured interviews

We have conducted interviews at digital finishing companies in Norway, Belgium and The Netherlands. The interview subjects were mainly cutting table operators, though also managers/owners. As the companies were relatively small, the latter category also in all cases were table operators, yet not on a daily basis. The interview questions were based on the results from the observation session and have been expanded based on finding between the interviews. We used a set with main questions and expanded while commencing the interview.

A.2.2.3 Questionnaire

We have developed a questionnaire in order to map the time operators use when operating the machines. It was sent to 100 digital finishing companies throughout the world.

A.2.2.4 Workshops

The technology provider catered for a workshop with employees with a computer science background. During this workshop technical boundaries were explored and details regarding the integration of the operator support system discussed.

A.2.2.5 Re-use of available data

We have gained access to a product guide describing which tools can be used for which materials, and for some of these also a set of settings for certain material types. However, the settings are relatively conservative as they pertain to a material family. Specific materials use material specific settings which can be much faster than the material family setting. For the most used specific materials, specific settings are available. Also an operator manuals of the current Esko machines with i-cut software has been used as an information source.

A.3 Results

A.3.1 Case study: As is situation

Input for the study uses the data gathering methods described above, in addition, one of the researchers took a table operator course to get a real hands-on feel of using the system.

In Fig. A.48 you see the repetitive and cyclic process of enhancing the input, which can be mapped to the IS research part of Fig. A.47; both Develop/build and justify/evaluate to ensure both relevance and rigor.

The methods have been applied to digital finishing companies in Norway, Belgium and The Netherlands.

Unfortunately, the response rate for the questionnaire was so low that we were unable to use the results as a pinpoint for the average type of operator and other information regarding machine use.

From the observation and interview activities, we learned that machine operation to a large degree is completely experience based and that the experience transfer is sub-optimal. Some factors that influence the choices are the quality of the material that is used, the wear and tear of the used actuator, the desired output quality (not all customers demand a high quality finish) as well as the time available between jobs. An ideal situation according to one of the shop managers is that the machine is in use continuously. We did a test using optimal speed settings with new actuators and high quality material vs the regular settings with a new actuator and high quality material. We found that the cutting speed in this specific case was 13 minutes vs 22 minutes. This supported the assumption that the operators do not use the optimal settings and that an operator support system indeed can be useful. For this specific case, relating to RQ1, we can state that there is a good effect in using the operator support system recommendations.

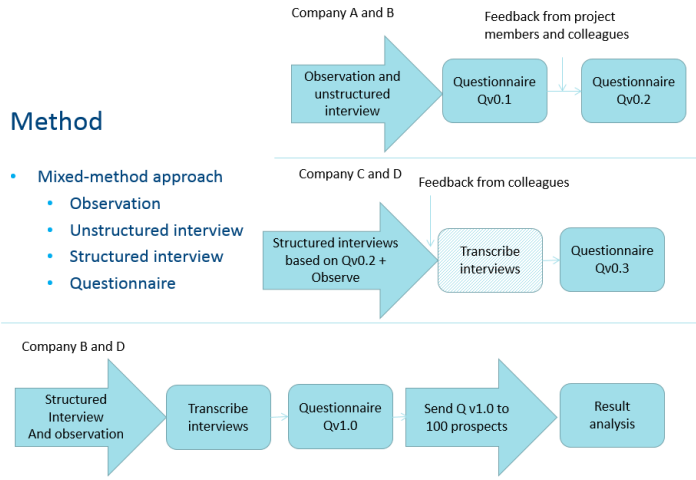


Figure A.48: Knowledge acquisition

Knowing the type of information the operators wish to use and how they wish to use it, we discussed the technical boundaries with the table and cutting table software provider. We gained access to subsets of the required information required to create an operator support system. All of the gathered information has been structured into a domain model. See the next subsection for more details.

A.3.2 Domain model

In order to model the domain, we need to map domain knowledge (for an impression, see Fig. A.49). The main parameters that need to be contained can be summarized as such:

/A cutting/milling **job** is performed by an **operator** on a **cutting table** which uses a **tool set** with different **actuators** on a **material type** following **patterns** stemming from a **design**. Considerations regarding the **speed** and **quality** of the job are done by the **operator** based on previous experiences and **customer demands**./

The previous sentences describe what the domain model needs to include on an overall level. In short, it needs to include all relevant information for an operator to be able to do a job in the fastest possible manner or with the highest quality possible. These two are not always mutually exclusive, though high speeds can sometimes lead to a lower end-product quality. In some cases, the lower quality is still within the quality assurance threshold.

- Some questions that the operator support system needs to be able to help answer are: Which settings should I avoid to use?
- What is the most optimal setting for this particular job with regards to either quality or speed?
- What is the maximum speed I can use?
- Will these settings break stuff?
- Which settings should I change?
- Will this actuator (bit/blade) work with this material?
- What are the limitations of this tool applied on this material?

These questions imply that we need to know about the properties of the materials, design, tools and table. During the domain knowledge gathering process, we have identified the relevant terms to include in the domain model. Due to space restrictions, we do not include the domain model in this paper, though some of it can be seen in the screenshots from MyCBR.

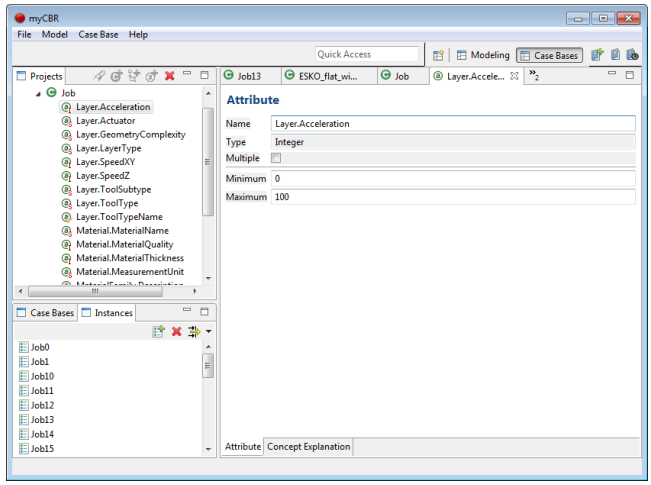
One of the results from the interviews shows that operators are more likely to trust a recommendation if an explanation is given. If the settings are presented following a pattern such as "in a similar case we have successfully applied the following settings with a satisfactory quality" followed by a question if the operator wishes to use these settings instead, the operators responded positively. However, without such explanation, the operators would not simply accept new settings.

