ROBUST ADAPTATION, MACHINE LEARNING DRIVEN ADAPTATION, AND THEIR IMPLICATIONS IN CASE-BASED REASONING

Xiaomeng Ye

Submitted to the faculty of the University Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in Luddy School of Informatics, Computing, and Engineering,

Indiana University

July 2022

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

David B. Leake, PhD

David Crandall, PhD

Chung-chieh Shan, PhD

Larry Moss, PhD

June 13th, 2022

Acknowledgments

I grew up in a highly structured society and culture where everything is by design. If it does not have a purpose, it is not worth pursuing. I was rebellious in heart and often questioned the education system I was in. When I grind my way to the top of the class, the school, even the city over 4 years in high school, I finally hardened my belief that it was not just the rebellious phase in adolescence (maybe a little bit), but rather a revelation of what I am not. However, I did not know what I am. In fact, it was a question that baffled me many years.

I escaped Gao Kao (the Chinese college entrance exam) by going to the US for college. I met Dr. Denise Byrnes in the College of Wooster. She showed me the fascinating depth of computer science in a computational theory class. With her, I worked as a teaching assistant and did my first legitimate research project. She also encouraged me to apply for graduate school. She was an anchor for my earlier days in the US, directing my energy to something bigger although I was still immature to fully understand that.

Still lost at the beginning of graduate school, I found every student knows exactly which professor they want to work for, except me. I developed anxiety and internet/game addiction. Things began to look up when I started to face my inner self, unsatiated and ignored by myself for many years.

My partner Kuo Deng and my daughter Nanzhi Lyra Ye are what ultimately drove my transformation. They give me purpose, hope, and a home. Kuo (who is a counseling psychologist herself) is the most insightful person, understands my deepest trouble and guides me to seek counseling. I want to thank my counselors, Farrell Hene and Elizabeth A. York. I want to thank Employee Assistance Program by Indiana University for its financial help. I also want to thank Middle Way House in Bloomington for the volunteer experience, through which I established some contact with the real world that is also beyond my comfort zone. Together with Kuo, we raised Lyra in our best

attempt to safeguard her nature. Lyra is a role model for me, demonstrating what a healthy mind is. Reliving childhood with her cures my own childhood experience. Kuo and Lyra, they shine.

I was fortunate to meet my advisor and mentor, Dr. David Leake. In my first couple of years working with Dr. Leake, I poked here and there, was unsure of research directions, and lacked insights. Even though those early ideas never came to fruit, Dr. Leake patiently bounced ideas with me weekly and I enjoyed his guidance and accompany. The trust from him eased my anxiety for productivity and allowed me to explore before settling onto topics I truly love. His huge knowledge pool of research, open-mindedness and receptiveness of new ideas, student-oriented advising style, laid-back attitude and incredible writing skill have been consistently supporting me through my doctoral years and multiple projects. Since we moved to Utah, I have been watching Lyra half of the day on weekdays while working during the rest of the day. Even so I am not stressed out, achieved a work-life balance and produced satisfying works. This would not be possible without Dr. Leake's understanding, respect and support.

Through my doctoral degree, I also received individual support from each of my committee members and fellows. I met with Dr. Larry Moss several times to discuss cognitive science research. I took Dr. Chung-chieh Shan's class and learned about probabilistic programming. I took Dr. David Crandall's computer vision class, worked as an associate instructor and am now working as a research assistant for Dr. Crandall. In my current research group led by Dr. Leake and Dr. Crandall, I gained lots of opportunities to collaborate and learn from others. Most noticeably, Ziwei Zhao has been a most reliable team member to work with.

My childhood and teenage years taught me who I am not, and my recent years working through a PhD has taught me who I am. I find myself through the connection with my family, my teachers, my fellows, and also people outside academia. I experience the sense of achievement by doing research I like and getting recognition from fellow researchers. I also began exercising, eating good food, being in the nature and treating my own body and mind. No man is an island and I owe my journey to myself and everyone I met along the way.

Preface

Certain chapters in this thesis are adapted from published works of the thesis author (Leake and Ye, 2019a, 2020; Leake et al., 2021a; Ye et al., 2021a; Leake and Ye, 2021b). Some chapters are under review for proceedings.

Xiaomeng Ye

ROBUST ADAPTATION, MACHINE LEARNING DRIVEN ADAPTATION, AND THEIR IMPLICATIONS IN CASE-BASED REASONING

Case-based reasoning is composed of four major aspects ("4 REs"): Retrieval, reuse, revision and retention. The reuse, or the case adaptation process, is critical to the flexibility of CBR systems and arguably the most difficult stage among the four. Traditional reuse makes use of single case adaptation rules, but suffers when the query is novel for the case base and extensive adaptation is needed.

This thesis focuses on the case adaptation process by first introducing Robust Adaptation (ROAD). Instead of applying a single adaptation rule, ROAD allows combining multiple adaptation rules in a sequence and applying them in order, forming a so-called adaptation path. Chaining multiple adaptation rules renders new challenges, and ROAD attempts to solve them by heuristic-guided path finding strategy. When multiple adaptation rules are combined, the reliability of the final adapted solution is not guaranteed. ROAD also attempts to uphold the reliability of the solution by resetting the adaptation path to a real stored case. This thesis demonstrates that ROAD has significant implications in not just reuse, but also the other aspects of the "4 REs". In the aspect of retrieval, resets in adaptation paths indicate room for improvement for more efficient case retrieval. In the aspect of reuse, quality of solutions reflects the reliability of adaptation rules involved. All the above information gained through using ROAD can be utilized for maintenance of the CBR system.

Additionally, this thesis explores combining machine learning techniques with CBR, and more specifically, the case adaptation process. Adaptation knowledge is hard to learn in general. The state-of-art approach, the case difference heuristic (CDH) approach, gathers case pairs to learn adaptation knowledge, which attributes the solution difference in a case pair to its problem difference. Traditionally CDH accumulates the adaptation knowledge in the format of rules.

When a query is sufficiently different from the retrieved case and adaptation is needed, CDH can find the rule with the most similar problem difference and apply the rule's solution difference to the retrieved solution, thus forming the final solution. Building upon the existing work of CDH, this thesis uses a neural network to learn the adaptation knowledge for regression tasks. Instead of accumulating rules from pairs, a neural network is trained to predict a solution difference from a problem difference. This thesis continues expanding the above idea into classification domains, resulting in a neural network learning adaptation knowledge for classification. Furthermore, these two models are unified under one overarching model that can learn adaptation knowledge for both classification and regression domains. We name this neural network based case difference heuristics (NN-CDH).

Existing work implemented the case retrieval process using machine learning techniques. NN-CDH takes one extra step of doing case adaptation using ML. In the experiments of combining ML-driven retrieval and adaptation, adaptation sometimes unexpectedly reduce the accuracy of the retrieved result. This is caused by the discrepancy between the retrieval and the adaptation process. The retrieval process might retrieve a similar case that is not easily adaptable. On the other hand, the adaptation process might be trained to adapt the difference between random pairs but not that between close pairs. This discrepancy appeared and was studied before in non-ML settings. Because they are now both implemented in ML techniques (e.g. neural networks), their training process hinges on their corresponding loss functions. This thesis tries to bridge the retrieval and the adaptation processes by two steps: 1) It integrates the loss of adaptation into the loss of retrieval so that the retrieval process is trained to retrieve adaptable result. This is inspired by the adaptation-guided retrieval approach. 2) It also optimizes the retrieval and the adaptation processes using alternating optimization. Instead of optimize one process fully between the other, this thesis optimizes both processes hand in hand. Lastly, we explore the possibility of using the case adaptation in the field of computation creativity. More specifically, the adaptation process can be used to not just solve a query, but also to adapt old cases to generate new samples.

Contents

Chapter 1

Background: Case-based Reasoning and Case Adaptation

1.1 Case-based Reasoning

Case-based Reasoning (CBR) is the AI methodology that solves new problems by adapting previous solutions to fit new circumstances (Riesbeck and Schank, 1989; Kolodner, 1993; Aamodt and Plaza, 1994; Leake, 1996). CBR is inspired by the cognitive science of problem-solving through reminding and remembering (Schank, 1982). In the spectrum of AI methods, CBR is memory-based, in contrast to rule-based (e.g. expert systems) or statistics-based methods (e.g. deep learning).

In the core of CBR, a case is composite of a problem description and a solution, representing a real-world episode. The problem description can be nominal/numerical features, text, constraints, or images. The solution can take many forms, such as a class label, a workflow, a design, a plan, or an explanation (Kolodner, 1992). The goal of a CBR system is to solve a new query, a case with problem description but without solution, by finding its missing solution. This goal is fulfilled by a cycle of four stages, or "4 REs" (Aamodt and Plaza, 1994; López de Mántaras et al., 2005):

- Retrieval: The system retrieves stored case(s) similar to the query case. Previous cases are stored in a collection called the *case base*.
- Reuse: The system reuses the solution to the retrieved case(s) and adapts it to solve the query case.
- Revise: The system evaluates the proposed solution (by evaluating the performance after applying the solution to the query, or by a domain expert). The system then revises the solution based on the evaluation.
- Retain: The system retains any new knowledge gained from solving the query. Upon a

successful evaluation, the query along with the proposed solution can be stored as a new case in the case base.

The retrieval stage is the first of the "4 REs". Based on the assumption that similar problems share similar solutions, a CBR system assumes the case most similar to the query contains the best or the closest solution to the query problem. By the same assumption, the similarity between two cases is often just the similarity between the problem descriptions of the cases. A function called the *similarity measure* (SM) assesses the similarity between two cases. A SM is often defined based on or as the complement of a distance measure, because the more distant two cases are away from each other, the less similar they are. For example, a common SM used in CBR is the Euclidean distance, where the distance between two cases, represented as two vectors $p = (p_1, p_2, ..., p_n)$ and $q = (q_1, q_2, ..., q_n)$, is

$$d(p, q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{1.1}$$

The reuse stage, or the case adaptation process, is critical in the flexibility of CBR systems, enabling stored cases to cover a range of new problems. When the retrieved case's solution does not apply to the query, adaptation to the solution is needed. There are in general three kinds of adaptations (Kolodner, 1993; López de Mántaras et al., 2005; Smyth and Cunningham, 1993): (1) Substitution adaptation modifies individual components of the retrieved solution; (2) Transformation adaptation modifies the overall structure of the solution; (3) Generative adaptation generates solution from ground up by replaying the process generating the retrieved solution, but with components such as placeholders replaced with information from the query. For example, CHEF, a CBR system for the cooking domain (Hammond, 1989), may utilize substitution adaptation to change an ingredient from "chickent" to "beef", and may also utilize transformation adaptation to reorder the step "stir fry beef" before "adding veggies" because "stir fry beef" can make the vegetable soggy (while "stir fry chicken" will not) (Hammond, 1989). In comparison with the two, generative adaptation is often more complicated and requires derivation strategies used to generate solutions, as seen in Prodigy/Analogy (Veloso, 1994).

The revise stage requires an extra component or a domain expert to evaluate the system's proposed solution or performance of other aspects of the system. In the event that the system fails to produce a satisfactory solution, the solution is revised. The reuse and revise stages are often grouped as one component as both processes improve the quality of the retrieved solution. A more creative revise stage may involve modifying the components of the system itself. For example, the ROBBIE system (Fox and Leake, 2001) revises the case indexing after evaluating the proposed solution, leading to faster and more adaptable solution in the future.

The retain stage, or retention, can be as simple as storing the solved query case. Case base maintenance (Leake and Wilson, 1998; Leake et al., 2001) is the more general process of the retain stage. Instead of simply storing newly solved case, case base maintenance may involve: (1) modifying existing cases and features, (2) deleting or adding cases, (3) modifying the representation or accounting of cases. The goal of case base maintenance can be more efficient case retrieval or better competence of the case base (Leake and Wilson, 1998; Smiti and Elouedi, 2011).

1.2 Rule-based Case Adaptation

In this thesis, we start by focusing on rule-guided case adaptation, or rule-based adaptation, where adaptation rules are designed to adjust solutions for each possible difference between old and new problems, enabling one-step adaptations of each difference.

Rule-based case adaptation is powered by adaptation rules. An adaptation rule describes how to modify a case. Each adaptation rule has an antecedent component describing the precondition needed to apply the rule, and an action component describing the effect of applying the rule. For example, an adaptation rule can be "when beef is available but chicken is not, replace chicken with beef". Rule-based case adaptation easily falls under the category of substitution adaptation. It may also involve transformation adaptation if a more sophisticated rule modifies the structure of the case.

Relying on one-step adaptation increases the burden of acquiring adaptation rules, which is well known as a classic problem for CBR. This problem is especially important when the case base is sparse or when the query is novel and far from stored cases (Craw et al., 2006b). Because of this, case adaptation is often considered the most difficult in the 4 REs of CBR. Many CBR systems do not even include a case adaptation process or do so minimally.

Because of the limitation of the case adaptation, the overall performance of a CBR system heavily relies on the result of case retrieval. If the system retrieves a case with perfect or close to perfect solution, then the query can be successfully solved. For example, if the task domain is a classification task and the case base contains sufficient amount of cases of every class, then the system can often retrieve a similar case with the correct class label for the query. However, if the task domain is a more complicated task, such as regression or design, or if the case base does not contain enough cases to cover the potential case space of the query, the CBR system may be incapable of retrieving a good solution and extensive adaptation is needed (Craw et al., 2006b).

Many existing efforts aim to reduce the burden of case adaptation. Leake et al. (2011) uses introspective reasoning and web mining to learn adaptation knowledge just in time of need. Leake et al. (1995) remembers previous adaptation episodes as adaptation cases, and reuses the most suitable adaptation case when needed. Craw et al. (2006b) learns adaptation knowledge from the case base, and its method is an extension of the case difference heuristic (CDH) (Hanney and Keane, 1996).

The case difference heuristic is the most relevant work for the purpose of this thesis, with specifics as described in Leake and Schack (2018). In this implementation of CDH, a pair of cases are selected to generate an adaptation rule. Their difference in problem descriptions (problem difference) is used as the antecedent of the rule and their difference in solution (solution difference) is used as the action of the rule. This process is repeated multiple times to generate a certain number of rules, forming an adaptation rule set. Given a query and a case retrieved from the case base, the system first calculates their problem difference. Then the system searches for the rule which has the most similar problem difference and is applicable to the retrieved case. Next the system applies the action of the rule by modifying the retrieved case's solution based on the solution difference of the rule. Lastly, the modified solution is used as the final proposed solution after the retrieve and reuse processes.

This thesis reuses the same implementation to generate adaptation rules, but the usage of rules is different, as detailed later.

The rule set generation process is influenced by following configuration parameters:

1. **Rule Count**: the number of rules to generate.

2. **Rule Specificity**: The level of rule specificity. The system may store only a portion of the features in the problem difference as the antecedent of a rule. For example, $rspec = 0.1$ if 10% of all feature differences between two cases are included in the rule.

3. **Rule Generating Distance**: the distance between pairs of cases generating rules. For example, if $ruleGenDist = 0.1$, rules are generated from cases whose difference is less than 10% of the maximum possible difference. A small $ruleGenDist$ value means that the rules are generated by pairs of cases close to each other.

1.3 Contribution and Layout of this Thesis

This work contributes to multiple research questions in the field of case adaptation and also the bigger field of CBR. The research efforts span a range of topics, in some cases with later chapters building directly on the previous ones, and in others where an earlier topic inspires a later one. More specifically:

1. Starting from traditional rule-based adaptation, Chapter 2 first proposes RObust ADaptation

(ROAD) to apply multiple rules in a chain to potentially cover for bigger problem differences and make bigger adaptation. Robust adaptation introduces new challenges, which are met with heursitic algorithms.

2. Chapters 3 and 4 examine the ramifications of ROAD. Chapter 3 shows that the case retrieval process can be tuned to better align with ROAD, so that the system can retrieve a case that requires easier adaptation. Chapter 4 shows that rules may not be compatible with each other and rules may not be reliable once combined with other rules. It proposes to curate the rule sets by retaining only the reliable/compatible rules.

3. Previous chapters show ROAD as an improvement to traditional rule-based adaptation and also new challenges for ROAD. Adaptation rules are symbolic and often rigid. Too few rules do not generalize well, and yet too many rules requires a collection effort to learn and store, and an extensive search for the right rule to apply. Chapter 5 explores the option of using neural networks to generalize rules and alleviate the burden of learning adaptation knowledge.

4. Both case retrieval and case adaptation can be implemented using machine learning techniques. Case retrieval can use a siamese neural network as a similarity measure for cases. Case adaptation, as described in Chapter 5, can be implemented using a neural network.

   Chapter 6 discovers that when both case retrieval and adaptation are driven by machine learning and trained for their own purposes, the two processes may not align: The retrieved result may not be adaptable or the adaptation process may worsen the result. Chapter 6 proposes to harmonize the two processes using alternating optimization.

5. All the previous chapters focus on strengthening case adaptation with new algorithms and examining their implications. Chapter 7 explores the possibility of using case adaptation for purposes beyond solving a target query. This chapter uses case adaption in a creative process of generating new unseen samples by adapting from seen ones. Oriented for the field of

computational creativity, this chapter proposes a creative sample generation method inspired by case adaptation. It also proposes an evaluation scheme that measures some aspects of creativity of sample generators.

6. Lastly, Chapter 8 summarizes the most important conclusions and future directions of this thesis.

Chapter 2

Robust Adaptation

2.1 Introduction

The case adaptation process is commonly driven by a list of differences between the new problem and the problem of the retrieved case, with adaptation rules retrieved according to the differences to adapt. Models of the adaptation process often assume that a suitable adaptation rule will be available for each difference the system may have to adapt, making adaptation for each difference a one-step process. However, in practice it may be infeasible to provide a CBR system with adaptation rules for every possible difference. Generating adaptation rules by hand is costly, and it may be difficult or impractical to anticipate or capture a sufficiently extensive rule set. Another issue is the potential need to address multiple differences between old and new problems. Rules may be tailored for specific differences, but the number of potential difference combinations aggravates the rule generation problem. In practice, CBR systems often address the problem of multiple differences by treating the effects of differences along particular problem features as independent and decomposing the difference set by generating a list of differences to adapt in turn, applying one rule per difference.

For numerical problems, the decomposition approach has been formalized by Fuchs et al. (2014a) as *differential adaptation*, which models case adaptation as the application of a sequence of adaptation operators. In spirit of Fuchs et al. (2014a), this dissertation research develops *Robust Adaptation* (ROAD), a search-based framework of the adaptation rule application process. ROAD allows the combination of multiple adaptation rules into an adaptation path. An adaptation path can cover the case differences that are unreachable by one-step adaptations. Additionally, ROAD pursues multiple alternative adaptation paths, applies a flexible chaining process, and integrates ongoing retrievals to help guard against the degradation of adaptation performance that can occur for long adaptation paths.

The main contribution of this chapter is:

$\boxed{\text{When the query is novel, ROAD is a better way to do case adaptation than existing methods.}}$

ROAD also opens up new avenues of improvement for other aspects of CBR: For example, reset histories of built paths can be used to improve adaptation guided retrieval; Accuracy of built paths can be used to assess the compatibility and reliability of rules involved; High traffic section of built paths can indicate valuable cases to retain.

2.2 Models of Applying Multiple Case Adaptation Rules

We begin by presenting three models of adaptation rule combination: adaptation by multiple independent one-step adaptation rules, adaptation by multi-step adaptation paths, and ROAD—robust adaptation by dynamically adjusted and reset adaptation paths.

Adaptation by multiple independent one-step adaptation rules: It is common for CBR systems to adapt a collection of differences by selecting adaptation rules for each one, applying each one, and combining the results. Given a set of independent features f_i of a case, and corresponding differences d_i between the value of the feature in the input problem and a retrieved case to adapt, and a collection of adaptation rules r_i, with each r_i applicable to difference d_i, the adaptation combination process combines the independent component results. For example, for a regression task (*e.g.* real estate appraisal), combination might simply sum the price effect of each difference.

The one-step approach assumes that there will be an adaptation rule suitable for each difference. However, this may not be the case. For example, if no sufficiently similar case is available, extensive adaptation may be required to bridge the distance between cases, with no predefined rules that bridge the gap. When a limited set of rules is available, a multi-step adaptation may be the only way to address certain feature differences, even for a single feature.

$$\begin{tikzcd}
B \arrow[r, "g"] & C \\
A \arrow[u, "f"] \arrow[ur, "h"'] &
\end{tikzcd}$$

Figure 2.1: Adapting from Solution A to Solution C through Adaptation Path (f, g).

Adaptation by multi-step adaptation paths: Path-based adaptation discards the assumption that there will exist an adaptation rule that directly addresses a given difference. Instead, the system may have rules that incrementally provide the desired solution. In the cooking domain, to change a recipe for regular pancakes to buttermilk pancakes, a cook might first adapt to use buttermilk (by replacing the leavening with buttermilk and baking soda) and then, if buttermilk is not available, adapt by substituting regular milk and vinegar for the buttermilk. This generates an adaptation path, the composition of a sequence of adaptation rules resulting in intermediate solution states (Fig. 2.1).