A.3.3 CBR

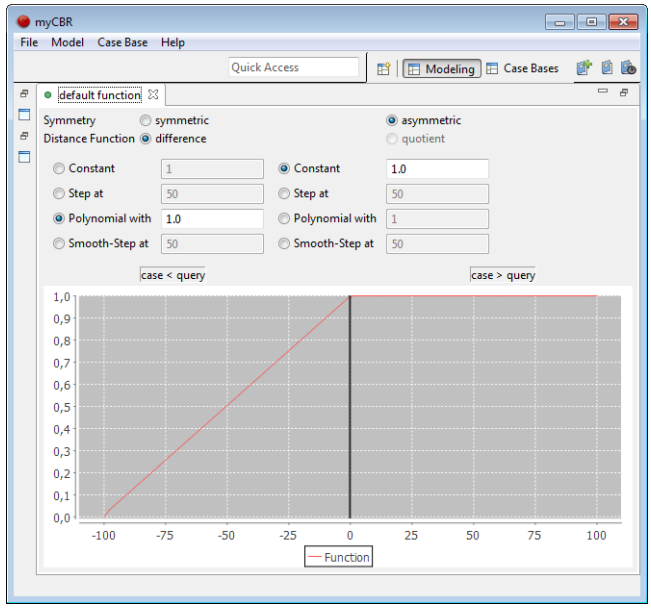
We have applied the domain model and created a CBR system prototype using MyCBR [168].

The initial case base has been made in close cooperation with experts from the company. Instances with proven cases in different levels of aptness have been entered. It is important to note that these cases are based on material family and not a specific instance of the material itself. As properties are supplier specific, different settings should be used. These settings will during the course of the use of the system be formed as cases.

The similarity features (Fig. A.49) are based on conversations with a tool and material experts. Each specific material combined with specific actuators have specific settings, also pertaining to the complexity of the output to be generated.



(a) Domain model



(b) Similarity measure

Figure A.49: A screenshot of the domain model (A.49a) and an example of a similarity measure (A.49b).

A.3.4 Validation

Testing has been done based on the different cases with each their rating. For material types two or more different cases have been entered in the initial case base, including an indication in the aptness. Similarity values and weights have been tuned in order to get the closest case to match. This was later tried with new cases and the results were satisfactory. A screenshot of matching results is shown in Fig. A.50.

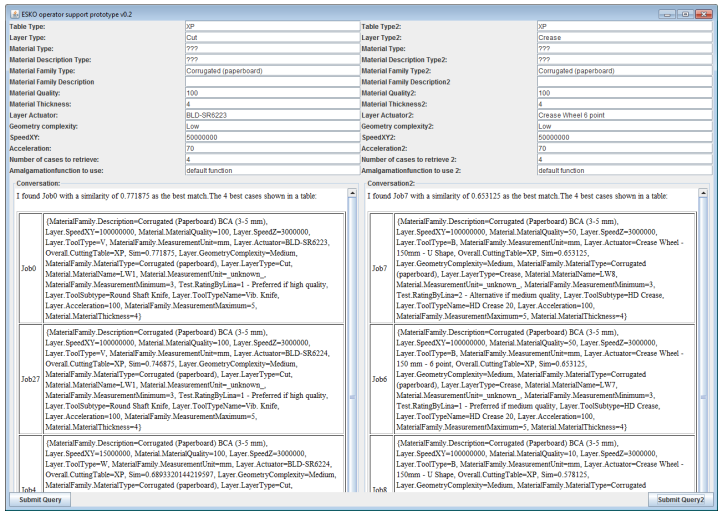


Figure A.50: Matching results screenshot

A.4 Discussion and lessons learned

The variety of knowledge elicitation methods we have used and the variety of companies visited may seem like an too rigorous information gathering, though we feel that in our case this was the right thing to do. It is time and resource demanding, though by presenting the various approaches, we hope to contribute to the knowledge gap that seems to exist concerning creating an initial case-base. Different situations cater for different methods of knowledge elicitation, and in many cases, a less rigorous approach might be sufficient. Creating a sound and valid foundation for the case template and case base is resources demanding. However creating a CBR system that is neither valid or useful is even more resource demanding. In general we can recommend to talk to the system owner and a variety systems users

multiple times in order to best understand the problem at stake and validate that the researchers (CBR system builders) really understand the problem that the CBR system is to solve in a manner that is useful for the end-users.

A.5 Conclusion and further work

This study has presented a use case for how to create a CBR system with focus on building the initial case base, and the case template or domain model. To create grounded basis for our CBR system, case template and domain model we; observed the operators, performed interviews with the operators, organized interactive workshops with the operators, collected questionnaires and utilized available product data.

These data sources all went into the design of the case base, case template and domain model. An initial validation at one of the companies shows that operators recognize and understand the CBR system inputs and outputs. This serves as an example use case that works toward solving the problems highlighted by [144]. With regards to the main motivating hypothesis of this work initial tests also shows increase in the operation of the machine that is augmented by the CBR system. In the next part of this project the system will be tested more thoroughly in terms of performance increase in the target domain of the CBR system. In addition, we will develop a method for abstracting and extracting high level knowledge from cases to be sent into a distributed case base to ensure both knowledge sharing across competing stakeholders while not disclosing competitive advantages.

A.6 Acknowledgments

The authors gratefully acknowledge the Norwegian Research Council and the BIA program for financial support of the project (partially through grant 235427) as well as the participating case companies, which together enabled this study.

References

[2] Aamodt, A. and Plaza, E. “Case-based reasoning: Foundational issues, methodological variations, and system approaches”. In: *AI communications* vol. 7, no. 1 (1994), pp. 39–59.

[22] Richter, M. M. “Knowledge containers”. In: *Readings in Case-Based Reasoning* vol. Morgan Kaufmann Publishers (2003).

[91] Yang, C., Farley, B., and Orchard, B. "Automated case creation and management for diagnostic CBR systems". In: *Applied Intelligence* vol. 28, no. 1 (Feb. 2007), pp. 17–28.

[95] Dufour-Lussier, V., Ber, F. L., Lieber, J., and Nauer, E. "Automatic case acquisition from texts for process-oriented case-based reasoning". In: *Information Systems* vol. 40, no. nil (2014), pp. 153–167.

[144] Öztürk, P. and Tidemann, A. "A review of case-based reasoning in cognition–action continuum: a step toward bridging symbolic and non-symbolic artificial intelligence". In: *The Knowledge Engineering Review* vol. 29, no. 01 (2014), pp. 51–77.

[145] Hinkle, D. and Toomey, C. "Applying case-based reasoning to manufacturing". In: *AI magazine* vol. 16, no. 1 (1995), p. 65.

[146] Nagendra Prasad, M. V., Lander, S. E., and Lesser, V. R. *On retrieval and reasoning in distributed case bases*. Generic. 1995.

[147] Plaza, E. and McGinty, L. "Distributed case-based reasoning". In: *The Knowledge Engineering Review* vol. 20, no. 03 (2005), pp. 261–265.

[148] Leake, D. B. and Sooriamurthi, R. *When Two Case Bases Are Better than One: Exploiting Multiple Case Bases*. Conference Paper. 2001.

[149] Roth-Berghofer, T. R. "Explanations and case-based reasoning: Foundational issues". In: *Advances in Case-Based Reasoning (In Peter Funk and Pedro A. González-Calero, editors)*. September 2004, pp. 389–403.

[150] Manzoor, J., Asif, S., Masud, M., and Khan, M. J. "Automatic case generation for case-based reasoning systems using genetic algorithms". In: *2012 Third Global Congress on Intelligent Systems*. Nov. 2012, nil.

[151] Shokouhi, S. V., Aamodt, A., and Skalle, P. *A semi-automatic method for case acquisition in CBR a study in oil well drilling*. Generic. 2010.

[152] Shokouhi, S. V., Skalle, P., and Aamodt, A. "An overview of case-based reasoning applications in drilling engineering". In: *Artificial Intelligence Review* vol. 41, no. 3 (2014), pp. 317–329.

[153] Cunningham, P. "CBR: Strengths and weaknesses". In: *Tasks and Methods in Applied Artificial Intelligence*. Ed. by Pobil, A. P. d., Mira, J., and Ali, M. Lecture Notes in Computer Science. 1998, pp. 517–524.

[154] Veloso, M. and Aamodt, A. *Case-Based Reasoning Research and Development: First International Conference, ICCBR-95, Sesimbra, Portugal, October 23-26, 1995. Proceeding*. Vol. 1010. 1995.

[155] Aamodt, A. *Modeling the knowledge contents of CBR systems*. Conference Paper. 2001.

[156] Cordier, A., Fuchs, B., Lieber, J., and Mille, A. *Failure analysis for domain knowledge acquisition in a knowledge-intensive CBR system*. Conference Paper. 2007.

[157] Cordier, A. "Interactive and opportunistic knowledge acquisition in case-based reasoning". Thesis. Université Claude Bernard-Lyon I, 2008.

[158] Watson, I. *Applying Knowledge Management: Techniques for Building Corporate Memories*. 2003.

[159] Richter, M. M. and Aamodt, A. "Case-based reasoning foundations". In: *The Knowledge Engineering Review* vol. 20, no. 03 (2005), pp. 203–207.

[160] Bach, K. *Knowledge Acquisition for Case-Based Reasoning Systems*. 2012.

[161] Bergmann, R. *Experience Management Foundations, Development Methodology, and Internet-Based Applications*. 2002.

[162] Tautz, C. *Costumizing Software Engineering Experience Management Systems to Organizational Needs*. 2000.

[163] Bergmann Ralph, e. a. *Developing industrial case-based reasoning applications: The INRECA methodology*. 2003.

[164] Kezunovic, M. and Rikalo, I. "Detect and classify faults using neural nets". In: *IEEE Computer Applications in Power* vol. 9, no. 4 (1996), pp. 42–47.

[165] Hevner, A. R., March, S. T., Park, J., and Ram, S. "Design science in information systems research". In: *MIS Quarterly* vol. 28, no. 1 (2004).

[166] Kubie, J., Melkus, L. A., Johnson, R. C., and Flanagan, G. a. "User-centred design". In: *IS Management Handbook: 7th Edition*. Ed. by Brown, C. V. and Topi, H. 7th. Boca Raton, FL, USA, 1999.

[167] Shluzas, L. A., Steinert, M., and Katila, R. "User-centered innovation for the design and development of complex products and systems". In: *Design Thinking Research*. 2014, pp. 135–149.

[168] Stahl, A. and Roth-Berghofer, T. R. “Rapid prototyping of CBR applications with the open source tool myCBR”. In: *European conference on case-based reasoning*. Springer. 2008, pp. 615–629.

Demonstrating the MYCBR Rest API

Kerstin Bach, Bjørn Magnus Mathisen, Amar Jaiswal

VI

VI

Abstract

Case-based reasoning (CBR) tools are important to reduce the effort of developing CBR systems. myCBR has been a tool for researchers and practitioners over the last ten years providing CBR system building blocks and functionality through the myCBR-SDK and means to develop CBR models in the myCBR-workbench. In this paper we present the myCBR Rest API which exposes the functionality of both myCBR-SDK and myCBR-workbench though a RESTful API. It includes the myCBR-SDK functionality to enable researchers the fast development and experimentation of CBR applications from not only Java, but from the programming language of the developers choice. Most of the myCBR-workbench functionality has also been exposed in the same fashion enabling users to programmatically create, modify and delete CBR models and case-bases, so that the Rest API also allows myCBR to act as a service, and be accessed by the client software through HTTP.

B.1 Introduction

CBR tools have been developed since the very beginning of the CBR research activities. The most general CBR tools developed and provided as bundled or open source software are COLIBRIStudio (and their predecessors COLIBRI, jCOLIBRI) [169], CBRworks [170] and its successor myCBR[168]. Furthermore there are more specific CBR tools targeting certain domains or case representations. For process-oriented CBR, the Collaborative Agent-based Knowledge Engine (CAKE) [171] has been introduced, while CREEK [172] is a tool for knowledge-intense CBR and (B)EAR [173, 174] focuses on the adaptation in CBR systems. In addition there are also application specific tools such as eXiT*CBR for medical diagnosis.

myCBR[13], which is the basis of this work, was developed by German Research Center for Artificial Intelligence (DFKI) and has been introduced as rapid prototyping tool for research and industrial applications. Initially, myCBR was introduced as an add-on to the Protégé ontology tool [175], but later re-implemented as a Java standalone tool and software development kit (SDK) [176]. Up until today, using myCBR still required the CBR system to be integrated in a Java environment.

Allowing users to model a CBR system using myCBR's workbench and then deploying the application as a web service would make it easier to build, test, compare and deploy CBR applications.