An adaptation path is a sequence of triples (c_i, a_i, c_{i+i}), where c_i is a case to adapt, a_i is an adaptation rule, and c_{i+1} is the result of applying a_i to c_i. In our formulation, as in that of D'Aquin et al. (2006) and Leake and Schack (2018), the adaptation rule modifies both the solution and the problem description, to keep the problem and solution descriptions consistent for a new complete case. As this new case is generated from adaptation and does not correspond to a situation encountered in the world, we call it a *ghost case* (Leake and Schack, 2018). The adaptation path retains provenance information about how each case in the sequence is derived, which could be used, e.g. for estimating result quality (Leake and Whitehead, 2007).

In an adaptation path, differences are not assumed to be independent; adaptations are performed in a sequence. Issues with interaction problems are handled when the CBR cycle revision step repairs the candidate solution (López de Mántaras et al., 2005).

In the adaptation path model of D'Aquin, Lieber, and Napoli, a similarity path is built first and then a corresponding adaptation path to modify the source solution to the target solution.

Badra et al. (2009) present an algorithm building an adaptation path by which the query is modified to match at least one case from the case base. Inspired by differential calculus, the differential adaptation model of Fuchs et al. (2014a) uses partial derivatives to make small variations in the problem and solution. Adaptation knowledge is generated by computing derivatives of every solution feature with respect to every problem feature. Differential adaptation can be applied to any numerical task domain provided that the dependencies between descriptors can be computed. Under differential adaptation, small-step adaptation is preferable to big-step adaptation as the former introduces less error using derivatives. This assumption motivates their multi-step differential adaptation.

Robust adaptation by dynamically adjusted and reset adaptation paths: The ROAD approach to adaptation paths is based on three principles. First, rather than dividing difference identification and adaptation into separate steps, after every adaptation rule application it re-assesses differences and selects the next adaptation rule to apply. Incremental choices of each adaptation rule are made in the context of the previous adaptations. Second, ROAD can simultaneously pursue multiple potential adaptation paths, enabling exploiting alternative adaptations and comparing competing alternatives. Third, ROAD's adaptation process is coupled to the case base: As adaptation proceeds, if an incremental solution is similar to an existing case, ROAD can "reset" the adaptation path to proceed from the existing case. This substitution effectively restarts solution generation from a known solution in the case base, reducing the required adaptation. In the common situation of imperfect adaptation knowledge, this is expected to increase solution quality.

2.3 Issues for Adaptation Paths and ROAD

Compared to applying single adaptations, adaptation path composition increases adaptation flexibility and coverage. However, moving from single adaptations to adaptation paths complicates multiple parts of the case adaptation process:

1. Estimating adaptability: CBR commonly uses similarity as a proxy for adaptability. This assumption has been questioned (Smyth and Keane, 1998), leading to methods for adaptation-guided retrieval (Díaz-Agudo et al., 2003; Leake et al., 1996; Nouaouria and Boukadoum, 2010; Smyth and Keane, 1998). However, when adaptation could in principle involve long sequences of adaptation steps, estimating adaptation cost could require consideration of many possible alternatives, making adaptation-guided retrieval extremely expensive.

2. Guiding path construction: If many alternative adaptation sequences could be considered, how to select them becomes important for adaptation efficiency and solution quality (e.g., if some rules are known to be more reliable than others or shorter paths are more desirable for quality (Leake and Whitehead, 2007)). In general, the adaptation path should approach the query case. In other words, the retrieved case is adapted to become closer to the query. This is not always true, as sometimes a path bypassing certain section is necessary even if the path will be longer than a direct path.

3. Terminating unpromising paths: When a path is judged to be unpromising (e.g., because its length suggests risk of excessive quality loss or explainability), it may be appropriate to terminate; heuristics are needed to determine when to terminate.

4. Handling adaptation interactions: If adaptations may interact, the choice of adaptations must be sensitive to side-effects of previous adaptations. In principle, in fully understood domains, it would be possible to use standard AI methods to manage such interactions. However, CBR is often applied to domains that are poorly understood or imperfectly formalized. An adaptation approach practical for such domains must use other methods. ROAD combines adaptation rules using heuristic-based methods. This thesis also addresses the interaction between adaptation rules in Chapter 4.

The ROAD model of dynamically adjusting and resetting adaptation paths raises additional issues:

1. Determining a strategy for exploring alternative adaptation paths: When multiple paths are pursued, any search strategy could be used to manage the choice of partial paths to extend (e.g., best-first, breadth-first).

2. Reconciling path intersections: If two adaptation paths lead to generating the same (or a nearby) ghost case, various strategies could be used for merging those results in the final path, such as selecting the single "best" path leading to the point of overlap or retaining both paths.

3. Resolving path conflicts and path reinforcements: If two adaptation paths lead to overlapping ghost cases with the similar problems and different (similar) solutions, the conflict could be evidence for reducing (increasing) path reliability.

4. Determining when to reset the search: As described in section 2.4.5, when the ghost cases in a path are similar to stored cases it may be beneficial to reset the path by replacing the ghost case with a new retrieval. This requires determining criteria for resetting the search.

2.4 ROAD's Strategies for Building Adaptation Paths

ROAD builds adaptation paths by a search process. Given a target problem p, the process begins by retrieving one or more most similar cases as a starting point. The starting point may be reset by new retrievals as the process progresses. The goal is to generate an adaptation path from the initial case to one applicable to the new situation, while minimzing a cost function $f : P \to \mathbb{R}^+$. The function f could reflect criteria such as minimizing solution error, minimizing path length (as a proxy for minimizing error, or for increasing explainability of the solution derivation), minimizing adaptation cost (i.e., path construction cost), or domain-specific criteria such as, in case-based planning, minimizing the execution cost of the generated plan. For example, for numerical domains, Fuchs et al. (2014a) propose building the adaptation path by hill climbing using the derivative to reflect the relationship between the variation of problem features and the variation of solution values.

ROAD uses four categories of strategies, for: 1) avoiding prohibited regions, 2) extending a single adaptation path, 3) initializing and pursuing multiple paths, and 4) preserving the reliability of paths.

2.4.1 Avoidance of Prohibited Regions (Dead Zones)

ROAD supports the avoidance of generation of ghost cases in certain regions of the problem space. We call these regions *dead zones*. For example, in the housing price prediction domain, local housing codes may prohibit houses in a certain area. Rather than allowing case adaptation to hypothesize ghost cases in that area, ROAD's can be provided with a test function to reject ghost cases there. This has two motivations. First, for an adaptation rule to hypothesize a case in such a region shows that the rule is missing portions of the relevant context; thus it might be expected to be less reliable in that region, making it reasonable to favor a path with all steps within the realm of possibility. Second, if the CBR system will present the adaptation path to the user as an explanation for its result, presenting an impossible intermediate step might undermine confidence in the explanation. Therefore ROAD builds adaptation paths bypasses dead zones.

We note that this prohibition is not required and might sometimes be undesirable. For example, for a domain in which solutions are hard to generate but easy to evaluate, the path might not be important to trust, and enabling adaptation to hypothesize impossible ghost cases might lead to creative solutions.

2.4.2 Extending a Single Adaptation Path

An adaptation path is extended by adapting the case at the head of the path, using an adaptation rule, and appending the triple of the original case, the rule, and the ghost case generated by adaptation to the path. At the beginning of the process, the initially retrieved case is the head of a path. Because multiple adaptation rules may apply to a single case, heuristics are needed to select the adaptation rule to apply. Our implementation of ROAD searches by a modified best-first process, favoring adaptation rules that result in cases most similar to the target case.

Inspired by Fuchs et al. (2014a), ROAD extends paths by a best-first process, but it differs in three respects. First, Fuchs et al. only addressed regression tasks, requiring the generation of differential numerical adaptation operators, with the sequence of chosen operators always generating the correct adaptation. ROAD, addresses adaptation for non-regression as well as regression tasks, and does not assume that a best-first process will necessarily produce the best path. Second, ROAD supports the simultaneous pursuit of multiple paths, with back-tracking as needed and with the ability to block consideration of portions of the candidate adaptation space (dead zones). Third, as described below, rather than simply applying the sequence of chosen adaptations, ROAD monitors the ongoing results of partial adaptations to potentially re-start adaptation from a nearby case.

When extending a path, ROAD first retrieves all adaptation rules applicable to the case at the head of the path. The rules are then applied, with the results ranked by applying the cost function (e.g., closeness of the problem of the new ghost case to the target problem). The rule with the best result is chosen and applied to the head (ties are broken arbitrarily). If the result has already been visited by an adaptation path, or if the result is impossible in the task domain (falls in a dead zone), then the next best rule is chosen.

The alternative to extend an adaptation path by breadth-first search is possible but too costly, because high availability of adaptation rules at each step leads to exponential growth in the search space of the whole path.

2.4.3 Heuristics for Initializing and Pursuing Multiple Paths

We considered three methods for the initial retrieval of source cases to use to initialize paths: (1) using 1-nn to retrieve one case and start one path from it, (2) using k-nn to retrieve k cases and starting one path from each of the k cases, and (3) using a k-nn to retrieve k cases but starting paths from a subset of those k cases selected for diversity. We expect (2) to perform better than (1) as combining solutions from multiple paths averages out errors in individual paths. We expect (3) to increase efficiency by reducing the number of paths considered by (2), with little quality loss because similar cases result in less independent paths–they involve similar adaptation steps toward the target–and less potential benefit from an ensemble of solutions. However, a trade-off is that we would expect substantially non-similar cases to result in lower performance. Selecting the diverse cases from the k nearest neighbors balances similarity and diversity. ROAD uses method (3).

ROAD stores all paths in a priority queue. In every iteration, the path with the highest priority is removed from the queue, extended, assigned a new priority, and added back to the queue. Using a priority queue ensures that computational resources are shared among all paths, thus avoiding the thread starvation problem. In ROAD, the priority of a path is defined as the inverse of the path length. Alternatively, the priority can be defined by the distance between the head and the target case, or confidence of a path.

2.4.4 Path Termination and Resetting as a Heuristic for Maintaining/Increasing Path reliability

One source of confidence in the results of CBR is that conclusions are derived from similar prior cases known to have correct solutions. In imperfectly understood domains, adaptation rules are generally imperfect, so adaptation is not guaranteed to generate correct solutions. Repeated application of unreliable adaptation rules may compound errors; in some contexts, reliability may be estimated by assigning a confidence decay rate to each adaptation rule and combining the decay effects of multiple rules (Leake and Whitehead, 2007).

ROAD uses several simple methods to reduce the chance of compound error. First, its greedy search process is aimed at finding a short adaptation path to the target (though as with any greedy search, the shortest path is not guaranteed). ROAD can limit adaptation path length, terminating a path when the maximum path length is reached. It also uses a novel method, path resetting, described in the following section, to bring the speculative path from adaptations back to case knowledge grounded in experience.

2.4.5 Resetting a Ghost Case with a Nearby Retrieval

ROAD's resetting process is triggered when heuristics suggest a potential solution quality problem. Resetting moves the head from the current ghost case to a similar case (reset case) in the case base. We note that changing to the reset case can result in a shorter or better adaptation path, than the path generated from the initially retrieved case (source case), even when the source case is the most similar to the query. Fig. 2.2 presents an intuitive example. In the figure, C_0 and C_1 are cases in the case base, where C_0 is the initially retrieved case to solve target case T. The solid arrows represent adaptation steps, with G_1, G_2 and G_3 ghost cases produced by adaptation.

Because C_1 is farther away from T than C_0 (the dashed arc), C_0 is initially used as the source case for an adaptation path. By greedy search, C_0 is adapted to G_1. The path further expands to G_2. Notice that even though C_1 is farther from T than C_0, it is closer to G_2 than C_0. At this point, the head of the path G_2 is better estimated by C_1 than by C_0. The solution of T can be predicted by either continuing adapting the path $C_0 - G_1 - G_2$, which may have accumulated error due to imperfect adaptation rules, or "resetting" the path to C_1 (with C_1's correct solution) and extending from there. The dashed arrow represents the reset to continue the path from C_1. After the reset, the path continues toward T by adapting C_1 into G_3.

ROAD triggers resetting in two conditions:

- *Cumulative quality decay:* Cases stored in the case base are associated with a perfect reliability score, as they are previously encountered real-world episodes. A decay function decreases a path's reliability based on each adaptation rule applied, with reset triggered when reliability

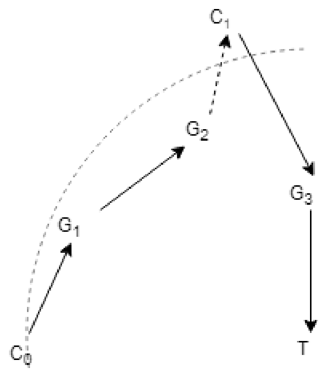


Figure 2.2: Illustration of Path Resetting

drops below a threshold. In the current implementation of ROAD, the reliability of a path is simulated by the inverse of the path's length since its most recent reset. The intuition is that the longer a path is, the less reliable it becomes, and a reset modifies the head of the path and restores its reliability to full.

- *Conflicting adaptation paths:* When the ghost cases at the heads of two adaptation paths are within a preset similarity threshold but their solution difference exceeds a difference threshold, the conflict puts their reliability in question. Because ROAD allows multiple adaptation paths to start from different (nearby) cases, ROAD only resets if the new solution difference exceeds their original solution difference. In other words, the disagreement between the two paths increases as they are extending toward the query. In this situation, at least one of the disagreeing paths is unreliable. ROAD chooses one of the disagreeing paths and reset it. The choice is a design question by the researcher, where ROAD can choose the longer path, or the path whose head is farther away from the target.

2.5 Evaluation

Our evaluation addressed the following questions:

1. How does the use of single adaptation paths vs. multiple adaptation paths affect solution

accuracy?

2. What are the trade-offs between one-step adaptation and multi-step adaptation, for varying degrees of case base sparsity around a target, and how is this affected by rule specificity?

3. How do rule specificity and locality affect the ROAD's performance?

4. How does resetting paths affect performance compared to not resetting?

5. How does the combined effect of multiple adaptation paths, multi-step adaptation, and resetting affect solution accuracy?

2.5.1 Experimental Design

Task and data: We tested the performance of ROAD for an automobile price prediction task, using the Kaggle automobile dataset (Kaggle, 2017). The original dataset contains 205 cases, each with 26 features. We removed the first two features because they concern insurance risk rather than attributes of a car. We also removed cases with missing features, leaving 193 cases. Every case has 10 nominal features and 13 numerical features, plus *price* as the solution.

Adaptation rule generation: Case adaptation rules were generated automatically for this task using a method based on the case difference heuristic (Hanney and Keane, 1996), with specifics as described in Leake and Schack (2018).

Their method generated adaptation rules that were considered applicable only when all nominal features in the source case used for learning were also present in the case to adapt. Such rules have high specificity (which corresponds to low generality). An alternative approach is to generate rules whose applicability is based on a smaller set of features, or even just one feature, making them more generally applicable (low specificity). Our rule generation procedure can be tuned to record only a subset of the features. A parameter *rule specificity* ($rspec$) governs the fraction of the original features retained in the adaptation rule (both in antecedents and in the features to adapt). The specific features for a given rule are chosen at random.

Similarity criteria: Similarity of nominal features is 1 if the features are identical and otherwise 0; all numerical features are assigned equal weight and are normalized by the range of feature values.

Controlling for case base sparseness: In general, the need for case adaptation depends on the density of the case base: When the case base is dense, with a high probability of nearby similar cases for any input problem, adaptation will be less crucial than when the case base is sparse and distant cases must be brought to bear. To study this effect, experiments were done both with the original case base, and with varying levels of case deletion in the neighborhood of the target case. Testing each target case, 0, 10, 25, 50, 75, 100, 125, and 150 cases around the target are removed from the case base to simulate situations where an initially retrieved source case is at a certain distance from the target.

Experimental parameters: Parameters regulate adaptation rule generation and adaptation path building for each experiment. Performance is measured by the relative error of *price*. Each experiment is carried out 100 times, with 10-fold cross validation for each trial. Cases are considered solved when either (1) no adaptation results in a ghost case closer to the target, or (2) the path length limit is reached. Settings for all experiments are listed in Table 2.1. Experimental conditions are determined by the following parameters (1–4 affect system behavior; 5–7 affect adaptation rule generation, as already explained in 1.2):

1. **k**: Number of cases retrieved as path starting points (also k in baseline k-nn).

2. **MaxLen**: Maximum allowed path length.

3. **ResetByDecay**: Enables/disables resetting when paths reliability decay. In the experiments, a path resets when a path grows longer than 3 steps since its initial retrieval or last reset.

4. **ResetByConflict**: Enables/disables resetting when paths disagree.

Table 2.1: Experimental Parameter Settings

#	k	maxLen	resetByDecay	resetByConflict	rcount	rspec	ruleGenDist
1	5	9	true	true	300	1.0	1.0
2	5	9	true	false	300	1.0	1.0
3	1	9	true	false	300	1.0	1.0
4	1	1	true	false	300	1.0	1.0
5	1	9	true	false	300	0.5	1.0
6	1	9	false	false	300	0.5	1.0
7	1	1	false	false	300	0.5	1.0
8	5	9	true	true	300	0.5	1.0
9	1	9	true	false	300	1.0	0.2
10	1	1	true	false	300	1.0	0.2
11	1	9	true	false	300	0.8	0.2
12	1	1	true	false	300	0.8	0.2
13	5	9	false	true	300	0.5	1.0
14	5	9	false	false	300	0.5	1.0

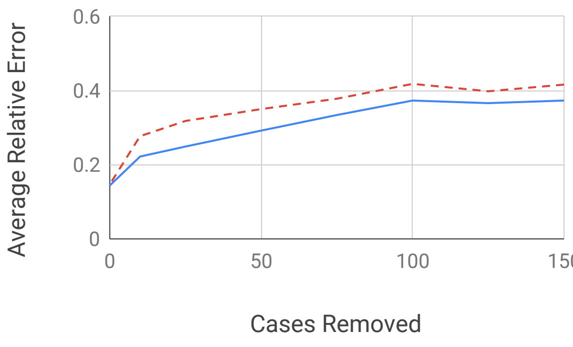
5. **Rcount**: Rule count.

6. **Rspec**: Rule specificity.

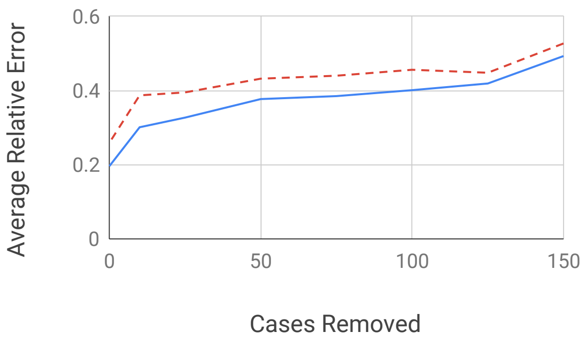
7. **RuleGenDist**: Rule generating distance.

2.5.2 Experimental Results

Question 1: Single adaptation path vs. combined results of multiple adaptation paths: To assess the value of combining the results of multiple adaptation paths from multiple starting points, we compared results when starting adaptations from a single case (experiments #3 and #5) and from five cases (#1 and #8), adapting each, and averaging the solutions, for two levels of rule specificity. Fig. 2.3 shows that using multiple adaptation paths lowered error at both levels of specificity.[1] This result is not surprising as k-nn naturally improves upon 1-nn by averaging multiple solutions.



(a) Standard Rule Specificity. #1 vs #3



(b) Low Rule Specificity. #8 vs #5

Figure 2.3: Error rate with increasing case base sparsity near target. Dashed red is one-path adaptation; solid blue is five-path adaptation.

Question 2: Tradeoffs of one-step adaptations and adaptation paths: To assess the tradeoffs between one-step adaptations and adaptation paths, we compared the performance of ROAD for one-step adaptation (experiment #4) and adaptation limited to nine adaptation steps (experiment #3), with adaptation rules reflecting the standard configuration of case difference heuristic rule generation (all differences reflected in the rules ($rspec = 1.0$), for a rule set including both short and long distance adaptations ($ruleGenDist = 1.0$)). As shown in Fig. 2.4(a), ROAD with the longer path length limit has lower error than ROAD with one-step adaptation. It might be expected that with a lower limit on distance covered, ROAD with longer path length limit will generate a longer path, which reduces path reliability and decreases benefit over one-step adaptation.

To illustrate the effect of rule distance coverage on the benefit of path-based adaptation,

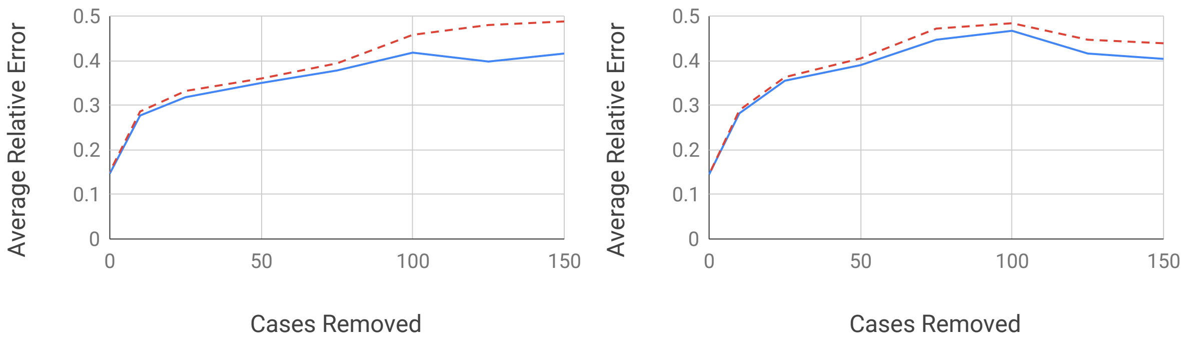
[1]In the experimental comparisons of this chapter, when cases have been removed, all differences are statistically significant ($p < .05$), except for: Fig. 4 (a) when more than 75 cases removed, Fig. 4 (b) when more than 25 have been removed, Fig. 5 (b), which has no significant difference, and Fig. 6 (b), when fewer than 150 cases removed.

Fig. 2.4(b) compares experiments #9 ($maxLen = 9$) and #10 ($maxLen = 1$) for a lower distance maximum ($ruleGenDist = 0.2$). Results are similar, with slightly higher error for rules with lower distance coverage when the region around the target case is less sparse. This suggests that the compounding of error from rules with less distance coverage may decrease the benefit of ROAD with long paths. For both levels of rule specificity, the error rate with adaptation paths increases more slowly than with one-step adaptation.

One interesting phenomenon observed in Fig. 2.4(b) is that for experiment #9, the error rate increases as the number of cases removed increases from 0 to 100, but decreases when the number of cases removed is 125 and 150. As more cases are removed, the initial retrieved cases become farther from the target case. Adaptation paths have to cover a larger distance (measured by similarity measure) to produce a ghost case close to the target case, and are therefore more prone to decay in reliability. As even more cases are removed, the initial retrieved case may reside on the boundary of the case space while the target case is in the middle. In such situations, an adaptation path often has only one direction to work toward the target case. The path is less likely to wander about, therefore leading to slightly less error. This phenomenon can be observed in experiments #9, #10, #11, #12. A common characteristic of these experiments is that *ruleGenDist* is set to 0.2, meaning the rules only cover small feature differences.

Question 3: Sensitivity of result quality to adaptation rule specificity and locality: To assess the dependence of ROAD on adaptation rule characteristics, we tested the effect of two factors: rule specificity and locality.

Rule specificity: To test the effect of rule specificity on the benefit of longer paths compared to one-step adaptations, we compared experiments #5 ($maxLen = 9$) and #7 ($maxLen = 1$), both of which use low specificity rules ($rspec = 0.5$) but allow non-local rules ($ruleGenDist = 1.0$). Non-local rules enable one-step adaptations to bridge long problem differences. Results are shown in Fig. 2.5(a). Here one-step adaptations outperform longer paths, which contrasts with Fig. 2.4,



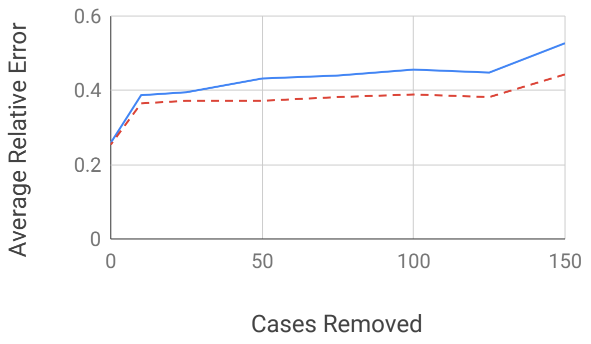
(a) Standard Rule Generating Distance. #3 vs #4 (b) Local Rule Generating Distance. #9 vs #10

Figure 2.4: Error rate with increasing case base sparsity near target. Dashed red is one-step adaptation; solid blue has max of nine adaptation steps.

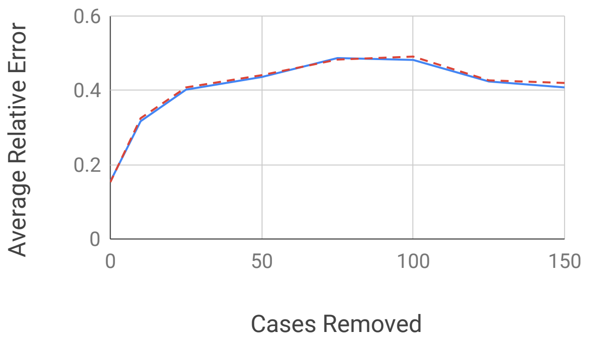
which shows ROAD with long paths consistently outperforms one-step adaptations for more specific rules. Our explanation is that decreasing rule specificity decreases rule accuracy. It also broadens rule applicability, and therefore increases average path length as more applicable rules are available to apply. For example, the average path length is 2.5 when $rspec = 1.0$, and 7.3 when $rspec = 0.5$. Longer paths further compound errors of individual rules. This nullifies the benefit of longer paths seen in the previous experiment.

Rule locality: In our tests, the rule generating distance parameter determines the maximum distance between the pair of cases from which an adaptation is generated, determining rule locality. To illustrate the effect of rule locality, we compared experiment #11 ($maxLen = 9$) and #12 ($maxLen = 1$) in Fig. 2.5(b), for $rspec = 0.8$ and $ruleGenDist = 0.2$. $ruleGenDist$ is set to 0.2 to favor generation of local adaptation rules, while $rspec$ of 0.8 increases rule reliability. A local rule generating distance limits one-step adaptation, because single rules cover shorter distances than paths with multiple rules. Here the performance of ROAD closely matches that of one-step adaptation for all sparsity levels.

Comparing to Fig. 2.5(a), experiments in Fig. 2.5(b) has higher $rspec$ and lower $ruleGenDist$,



(a) Standard Rule Generating Distance, Low Rule Specificity. #5 vs #7



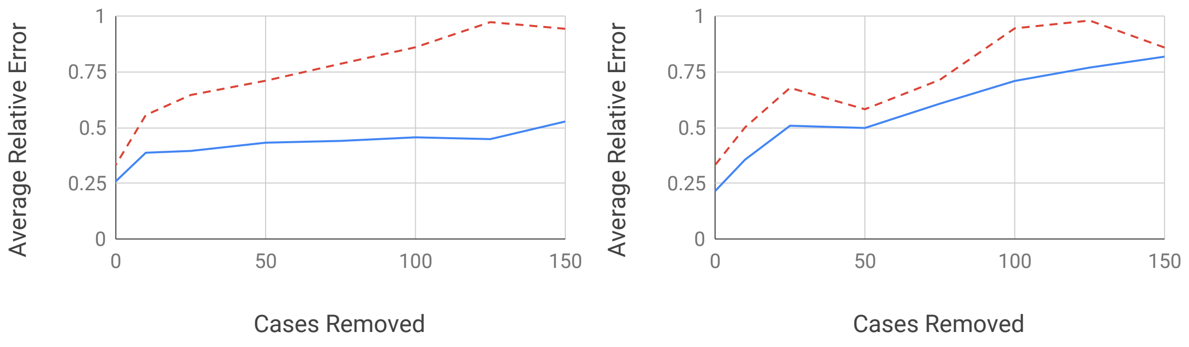
(b) Local Rule Generating Distance, High Rule Specificity. #11 vs #12

Figure 2.5: Error rate with increasing case base sparsity near target. Dashed red is one-step adaptation; solid blue has max of nine adaptation steps.

favoring ROAD over one-step adaptation. If the two parameters are tuned further to benefit ROAD, we will get Fig. 2.4, where ROAD starts outperforming one-step adaptation.

Question 4: Effect of path resetting: We proposed two triggers for adaptation path resetting: (1) When a cumulative measure of path reliability drops below a threshold, or (2) when the solutions of two paths disagree. We conducted experiments to assess the benefit of resetting under each strategy. Fig. 2.6(a) shows the error as a function of increased sparsity near the adaptation target, with and without resetting at the decay threshold (experiments #5 and #6). Resetting results in a substantial drop in error. Fig. 2.6(b) shows the error as a function of increased sparsity near the adaptation target, with and without resetting for conflicts (experiments #13 and #14). Here resetting benefits as well, but less uniformly.

Question 5: Combined effect: A final question is the combined effect when the individual aspects of ROAD are combined. Figure 2.7 compares accuracy for basic adaptation—a single adaptation path, at most one adaptation applied, no resetting—with ROAD using five paths, a maximum of nine adaptation steps, and both resetting mechanisms. It compares these for two types of adaptation rules generated from random case pairs, complete rules (rSpec=1 and rule-



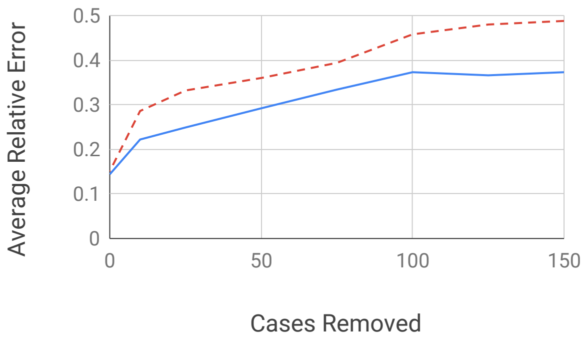
(a) No Resetting vs. Resetting by Decay. #5 vs #6 (b) No Resetting vs. Resetting by Conflict. #13 vs #14

Figure 2.6: Error rate with increasing case base sparsity near target. Dashed red is without resetting; solid blue is with resetting.

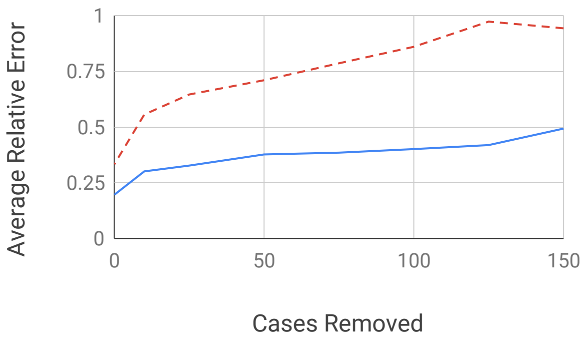
GenDist=1), in Figure 2.7(a), and partial rules (rSpec=0.5 and ruleGenDist=1) in Figure 2.7(b). (These respectively compare the results of experiments #1 and #4, and #6 and #8; note differing scales because rule characteristics heavily influence accuracy.) The experiments show substantial benefits for the full ROAD configuration, especially with partial rules.

2.6 Conclusion

This chapter presented ROAD, a model of robust adaptation. ROAD uses heuristics to guide generation of multi-step adaptation paths and reset the adaptation path to nearby stored cases when path reliability decays or when multiple adaptation paths suggest conflicting results. The resetting process is a novel way to help ground uncertain adaptation in the real experience of the case base. Evaluations support the benefit of the approach, and especially the benefit of resetting. The evaluation demonstrated, as expected, that the benefit of ROAD depended on case base sparsity and adaptation rule quality: 1) If cases are available near a target, there is less need for extensive adaptation; 2) If rules are unreliable, ROAD may compound errors (although resetting alleviates this problem).



(a) Complete Rules. #1 vs #4



(b) Partial Rules. #8 vs #6

Figure 2.7: Error rate with increasing case base sparsity near target. Dashed red is single case retrieval with at most one adaptation; solid blue is ROAD with combined methods.

Algorithm 1 ROAD Overall Procedure

```
Input:
*CB:* case base
*target:* the input query
*settings parameters:* from section 2.5.1
*rules:* adaptation rules

Output: a prediction for the solution of the target case
$sourceCases \leftarrow intialRetrieval(CB, target, k)$
$paths = new\ priorityQueue()$
$paths.addAll(createPathsFrom(sourceCases))$

$finishedPaths = []$
while not $paths.isEmpty()$ do
    $p \leftarrow paths.pop()$
    $resetIfDecay(resetByDecay, CB, p)$
    $resetIfConflict(resetByConflict, CB, target, p, paths, finishedPaths)$
    $step(path, rules)$
    if $done(path, maxLen)$ then
        $finishedPaths.add(path)$
    else
        $paths.add(path)$

$solutions = []$
for all $path$ in $finishedPaths$ do
    $solutions.add(path.head.solution)$
$prediction \leftarrow average(solutions)$
return $prediction$
```

Chapter 3

The Implication of ROAD in Case Retrieval: Using Adaptation Paths to Improve Similarity Measure

Although ROAD starts out as an improvement for the reuse stage of CBR, it also holds great promise in other 3 "RE" stages by introspective reasoning. In this chapter, we focus on the implication of ROAD in the retrieval stage. More specifically, after adaptation paths are built after using ROAD, we can potentially improve the similarity measure based upon the paths generated.

When a path is reset to a new source case, the path finding process continues from the new source case, and the path built until this point is forgone. If the case retrieval mechanism is updated so that the reset case is initially retrieved, then the system may circumvent the work in building the path from the original source case until the reset case. In this chapter, we examine the effect of updating a similarity metric based on the hindsight of resets in paths.

The efficiency of similarity-based search with resetting depends on the similarity measure being a good proxy for adaptability. However, the correspondence between similarity and adaptability is not guaranteed (Smyth and Keane, 1998). When similarity distances diverge from true adaptation distances, a similarity-based adaptation path may proceed through ghost cases that are not easily adaptable, resulting in longer paths. This chapter presents a maintenance method that use information from the ROAD adaptation process to refine the system's similarity measure, bringing it closer to reflecting true adaptability. We call this *Reset-induced Similarity Adjustment* (RISA). RISA uses knowledge of the final path to determine which prior cases should have been retrieved to minimize adaptation cost, and adjusts similarity criteria accordingly to improve future retrievals. This can be seen as ongoing CBR system maintenance (Wilson and Leake, 2001) of similarity criteria, based on failures revealed by resetting. The RISA approach can be applied to any similarity measure that supports adjusting distances between pairs of cases (e.g., based on a ranking loss function).

Experimental results in this study support that RISA, in conjunction with a local weighting scheme, can improve the similarity measure to produce shorter adaptation paths requiring fewer resets.

3.1 Reset-induced Similarity Adjustment

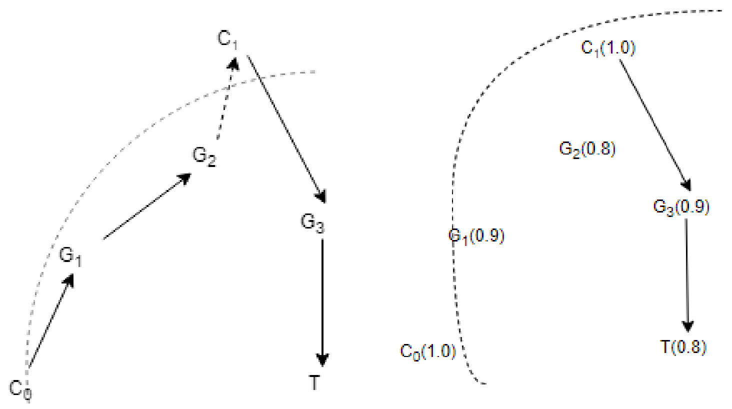
The quality of a CBR system's retrieval plays a critical role in system performance. Retrieval is generally based on similarity, which is used as a proxy for adaptability: the goal of retrieval is to retrieve the most adaptable cases (Smyth and Keane, 1998). With a perfect similarity measure, ROAD would never need to reset a path. Consequently, when resets are needed, it reveals deficiencies in the similarity measure. These are opportunities for similarity learning.

RISA uses generated adaptation paths to guide similarity learning. The goal of learning is to adjust the similarity measure so that the case to which the path was reset will become the initial retrieval in the future, enabling adaptation to be performed with fewer steps and decreasing the processing cost of future adaptations. The example in Figure 3.1 illustrates the potential benefit of ROAD for adaptation path length. In the figure, ROAD would generate the same solution regardless of whether it retrieves C_1 or C_0 in its initial retrieval. However, if the system starts by retrieving C_1, it would avoid the effort of building the path from C_0 to C_1.

3.1.1 The Reset-induced Similarity Adjustment Algorithm

The RISA algorithm is shown in Algorithm 2. RISA takes as input (1) information recorded about resets during adaptation, and (2) a procedure to adjust feature weightings for similarity based on the stored information.

Information recorded about resets: To support RISA, the ROAD implementation was augmented with instrumentation to record its reset behaviors. Each time the system resets a path,



(a) Before RISA (b) After RISA

Figure 3.1: RISA modifies similarity measure to retrieve more adaptable cases. Dashed arc marks a certain similarity distance from the target query.

it also stores a path segment record of the form $(C_{start}, C_{reset}, T)$ where C_{start} is the case most recently retrieved prior to the reset, C_{reset} is the case retrieved by adapting and resetting from C_{start}, and T is the target case. For the first reset record for a path, C_{start} is the case the CBR system retrieved for the original problem. In each subsequent path segment record that is generated during resetting, C_{start} is the case retrieved for the previous reset—from which the path is continuing—and C_{reset} is the case to which it is reset.

There are two situations in which a path segment record may not include a reset: When the path from the initially retrieved case can be pursued to the target without resetting, and when the path from a reset case can be pursued to the target without further resetting. In those cases the record uses *null* for C_{reset}. The presence of C_{reset} indicates a potential defect in the similarity measure. One strategy for addressing similarity defects, pursued in this chapter and elsewhere (Bonzano et al., 1997), is to adjust feature weights. Other issues such as insufficient vocabulary knowledge and noisy cases might also lead to resets, but are beyond the scope of this chapter.

Adjusting feature weights: For a record $(C_{start}, C_{reset}, T)$ produced by a path reset, RISA adjusts similarity criteria to increase the similarity of C_{reset} and T (by pulling them closer), and to decrease the similarity of C_{start} and T (by pushing them away from each other). As a result, the case retrieval process is more likely to retrieve C_{reset} directly for future problems similar to T.

A potential issue is that this adjustment may have ramifications for other retrievals, possibly affecting situations in which prior retrievals were correct. Consequently, for a $(C_{start}, null, T)$ record produced by a path not involving reset, RISA pulls C_{start} and T closer, to help preserve the current correct retrieval. The goal is to preserve the ability to generate high quality adaptation paths for similar starting and ending points in the updated similarity measure.

In general, there are many ways in which the push/pull effect could be achieved. For example, a feature weight updating policy can adjust the similarity distance between two cases. In our testbed RISA system for evaluation, we follow the approach of ISAC (Bonzano et al., 1997). To pull two cases closer, ISAC increases weightings of their matching features and decreases weightings of differing features. Similarity scores between the two cases are thus increased. Similarly, to push two cases away from each other, ISAC decreases weightings of matching features and increases weightings of unmatching features. ISAC adjusts feature weightings using the update formula:

$$w_i(t+1) = w_i(t) \pm \delta * \frac{F_c}{K_c}, \tag{3.1}$$

where $w_i(t)$ is the i-th feature weighting at time step t. δ is a fixed value. F_c is the number of times the case has been "falsely retrieved"—retrieved when another case would have been more suitable—and K_c is the number of times the case has been successfully retrieved. In the following evaluation, we apply this update procedure in ROAD, with failed retrievals corresponding to retrievals prompting a reset, and successful retrievals those for which no reset was needed.

Algorithm 2 Reset-induced Similarity Adjustment

```
Input:
Paths: records of paths ($(C_{start}, C_{reset}, T)$). If the path is never reset $C_{reset} = null$
SM: similarity measure
Pull(SM,A,B): Updating procedure for SM that pulls A and B closer
Push(SM,A,B): Updating procedure for SM that pushes A and B away
Output:
SM: the modified similarity measure

for all $(C_{start}, C_{reset}, T)$ in $Paths$ do
    if $C_{reset} = null$ then
        $Pull(SM, C_{start}, T)$
    else
        $Push(SM, C_{start}, T)$
        $Pull(SM, C_{reset}, T)$
return $SM$
```

3.1.2 Evaluation of RISA

The evaluation of RISA tested the effect of its similarity adjustment on adaptation efficiency and solution accuracy. Adaptation efficiency was measured by the ability to generate shorter adaptation paths and to decrease the number of resets required during adaptation. Solution accuracy was measured by relative error for a numerical prediction task.

The criteria for adaptation efficiency directly measure the ability of the system to retrieve adaptable cases; we expected RISA to increase the system's ability to do so. We did not expect a strong effect on accuracy, but sought to observe whether the decreased path length brought accuracy benefits. The experiments tested RISA for two alternative similarity schemes: Global weighting and local weighting. Because RISA's feature weight adjustments could have unexpected side-effects on retrievals of distant cases when the adjusted weights are global, we expected better performance for local weightings.

Experimental Design

Task Domain: The evaluation task was automobile price prediction, using the Kaggle automobile dataset (Kaggle, 2017). Because the need for adaptation paths and difficulty of adaptation are affected by case-base sparsity, the experiments simulated varying levels of sparsity by removing the closest N cases to the target case before each trial, for varying N. For additional discussion of the data set and the case removing process, see Chapter 2.

Similarity measure: The similarity between two cases is a weighted sum of feature similarity, with each feature weighted by either global or local weighting. Similarity of nominal features is 1 if they are identical and 0 otherwise; similarity between numerical features is their absolute difference normalized into $[0, 1]$. All feature weights were initialized to the same value.