[13] http://mycbr-project.org

Much of the current success of Machine Learning can be linked to the availability of machine learning models in tools like Scikit-learn [177] and Keras [178]. Such tools allows researchers and developers to use the methods more easily and provide reproducible results [179]. Currently successful Machine Learning APIs are DialogFlow[14], TensorFlow.js[15] or the Microsoft Cognitive Toolkit[16]. They all have in common that users interact with them via web pages or web services. Users of such services take responsibility for the provision of data, configuration of each method's parameters, and validation and verification of the results. The services provide the functionality of the core methods.

RESTful (REpresentational State Transfer) Application Programming Interface (API) [180] is the current state-of-the-art to provide web services. Restful APIs, or Rest APIs, have been developed and their services have been deployed in industry over many years. However, in research they have not received a lot of attention. We believe, however, that implementing research prototypes using a Rest API can be a way forward to ease the development of systems in general, and CBR systems in particular.

B.2 MYCBR Rest API

MYCBR applications are typically built as a Java application on top of the MYCBR-SDK Java library. MYCBR-REST is designed to expose MYCBR's modeling functionality (e.g. creating concepts and similarity functions) and MYCBR's runtime functionality through a HTTP REST API. This enables the user to programatically access the features of what was previously exposed through two different tools (MYCBR workbench and MYCBR-SDK) into an API. This API is also conveniently accessible from all programming languages that support accessing HTTP REST APIs. Figure B.51 shows the architecture of this design.

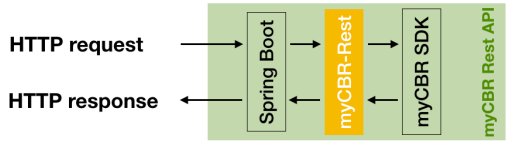


Figure B.51: Components of the MYCBR-REST architecture.

[14] https://dialogflow.com/
[15] https://www.tensorflow.org/js/
[16] https://docs.microsoft.com/en-us/cognitive-toolkit/

The Rest API has been implemented using the Spring Boot Framework and configured to expose its documentation via a tool called swagger. From swagger a developer can use the interactive documentation for testing requests and developing applications.

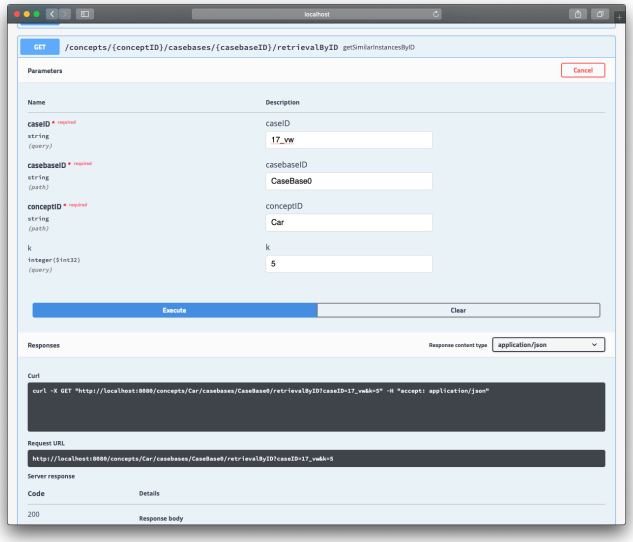


Figure B.52: Example of the Swagger tool that shows how a Rest call can be tested against a CBR application

During the demo, we will show how use the Rest API to obtain information about the case representation, carry out the retrieval as well as evaluating the results. We will provide examples using Python 3, however the Rest API can be used with all programming languages languages that support Rest API and parsing JSON objects.

B.2.1 Retrieval

Figure B.53 shows how the retrieval against the Rest API can be implemented. The JSON data returned by the API can be directly included in data frames and further evaluated from there.

B.3 Experiments and Applications

The myCBR Rest API described in this demo has been used to build and experiment with CBR systems over the last years. In the following, we will describe the different areas of application.

```
In [2]: server = 'localhost'

        port = '8080'
        url = 'http://' + server + ':' + port + '/'

In [3]: def retrieve_k_sim_byID_content(concept, casebase, amalFct, queryID, k):
            raw = pd.DataFrame(requests.get(url + 'concepts/' + concept + '/casebases/'
                                            + casebase + '/retrievalByIDWithContent?amalgamation%20function='+amalFct+'&caseID='
                                            + queryID + '&k=' + k ).json())
            results = raw.apply(pd.to_numeric, errors='coerce').fillna(raw).sort_values(by='similarity', ascending=False)
            returned_df = results.apply(pd.to_numeric, errors='coerce').fillna(results)
            return returned_df

In [4]: # Find most similar cases and get the full case content for them
        retrieve_k_sim_byID_content('Car', 'CaseBase0', 'CarFunc', '17_vw', '5')
```

Out[4]:

	Body	CCM	Car Code	Color	Doors	Gas	Manufacturer	Miles	Model	Power	Price	Speed	Year	ZIP	caseID	similarity
0	sedan	1900	17	dark_blue	4	diesel	vw	71433	passat	90	22099	183	1995	6	17_vw	1
1	sedan	1900	891	blue	4	diesel	vw	68993	passat	110	23599	183	1995	2	891_vw	0.658371
2	sedan	1900	532	dark_blue	4	diesel	vw	13701	passat	90	33199	183	1996	3	532_vw	0.658114
3	sedan	1900	602	blue	4	gasoline	vw	80748	passat	193	26499	183	1996	1	602_vw	0.632703
4	sedan	1900	467	dark_blue	4	gasoline	vw	91911	passat	100	16699	183	1995	8	467_vw	0.632574

Figure B.53: Example of the Python Code carrying out a retrieval using the Rest API

B.3.1 Application in Research Projects

Up to today the integration of a CBR component as a web service required a Java-based framework. With the presented work, a CBR engine can be developed and easily deployed. One of the currently running CBR applications developed in the SELFBACK project [181] uses this infrastructure. Here the MYCBR-driven engine is the core of the decision support system to compare patient profiles and generate self-management plans for low back pain patients. Moreover the extended RESTful services are used to monitor and evaluate the patient's progress and provide responses to clinicians.

Furthermore, research prototypes have been developed in the last three years. Skjold et al. [182] presented an application focusing on adapting sandwich recipes using CBR[17]. In this application the Rest API has been used as the backend to carry out the similarity-based retrieval and building ephemeral case bases for the adaptation processes. In Engin et al. [183], Verma et al. [184] and Jaiswal et al. [185] the Rest API was used in combination with Python to prototype and evaluate different CBR methods.