Global vs. local similarity: We test the effect of RISA for two feature weighting methods: (1) Global weighting relies on a single set of feature weights applied for all similarity comparisons; (2) Instance-specific weighting allows each case to have its own set of feature weightings, used for comparisons to that case (Aha and Goldstone, 1992; Bonzano et al., 1997; Friedman, 1994).

Adaptation Rules: Adaptation rules were generated automatically from the case base using the case difference heuristic (CDH) approach (Hanney and Keane, 1996). This approach compares pairs of cases and generates rules that apply when a retrieved case and target case have similar problem differences, and adjusts the solution of the retrieved case according to the solution difference in the case pair from which the rule was learned. The process used here follows the algorithm in Leake and Schack (2018). The rule set generated depends on parameters described in Section 1.2.

300 rules are generated from pairs of random cases ($ruleGenDist = 1.0$), and half of the feature differences between two cases are used ($rspec = 0.5$). These configuration parameters were chosen

#	Rule Specificity	Rule Gen Dist	Avg Path Len	Avg Path Len after Last Reset
1	0.5	1.0	6.523	2.208
2	1.0	1.0	2.476	1.079
3	1.0	0.2	1.992	0.830
4	0.8	0.2	2.111	0.974

Table 3.1: Rule Set Configurations and Corresponding Path Lengths

based on a simple preliminary experiment to identify rule characteristics for which path lengths were high and increased efficiency would be most useful. In every run of the preliminary experiments, the closest 150 stored cases were removed around the query case to increase the need for longer paths. The average length of full paths and the average length of the paths after their last resets are recorded. The difference between the two measures shows the potential saving in efficiency: If the last reset case were retrieved directly, the system would avoid building the path from the original source case to the last reset case. The preliminary experiment used four rule sets of 300 rules. As shown in Table 3.1, rule set #1 has the longest average length and the biggest proportional benefit, so was chosen as the testbed rule set.

ROAD configuration for experiments: All experiments are based on adaptation path generation by the ROAD system, described in Chapter 2. The performance of ROAD depends on multiple parameters. The test version of ROAD had the following configurations: (1) Multiple adaptation paths are generated simultaneously (at most 5 paths); (2) Maximum path length is 10; (3) Paths are reset when reliability decays below a threshold or when two paths disagree on the solutions.

Experimental Results

Effect of RISA with Global Weighting Using every case in the case base as a target, by 10-fold cross validation, Table 3.2 and Table 3.4 show that the effect of RISA with global weighting

	Number of Cases Removed	150	125	100	75	50	25
Before RISA	Average Path Length	6.975	6.687	6.204	6.101	6.064	5.431
	Sd of Path Length	1.670	1.682	1.666	1.683	1.688	1.663
After RISA	Average Path Length	**6.581**	6.572	**5.850**	**5.548**	**5.651**	5.195
	Sd of Path Length	1.671	1.687	1.620	1.643	1.648	1.666
P Value		**.021**	.505	**.035**	**.001**	**.015**	.17

Table 3.2: Effect of RISA on Path Length under Global Weighting

	Number of Cases Removed	150	125	100	75	50	25
Before RISA	Average Error	0.481	0.461	0.388	0.410	0.282	0.275
	Sd of Error	0.816	0.814	0.724	0.890	0.544	0.527
After RISA	Average Error	0.465	0.452	0.406	0.419	0.319	0.345
	Sd of Error	0.734	0.799	0.848	0.868	0.672	0.747
P Value		.83	.92	.82	.92	.55	.29

Table 3.3: Effect of RISA on Error under Global Weighting

consistently decreases average path length and the number of resets. As shown, most differences are significant ($p < 0.05$).

Table 3.3 shows the effect of RISA with global weights on the average relative error. After RISA, the effect on error is mixed. The rates tend to become worse, but the differences are not significant. We hypothesize that this is due to the coarse-grained nature of updating weights for global weighting. Updating global weighting influences the similarities between all cases, which can have adverse effects on similarities for cases other than those prompting the updates. Thus with global weighting, RISA decreases path lengths as desired, but with possible degradation of accuracy.

	Number of Cases Removed	150	125	100	75	50	25
Before RISA	Total Number of Resets	221	261	224	225	248	160
After RISA	Total Number of Resets	177	248	202	182	196	145

Table 3.4: Effect of RISA on Resets under Global Weighting

	Number of Cases Removed	150	125	100	75	50	25
Before RISA	Average Error	0.438	0.381	0.430	0.410	0.340	0.293
	Sd of Error	0.691	0.617	0.835	0.832	0.683	0.529
After RISA	Average Error	0.451	0.393	0.372	0.410	0.321	0.265
	Sd of Error	0.709	0.631	0.673	0.828	0.700	0.501
P Value		.85	.85	.45	.99	.78	.60

Table 3.5: Effect of RISA on the Error under Instance-Specific Weighting

Effect of RISA with Instance-specific Weighting To address the issue of side-effects for adaptation, we tested RISA for local weighting. With local weighting, updating the feature weighting for a case only influences the similarities between this case and other cases, but not the similarities among other cases.

Table 3.6 and Table 3.7 show the effects of RISA on efficiency in the instance-specific weighting configuration. In all runs, the number of resets and average path length consistently drop. Most path length results are significant, while the change in error is statistically insignificant, as shown in Table 3.5.

	Number of Cases Removed	150	125	100	75	50	25
Before RISA	Total Number of Resets	296	261	254	232	233	181
After RISA	Total Number of Resets	261	238	237	201	219	160

Table 3.6: Effect of RISA on Resets under Instance-Specific Weighting

	Number of Cases Removed	150	125	100	75	50	25
Before RISA	Average Path Length	7.154	6.712	6.502	6.218	6.003	5.794
	Sd of Path Length	1.706	1.649	1.654	1.698	1.689	1.718
After RISA	Average Path Length	**6.652**	**6.278**	6.205	**5.737**	**5.672**	5.494
	Sd of Path Length	1.645	1.585	1.618	1.631	1.607	1.673
P Value		**.004**	**.009**	.08	**.005**	**.05**	.08

Table 3.7: Effect of RISA on Path Length under Instance-Specific Weighting

3.2 Related Work in Learning to Refine Similarity Criteria

RISA is a learning method for refining similarity criteria. An extensive body of CBR research has addressed similarity learning (see, for example, Wettschereck et al. (1997), and, for a recent overview, Mathisen et al. (2019)). RISA differs in aiming to refine existing similarity criteria on the fly, rather than generating a similarity measure from scratch. RISA's failure-driven method for learning to refine similarity criteria is most closely related to Bonzano et al. (1997), which adjusts similarity weights in response to failed and successful retrievals, and whose updating strategy is applied by RISA. This work is also in the spirit of work on building similarity measures by Xiong and Funk (2006), which adjusts local similarity to reflect the utility of pairs of cases.

3.3 Conclusion

This chapter presents an initial investigation of applying information from adaptation paths to improve efficiency of path-based adaptation by focusing on similarity measures. By learning from path generation failures, as shown by path resets in ROAD, RISA helps align the similarity measure with adaptability. Experimental results support its value for retrieving more adaptable cases.

Chapter 4

The Implications of ROAD for Adaptation Rule Selection

In the introspective reasoning regarding the reuse stage of CBR, paths built by ROAD may be used to assess the compatibility and reliability between rules. The interaction between rules may not be uniform: Some rules cannot be combined, some rules can be combined but is unreliable, while some rules can be combined and the adaptation is still faithful to the task domain. Given a case base and an adaptation rule set, ROAD can simulate path building without resetting (because resetting upholds the reliability of paths and therefore the final result does not reflect the net effect of the rules used) and check the reliability of built path by comparing the final solution with the head's nearest neighbor.

Following RISA in Chapter 3, we provide a second maintenance method compacts the set of adaptation rules. Especially with the use of large-scale automatic rule generation methods, large sets of adaptation rules may be generated, making rule filtering potentially important to CBR system performance (Jalali and Leake, 2014). Also, in path-based adaptation using heuristic search, decreasing the number of rules to consider decreases the branching factor—and consequently, the computational cost—of the search. To compact the adaptation rule set, we propose *Compatibility-based Adaptation Rule Selection* (CARS), which prioritizes adaptation rules for retention based on analysis of pairwise compatibility of adaptation rules in adaptation paths.

4.1 Compatibility-based Adaptation Rule Selection

Commonly, adaptation rules are assumed to be independent, and selected without regard for its interactions with other rules. However, it is well known that rules may not capture all aspects of a situation, resulting in uncertain outcomes, potentially resulting in degradation of adaptation results (Leake and Whitehead, 2007). An adaptation path might further compound the cumulative error by applying multiple rules. In response, we propose a method for using information about the interactions of adaptation rules to learn which adaptation rules to favor.

Given a case base and an adaptation rule set, it is possible to estimate reliability of adaptation rules by building paths with resetting disabled, and then comparing the final solution from the path with the actual solution of the nearest neighbor. The path's error can then be used to represent the reliability of the rules involved. This is the approach of CARS.

4.1.1 Compatibility-based Adaptation Rule Selection Algorithm

Algorithm 3 shows the process that CARS uses to assess rule compatibility. It computes a rule compatibility matrix representing the compatibility between every pair of rules, calculated from a set of adaptation paths of length 2, generated based on test adaptations of a selected set of cases. This set of cases could be the entire case base or a subset (for efficiency).

Given *rcount* rules, the compatibility matrix has dimension $rcount \times rcount$. The entry (i, j) in the matrix records the compatibility between $rule_i$ and $rule_j$ if $rule_i$ and $rule_j$ can be applied in sequence to a case. If adaptation rules are commutative (e.g., if each rule is multiplying a numerical solution value by a feature difference in a single dimension), entry (i, j) is equal to entry (j, i) and the matrix is symmetric. However, in many domains rules depend on each other and must be applied in a particular order (e.g., recipe generation).

CARS estimates compatibility between two rules based on an estimated error value after the two rules are successively applied to a case in a 2-step adaptation path. Error is estimated by retrieving the stored case closest to the ghost case generated by the adaptation path, and comparing the stored case and ghost case solutions. After computing the compatibility matrix, CARS uses the average value of each row as a proxy for the overall reliability of the corresponding rule.

CARS compresses the adaptation rule set by retaining only the most reliable rules. To do so, it sorts all rules based on reliability and trims the rule set by retaining only those falling above a selected percentile of the original rule set.

4.1.2 Evaluation of CARS

The evaluation of CARS addresses two questions:

- How does CARS rule deletion affect efficiency of adaptation path generation?
- How does CARS rule deletion affect solution accuracy?

Efficiency is measured in three ways: number of resets prompted by path reliability decay, number of resets prompted by disagreement between alternative paths being explored, and average adaptation path length. Tests assess the effect of CARS both on ROAD and on a baseline ablated version of ROAD that performs single-rule adaptation instead of using adaptation paths.

Experimental Design

As in the previous experiment, tests used the Kaggle automobile data set (Kaggle, 2017). In all runs, after building the compatibility matrix based on the entire case base, 75 cases around the target query are removed to simulate situations where multiple adaptations are needed but the adaptation paths have moderate average length. The experiment uses the rule set configuration ($Rcount = 300$, $RuleGenDist = 1.0$, $Rspec = 0.5$) with 10-fold cross validation. In the experiments, CARS is applied to assess compatibility and retain the top 80%, 60%, 40%, and 20% of rules; these conditions are compared to a baseline of 100% retention. We note that because of the case removals to simulate a sparse case base, as well as the rule retention and path building mechanisms in ROAD, the case bases and rules used to solve the test problems are different from those used when building the rule compatibility matrix.

Experimental Results

Table 4.1 shows the experimental results. We observe:

- The number of resets consistently decreases when fewer rules are used. With fewer rules the path searching algorithm explores fewer options, decreasing the number of prior cases near the path. In addition, we hypothesize that because the retained rules are more reliable, paths tend to agree with each other more often, decreasing resets due to path disagreement.

- The average path lengths become smaller as fewer rules are retained, because of fewer resets.

- The error improves markedly as the worst 20% of the rules are deleted (80% retention). Observed error is better than the initial rule set for 60% and 40% retention, with the benefit dropping compared to 80% retention, but the differences below 80% are not statistically significant.

Thus the results support the efficiency benefits of rule set compression, as well as improved accuracy at low compression rates. Effects for higher compression rates are an interesting question for future study. We expect a tradeoff between higher average rule quality and lower coverage with the compressed rule sets, resulting in decreasing accuracy when the rule set is too sparse. However, refining the reliability estimate process might help to delay serious accuracy loss.

In summary, using CARS increases the efficiency of both the baseline and ROAD by reducing the number of resets and reducing path lengths, with a stronger effect on ROAD. Also, deletion of an initial set of the worst rules benefits accuracy for both. Beyond that deletion level there is weak or no significant effect on the accuracy until accuracy falls for very high deletion levels.

4.2 Related Work

CARS is a method for prioritizing adaptation rules for retention.

Hanney et al. (1995)'s initial proposal for generating adaptation rules by the case difference heuristic also proposed selective retention, based on the frequency with which particular rules were generated from cases . Adaptation rule maintenance to remove duplicates and resolve conflicts was

Percent of Rules Retained	100%	80%	60%	40%	20%
Using Single-Rule Reliability (baseline)					
Resets due to Reliability Decay	227	152	99	37	14
Resets due to Path Disagreement	137	81	72	42	22
Average Error	0.372	0.268	0.323	0.324	0.427
SD of Error	0.407	0.229	0.321	0.281	0.365
P Value (Comparing to no Trimming)	N/A	.002	0.19	0.18	0.16
Average Path Length	6.126	4.689	3.841	2.662	1.932
SD of Path Length	3.011	2.575	2.366	1.630	1.200
P Value (Comparing to no Trimming)	N/A	<.001	<.001	<.001	<.001
Using Reliability from Compatibility Matrix (CARS)					
Resets due to Reliability Decay	227	49	17	1	0
Resets due to Path Disagreement	137	83	41	10	1
Average Error	0.372	0.303	0.305	0.356	0.420
SD of Error	0.407	0.267	0.244	0.281	0.316
P Value (Comparing to no Trimming)	N/A	.049	.05	.65	.19
Average Path Length	6.126	3.322	2.346	1.548	1.231
SD of Path Length	3.011	2.053	1.366	0.719	0.468
P Value (Comparing to no Trimming)	N/A	<.001	<.001	<.001	<.001

Table 4.1: Performance by Rule Retention Level

proposed by Li et al. (2007). Additional work focuses on how to prioritize rule selection, without deleting rules from the rule set (Leake and Dial, 2008), and on combining systematic accuracy testing with retention of top-ranked rules (Jalali and Leake, 2014). The current work differs in addressing adaptation paths, rather than only individual rules, and in focusing on minimizing adaptation path length.

4.3 Conclusion and Limitation

CARS apply information from adaptation paths to improving efficiency of path-based adaptation by focusing on adaptation rules. By favoring pairs of rules that have participated in successful adaptation paths, CARS compresses the adaptation rule set while retaining useful rules. Experimental results support its value for increasing adaptation efficiency and that deletion of least reliable rules can improve accuracy, subject to an efficiency/coverage tradeoff.

In the current implementation of CARS, the concepts of reliability and compatibility are mixed. It is possible to distinguish them and implement a more fine-grained version of CARS. For example, the ROAD system in its building of adaptation paths may search only compatible rules and prioritize reliable rules. This can be especially helpful in a domain where certain rule can be compatible with few rules but highly reliable when used in a path. However our test data set and rule set do not have such a characteristic for us to test this more fine-grained version of CARS.

Algorithm 3 Assessing Rule Compatibility

```
**Input:**
*CaseSet:* Cases for testing
*rules:* List of adaptation rules
**Output:** Rule compatibility matrix

$R = size(rules)$, $N = size(CB)$
Initialize matrix $M$ of size $R \times R$
for $i \leftarrow 1$ to $R$ do
    for $j \leftarrow 1$ to $R$ do
        $M[i][j] \leftarrow undefined$
for $i \leftarrow 1$ to $R$ do
    for $j \leftarrow 1$ to $R$ do
        $totalError = 0$
        $errorCount = 0$
        for all $case$ in $CaseSet$ do
            if $rule[i].isApplicableTo(case)$ then
                $ghost1 = rule[i].applies(case)$
                if $rule[j].isApplicableTo(ghost1)$ then
                    $ghost2 = rule[j].applies(ghost1)$
                    $target = CaseSet.nearest(ghost2)$
                    $totalError+ = errorInSolution(ghost2, target)$
                    $totalCount+ = 1$
                else
                    continue
            else
                continue
        if $totalError \neq 0.0$ then
            $M[i][j] = totalError/totalCount$
return $M$
```

Chapter 5

Neural Network Based Cased Difference Heuristics: Supporting the Adaptation Process with Neural Networks

5.1 Introduction

Chapter 2 through Chapter 4 discusses a method of applying multiple case adaptation rules. Rule-based adaptation is an intuitive and straightforward adaptation method in CBR. However, obtaining the adaptation knowledge needed to adapt prior solutions is a classic challenge. In response, extensive research has explored the use of machine learning methods to acquire case adaptation knowledge for both classification (D'Aquin et al., 2007; Jalali et al., 2017a) and regression (Fuchs et al., 2014b; Jalali and Leake, 2015; Liao et al., 2018; Patterson et al., 2002; Policastro et al., 2006). An interesting recent direction is the use of neural network methods for case difference heuristic (CDH) learning of adaptations for case-based regression.

Neural network architectures for case adaptation have been the subject of considerable study within the case-based reasoning community. However, gaps remain both in architectural capabilities and in fundamental questions of system behavior. First, existing architectures focus on adaptation of numeric attributes; adapting nominal attributes is an open challenge, made harder because of the lack of standard methods for expressing such adaptations in network architectures. Second, some proposed neural network adaptation architectures take as input a query and retrieved case and generate a solution, which raises a question about whether they truly learn adaptation——in principle, they could learn simply to ignore the prior case and generate a solution from scratch. Third, the performance benefit of using CBR systems with neural network components over other models is unclear. This chapter presents research on these questions.

This chapter introduces a new method for expressing the difference between two one-hot encoded nominal values as a vector, and extends the NN-CDH adaptation approach (Leake et al., 2021b) to predict a solution difference given a problem difference, where the problem/solution difference can involve nominal attributes (features or labels). Unlike the earlier variation of NN-CDH for classification, this new extension guarantees the model is learning adaptation knowledge. Extensive experiments on multiple regression and classification data sets, and on controlled artificial data sets, compare the extended NN-CDH with other models. Some trends of results paralleled those obtained in previous tests (Leake et al., 2021a; Ye et al., 2021b), providing additional support for them as general characteristics of neural network adaptation.

The experiments suggest that NN-CDH can be most useful when (1) the retrieval is relatively good to provide a good initial start for adaptation and not out of synchronization with the adaptation and (2) when queries are relatively novel so adaptation is needed beyond retrieval and yet not too novel, for adaptation needs to go beyond the learned adaptation knowledge.

The results reveal two important lessons: (1) Good case retrieval alone can surpass other ML methods such as neural networks. The adaptation could worsen the retrieval result if the two processes are not in synchronization. This issue is observed in this chapter but beyond the scope of this chapter. More discussion on this issue can be found in Leake and Ye (2021a); (2) Directly integrating neural networks into a CBR system may lead to comparable, but often not better results than the counterpart neural network. To actually surpass statistical AI such as the neural network, we believe symbolic knowledge will be necessary.

5.2 Background

5.2.1 Neural Network Adaptation Learning by the Case Difference Heuristic

CBR researchers have explored many machine learning techniques for acquiring case adaptation knowledge. Because of the difficulty of codifying adaptation knowledge, especially in poorly understood domains, there has been particular interest in data-driven methods (Corchado and Lees, 2001; Craw et al., 2006a; F. Zhang et al., 2004; Liao et al., 2018; Policastro et al., 2006). Hanney and Keane's case difference heuristic approach (Hanney and Keane, 1996) is a method for learning adaptation knowledge from the case base, by comparing pairs of cases and attributing the solution difference to the problem difference. Originally, the CDH approach was applied to learning adaptation rules, but other variants have been explored. Craw et al. (2006a) present a method that stores case pairs and applies the most similar pair with the most similar problem difference when an adaptation is needed. Liao, Liu, and Chao (Liao et al., 2018) train a neural network on case pairs to predict solution difference based on problem difference. Leake, Ye, and Crandall (Leake et al., 2021a) take a similar approach but train the neural network with adaptation context in addition to problem/solution differences. This method is called the neural network-based case difference heuristic (NN-CDH) method.

5.2.2 Case Adaptation Involving Nominal Attributes

The work mentioned in Section 5.2.1 all focus on case adaptation for regression tasks, based on numeric attributes. The hesitation of the research community over case adaptation involving nominal attributes is expected. Nominal attributes hide some information, so it is convenient when the information is not relevant. For example, imagine three apartments with one bed, two beds, and three beds, and of the price 700, 900 and 1200 respectively. We define two class labels as "cheap" for prices under 1000 and "expensive" for prices over 1000. From the perspective of machine learning in general, classification might be an easier problem than regression. In our example, it is easier for a system to predict that the rent for a one-bed apartment and a two-bed apartment are both "cheap" than to predict their price numbers.

As noted by Craw et al. (Craw et al., 2006a), adaptation involving nominal attributes is difficult compared to numerical attributes because there is no natural calculation for symbolic similarity (for retrieval) or dissimilarity (for adaptation). Some previous works have studied non-neural methods for learning case adaptation involving nominal attributes. For example, Jarmulak, Craw and Rowe (Jarmulak et al., 2001) present a case-based method that learns adaptation cases for both numeric and nominal features. Jalali, Leake, and Forouzandehmehr (Jalali et al., 2017b) combine a statistical method with an ensemble approach. They use the ordered pair of two values of a nominal feature x (e.g., (t_i, t_j)) as an adaptation rule that modifies the value from t_i into t_j. As a precondition of applying a rule such as (t_i, t_j), the system must check that the source case's x feature is of value t_i before modifying it into t_j. Craw et al. (Craw et al., 2006a) propose two methods to adapt nominal attributes: 1) a coarse method indicating whether a nominal attribute should remain the same or change, 2) a multi-class adaptation which proposes the target solution directly.

5.2.3 Initial Version of NN-CDH for Classification

The initial version of NN-CDH for classification followed the second route proposed by Craw et al. (Craw et al., 2006a), Ye et al. (Ye et al., 2021b) modify NN-CDH into C-NN-CDH for classification task domains. The C-NN-CDH approach predicts a target solution (class label) based on a source case and a target problem. It was tested in comparison to other classification algorithms including SVMs and neural networks. Those experiments supported that C-NN-CDH can achieve state-of-art case adaptation and classification results.

However, C-NN-CDH operates differently from other CDH methods (including NN-CDH) that work directly with problem/solution differences. C-NN-CDH takes a source case and a target problem as inputs. It is not guaranteed to make use of the source case to truly perform adaptation; in principle it could learn to find a target solution based on the target problem only. In response, this chapter proposes a single adaptation model that learns from differences of nominal features/labels and therefore guarantees the learning of adaptation knowledge.

NN-CDH can handle nominal feature differences but not nominal label differences. To address this, we use the following encoding scheme. If a nominal feature is one-hot encoded into a vector of numbers (with a single "1" and multiple "0"s), it can be treated as multiple numerical features and processed by the neural network. Two such vectors can be subtracted from one another to form a nominal difference, which can be then passed to the input layer of NN-CDH. However, NN-CDH's last layer uses a single neuron with the linear activation function to predict a numerical output. This numerical output is the solution difference in a regression task. This output neuron cannot represent a nominal difference in the solution.

Both the studies on NN-CDH and C-NN-CDH methods (we will refer to both as NN-CDHs for short) compared CBR systems using NN-CDHs with their counterparts, neural network systems that predict target solution directly from target problem (Leake et al., 2021a; Ye et al., 2021b). Ye et al. (Ye et al., 2021b) show that NN-CDH performs comparably to the counterpart neural network system on certain data sets. Leake, Ye and Crandall (Leake et al., 2021a) hypothesize that NN-CDHs can outperform end-to-end network methods, for example, when the query is novel in a high dimensional space. However more evidence is needed to assess this hypothesis. This chapter reports experiments to elucidate this behavior.

5.2.4 Siamese Networks for Similarity Assessment

Siamese networks (Bromley et al., 1993) are a neural network architecture that can predict the distance between two input samples. A siamese network is composite of two identical feature extraction networks and a subtraction layer. Given two input vectors (in our context, two feature vectors, one describing a query and the other the problem addressed by a prior case), the feature extraction networks extract their features and the subtraction layer compares their features and outputs a value indicating the distance (often using a sigmoid function) between the two cases. Siamese networks have been used as the similarity measure in case retrieval (Martin et al., 2017; Mathisen et al., 2019), with good performance. This chapter refers to a KNN retrieval process that uses a siamese network as similarity measure as *SN retrieval*.

5.2.5 The Relationship Between Retrieval and Adaptation

Smyth and Keane observed that for efficient adaptation, CBR retrieval and adaptation knowledge must be tightly connected (Smyth and Keane, 1998). Leake and Ye (Leake and Ye, 2021a) showed that even when strong models are trained for both retrieval and adaptation, adapted solutions might be less accurate than the retrieved solution if the retrieval and adaptation knowledge are not in synchronization with each other—in other words, if the retrieved cases are close to the real solution by the similarity metric but not easily adaptable, or if the adaptation model is not trained to adapt such retrieval results. The experiments in this chapter tested a CBR system adapting results from either 1-NN retrieval or SN retrieval. The experiments further support that when both the retrieval model and the adaptation model are well trained individually but not in accordance with each other, adaptation may worsen the result of retrieval (Leake and Ye, 2021a).

5.3 NN-CDH for both Classification and Regression

This chapter extends NN-CDH to perform adaptation for both regression and classification task domains. If the domain is regression, the model works identically to the original NN-CDH (Leake et al., 2021a). The following first explains the case adaptation process applicable to both classification and regression domains, then introduces a network method for handling nominal difference, and last discusses the neural network structure of NN-CDH and how it works with both numerical and nominal differences.

5.3.1 The General Model of Case Adaptation

Given a query with a target problem description, the standard CBR adaptation process first retrieves a case whose problem is similar to the query. The retrieved case contains a retrieved problem and a retrieved solution. The system calculates a problem difference and an adaptation context to adapt this retrieved case toward the query. The problem difference is the difference between the retrieved problem and the query problem. Last, the retrieved solution is modified according to the solution difference.

NN-CDH augments the difference information by using the retrieved problem as context for the adaptation. It uses both the adaptation context and the problem difference to predict a solution difference.

If the system task is a regression task, NN-CDH calculates the difference of two values over a numerical attribute by subtracting the two values. However, in classification tasks, each case contains a solution description that is a nominal label (Multi-class labeling is not discussed in this chapter). The neural network of NN-CDH needs to work with outputs of differences of nominal attributes, or nominal differences. It does this as described in the following section.

5.3.2 1-Hot/1-Cold Nominal Difference

Given a nominal-valued attribute n that can take any of a set d different possible values for that attribute, one-hot encoding converts n into a group of d binary bits $\{n_1, n_2, ...n_d\}$, where n_i is '1' and all others bits are '0's. Given two nominal values $n = \{n_1, n_2, ...n_d\}$ and $m = \{m_1, m_2, ...m_d\}$ of the same attribute, their 1-hot/1-cold nominal difference (or "nominal difference" for short) is defined as

$$n - m = \{n_1 - m_1, n_2 - m_2, ...n_d - m_d\} \tag{5.1}$$

Given two cases in a classification task, their problem difference is calculated as a vector of individual feature differences concatenated, which may or may not involve nominal differences; Their solution difference is the nominal difference of their class labels. If the two cases are of the same class, then the solution difference is all '0's, indicating no difference between their solutions; Otherwise, the solution difference contains exactly one '1' and one '-1' (hence the name "1-hot/1-cold"), indicating changing from one class into another. Nominal differences are vectors that can be either input or output of NN-CDH, allowing NN-CDH to learn the projection from problem difference to solution difference in a classification task.

As a side benefit, the problem difference may contain both nominal and numerical differences, allowing interaction between the two categories of feature differences. This benefit is similar to that of a classification neural network which learns one-hot encoded nominal values and numerical values together.

5.3.3 Neural Network Structure of NN-CDH

Given a neural network structure for a given domain and task, the network structure of the NN-CDH for it is similar. The main difference is in its first and last layers. The first layer needs to have more neurons to accommodate both adaptation context and problem difference. The last layer needs to express predicted solution difference. The last layer is a dense layer using the tanh activation function, as opposed to the linear activation function in the original NN-CDH.

For regression, the last layer is a single neuron, producing a solution difference between -1 and 1 to increase or decrease a retrieved solution. For classification, the last layer is a dense layer with d neurons, where d is the dimension of a one-hot encoding of the classification label. Each neuron can be a value between -1 (indicating "change into this class") and 1 (indicating "change away from this class").

5.3.4 Training and Adaptation Procedure

The training and adaptation procedures follow from the original NN-CDH, but with some modification. This is illustrated in Algorithm 4.

During training, pairs of training cases are assembled as training data for NN-CDH. For each pair, their problem difference and solution difference are calculated. One problem description of the two cases is used as the adaptation context. The problem difference and the adaptation context are concatenated to form the input of the NN-CDH and the solution difference is the expected output of the NN-CDH. Lastly, the NN-CDH is trained using backpropagation and the mean squared error as the loss function.

During adaptation, the NN-CDH predicts a solution difference based on a problem difference and an adaptation context. Depending on the task domain, the retrieved solution is modified according to the solution difference in one of two ways. If it is a regression task, the retrieved solution subtracting the predicted solution difference forms the final solution (same as the original NN-CDH). If it is a classification task, the retrieved solution is an one-hot encoding of the class of the retrieved case. The retrieved solution subtracts the predicted solution difference element by element. The maximum bit of the resulting vector is used to determine the final solution.

5.4 Evaluation

Our evaluation addresses the two main questions: Does NN-CDH consistently improve the result of the CBR retrieval, and how does a CBR using NN-CDH perform when compared to a neural network of equivalent computational power?

5.4.1 Systems Being Compared

This chapter tests six different methods and compares their performance in terms of prediction accuracy. They are: 1) a baseline neural network, 2) k-nearest neighbor (k-NN), 3) a CBR system whose retrieval is either 1-NN retrieval or SN retrieval and whose adaptation is either a rule-based CDH or an NN-CDH. The rule-based CDH is a baseline adaptation method (referred as NN_A in the work of Craw et al. (Craw et al., 2006a)) where the case pairs are stored in an adaptation case base. The rule-based CDH uses a non-optimised 1-NN retrieval to select the case pairs based on the adaptation context and problem difference and applies the solution difference as the adaptation. Because there are two variations of retrieval and two of adaptation, four variations of CBR systems are tested. The six different methods are referred as "all models" in the following text.

Algorithm 4 Pseudocode for the Training and Usage of NN-CDH

```
 1: procedure Training($cases$)
 2:     $pairs \leftarrow$ assembled pairs of $cases$
 3:     $NN\text{-}CDH \leftarrow$ new neural network
 4:     CDH_data $\leftarrow \{\}$
 5:     for each $source$ and $target$ in $pairs$ do
 6:         CDH_data.append(
 7:         [$prob(source)$,$prob(source) - prob(target)$:$sol(source) - sol(target)$]])
 8:     $NN\text{-}CDH$.fit(CDH_data)
 9:     return $NN\text{-}CDH$
10: procedure Adapt($query$,retrieval)
11:     $retrieved \leftarrow$ retrieval$(CB, query)$
12:     $sol\_diff \leftarrow NN\text{-}CDH$.predict$(prob(retrieved), prob(retrieved) - prob(query))$
13:     if Task is regression then
14:         $r \leftarrow sol(retrieved) - sol\_diff$
15:     else if Task is classification then
16:         $r \leftarrow sol(retrieved) - sol\_diff$   #element-wise subtraction
17:         $r \leftarrow argmax(r)$
18:     return $r$
```

The neural networks may have more or fewer layers and neurons per layer to accommodate the varying complexity of different data sets. In our implementation, the neural networks have 2-4 hidden layers, each of which has 8-128 neurons. For every task domain, the baseline neural network is almost identical to the NN-CDH and they share the same number of layers, neurons and activation functions, except for the first layer because NN-CDH takes in adaptation context and problem difference. This is to ensure fair comparison of the two models because they share similar complexity. The structure of the NN-CDH was already discussed in Section 5.3.3. The last layer of the baseline neural network uses a sigmoid activation function for regression or softmax activation function for classification. The baseline neural network model is trained with the loss function of mean squared error in regression or with the loss function of categorical cross entropy for classification. Such designs of neural networks for regression and classification are widely used.

Both k-NN and 1-NN are default implementations from the scikit-learn package (Pedregosa et al., 2011). They use Euclidean distance over problem feature values to calculate distance between cases. All cases are weighted equally. Our k-NN used k=3. When the CBR system uses a siamese network over the Euclidean distance for similarity measure, a separate siamese network is trained to predict the similarity of two given cases.

The siamese network for case retrieval measures the similarity between two cases. The feature extraction network is composite of three dense layers (of dimension 32, 32 and 16) with dropout layers (dropout rate = 0.1) in between. For harder problems, the number of neurons in each layer is multiplied by 4. The two features extracted from two cases are then passed to the subtraction layer which outputs the element-wise feature distance. Last, this is passed into a final dense layer to output a single similarity score. The final layer uses the ReLU activation function for regression or the sigmoid activation function for classification. The siamese network is trained with the mean average error loss for regression or the contrastive loss for classification.

5.4.2 Assembling Case Pairs for Training

While the baseline neural network and the k-NN are trained from cases directly, both the siamese network and the NN-CDH need case pairs for training. Using all possible pairs is unrealistic as a case base of n cases would have n^2 pairs. Multiple strategies for case pair assembly exist (Jalali and Leake, 2013). In all experiments of this chapter, we use n neighboring pairs (each case is paired with its nearest neighbor) and $10n$ random pairs (each case is randomly paired with another case). From these pairs, 90% are used for training and the rest for validation. We observed that this design choice can heavily influence the models, for example, a model may not accommodate two highly different input cases if the model is not trained with enough random pairs. However, this design choice is not the focus of this chapter.

To train the siamese network, we assemble pairs of cases' problem descriptions as input and determine their distance as the expected output of the siamese network. For regression, the distance is the absolute distance between the two cases' solutions. For classification, the distance is 0 if the two solutions are the same and 1 otherwise. To train the NN-CDH, we use problem differences and solution differences of the case pairs.

5.4.3 Data Sets

This experiment involves 5 regression data sets (Airfoil, Car, Student Performance, Yacht, Energy Efficiency) and 5 classification data sets (Credit, Balance, Car, Yeast, Seeds). As a step in data preprocessing, all numeric attributes are scaled to the range of [0,1] and all nominal attributes are one-hot encoded. This way the expected output values match the output range of the NN-CDH. Each data set is tested with 10-fold cross validation, and on three different settings:

- The normal setting: The standard setting where 90% of the cases are used as the training set (out of which 90% of the cases are used for training and the rest for validation) and 10% as the test set. The case pairs are assembled from the training set. The models are first trained and then tested on the whole test set.

- The novel setting (X): Similar to the normal setting with the difference that: For every test case, we remove its top $R\%$ neighbors (based on euclidean distance on their *problem* descriptions) in the train set to form a trimmed train set. All models are trained on the *trimmed train set* and then tested on *that single test case*. This is a direct reuse of a previous novel setting experiment (Leake et al., 2021a).
- The novel setting (Y): Similar to the novel setting (X), we still remove top $R\%$ neighbors of a test case but the Euclidean distance is based on the solution description rather than the problem description. All models are trained on the trimmed train set and then tested on that single test case. This is a modification of another previous novel setting experiment (Ye et al., 2021c).

The novel setting (X) simulates when the models are not trained with cases whose problem descriptions are similar to the query problem. The novel setting (Y) simulates when the models are not trained with cases sharing similar solutions as the query solution. The novel setting (Y) is arguably harder than the novel setting (X) because a CBR system in the later setting may still retrieve a case with good enough solution. The results for 1-NN retrieval under the novel setting (Y) are not reported because the trend is already clear in the novel setting (X). Moreover, the novel setting (Y) does not apply to classification data sets because two cases can be very different but still share the same class label. Removing cases based on class label does not necessarily make the query novel.

The data sets and various settings have been used on previous studies (Leake et al., 2021a; Ye et al., 2021b,c) while this chapter tests both classification and regression using the same models. We chose $R\%$ as 40%. For the normal setting of simpler data sets, experiments run multiple times and the average is taken. However, the novel settings have very high computational costs and even one run on the whole test set is extremely expensive, because each test case requires a re-training of all the models. We resort to randomly choosing only 50 cases from a test set for testing.

Because of the nature of the novel settings, the performance of any model on test cases can vary tremendously as the novelty and difficulty of the test cases vary. Therefore almost all the differences in novel settings are not statistically significant.

Experimental Results on Real Data Sets

	Setting	Retrieval	Neural Network	k-NN	Retrieval	Rule CDH	NN-CDH
Airfoil	Normal	1-NN	7.42%	6.87%	6.94%	5.80%	5.71%
	Normal	Siamese	7.60%	6.97%	6.49%	17.06%	8.82%
	Novel(X)	1-NN	9.34%	16.88%	17.99%	23.85%	12.02%
	Novel(X)	Siamese	9.51%	16.20%	8.45%	18.16%	13.84%
	Novel(Y)	Siamese	8.90%	16.49%	17.18%	22.29%	13.44%
Car	Normal	1-NN	1.52%	1.95%	1.63%	1.81%	1.38%
	Normal	Siamese	1.47%	1.92%	2.19%	6.89%	1.72%
	Novel(X)	1-NN	5.00%	7.42%	8.48%	8.87%	4.31%
	Novel(X)	Siamese	5.41%	7.36%	2.76%	8.45%	3.61%
	Novel(Y)	Siamese	5.06%	7.05%	6.20%	9.32%	4.77%
Student Performance	Normal	1-NN	21.61%	25.47%	31.59%	31.03%	29.63%
	Normal	Siamese	21.38%	25.64%	26.39%	31.66%	30.62%
	Novel(X)	1-NN	16.42%	18.73%	23.30%	27.00%	24.33%
	Novel(X)	Siamese	16.32%	19.30%	21.00%	21.90%	25.82%
	Novel(Y)	Siamese	23.73%	26.17%	32.00%	28.70%	33.38%
Yacht	Normal	1-NN	7.53%	13.77%	11.48%	6.85%	8.05%
	Normal	Siamese	5.94%	13.87%	2.18%	17.11%	7.71%
	Novel(X)	1-NN	10.50%	13.15%	16.72%	26.96%	10.50%
	Novel(X)	Siamese	10.14%	12.96%	3.52%	24.25%	6.43%
	Novel(Y)	Siamese	15.64%	19.56%	13.68%	19.77%	23.22%
Energy Efficiency	Normal	1-NN	7.36%	7.53%	14.62%	15.06%	13.04%
	Normal	Siamese	7.20%	7.55%	2.17%	12.51%	4.89%
	Novel(X)	1-NN	17.96%	23.46%	25.83%	22.29%	14.88%
	Novel(X)	Siamese	17.82%	23.44%	16.40%	22.41%	13.20%
	Novel(Y)	Siamese	17.07%	24.19%	23.72%	20.74%	12.95%

Table 5.1: Error Rates of Models on Regression Data Sets

We observed the error rate of models on regression in Table 5.1 (The lower the better) and accuracy of models on classification in Table 5.2 (The higher the better). It is important to note that the variance is very high (not shown in tables) in any experiment under novel settings. The

	Setting	Retrieval	Neural Network	k-NN	Retrieval	Rule CDH	NN-CDH
Credit	Normal	1-NN	86.06%	85.34%	85.34%	80.37%	81.10%
	Normal	Siamese	85.91%	85.06%	80.78%	75.73%	76.34%
	Novel(X)	1-NN	77.80%	58.60%	58.60%	58.80%	70.20%
	Novel(X)	Siamese	68.00%	48.00%	70.00%	52.00%	76.00%
Balance	Normal	1-NN	97.21%	79.00%	79.00%	72.21%	97.15%
	Normal	Siamese	97.12%	79.44%	98.40%	73.66%	97.28%
	Novel(X)	1-NN	84.00%	46.00%	46.00%	56.00%	70.00%
	Novel(X)	Siamese	88.00%	46.00%	90.00%	42.00%	70.00%
Car	Normal	1-NN	99.87%	86.36%	86.36%	79.21%	99.27%
	Normal	Siamese	99.71%	84.84%	97.63%	71.12%	97.05%
	Novel(X)	1-NN	82.00%	52.00%	52.00%	52.00%	84.00%
	Novel(X)	Siamese	78.00%	60.00%	82.00%	64.00%	62.00%
Yeast	Normal	1-NN	58.83%	54.65%	54.65%	50.68%	52.90%
	Normal	Siamese	58.84%	54.42%	48.18%	45.19%	49.03%
	Novel(X)	1-NN	42.00%	40.00%	40.00%	38.00%	34.00%
	Novel(X)	Siamese	40.00%	26.00%	26.00%	26.00%	30.00%
Seeds	Normal	1-NN	93.71%	92.67%	92.67%	89.33%	94.29%
	Normal	Siamese	94.76%	93.33%	94.29%	90.48%	95.71%
	Novel(X)	1-NN	46.00%	26.00%	28.00%	28.00%	48.00%
	Novel(X)	Siamese	42.00%	14.00%	44.00%	32.00%	40.00%

Table 5.2: Accuracy Rates of Models on Classification Data Sets

difference between the performances of models is not statistically significant under novel settings. However, the general trend still reveals some interesting comparisons between the models.

1. SN retrieval is almost always better than 1-NN retrieval.

2. The relation between retrieval methods and NN-CDH is complicated:

   (a) NN-CDH consistently improves the result of 1-NN retrieval in regression, but less so in classification. We believe this is largely because (1) it is easier to retrieve a case with the same label in classification than to retrieve a case with the exact same solution in regression, and (2) nominal attributes hide subtle differences between cases (ex. a major problem difference may lead to no class change or a minor problem difference may lead to a class change) and therefore case pairs are harder for NN-CDH to learn.