In Mathisen et al. [50] Rest API is used to evaluate the performance of different neural network architectures used as a amalgamation function. The neural amalgamation function can be used to add different pre-trained neural networks to be used a global similarity function.

[17] The prototype of IntelliMeal is available at http://hv-6151.idi.ntnu.no

B.3.2 Application in Education

During the last two years while the API has been developed, we introduced it to students who then implemented CBR systems during courses at Master's and PhD level. Especially web-based applications such as IntelliMeal ([182]) were created. In the beginning of the courses we gave an introduction explaining on how to create a myCBR project using the myCBR tutorial[18], followed by the introduction to the myCBR Rest API similar to the examples given in the aforementioned iPython Notebook. Until now, about ten myCBR applications have been developed and experimented with at NTNU using the introduced approach while four more are currently under development.

Compared to the Java-only approach we see that the time until the first prototype is up and running is dramatically reduced. Also, for a basic CBR system, no Java knowledge is needed and students can focus on case representations, similarity measure development or adaptation strategies. The students found it convenient to interact through the REST interface, so that the integration of a CBR component in a more comprehensive data processing pipeline is feasible.

B.4 Conclusion and Outlook

This demo presents a further development of the myCBR tool that allows developers and researchers easier prototyping, integration and deployment of CBR systems. The myCBR Rest API creates CBR services that can be deployed and used anywhere as well as it allows a systematic evaluation of the CBR systems. All content provided in this paper is made available on GitHub under LGPL to be shared with the community[19]. In conclusion, the myCBR Rest API provides a flexible framework for creating CBR systems and developing new components. Moreover, it lowers the entry bar to experiment with Case-Based Reasoning and compare it with other AI methods.

Acknowledgement

This work has been supported by the Norwegian Open AI Lab as well as NTNU's rector's funds for multidisciplinary research. Further, parts have been conducted within the selfBACK research project, which has received funding from the European Unions Horizon 2020

[18] http://mycbr-project.org/tutorials.html
[19] https://github.com/orgs/ntnu-ai-lab/teams/mycbr/repositories

research and innovation programme under grant agreement No 689043 and the EXPOSED Aquaculture centre for research-based innovation funded by the centre partners and the Research Council of Norway.

References

[50] Mathisen, B. M., Aamodt, A., Bach, K., and Langseth, H. "Learning similarity measures from data". In: *Progress in Artificial Intelligence* (Oct. 2019), pp. 129–143.

[168] Stahl, A. and Roth-Berghofer, T. R. "Rapid prototyping of CBR applications with the open source tool myCBR". In: *European conference on case-based reasoning*. Springer. 2008, pp. 615–629.

[169] Díaz-Agudo, B., González-Calero, P. A., Recio-García, J. A., and Sánchez-Ruiz-Granados, A. A. "Building CBR systems with jcolibri". In: *Science of Computer Programming* vol. 69, no. 1 (2007). Special issue on Experimental Software and Toolkits, pp. 68–75.

[170] Schulz, S. "CBR-Works - A state-of-the-art shell for case-based application building". In: *Proceedings of the 7th German Workshop on Case-Based Reasoning, GWCBR'99, Wrzburg*. 1999, pp. 3–5.

[171] Bergmann, R., Gessinger, S., Görg, S., and Müller, G. "The collaborative agile knowledge engine CAKE". In: *Proceedings of the 18th International Conference on Supporting Group Work*. GROUP 14. New York, NY, USA, 2014, pp. 281–284.

[172] Aamodt, A. "Knowledge-intensive case-based reasoning in CREEK". In: *Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004, Madrid, Spain, August 30 - September 2, 2004, Proceedings*. 2004, pp. 1–15.

[173] Jalali, V. and Leake, D. "CBR meets big data: A case study of large-scale adaptation rule generation". In: *Case-Based Reasoning Research and Development*. Ed. by Hüllermeier, E. and Minor, M. Cham, 2015, pp. 181–196.

[174] Jalali, V. and Leake, D. "Enhancing case-based regression with automatically-generated ensembles of adaptations". In: *J. Intell. Inf. Syst.* vol. 46, no. 2 (Apr. 2016), pp. 237–258.

[175] Musen, M. A. "The Protégé project: A look back and a look forward". In: *AI Matters* vol. 1, no. 4 (June 2015), pp. 4–12.

[176] Bach, K. and Althoff, K.-D. “Developing case-based reasoning applications using myCBR 3”. In: *Case-Based Reasoning Research and Development*. Ed. by Agudo, B. D. and Watson, I. Berlin, Heidelberg, 2012, pp. 17–31.

[177] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. “Scikit-learn: Machine learning in Python”. In: *J. Mach. Learn. Res.* vol. 12 (Nov. 2011), pp. 2825–2830.

[178] Chollet, F. et al. *Keras*. https://keras.io. 2015.

[179] Recio-Garca, J. A., Daz-Agudo, B., and González-Calero, P. A. “The COLIBRI open platform for the reproducibility of CBR applications”. In: *Case-Based Reasoning Research and Development - 21st International Conference, ICCBR 2013, Saratoga Springs, NY, USA, July 8-11, 2013. Proceedings*. 2013, pp. 255–269.

[180] Fielding, R. T. “Architectural styles and the design of network-based software architectures”. AAI9980887. PhD thesis. University of California, Irvine, 2000.

[181] Mork, P. J. and Bach, K. “A decision support system to enhance self-management of low back pain: Protocol for the selfBACK project”. In: *JMIR Res Protoc* vol. 7, no. 7 (July 2018), e167.

[182] Skjold, K., Øynes, M., Bach, K., and Aamodt, A. “IntelliMeal - Enhancing creativity by reusing domain knowledge in the adaptation process”. In: *ICCBR 2017 Workshops Proceedings, Trondheim, Norway, June 26-28, 2017*. 2017, pp. 277–284.

[183] Engin, H. K., Nadim, F., Carotenuto, P., and Bach, K. “Estimation of pile capacities using case-based reasoning (CBR) method”. In: *Proceedings of 4th International Symposium on Computational Geomechanics*. 2018.

[184] Verma, D., Bach, K., and Mork, P. J. “Modelling similarity for comparing physical activity profiles - a data-driven approach”. In: *Case-Based Reasoning Research and Development*. Ed. by Cox, M. T., Funk, P., and Begum, S. Cham, 2018, pp. 415–430.

[185] Jaiswal, A., Bach, K., Meisingset, I., and Vasseljen, O. “Case representation and similarity modeling for non-specific musculoskeletal disorders - a case-based reasoning approach”. In: *FLAIRS-32 Conference*. 2019.