   (b) NN-CDH often fails to improve the result of SN retrieval. When the retrieval process is very good but not in synchronization with the adaptation, in this case NN-CDH, the adaptation model does not necessarily improve the retrieval result. See discussion in Section 5.2.5.

3. Neural network, SN retrieval and NN-CDH all may achieve best performance in various settings of different data sets. It is unclear which model will be most suitable in a given situation but some general trends are observed:

   (a) Under normal settings, the neural network is often best performing. Under novel settings, performance of all models degrades.

   (b) Under the novel setting (X), 1-NN is much worse than in the normal setting, but siamese retrieval performs relatively well and is often best.

   (c) Under the novel setting (Y), the siamese retrieval also suffers. We observe that NN-CDH may improve (but sometimes worsen) the retrieval results.

   (d) The neural network and the NN-CDH have comparable performance. This is because the two models share the same structure and computational power, although trained with different data for different purposes. Often NN-CDH is slightly worse. This may be because NN-CDH is working on the retrieval result. If the retrieval result is bad or if the retrieval and the adaptation are not in synchronization, it is harder to adapt.

   (e) As the neural network and the NN-CDH are similar in terms of problem solving power, if the neural network is underperforming (for example, if it performs worse than the retrieval method), then NN-CDH is likely underperforming as well and likely to worsen the retrieval result.

5.4.4 Artificial Data Sets

Considering the nature of different models, we hypothesize that the locality of the task domain (whether local regions follows a specific pattern that is sufficiently different from the global landscape) is one factor causing disparity between the performances of models. Obviously, there are many other factors of a data set that might influence a model's performance, for example, the sparsity of feature values, the dimension, and the time-spatial relationship between features.

To further study the trends and find scenarios where one model may outperform the others, we created a way to generate artificial data sets with variable locality.

The artificial cases take the form of

$$\{x_1, x_2, ...x_k : y\}, 0 \leq x_i \leq 1$$

where $\{x_1, x_2, ...x_k\}$ is a problem description with k features and y is the solution. k weights $\{w_1, w_2, ...w_k\}$ and k biases $\{b_1, b_2, ...b_k\}$ are randomly sampled from 0 to 1, each corresponding to one feature. For each case, x_i are randomly sampled from 0 to 1 and y is determined by two steps: 1) Find the first integer i such that $x_1 \leq i/k$; 2) Calculate the value of y as $y = w_i * x_i + b_i$.

This data set can be converted to include nominal features and even nominal solutions. For example, the first feature x_1 can be converted to an nominal feature of k possible values. If converted this way, the data set shows a perfect example where the nominal and numerical features are independent and yet interact, as the first feature (nominal) determines which numerical feature to use in calculating the solution.

On a high level, the first feature x_1 is globally used while the other features are only locally used depending on the value of x_1. Therefore this data set demonstrates a good example of task domains involving both global and local landscapes. By tuning the number of features k, we can modify the locality of the data set. When $k = 1$, the data set follows a single global rule; When k is large, the data set follows many local rules.

We generate data sets of 1000 cases respectively using $k = 3$, 5, and 7. For each data set, we test it with all models and three settings: normal, novel (X), and novel (Y). For each novel setting, we also vary $R\%$ to be 10%, 20%, and 30% in order to render gradual influences of the novel settings. To ensure fair comparison between models, we use the same seed to generate random cases for each choice of k and to select test queries.

Experimental Results on Artificial Data Sets

We observed the performance of models in Table 5.3 where the CBR system uses 1-NN retrieval and in Table 5.4 where the CBR system uses SN retrieval. We recorded both the error rate and standard deviation of each model. As a reminder, due to technical constraint, each model is tested with 50 samples under each novel setting. Additionally, the standard deviation of all models are extremely high in novel settings. This renders all comparison in novel setting statistically insignificant, but we still believe the average error rates show a trend for comparing the models.

Many of the observations mesh with those in Section 5.4.3. We also observed additional phenomena in the experiments with the artificial data sets:

1. To our surprise, the baseline network performs relatively well (and sometimes best) in data sets with high locality. This contradicts our hypothesis.

2. Rule-based CDH always downgrades the retrieval result. Rule-based CDH finds the best case pairs to apply using the naive 1-NN, which works poorly in this special domain where only two features matter.

3. All models perform better in low-dimension data sets than high-dimension ones. 1-NN performs well in low-dimension spaces. The NN-CDH further improves the result of 1-NN retrieval, and often achieves the best performance. However, as the dimensionality increases, only two features are relevant and 1-NN retrieves worse results, making adaptation harder for NN-CDH. We observe when $k = 7$, the neural network outperforms 1-NN retrieval and

	Setting	Neural Network	k-NN	1-NN Retrieval	Rule CDH	NN-CDH
3 features	normal	17.47%(5.37)	9.677%(1.55)	9.818%(2.81)	12.31%(2.42)	9.906%(3.35)
remove on X	novel 0.1	17.23%(16.4)	14.86%(15.3)	18.57%(19.6)	24.90%(27.3)	9.148%(12.0)
	novel 0.2	19.90%(15.8)	19.09%(19.9)	23.94%(21.4)	27.62%(25.1)	11.41%(13.3)
	novel 0.3	22.42%(16.0)	21.00%(17.5)	24.40%(22.3)	30.25%(26.3)	15.41%(15.7)
5 features	normal	12.41%(1.66)	16.87%(2.13)	20.68%(1.98)	25.92%(2.12)	11.82%(2.82)
remove on X	novel 0.1	11.39%(9.71)	17.76%(14.7)	19.28%(22.6)	26.61%(23.4)	9.961%(9.01)
	novel 0.2	12.68%(11.6)	22.81%(15.7)	24.98%(23.7)	31.35%(23.9)	12.61%(11.8)
	novel 0.3	13.08%(11.9)	21.13%(18.7)	24.36%(22.4)	35.70%(27.8)	12.89%(11.7)
7 features	normal	24.42%(2.69)	25.70%(1.87)	32.08%(2.09)	39.37%(3.18)	25.49%(2.58)
remove on X	novel 0.1	20.02%(13.6)	25.53%(15.8)	34.56%(22.7)	37.55%(29.3)	23.15%(14.5)
	novel 0.2	21.38%(12.2)	25.70%(19.1)	33.92%(23.3)	40.67%(26.6)	23.84%(20.0)
	novel 0.3	22.98%(12.2)	28.35%(16.1)	31.02%(20.7)	36.22%(22.6)	21.82%(15.4)
3 features	normal	17.47%(5.37)	9.677%(1.55)	9.818%(2.81)	12.31%(2.42)	9.906%(3.35)
remove on Y	novel 0.1	15.59%(17.6)	13.77%(19.8)	15.27%(22.6)	14.24%(23.7)	10.63%(17.3)
	novel 0.2	20.65%(18.0)	17.97%(22.5)	21.64%(23.8)	21.19%(25.4)	16.74%(21.8)
	novel 0.3	22.33%(19.2)	24.68%(21.8)	26.23%(23.2)	25.05%(24.9)	19.62%(22.0)
5 features	normal	12.41%(1.66)	16.87%(2.13)	20.68%(1.98)	25.92%(2.12)	11.82%(2.82)
remove on Y	novel 0.1	12.11%(12.3)	16.81%(14.9)	18.15%(19.6)	21.64%(23.1)	11.83%(11.8)
	novel 0.2	18.73%(17.3)	24.74%(18.0)	28.06%(21.5)	26.95%(20.2)	20.88%(18.5)
	novel 0.3	23.25%(20.7)	31.04%(19.4)	33.68%(19.6)	34.40%(19.2)	21.99%(19.2)
7 features	normal	24.42%(2.69)	25.70%(1.87)	32.08%(2.09)	39.37%(3.18)	25.49%(2.58)
remove on Y	novel 0.1	29.76%(14.4)	29.85%(19.4)	33.85%(22.2)	37.84%(21.4)	31.37%(20.5)
	novel 0.2	32.92%(17.9)	35.60%(19.3)	38.07%(20.4)	38.89%(22.9)	39.12%(20.7)
	novel 0.3	35.73%(15.9)	39.03%(17.8)	40.11%(18.6)	41.49%(21.0)	39.42%(19.8)

Table 5.3: Error rates (and standard deviation) of Models on the Artificial Regression Data Sets. CBR uses 1-NN retrieval and adaptation is based on the retrieval result.

NN-CDH consistently. This trend is not as obvious when the retrieval is based on the siamese network.

4. SN retrieval can consistently outperform other models, except under novel settings (Y). Under a novel setting (Y), even the best possible neighbor's solution will be different from the target solution, as all cases with close solutions are removed. In such situations, we see the neural network and NN-CDH can outperform SN retrieval.

5. The artificial data set can be perfectly solved if a symbolic system somehow learns the rule to

	Setting	Neural Network	k-NN	SN Retrieval	Rule CDH	NN-CDH
3 features	normal	16.93%(4.94)	9.677%(1.55)	4.338%(1.09)	15.28%(4.23)	9.087%(3.05)
remove on X	novel 0.1	16.83%(16.5)	14.86%(15.3)	8.870%(15.4)	21.35%(29.8)	8.562%(10.9)
	novel 0.2	17.36%(14.3)	19.09%(19.9)	11.36%(15.8)	32.46%(27.5)	13.35%(12.3)
	novel 0.3	22.95%(14.9)	21.00%(17.5)	16.28%(18.0)	36.49%(25.8)	19.17%(18.0)
5 features	normal	13.01%(2.44)	16.87%(2.13)	7.322%(1.72)	25.59%(2.94)	12.51%(2.68)
remove on X	novel 0.1	11.03%(10.6)	17.76%(14.7)	6.017%(10.1)	28.25%(22.7)	9.115%(8.58)
	novel 0.2	11.38%(11.3)	22.81%(15.7)	6.954%(9.07)	25.10%(21.2)	12.42%(10.4)
	novel 0.3	12.72%(10.9)	21.13%(18.7)	9.769%(13.1)	22.93%(21.4)	12.48%(11.0)
7 features	normal	21.16%(2.38)	25.70%(1.87)	17.92%(4.98)	36.13%(3.47)	23.44%(2.90)
remove on X	novel 0.1	21.60%(13.3)	25.53%(15.8)	19.76%(19.8)	32.57%(25.6)	24.73%(19.0)
	novel 0.2	22.37%(13.0)	25.70%(19.1)	22.13%(20.3)	40.46%(33.8)	25.75%(17.9)
	novel 0.3	25.39%(10.8)	28.35%(16.1)	23.00%(20.6)	38.59%(27.1)	27.47%(18.3)
3 features	normal	16.93%(4.94)	9.677%(1.55)	4.338%(1.09)	15.28%(4.23)	9.087%(3.05)
remove on Y	novel 0.1	15.02%(18.3)	13.77%(19.8)	10.46%(13.9)	13.31%(16.6)	9.129%(11.6)
	novel 0.2	22.83%(19.4)	17.97%(22.5)	18.77%(21.5)	25.64%(24.3)	16.33%(24.6)
	novel 0.3	21.29%(20.2)	24.68%(21.8)	22.06%(18.3)	28.12%(25.8)	21.04%(23.8)
5 features	normal	13.01%(2.44)	16.87%(2.13)	7.322%(1.72)	25.59%(2.94)	12.51%(2.68)
remove on Y	novel 0.1	10.88%(10.4)	16.81%(14.9)	6.869%(8.68)	21.01%(20.2)	9.752%(9.32)
	novel 0.2	19.78%(17.1)	24.74%(18.0)	17.66%(19.0)	33.89%(26.6)	20.01%(16.9)
	novel 0.3	22.53%(20.2)	31.04%(19.4)	23.35%(20.4)	34.45%(24.1)	24.40%(22.3)
7 features	normal	21.16%(2.38)	25.70%(1.87)	17.92%(4.98)	36.13%(3.47)	23.44%(2.90)
remove on Y	novel 0.1	29.28%(15.8)	29.85%(19.4)	27.28%(22.3)	35.69%(22.4)	33.14%(21.8)
	novel 0.2	34.26%(17.3)	35.60%(19.3)	36.82%(21.0)	46.45%(27.8)	40.74%(22.9)
	novel 0.3	36.16%(15.5)	39.03%(17.8)	40.23%(20.4)	41.37%(22.8)	41.06%(20.7)

Table 5.4: Error rates (and standard deviation) of Models on the Artificial Regression Data Sets. CBR uses SN retrieval and adaptation is based on the retrieval result.

1) find the first integer i such that $x_1 \leq i/k$; and 2) Calculate the value of y as $y = w_i * x_i + b_i$. If $w_i = 1$ and $b_i = 0$ for all is, the data set becomes straightforward for human intuition. However, none of the systems tested achieve a similar level of proficiency in normal or novel settings.

5.5 Conclusion

This chapter proposed a way to express the difference between nominal values in a vector, and extended NN-CDH to carry out case adaptation for both classification and regression domains.

Experiments with both real and artificial data sets align with results from previous studies of NN-CDH. In general, NN-CDH achieves comparable performance to its counterpart, the traditional neural network. Moreover, SN retrieval may outperform both neural network and NN-CDH, and NN-CDH can worsen the result from SN retrieval. This illustrates the need to harmonize retrieval and adaptation methods (Leake and Ye, 2021a).

In extensive experimental results, integrating neural network methods into CBR did not give CBR power beyond the neural network, even on data designed to test a hypothesized advantage for CBR. The artificial data set, which can be perfectly solved using simple rules, illustrates a situation where neither the neural network or knowledge-light CBR learns the underlying abstraction well. On the other hand, CBR generally achieves performance comparable to the network, making it a competitive option, especially in tasks for which the intrinsic interpretability of CBR would make it preferable to a network approach.

Symbolic CBR provides the opportunity to integrate knowledge and task-optimized representations. We envision that a balanced blend of domain knowledge, brought to bear by symbolic AI methods, with network-learned methods, has the potential to achieve a performance edge. We consider the study of such integration to be a key step in advancing the performance of case adaptation (Leake and Crandall, 2020).

Chapter 6

Post ML-driven CBR: Harmonizing Case Retrieval and Adaptation with Alternating Optimization

In Chapter 3, RISA uses hindsight of adaptation results to improve the similarity assessment. RISA aims to build a form of adaptation-guided retrieval (AGR), where the retrieval method retrieves source cases adaptable toward the target query. The other side of the coin is retrieval-guided adaptation, where the adaptation method can adapt retrievable cases. In fact, almost all adaptation methods are by default retrieval-guided adaptations, because adaptation by definition adapts retrieved cases. In Chapter 5, we explored the possibility of implementing both retrieval and adaptation using ML techniques. This chapter takes a step further: it proposes a training scheme that guides both the retrieval method with the system's adaptation ability and the adaptation method with the retrieved cases. The end goal is that both retrieval and adaptation are aware of each other and work in harmony.

Case-based reasoning (CBR) research has developed numerous methods for learning to improve case retrieval and adaptation knowledge. Learning for each type of knowledge is usually pursued independently. However, it is well known that the knowledge containers of CBR are tightly coupled, in that changes in one can affect requirements for another, which suggests potential benefit for coupling learning across knowledge containers. This paper proposes applying alternative optimization to learn retrieval and adaptation knowledge together, in order to harmonize their behaviors. For a testbed system using neural network based similarity and adaptation, this study compares alternative optimization, independent learning, and learning by prioritizing adaptation for adaptation-guided retrieval. Results support that alternative optimization can help to balance both components and achieve good performance.

CBR systems solve a new problem by retrieving a similar prior case and applying its solution to the new situation, potentially adapting the solution to address differences between the old and new problem situations (e.g., López de Mántaras et al. (2005)). Much CBR research has focused on learning to improve case retrieval, often through refinement of case indices and similarity criteria. Another current of research has focused on learning to improve case adaptation. Recently, considerable interest has emerged in the use of network models to learn case retrieval and adaptation.

It is natural to expect that strengthening any component of a CBR system will strengthen the system as a whole. This assumption implicitly underlies research that focuses on learning for a single knowledge container such as similarity knowledge or adaptation knowledge. However, it is well known that similarity/retrieval and case adaptation knowledge containers are intimately connected: Smyth and Keane (1998) showed that an adaptation-guided retrieval (AGR) approach to case-based planning, which bases retrieval on adaptability, can increase efficiency of plan generation. Richter (1998) observed that strengths in one form of knowledge can compensate for weaknesses in another; a consequence is that it may be valuable to focus learning in one form of knowledge to address gaps in another (*e.g.*, responding to gaps in the case base by learning the adaptation knowledge most useful to alleviate the gaps). Uncoordinated learning may even be harmful: Leake et al. (1997) present a study in which uncoordinated case and adaptation learning degrades system efficiency, but overall efficiency is improved when case and adaptation knowledge are coordinated by learning adaptation-based similarity.

Even if equivalent efficiency or solution quality can be achieved by learning either retrieval or adaptation knowledge, the type of knowledge learned can affect explainability. A system with very strong adaptation might be capable of successfully adapting very distant cases. However, if retrieved cases are to be used to explain system conclusions to a user (*e.g.*, Nugent and Cunningham (2005)), explanations based on the distant cases might be less compelling than those based on more intuitively relevant cases. Therefore it may be preferable to learn to retrieve closer cases rather than learning stronger adaptation, even if both provide equivalent solution quality. On the other hand, if the user's conception of similarity is not important, it may be appropriate to retrieve the most adaptable cases, no matter how dissimilar they appear to the user, to increase system efficiency. If retrieval and adaptation are trained independently, or in a fixed sequence of one before the other, the balance between the two cannot be optimized for such considerations.

As an alternative to learning retrieval and adaptation knowledge independently or successively in a fixed sequence, this paper proposes applying alternating optimization (AO) (Bezdek and Hathaway, 2002b). Use of AO coordinates learning of retrieval and adaptation knowledge to harmonize their behaviors according to criteria for a desired balance. This method is applicable to any CBR system in which gradient descent is used to train retrieval and adaptation, as for network-based retrieval and adaptation approaches.

The paper presents an application of the AO approach to a regression task, using network-based learning methods for both retrieval and adaptation. In the testbed system, retrieval is based on a Siamese network for similarity measure and adaptation is based on a neural network based case difference heuristic (NN-CDH) approach for adaptation learning (Leake et al., 2021b). Experiments compare prediction error for three training schemes. The first is independent training of the two stages. The second is training adaptation first and then training retrieval. As this training gives precedence to adaptation, it provides fully adaptation-guided retrieval (AGR), and will be referred to as AGR training. The third is alternating optimization. The schemes are tested for five data sets. Experimental results show that under independent training, the two components may be poorly balanced; for example, retrieval may be strong while adaptation may provide little benefit or sometimes may even worsen the initial solution provided by retrieval. Results also show that under AGR training, retrieval may learn to retrieve cases that are adaptable but that have distant solutions, decreasing explainability. AO generally decreases the incidence of undesirable behaviors. Because AO harmonizes the retrieval stage and adaptation stage, the CBR system retrieves cases similar to queries and requiring limited adaptation, resulting in good accuracy.

The paper first presents background on retrieval and adaptation learning and introduces alternating optimization. It then presents loss functions for retrieval and adaptation and an algorithm for training them together in the AO framework. It closes with an evaluation of a testbed system, guidelines for application of AO for CBR components, and future directions.

6.1 Background

Learning for Retrieval and Adaptation: An extensive body of CBR research has studied machine learning methods to improve retrieval and case adaptation. Retrieval learning is generally focused on adjusting feature weights to improve similarity assessment (see Wettschereck et al. (1997) for a survey). Many such methods can be seen as adjusting parameters to optimize accuracy. Recently, there has been much interest in network-based similarity learning by optimizing a loss function. For example, Martin et al. (2017) use a Siamese network to learn a similarity measure for retrieval; Mathisen et al. (2019) propose using an extended Siamese network, which is the basis of the retrieval method in the testbed system presented in this paper.

Likewise, substantial research has been done on learning to improve case adaptation (*e.g.*, Craw et al. (2001); D'Aquin et al. (2007); Hanney and Keane (1996); Jalali and Leake (2016); Leake et al. (1997); Leake and Ye (2019b); Leake et al. (2021b); McDonnell and Cunningham (2006); Minor et al. (2014); Shiu et al. (2001)). Again, network methods have prompted strong interest. Policastro et al. (2003) use an adaptive resonance theory neural network for retrieval and multi-layer perceptron for adaptation. Leake et al. (2021b), Liao et al. (2018), and Policastro et al. (2006) use neural networks to learn adaptation knowledge from pairs of cases, using methods based on the case difference heuristic approach (Hanney and Keane, 1996).

In existing work learning of retrieval and/or adaptation knowledge is coordinated in either of two ways. First, training of retrieval and adaptation may be *independent*, where one is trained without using knowledge of the other (Policastro et al., 2003; Craw et al., 2006a). Second, they may be trained *in-order*, with one trained (or pre-defined) before the other and the latter trained with the knowledge of the former (Petrovic et al., 2016).

Alternating Optimization: As defined by Bezdek and Hathaway (2002b), alternating optimization (AO) is an iterative procedure to minimize a scalar field $f : R^s \to R$, under certain assumptions, in the nonlinearly constrained optimization problem:

$$\min_{x \in R^s} \{f(x)\}, \text{ subject to constraint functions}$$
$$c_i(x) = 0, i = 1, ..., k; \text{ and}$$
$$c_i(x) \geq 0, i = k+1, ..., m.$$

AO partitions the parameters $x = (x_1, ..., x_s)$ into t non-overlapping subsets $(X_1, ..., X_t)$, with $X_i \in R^{p_i}$ for $i = 1, ..., t$ and $\sum_{i=1}^{t} p_i = s$. p_i is an integer that controls the number of parameters in subset X_i. AO minimizes $f(x)$ iteratively. Each iteration includes t time steps. At each time step i $(i = 1, ..., t)$, instead of attempting to optimize all parameters x of a model, AO minimizes f by optimizing X_i, while keeping all other subsets $X_j (j \neq i)$ fixed. After one iteration, all subsets X_i for $i = 1, ..., t$ are optimized exactly once. AO can carry out multiple iterations, until x converges and cannot be further optimized, or until a fixed limit of iterations is reached. Examples of alternating optimization algorithms are k-means clustering and the expectation-maximization algorithm.

6.2 Alternating Optimization of Retrieval and Adaptation

Case-based reasoning has four processing stages: retrieval, reuse (adaptation), revision and retention. If each stage is considered as operating based on a collection of parameters x_i, then in principle, all stages could be refined in an AO process over all parameters $x = (x_1, x_2, x_3, x_4)$. This paper focuses on optimizing the retrieval function R and adaptation function A. Let CB be the case base, q be a query from a query set Q, x_r be the parameters of the retrieval function R, x_a be the parameters for the adaptation function A, and f be a loss function for the CBR system as a whole (*e.g.*, error of the system solution for q). The retrieval function R retrieves case(s) r from the case base CB according to parameterization x_r (for example, x_r could determine a similarity measure).

$$r = R(q, CB, x_r), r \in CB$$

The adaptation function A produces a final solution a by adapting the solution of retrieved case r to solve the query q according to x_a (for example, for network-based adaptation, parameters could determine adaptation network weights).

$$a = A(q, r, x_a)$$

Then the problem of optimizing the CBR system is equivalent to finding x_r and x_a to minimize the loss f over Q. This can be formulated as

$$\arg\min_{x_a, x_r} \sum_{q \in Q} \{f(A(q, r, x_a), q)\}, \tag{6.1}$$

The proposed AO process trains the parameters for retrieval and adaptation in a back-and-forth manner. A training iteration involves two steps, first seeking x_r according to:

$$\arg\min_{x_r} \sum_{q \in Q} \{f(A(q, R(q, CB, x_r), x_a), q)\}, \text{ while } x_a \text{ is fixed,} \tag{6.2}$$

and then setting x_a according to:

$$\arg\min_{x_a} \sum_{q \in Q} \{f(A(q, r, x_a), q)\}, \text{ while } x_r \text{ is fixed.} \tag{6.3}$$

The optimization runs multiple iterations, alternating between steps applying (6.2) and (6.3), until x_r and x_a both converge (*e.g.*, there is no significant update within a preset number of iterations) or an iteration limit is reached.

Goals for performance of CBR systems can be finer-grained than overall criteria such as system accuracy. For example, a system designer could be concerned about the usefulness of cases provided by the retrieval function as explanations, about efficiency of retrieval, about the efficiency or explainability of adaptation given a retrieval, or about some combination. To reflect various goals, we refine (6.2) for optimizing retrieval as:

$$\underset{x_r}{\arg\min} \sum_{q \in Q} \{\alpha g(r, q) + (1 - \alpha) f(A(q, r, x_a), q)\}, r = R(q, CB, x_r) \text{ while } x_a \text{ is fixed,} \tag{6.4}$$

where f is the adaptation loss (*e.g.*, measuring the adaptability of the retrieved case), while g is the retrieval loss (*e.g.*, reflecting whether the case is a convincing explanation). In our testbed implementation, f calculates the error of the adapted solution for q, and g calculates the error of the retrieved solution for q.

The metaparameter α controls the emphasis between the two losses during the training of R. If $\alpha = 0$, then formula (6.4) is reduced to formula (6.2), where the retrieval loss is ignored and retrieval R is trained to retrieve the case that minimizes the adaptation loss. If $\alpha = 1$, then adaptation loss is ignored and retrieval R is trained to retrieve the case that minimizes the retrieval loss.

As shown in the evaluation, setting $\alpha = 0$ or $\alpha = 1$ may be problematic, leading to a form of codependence of the two stages. If the retrieval step converges rapidly and can retrieve very similar cases, little learning may be required from adaptation, potentially resulting in adaptation that struggles to handle novel cases. If the adaptation step converges rapidly and can adapt a wide range of differences, little learning may be required from retrieval, because even retrievals distant from the query result in accurate solutions. In that case, the retrievals may be less compelling as explanations (in the extreme, a system could retrieve the same case for all problems, for adaptation to generate the solution from scratch). To mitigate the codependence effect, we choose an α value so that formula (6.4) considers both retrieval loss and adaptation loss. Alternation between training retrieval and training adaptation also helps mitigate codependence.

6.3 Testbed System Design

To study the impact of alternating optimization on harmonizing retrieval and adaptation, we designed a testbed system for case-based regression, with optimization based on solution accuracy. We note that other loss functions would be possible. For example, previous research on adaptation-guided retrieval focused on adaptation efficiency (Smyth and Keane, 1998).

6.3.1 Loss Function

As loss function f, we use squared error of the final solutions (post-adaptation) compared to the actual solutions.

$$f(a, q) = (a - sol(q))^2. \tag{6.5}$$

Formula (6.3) becomes

$$\arg\min_{x_a} \sum_{q \in Q} \{(A(q, r, x_a) - sol(q))^2\}, \text{ while } x_r \text{ is fixed,} \tag{6.6}$$

As loss function g, we use squared error between the retrieved solutions and the actual solutions. This could be considered as a proxy for suitability as an explanation: If the error is already small, the retrieved case is a good candidate to explain the solution for the query. We note that in general the suitability as an explanation also depends on the user's criteria for problem similarity (which may differ from that of the system).

Retrieved cases within a set error threshold E are considered correct, with loss set to zero.

$$g(r, q) = max(0, (sol(r) - sol(q))^2 - E^2)$$

Formula (6.4) is thus expanded into

$$\arg\min_{x_r} \sum_{q \in Q} \{\alpha \cdot max(0, (sol(r) - sol(q))^2 - E^2) + (1 - \alpha)(A(q, r, x_a) - sol(q))^2\}. \tag{6.7}$$

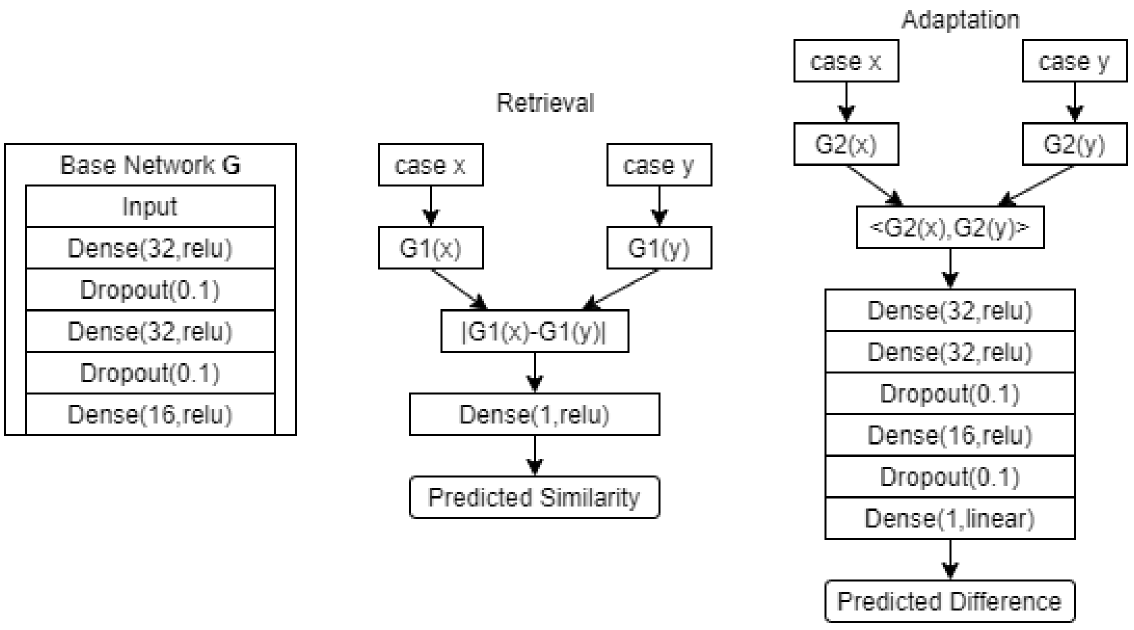


Figure 6.1: Network Architecture of the Testbed System. Left: Base network used to extract features. Middle: Extended Siamese network for similarity. Right: Extended Siamese network for solution difference prediction

6.3.2 Testbed Retrieval and Adaptation

In the testbed system, the retrieval function R performs 1-nearest neighbor retrieval. The similarity measure is learned using a Siamese network following the example of eSNN (Mathisen et al., 2019). This network involves a base network (which extracts features from input cases) and an element-wise distance layer (which subtracts the two features), followed by a fully connected layer for final output. It is trained by backpropagation using formula (6.7) as the loss function, where the parameter x_r represents the weights and biases of the network.

The adaptation function A is an NN-CDH, as discussed in Section 6.1. It involves a Siamese base network (not related to the Siamese network used in retrieval) and an adaptation network. The Siamese base network is trained to extract feature vectors from the input query q and retrieved case r, and the adaptation network takes as input both feature vectors and outputs a predicted difference d of the solutions of the two cases. The predicted difference is added to the retrieved solution to produce the final solution. The NN-CDH adaptation network is trained through back propagation using formula (6.6) as the loss function, where the parameter x_a contains the weights and biases of the NN-CDH. The networks used in retrieval and adaptation are illustrated in Figure 6.1.

The retrieval function and adaptation functions were chosen based on several factors: They can be independent of or dependent on each other, allowing comparison between the paradigms; they can be trained using formula (6.3) and formula (6.4); and they are both powerful enough to solve majority of the queries in the experiment datasets.

6.3.3 Testbed Training and Testing Procedures

A general training process for retrieval and adaptation, applied in the testbed system, is described in Algorithm 5. Given a query q, the retrieval stage collects a batch of case pairs by retrieving neighboring cases from CB. Retrieval is based on the similarity measure determined by x_r. The batch is then used to train x_a and x_r. Afterwards, retrieval collects a new batch based on the updated x_r and the batch is used for the next iteration of training. This process repeats until either a predefined iteration limit is reached or the parameters x_a and x_r converge. The steps to update x_a and x_r, lines 17 and 18, are only described at a high level because these steps vary across specific adaptation and retrieval components. In our testbed system this training is done by backpropagation to minimize loss functions (6.6) and (6.7). The system alternates between training retrieval and training adaptation across batches, with respective loss functions enabling consideration beyond overall performance, for example, explainability or adaptation efficiency.

Parameters in the algorithm enable fine tuning. The parameter *num_neighbor* controls the number of neighboring cases to retrieve for a query. The retrieval and adaptation stages train from pairs of a query and its neighbor(s). In principle, pairs of random cases can be used to provide more comprehensive training samples (Jalali and Leake, 2016; Leake and Ye, 2019b), but this is not used here. Lines 17 and 18 could be exchanged depending on the choice of whether to train retrieval or adaptation first.

AO training is a general version of the traditional training scheme used to learn retrieval or adaptation in a CBR system. If *batch_size* is set to the size of CB in line 6, then retrieval and adaptation are only trained once. We tested two training schemes for comparison with AO: First, when $\alpha = 1$ and $batch\_size = size(CB)$, the retrieval and adaptation stages are trained independently of each other (**independent training**). Second, when $\alpha = 0$ and $batch\_size = size(CB)$, adaptation is trained first and then the retrieval is trained solely based on adaptation loss (**adaptation-guided retrieval (AGR) training**).

For independent training or AGR training, x_r and x_a are trained until convergence in lines 17 and 18. However, AO uses parameter *R_epochs* and *A_epochs* to respectively train x_r and x_a only for a set number of epochs before alternating. Forcing more frequent alternation is intended to prevent one parameter from converging too fast, reducing risk of a local minimum.

In this study, we do not explicitly implement a scheme for retrieval-guided adaptation (the counterpart of AGR). In Algorithm 5, a query is paired with a neighboring case found through the *Retrieval* procedure, and the case pair is in turn used for training both retrieval and adaptation. However, this is not strictly retrieval-guided adaptation as pairs are retrieved before x_r is trained, so adaptation is trained with pairs assembled from an untrained retrieval.

Algorithm 6 illustrates the testing mode for the system, which applies trained retrieval followed by trained adaptation.

6.4 Evaluation

This section evaluates the performance of AO in comparison to independent training and AGR training in terms of quality of retrieved solution, quality of adapted solution, and incremental benefit from adaptation.

Algorithm 5 Training for Retrieval and Adaptation

```
1:  procedure R($q, CB, num\_neighbor, x_r$)
2:      $r \leftarrow \{\}$
3:      sort $CB$ by similarity to $q$, using similarity measure determined by $x_r$
4:      $r$.append($num\_neighbor$ top neighbors of $q$ from $CB$)
5:      return $r$
6:  procedure AO_Train($x_r, x_a, batch\_size, CB, R\_epochs, A\_epochs, R\_loss, A\_loss$)
7:      $pairs \leftarrow \{\}$
8:      $batch\_counter \leftarrow 0$
9:      for each $q_i$ in $CB$ do
10:         $remaining\_cases \leftarrow CB$ not including $q_i$
11:         $r \leftarrow$ R($q_i, remaining\_cases, num\_neighbor, x_r$)
12:         for each $r_j$ in $r$ do
13:             $pairs$.append($< q_i, r_j >$)
14:         $batch\_counter \leftarrow batch\_counter + 1$
15:         if $batch\_counter = batch\_size$ then
16:             $batch\_counter \leftarrow 0$
17:             Train $x_a$ with $pairs$ using $A\_loss$ for $A\_epochs$
18:             Train $x_r$ with $pairs$ using $R\_loss$ for $R\_epochs$
```

Algorithm 6 Using the CBR System in Testing Mode

```
1:  procedure Solve($q, CB, x_r, x_a$)
2:      $r \leftarrow$ R($q, CB, 1, x_r$)
3:      Return $A(q, r, x_a)$
```

6.4.1 Experimental Settings

This study carries out experiments on five regression data sets. Four of these data sets, Energy Efficiency (EE), Yacht Hydrodynamics (YH), Student Performance in Math (SP), and Airfoil Self-noise (AS) are from UCI machine learning repository (Dua and Graff, 2017). EE, YH, and AS are datasets of physics phenomena, and SP is a dataset about predicting students' math grades with social and economical attributes. The fifth data set is an artificial data set (Ar). Ar has 4 attributes $\{x_1, x_2, x_3, x_4\}$, each in $[0, 5]$, and a target value y set by:

$$y = \sum_{i=1}^{4} 3 - |x_i - 3|$$

This is a simple data set for which neighboring cases have similar solutions and adaptation can be done with two simple adaptation rules:

$$\Delta y = \begin{cases} \Delta x_i & \text{if } x_i \leq 3 \\ -\Delta x_i, & \text{otherwise} \end{cases}$$

The testbed system is trained by Algorithm 5 and tested by Algorithm 6. For replicability, the source code and parameter settings are available online.[1]

For experimental runs, each data set is split into a training set (90%) and a test set (10%). The testbed system is trained under three different training schemes, and then tested on the test set. Under independent and AGR training, 10% of the pairs collected are used for validation. Under AO, the training happens per batch and 10% of the pairs in a batch is often too small for validation. For this reason, under AO, 10% of the cases in the training set are separated and used as validation cases. Each validation case *val_case* forms a validation pair with the case base by using the function Retrieve($val\_case, CB, 1$). Note that with this experimental design, the three schemes use same number of case pairs in training and in validation. They are also tested on the same number of cases during testing.

[1]https://github.com/Heuzi/AOtrainingCBR

In all experiments, we used $num\_neighbor = 1$ in all the retrievals. We train the test-bed system using three training schemes: Independent training, AGR, and AO. Depending on the performance of independent and AGR training, we fine tune E and α to balance the emphasis between retrieval and adaptation in AO. These detailed parameter settings are omitted because they vary across the data sets but can be found in the source code. Instead, Section 6.5 provides general guidelines on how to set the parameters. We evaluate the testbed system performance by mean square prediction error.

We note that the training procedures may not be optimal for each individual training scheme, but this approach ensures fairness between training schemes, making it suitable for comparing their effects.

6.4.2 Experimental Results

All experimental results are shown in Table 6.1. Each row represents the results over one data set. The histograms provide comparative information about accuracy with retrieval alone, with retrieval followed by trained adaptation, and the accuracy difference between the two, under the three training schemes

The X axis of each histogram reflects a level of error. Between rows, different scales are used for different data sets, and outliers are included as the left- and right-most buckets. In the first two columns, bars in each histogram reflect the number of trials of a given method for which the error was at a given level. Consequently, higher bars to the left of the histograms reflect better performance. In the third column, bars represent the number of times the difference between error after adaptation and error of the solution of the retrieved case was at a particular level, i.e., how adaptation changes error. The bars for independent training are black, for AGR are light gray, and for AO are dark gray.

Ideally, retrieval will retrieve a similar case and adaptation will improve the solution. This is

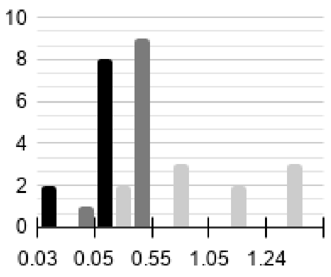
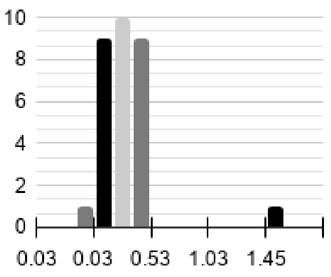
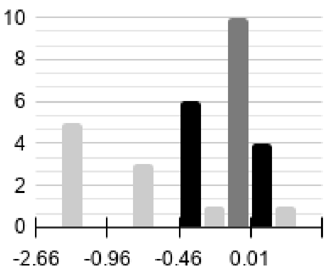
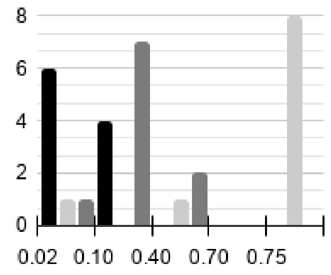
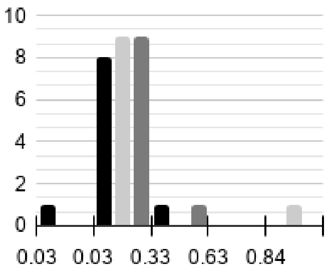
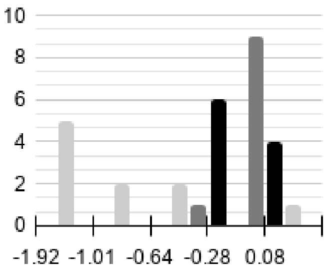
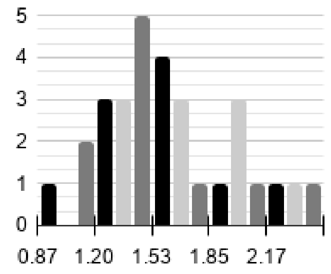
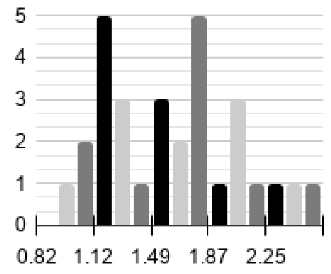
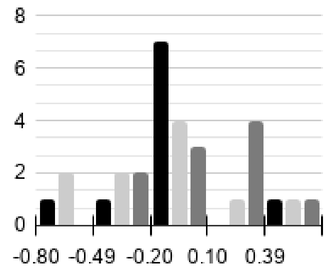
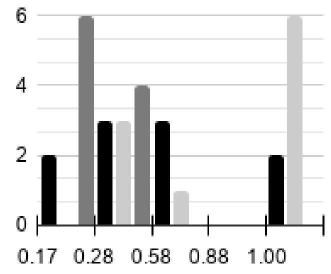
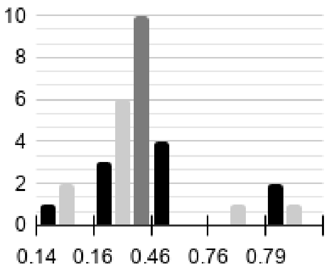
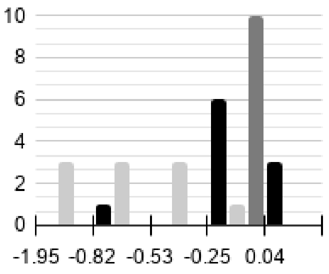
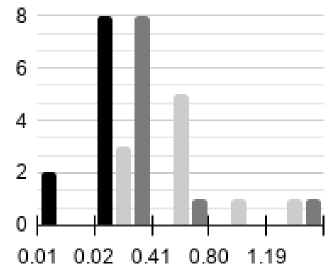
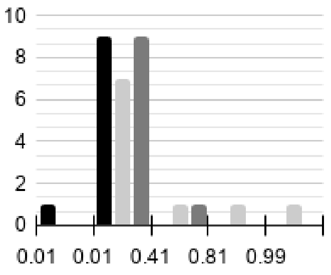
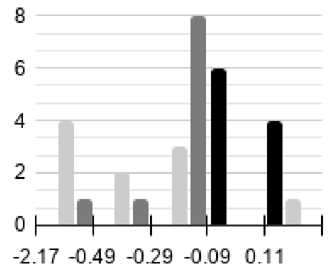
Data	Retrieval Only	After Adaptation	Δ by Adaptation
EE			
YH			
SP			
AS			
Ar			

Table 6.1: Black: Independent training; Light gray: AGR training; Dark gray: AO

	EE	YH	SP	AS	Ar
Independent	0.291	0.172	0.123	1.58	0.693
AGR	0.149	0.226	0.306	1.74	0.323
AO	0.106	0.164	0.158	1.61	0.209

Table 6.2: Average Error by Systems under Different Training Schemes

shown for all runs of AO for EE and AS, for the overwhelming majority of runs for YH, and for the majority of runs for SP and Ar. However, independent training and AGR training show less ideal effects as follows:

A **Independent training scheme (black bars)**: Here adaptation does not necessarily improve the results of retrieval. The retrieval stage may retrieve a case with a solution close to the real solution that the adaptation stage changes for the worse. This can be observed for the majority of runs with EE, YH, Ar, and SP.