Use Case applying machine-learning techniques for improving operation of the distribution network

Jørn Foros[20], **Maren Istad**[21], **Andrei Morch**[22], **Bjørn Magnus Mathisen**[23]

VII

VII

Abstract

This paper discusses the use of machine learning (ML) techniques to improve fault handling in distribution networks. The paper includes a short survey on the use of ML techniques in fault handling and shows that little published work has been done on using weather data and smart metering data as data sources. It can be argued that this is desired to increase the performance and usability of ML in operational support systems. Previous work also focuses almost exclusively on statistical machine learning aiming to replace traditional simulation models, overlooking other ML methods which can support operations. Here it is illustrated that Case based reasoning (CBR) can be used to aid the decision-making for example, when trying to restore service after an outage. The paper also describes the use of experience databases to aid the operator during fault handling. To illustrate potential use of ML and CBR, the paper presents a use case for future fault handling in low voltage distribution network and discusses the usefulness of this approach. This example shows that implementation of ML techniques in daily operation can be expected to contribute to reduction of costs for the network companies and increased security of supply for the customers.

C.1 Introduction

A distribution system operator (DSO) naturally wants to avoid outages in the power supply. Outages can cause large costs for repairs and incur penalties like cost of energy not supplied (CENS). In addition, outages are inconvenient for the customers and might damage the reputation of the DSO. During localisation and repair of the faults, the personnel can suffer injuries, especially during demanding weather conditions such as strong winds and snow. Hence, improving the precision of fault localization is desirable to reduce the outage duration and increase personnel safety.

Localising a fault can at present be time consuming and it is difficult to provide customers with precise and timely information during the outage. All customers connected to the distribution network in Norway must have a smart meter installed by 1.1.2019, and this provides new source of information for the DSOs that can be useful for fault handling, as well as other applications. Smart meters can notify the DSO when power is lost, and this is a large improvement

[23]SINTEF Energy Research – Norway SINTEF Energy Research - Norway SINTEF Energy Research - Norway
jorn.foros@sintef.no maren.istad@sintef.no andrei.morch@sintef.no

compared to current practice where the DSO rely on customers alerting them. At the same time development of information and communication technologies (ICT) and cost reductions for computing power allows application of advanced techniques such as machine learning (ML) to resolve many operational issues, including fault handling, more efficiently. The planned introduction of CENS for households by 1.1.2020[24] is expected to increase the focus on fault localization and repair in the low voltage distribution network. According to a white paper [186], analytics is becoming a part of core business processes for an increasing number of utilities. Three high priority areas for analytics are reported to be energy forecasting, smart metering analytics and asset management. In a review paper [187] four analytics areas for smart metering data are highlighted; load analysis, load forecasting, load management and miscellaneous, with the latter including outage management.

This paper starts with a short survey of the application of machine learning techniques in fault handling. The perpetual challenge of outage and fault management for DSOs is then discussed. Finally, the paper presents and discusses a use case for improved future fault handling in low voltage distribution networks utilising ML techniques. The paper constitutes the first results of the activity "Smart Grid Operation", which is a part of the Norwegian program Centre for Intelligent Energy Distribution (CINELDI), which is a Centre for Environmentally-Friendly Energy Research (2016-2024) [25].

C.2 Machine Learning Techniques In Fault Handling

Machine learning has been applied within the energy domain to solve several types of problems such as predicting power generation from solar panels [188], detecting cyber-attacks in the grid, optimizing power consumption in large data centres and predicting failures in grid components [189]. Here we focus on the handling of faults in the distribution network from the perspective of the DSO. Thus, we have chosen to specifically identify machine learning techniques previously used for identifying causes and locations of faults. Most ML methods train models to fit a set of data (the training data). The output from the ML method can be seen as a function of the ML method and the training data. Thus, the results from ML methods are very dependent on the data used to train the models. As a result, the current state

[24] https://www.nve.no/reguleringsmyndigheten-for-energi-rme-marked-og-monopol/okonomisk-regulering-av-nettselskap/aktuelle-prosjekter/kile-for-husholdninger/

[25] https://www.sintef.no/cineldi

of the art of applying ML to fault handling has three main features: The machine learning method, the specific problem that is targeted, and the data used to train the model.

A short survey on the use of ML techniques for fault handling in the power system domain has been conducted. In short, the survey (for details see [190]) finds that a multitude of ML methods are being applied within fault handling: Fuzzy systems [191], expert systems [192], artificial neural networks (ANN) [193], support vector machines (SVM) and Q-learning [194]. The most prominent method by a large margin was ANN, see e.g. [193, 195, 196, 197] followed by SVM, see e.g.[193, 195, 197, 198]. The most targeted problem was fault location, see e.g. [193, 196, 198].

The survey found that most of the research done on applying ML to fault handling has been done on transmission networks, as is also seen in the survey done by Ferreira et al. [199]. Some of that knowledge is transferable to distribution networks, which is the main focus of this paper. In terms of sources for training data, mainly simulated data has been used, and very little data collected from the real world. This is probably because the number of real outages is limited. Surprisingly, we found little research done on using smart meters as a data source for fault handling.

Fault handling is in general comprised of four phases; detection, location and diagnosis, and finally repair/response. Most of the research found in our survey was focused on detecting, locating and classifying (diagnosing) the faults. Very little work has been done on applying ML to suggest responses to faults. This contributes to explaining the absence of some ML methods such as case-based reasoning (CBR)[2]. CBR is well suited for mapping similar problems (or cases) to similar solutions, especially when solutions are best described with text and not numbers. CBR is inspired by psychological models, as humans use past experiences (cases) to solve new problems. Presented with a new problem (case) a CBR system will search it is case-base (a repository of stored cases) for similar cases, then present the user with the solution to the most similar case adapted to the new case. The new solution will be stored in the case-base along with the original case description.

Based on the survey, we propose a new general architecture for fault handling in distribution networks that takes advantage of several data sources, such as smart meter data and weather data, and state-of-the-art machine learning techniques including CBR. The architecture is shown in Figure C.54.

C.3 Outage and Fault Management - an example for application of new techniques

Managing an outage in distribution networks and restoration of power supply are important tasks for a DSO. Different ways of doing this have been described in numerous use cases. A use case is commonly defined as a list of steps defining interactions between different actors in order to achieve a certain goal. Therefore, considering changes in operation of distribution networks in the future, e.g. for the time horizon 2030-2040, it is reasonable to expect that the triggering event (outage for one or several customers) and the final result or goal (restoration of the power supply) will remain unchanged. The composition of interactions among the involved actors from the initiating event to the goal is however going to be modified in the future by applying the most up-to-date technologies.

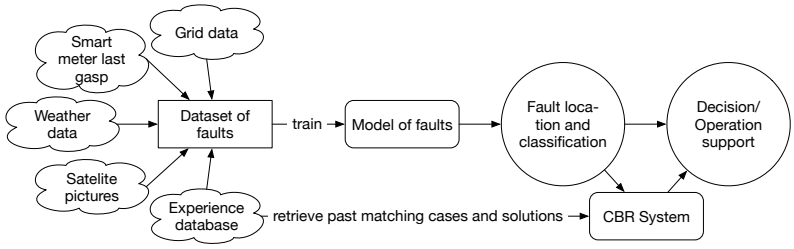


Figure C.54: A proposed architecture for fault handling in distribution networks based on machine learning techniques.

The way of dealing with faults and outages has certain limitations at present, which can be improved:

Fault handling:

- Localisation of the fault can be very demanding, especially in rural areas and in bad weather conditions
- Prioritisation of customers for reconnection during restoration of power supply can be necessary in case there are multiple faults involving many customers, such as during severe weather conditions

Work processes:

- Mobilisation and dispatching of working crews can be significantly improved if the fault localisation and fault type are well

predicted. This may be especially relevant during for example bad weather and holiday periods with high electric loads

- Automation of (the mandatory) registration of faults. Today this requires manual work that is time consuming and potentially error prone.

At the same time, there is an emerging desire at DSOs to utilise the smart metering infrastructure as much as possible in order to improve planning and operation of the distribution network. Until recently, observability of the low voltage (LV) distribution network was minimal, and DSOs were normally informed about outages directly by the affected customers. Installation of smart meters brings an opportunity to modify the process by utilising the meters so-called last gasp function. This function sends an alarm to the DSO in case the voltage drops below a pre-defined threshold or in case of an outage. Combining signals from several smart meters can allow the DSO to identify the affected area and the actual fault location more accurately [200].

The working hypothesis for this paper is that ML techniques, together with improved data availability from e.g. smart meters, can substantially improve fault handling in the future. The following use case tries to address how the above-mentioned limitations can be resolved by deploying the new techniques and data.

C.3.1 Use Case: Fault handling in low voltage distribution network in 2030/2040

The use case is inspired by the proposed architecture in Figure 1 and focuses on fault handling in the low voltage distribution network in the coming decades. In Figure C.55 a flow chart for the use case is provided. The triggering event for the use case is an outage, that may be detected in three possible ways: By smart meters, by breaker operation at the secondary substation, or by a customer. The number of outage reports/complaints from customers is expected to decrease in the future, as the use case will enable DSOs to rapidly inform the customers about an outage and how it is being handled.

Additionally, the use case will enable automatic and precise fault localization and fault handling. The ultimate objective of the use case is to improve the security of supply in low voltage networks by reducing the time to restoration after an outage.

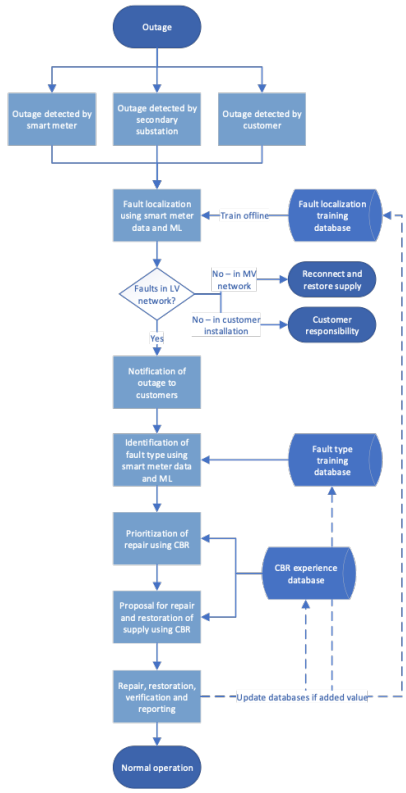


Figure C.55: Flow chart for fault handling in low voltage distribution networks in 2030/2040 utilising ML

C.3.1.1 Description of the Use Case

After receiving the alarm, fault localization occurs automatically by analyzing data from smart meters using machine learning algorithms. Although focus here is on machine learning, other methods may also be relevant for fault localization. With the large number of smart meters, both redundancy and missing or bad data is important to be properly handled [201]. Another concern is that training of the ML model may pose a challenge due to limited number of experienced outages. This problem may be mitigated by utilising simulated data as discussed in the above ML survey.

If only one smart meter reports an outage, the fault is probably in that installation, and it is then the customers' responsibility to rectify this. If the fault is located to the medium voltage (MV) distribution network, it is for simplicity assumed that automatic reconnection and

restoration is possible. The remainder of the use case hence applies to the LV distribution network.

After notification of customers, the fault type is identified using ML techniques. Here several data sources may be relevant, including grid data, weather data, satellite pictures, and smart meter data. Machine learning algorithms are well suited to take into account data from several data sources. Identification of fault type includes identifying the component (cable, overhead line, substation) and phases (one phase, multiple phases, grounding) involved, as well as the cause of the fault.

Following identification of fault location and fault type, the operator needs to identify appropriate actions. For multiple faults, this may include prioritization of repairs. A CBR system can use the predicted location and cause of a fault as a new problem description to search within an experience database for similar problem descriptions. If one is found, the previous solution can be retrieved, adapted and presented to the user of the system. If the retrieved solution is straightforward and recognized as identical to the current problem it can lead to automatic actions such as resource allocation, prediction of type of repair needed, and issuing of work orders. Prioritization of repairs may be especially relevant in the future, when high penetration of renewables (PVs etc.) may enable customers to be self-sufficient for a while during the outage such that they can be given a lower priority.

Finally, it is verified that normal supply has been restored and the resolved outage case is documented in the experience database if it provides added value compared to the data already stored in the database. The fault location and fault type are also stored in the ML training databases.

C.4 Discussion and Limitations

Being set in the years 2030 – 2040, the use case relies on a number of assumptions and prerequisites. Smart meters must be installed with data reporting capabilities of enough resolution, including last gasp-functionality. This requires that the meters have sufficient battery supply to send data also during an outage. Time stamping of data must be well-synchronized between all smart meters (e.g. using GPS) in order to be enough for improved fault localisation. There must be communication between systems that enables transmission of data at sufficiently high frequency, speed, bandwidth, and with sufficiently low latency to conduct outage analysis timely. (E.g., today's radio mesh technology for communication from smart meters is replaced by other technology, such as 5G).

It is in the use case assumed that all outage handling takes place centrally, i.e. in the DSO control centre. Alternatively, part of this functionality may be decentralized. The system responsible for collecting smart meter data, e.g. located in the secondary substation, could also include functionality for localizing the fault and identifying the fault type. The present study does not consider complete automation of the LV distribution network e.g. using remotely-operated breakers. Such breakers may allow to combine the use case with self-healing techniques and corresponding analytics.

The functionality included in the present use case may also be useful in similar use cases for improving the operation of the distribution network. An example is anomaly detection. Such a use case can benefit from the same machine learning algorithms and data, with the objective to identify unnormal conditions by comparing with normal behaviour. An advantage is that in such a scheme, the ML algorithm can be trained with data from normal operation, of which there is an abundance. The challenge then becomes to evaluate the severity of the anomalies, i.e. what constitutes an anomaly severe enough that some action needs to be taken. CBR techniques can be a nice fit for this decision-making challenge.

C.5 Conclusions and future work

ML appears to be a viable tool for improving the most common processes in fault handling, such as decision support and fault type classification. Deployment of ML will require that certain technological prerequisites are met, such as availability of smart metering with required functionality, availability of sufficient ICT infrastructure, and computational power.

Further work may include integration of other technologies such as self-healing and new types of customers as for example prosumers into the use case. The ML- and CBR-based support can provide probabilities of potential faults and causes of these, and hence thus give input to long term planning of required manpower and materials. The presented use case does not include exceptions from the steps in Figure C.55, such as missing data from smart meters, but these exceptions are important to identify in further work.

The use case has not been tested. An option for testing is simulated tests of the last gasp functionality for fault localization. For testing of ML functionality, a first step is to prepare an appropriate data set for simulation of outages or increase the number of outages by sharing information among different DSOs. The sharing of information can also increase the number of cases in the databases

of Figure C.55.

Deployment of new techniques will probably require substantial efforts and investments and is unlikely to happen without sufficient incentives for the network operators, such as e.g. CENS.

C.6 Acknowledgements

The Centre for Intelligent Energy Distribution (CINELDI) receives funding from the Research Council of Norway. The authors appreciate contributions from CINELDI partners and in particular Hafslund Nett AS.

References

[2] Aamodt, A. and Plaza, E. "Case-based reasoning: Foundational issues, methodological variations, and system approaches". In: *AI communications* vol. 7, no. 1 (1994), pp. 39–59.

[186] SAS. *Utility analytics in 2017: Aligning data and analytics with business strategy*. Tech. rep. 2017.

[187] Wang, Y., Chen, Q., Hong, T., and Kang, C. "Review of smart meter data analytics: Applications, methodologies, and challenges". In: *IEEE Transactions on Smart Grid* vol. 10, no. 3 (2018), pp. 3125–3148.

[188] Sharma, N., Sharma, P., Irwin, D., and Shenoy, P. "Predicting solar generation from weather forecasts using machine learning". In: *2011 IEEE international conference on smart grid communications (SmartGridComm)*. IEEE. 2011, pp. 528–533.

[189] Rudin, C., Waltz, D., Anderson, R. N., Boulanger, A., Salleb-Aouissi, A., Chow, M., Dutta, H., Gross, P. N., Huang, B., Ierome, S., et al. "Machine learning for the New York City power grid". In: *IEEE transactions on pattern analysis and machine intelligence* vol. 34, no. 2 (2011), pp. 328–345.

[190] Morch, A. Z., Istad, M., Ingebrigtsen, K., Garnås, S., Foros, J., and Mathisen, B. M. *Use cases for future (2030-2040) smart distribution grid operation*. Tech. rep. 2019.

[191] Lee, H.-J., Park, D.-Y., Ahn, B.-S., Park, Y.-M., Park, J.-K., and Venkata, S. "A fuzzy expert system for the integrated fault diagnosis". In: *IEEE Transactions on Power Delivery* vol. 15, no. 2 (2000), pp. 833–838.

[192] Yang, C., Okamoto, H., Yokoyama, A., and Sekine, Y. "Expert system for fault section estimation of power systems using time-sequence information," in: *International Journal of Electrical Power & Energy Systems* vol. 14 (1992), pp. 2–3.

[193] Thukaram, D., Khincha, H., and Vijaynarasimha, H. "Artificial neural network and support vector machine approach for locating faults in radial distribution systems," in: *IEEE Transactions on Power Delivery* vol. 20, no. 2 (2005), pp. 710–721.

[194] Q.Wei, D. and Shi, G. "A novel dual iterative Q- learning method for optimal battery management in smart residential environments," in: *IEEE Transactions on Industrial Electronics* vol. 62, no. 4 (2015), pp. 2509–2518.

[195] J. Morales, E. O. and Rehtanz, C. "Classification of lightning stroke on transmission line using multi- resolution analysis and machine learning," in: *International Journal of Electrical Power & Energy Systems* vol. 58 (2014), pp. 19–31.

[196] Rafinia, A. and Moshtagh, J. "A new approach to fault location in three-phase underground distribution system using combination of wavelet analysis with ANN and FLS," in: *International Journal of Electrical Power & Energy Systems* vol. 55 (2014), pp. 261–274.

[197] Samantaray, S., Dash, P., and Panda, G. "Distance relaying for transmission line using support vector machine and radial basis function neural network," in: *International Journal of Electrical Power & Energy Systems* vol. 29, no. 7 (2007), pp. 551–556.

[198] Ekici, S. "Support Vector Machines for classification and locating faults on transmission lines," in: *Applied Soft Computing* vol. 12, no. 6 (2012), pp. 1650–1658.

[199] Ferreira, V. "A survey on intelligent system application to fault diagnosis in electric power system transmission lines," in: *Electric Power Systems Research* vol. 136 (2016), pp. 135–153.

[200] Ingrid, M. *Smartgrid conference*. Danske Bank. May 2, 2019. (Visited on 05/04/2019).

[201] Jiang, Y. "Outage management of distribution systems incorporating information from smart meter," in: *IEEE Transactions on power systems* vol. 31, no. 5 (2016), pp. 4144–4154.

ISBN 978-82-326-5184-9 (printed ver.)
ISBN 978-82-326-5625-7 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

NTNU

Norwegian University of
Science and Technology