B **AGR training scheme (light gray bars)**: Here the retrieval stage generally provides poor initial solutions while the adaptation stage makes substantial corrections (*i.e.*, results in a large negative change to the error). This can be observed for all data sets.

Even when adaptation can generate a successful solution, if moderate adaptation from the retrieved solution is important for explanation, the large adaptations in this condition are undesirable. There are also extreme scenarios in which retrieval is so poor that even a strong adaptation stage generates bad solutions, as seen in experiments with YH, SP, AS, and Ar.

In comparison, under AO, the retrieval stage generally provides a good initial case, and adaptation further modifies the solution to be closer to the correct solution. In all experiments except with SP, AO shows this reliable behavior while the other two training schemes suffer codependence effects to some degree. As shown in Table 6.2, the average errors under AO are often better than those of other schemes.

AO decreases the codependence effects between retrieval and adaptation according to parameter settings in Algorithm 5. To alleviate effect (A), training is done on batches of training samples, instead of all the samples; Lines 17 and 18 train for a set number of epochs/steps instead of until convergence, so that neither retrieval nor adaptation fully assumes the accuracy burden before the other is trained. To alleviate effect (B), formula (6.2) is expanded into formula (6.7), and fine-tuning α and E enables more control over how each stage is trained.

6.5 Guidelines for Applying AO to Train CBR Components

The experiments illustrate the benefit of AO in the testbed system. This section provides general questions and guidelines for applying AO to other CBR systems.

- **When can AO be applied?**

  AO is applicable if both retrieval and adaptation have training procedures which iteratively minimize loss functions. If loss functions can be formulated for formula (6.3) and formula (6.4), then AO can be used to harmonize training.

- **When may AO be useful?**

  AO may be useful in circumstances such as (1) when the effective scope of adaptations is limited, so that the effectiveness of adaptation depends strongly on the cases from which adaptation starts (in contrast, if adaptation can succeed from a wide range of starting points, AO would be expected to be less useful), (2) when successive training might lead to a local minimum in overall loss (*e.g.*, error or adaptation cost), but not a global minimum, (3) when it is desirable to control the balance between contributions of retrieval and adaptation, *e.g.*, to increase explainability by retrieving cases requiring only small adaptations.

- **How can a CBR system be trained with AO?**

  Applying Algorithm 5 requires three steps:

1. Design the loss functions f and g for formula (6.3) and formula (6.4). The adaptation loss f reflects quality of the adapted solution and/or adaptation process (*e.g.*, solution error, adaptation cost, or a combination), and the retrieval loss g reflects quality of the retrieved case (*e.g.*, accuracy of the retrieved case solution without adaptation, or explainability factors such as similarity between the problems of the query and retrieved case and closeness of the retrieved solution to the system solution). The loss functions should be compatible with training processes for the retrieval and adaptation stages, such that gradient descent can guide training.
2. Choose α and E. This can be done by training retrieval and adaptation independently and noting losses induced by g and f. The choice of α may emphasize training the process with more room for improvement or be based on other criteria.
3. Determine *batch_size* and the training steps (lines 17 and 18). These design choices influence how fast x_r and x_a converge. For a goal of balanced training, these should be tuned so that retrieval and adaptation converge at a similar rate.

- **Does AO guarantee an optimal solution?**

  No. As detailed in Bezdek and Hathaway (2002a), the convergence of AO relies on the assumption that each subproblem will converge. In our study, this means that AO will converge if both training of retrieval and adaptation can converge. Additionally, AO may converge to a local minimum or saddle points instead of a global minimum. However, these issues are beyond the scope of this study.

6.6 Future Work

AO and Explainability: In this study we used the error of the retrieved solution as a proxy for the explainability of the system solution. Other factors could affect explainability, such as problem similarity according to user criteria (which might differ from system criteria) or the adaptation distance. Such criteria and the benefit of AO for their optimization would be interesting to investigate.

Alternative tasks and loss functions: It would be interesting to examine the comparative value of AO for tasks beyond regression and for other loss functions (*e.g.*, efficiency, solution execution cost). A challenge for a broader class of tasks is the development of appropriate loss functions and training procedures.

Extending AO to the Full CBR Cycle: The work in this paper explores AO for retrieval and adaptation. However, whether to retain a new case in the case base depends strongly on, and also influences, both retrieval and adaptation capabilities. Consequently, an interesting direction is an extended version of AO to harmonize retention as well.

For example, retention training could optimize to retain cases for which changing adaptation capabilities are most accurate (or most efficient), or could adjust indexing as the similarity criteria for retrieval change. If a suitable loss function and training process could be defined, this could be done within the AO framework; if not, adjustments of retention could still be interleaved with AO for retrieval and adaptation. This raises questions of processing cost and how to determine when the additional processing would be worthwhile.

6.7 Conclusion

Extensive research has examined learning methods for improving particular CBR knowledge containers such as retrieval and adaptation knowledge. However, it is well known that the knowledge containers of CBR are closely connected and overlapping. More knowledge in one sometimes compensates for less in another (Richter, 1998) and coordination between containers can improve overall performance (Leake et al., 1997; Smyth and Keane, 1998). The interaction between retrieval and adaptation has primarily been addressed by treating one component as fixed and adjusting the other. In such scenarios, it is possible that the overall system will reach an undesirable balance between retrieval and adaptation, or will reach a local performance maximum instead of a global one. By iteratively harmonizing both, alternating optimization can achieve a combination of good retrieval and good adaptation. In our experiments, independent training and AGR training suffer codependence effects in all five data sets while AO works well in four of the five. This work also suggests the potential to explore using AO to harmonize other processes of CBR.

Chapter 7

Using CBR Adaptation to Generate Creative Samples from Limited Data

7.1 Introduction

Methods in CBR have potential in applications beyond CBR. In Chapter 5, we explore CDH-inspired adaptation method. As CBR assumes that similar problems share similar solutions, CDH assumes that similar problem difference induces similar solution difference.

This assumption by CDH aligns with the class-to-class (C2C) methodology Ye (2018a) which studies the pattern of change from one class to another. C2C's assumption states that the similarity and difference of features from one class to another is consistent.

In a classification task domain, a sample can be considered as a case, by considering its features as a problem description and its class label as a solution description (The terms "sample" and "case" are interchangeable in this chapter). Therefore the assumptions of CDH and C2C are equivalent in this context. In this chapter, we demonstrate a sample generation method inspired by CDH-style adaptation. This sample generation method can generate creative samples even if the model is trained with limited data.

Generating novel items with desired characteristics requires creativity. One method to achieve this is through creative transformations. Deep learning network methods provide an interesting potential substrate for this task. this chapter presents a method for network-based generation of novel images by applying variational autoencoders (VAEs) to learn features, which are then perturbed based on a class-to-class (C2C) method for learning of inter-class similarity and difference information, enabling generating creative samples. Our method learns the pattern between classes, applies this pattern to samples of a source class, and generates new samples of a target class. This study also proposes a general approach to evaluating the creativity of sample generators for classification domains, by evaluating the samples generated by the generator trained in a one-shot setting. The evaluation approach requires only classification labels but not human assessments of creativity. An experiment in two image domains supports that the samples generated by our method satisfy two of Boden's creativity criteria: being valuable (falling into desired categories) and novel (samples show high variance), even in a one-shot setting where training samples of a target class are limited. The evaluation illustrates a general approach to evaluating the creativity of sample generators for classification domains in a one-shot setting.

Advances in machine learning have yielded many successful deep generative models (Pan et al., 2019). Such models generate samples conforming to the distribution of the training data, leading to samples that are "authentic" in the sense of substantially sharing the properties of real examples. A surge of research in computational creativity is applying generative deep learning methods while inducing novelty—and even surprise—for the sake of creativity, as described in the surveys of Franceschelli and Musolesi (2021) and Broad et al. (2021). For example, the creative adversarial network proposed by Elgammal et al. (2017) is a generative adversarial network (GAN) that generates artwork that is realistic but also deviates from style norms. Similar work has induced creativity in GANs by introducing additional goals (loss functions) beyond the original adversarial loss (e.g. StyleGAN (Karras et al., 2018)). Following StyleGan, Nobari et al. (2021) proposed a systematical method to modify GANs to automatically generate novel designs without human intervention. Generally, variational auto encoder (VAE) methods are less suited for creativity tasks because their reconstruction loss aims to mimic the data distribution within a learned latent space and it is difficult to reflect other goals in the corresponding loss function. However, these latent spaces can be manipulated to induce creative results (e.g. MusicVAE (Roberts et al., 2018b) and sketchRNN (Ha and Eck, 2017)).

Characterizing the creativity of AI systems requires criteria for assessing the creativity of a process or of a system's results within a task context. Developing such criteria is nontrivial and has received considerable attention (Wiggins, 2021). Boden (1991) provides three criteria for assesssing the creativity of outputs of a process: value, novelty, and surprise. Many researchers have continued this school of thought, refining and expanding on these criteria (Wiggins, 2006; Draper, 2010).

This chapter addresses creativity as it applies to generating new samples for a target class when training samples are limited. We consider a sample (generated or not) to be *valuable* if it fits in the target class, and *novel* if it is different from the observed samples of the target class. According to Boden, surprise can happen when the sample is unexpected (which requires a prior expectation entity). We do not consider surprise when evaluating our model.

This chapter makes two contributions. First, we present an algorithm for generating creative samples in a classification domain. The algorithm uses a method we call a *class-to-class variational autoencoder* (C2C-VAE), which learns a latent space of the difference patterns between samples of all classes. The C2C-VAE then samples new differences from this latent space, and applies the difference to existing samples in the original conceptual space to generate new samples. Second, we address the general question of how to evaluate the creativity of sample generators for classification in a one-shot setting. We propose an approach for evaluating creativity of sample generators, which we call GOF/TOM — "Generated On Few, Tested On Many." A generator is trained in a zero, one, or few-shot setting where samples of a target class are trimmed from the training set. The generator is then used to generate new samples of the target class. Meanwhile, an oracle is trained on the *untrimmed* dataset to evaluate the generated samples. Because the generator has limited examples of the target class, its ability to generate satisfactory unseen samples of the target class can be used as measure for its creativity. This approach is used to evaluate the C2C-VAE.

We begin by discussing related techniques for generating and measuring computational creativity. We then present our C2C-VAE approach for generating creative samples, and introduce our GOF/TOM approach for evaluating creativity. Finally, we evaluate C2C-VAE on two data sets, MNIST (LeCun and Cortes, 2010) and Fashion-MNIST (Xiao et al., 2017), using the GOF/TOM approach. In the two data sets, C2C-VAE successfully generates samples that are valuable and novel with respect to its training data, making a case for the potential of C2C-VAE as a creative approach. We examine the limitations of C2C-VAE and propose methods for addressing them in future work.

7.2 Background

7.2.1 Active Divergence with Generative Deep Learning

In her seminal work, Boden (1991) identifies three forms of creativity: combinatorial, exploratory and transformational. Combinatorial creativity generates new ideas by combining old ones. Exploratory and transformational creativity both involve a conceptual space, where the former explores the conceptual space while the later alters it, potentially causing a paradigm shift (Wiggins, 2006; Franceschelli and Musolesi, 2021).

Franceschelli and Musolesi ((Franceschelli and Musolesi, 2021)) consider VAEs and GANs to perform exploratory creativity, as they both sample from a conceptual space. GANs can be also be transformational. As an example, in CANs (Creative Adversarial Networks), the discriminator determines both whether a sample image is art or not and its artistic style, while the generator tries to generate art and also generates deviations from original style norms. GANs can even be combinatorial. For example, StyleGAN can achieve style mixing by combinig the latent codes of two samples at multiple different levels of detail.

A CycleGAN is a image-to-image translation technique that can translate an image of one class into an image of another, e.g. modifying the image of a horse h into that of a zebra z using a translation function Z (or conversely from a zebra into a horse, using a function H). A CycleGAN is trained with two loss functions: 1) An adversarial loss trains the generators Z and H to generate quality images (so that a horse h can be translated into a realistic zebra $Z(h)$); 2) A cycle-consistency loss ensures the transition can go back-and-forth (so that the horse-translated zerba $Z(h)$ can be translated back to a horse $H(Z(h))$ similar to the original horse h). The artist Helena Sarin uses CycleGAN to generate creativity-related artwork (NVIDIA, 2021).

Our proposed model is based on variational autoencoders (VAEs) (Kingma and Welling, 2013). A VAE is comprised of an encoder and a decoder, both implemented as neural networks. The encoder takes samples as inputs and compresses them into a Gaussian distribution of lower-dimension embedding vectors in a latent space. The decoder takes an embedding vector and recovers the original input sample. Regularity—the property of similar samples having similar representations—is encouraged in the latent space because a sample is encoded as a distribution of embeddings, instead of a single embedding as in autoencoders (the forerunners of VAEs).

Features extracted by VAE can be manipulated by perturbation and even vector arithmetic for creative results. For example, MusicVAE (Roberts et al., 2018b) has the ability to "adjust the number of notes in a melody by adding/subtracting a note density vector to/from the latent code" (Roberts et al., 2018a). Similarly, sketchRNN can "subtract the latent vector of an encoded pig head from the latent vector of a full pig, to arrive at a vector that represents a body. Adding this difference to the latent vector of a cat head results in a full cat (i.e. cat head + body = full cat)" (Ha and Eck, 2017). The effects of such modification over VAE embeddings are not guaranteed and only partially understood. As noted by Ha and Eck (2017), such analogy is only possible when the embedding distribution is smooth and any interpolation between two embeddings is coherent. This study attempts to model the differences between pairs of embeddings extracted by VAE.

In the taxonomy of active divergence by Broad et al. (2021), this study proposes a method of chaining models. The method is a combination of a standard VAE with a secondary VAE (C2C-VAE) that explores the learned representation of feature differences.

7.2.2 Class-to-class Approach

Classification methods commonly consider the similarity of new instances to instances in a class. The Class-to-class (C2C) approach considers both similarity and difference. It assumes that there exist inter-class patterns between each pair of classes, and the samples from the two classes are consistently similar in some features and different in some other features. For example, zebras and horses have the similarity of both belonging to the Equidae family, and the difference that zebras have stripes while horses do not. The inter-class patterns, once learned, can be used to classify a query based on instances from another or multiple other classes (Ye, 2018a; Ye et al., 2020, 2021a).

We hypothesize that inter-class patterns can also be used in computational creativity. A system that learns inter-class difference patterns can intentionally apply the patterns to modify a sample. For example, knowing that zebras have stripes and horses do not, the system can modify a horse image by replacing its texture with black-and-white stripes and thus create a new zebra image.

The C2C approach is highly related to GAN methods. For example, CycleGAN is trained on unpaired image-to-image data from one class to another and can generate zebra images from horse images (Zhu et al., 2017). GAN methods are mostly end-to-end. For example, CycleGAN generates an output image from an input image, and the inter-class pattern is integrated into the procedure of the model and is applied automatically in the forward pass of the neural network; C2C methods work with the inter-class pattern directly. For example, the method to be presented in this study uses the feature differences between two samples as both inputs and expected outputs of an variational autoencoder. This difference provides more flexibility to introduce creativity. More specifically, our approach can choose an inter-class pattern as the modification and also choose a sample to apply this modification.

7.2.3 Measurement of Creativity

Franceschelli and Musolesi (2021) (2021) survey multiple creativity measures implemented via machine learning algorithms. Our method, GOF/TOM ("generated on few, tested on many"), fits within the formalization of the generate and test framework (Toivonen and Gross, 2015), in which the system uses a generative function to generate samples and an evaluation function to evaluate the samples. The authors describe three works (Varshney et al., 2013; Norton et al., 2010; Morris et al., 2012) that fit in this framework.

Varshney et al. (2013) proposes a system that generates creative recipes. The novelty of a recipe is evaluated based on Bayesian surprise, the difference between a prior probability distribution of recipe and a posterior probability distribution after a new recipe is observed. The value of a recipe is evaluated by a model predicting pleasantness of scent from its ingredients and flavor compounds in those ingredients.

(Ritchie, 2007) describes that creativity can come from an *inspiring* set, which is a set of usually highly valuable samples used to train or configure the generator. (Gervás, 2011) expands on this by splitting an inspiring set into a learning set, which informs the construction of the generator, and a *reference* set, which is used to evaluate the novelty of generated samples. Similarly, (Morris et al., 2012) uses an inspiring set (crockpot recipes) to generate samples via a genetic algorithm and to evaluate the quality of generated sample by training a multilayer perceptron to predict user ratings from a sample.

7.2.4 Creativity Inspired Zero-Shot Learning

The goal of a zero-shot learning task for classification is to train on seen classes and then predict the class label of a sample from an unseen class (or samples from seen and unseen classes in generalized zero-shot learning). (Elhoseiny and Elfeki, 2019) implemented a creativity inspired zero-shot learning algorithm. In that work, both visual and semantic information are available for seen classes but only semantic descriptions are available for unseen classes. The authors introduce a creativity inspired zero-shot learning method which trains a discriminator to differentiate between real and fake images and also classifies an image into seen classes. It also trains a generator to generate realistic images based on texts describing seen classes and realistic yet hard-to-classify (high entropy over seen classes) images from "hallucinated" texts. This training goal drives the generator to explore the latent space of texts with two objectives: 1) Generate samples that are realistic; 2) Generate samples from hallucinated texts. These properties enable the generator to generate realistic images based on the descriptions of unseen classes.

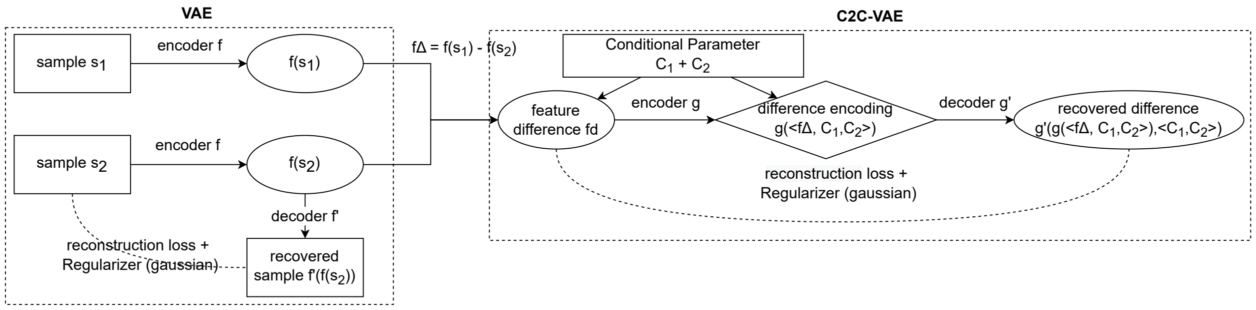


Figure 7.1: A VAE extracts (recovers) a feature from (to) a sample. A C2C-VAE extracts (recovers) an embedding from (to) a feature difference. $L1$ entities are marked with rectangles, $L2$ and $L2'$ entities with circles, and $L3$ entities with diamonds.

7.3 Creative Sample Generation with a Class-to-class Variational Autoencoder

This chapter proposes a class-to-class variational autoencoder (C2C-VAE) approach to generating creative samples in the context of classification—that is, generating creative samples falling within desired categories (e.g., generating images for creative versions of a given letter of the alphabet). For this task, the C2C-VAE approach learns the difference pattern between pairs of samples of two different classes. Four spaces are involved in this task: the original sample space $L1$, the feature space $L2$, the space of feature differences $L2'$, and the space of feature difference embeddings $L3$. As a precondition, C2C-VAE relies on a means to transition between $L1$ and $L2$, more specifically, to extract a feature $f(s)$ from a sample s and also to recover a sample s from feature $f(s)$.

For our testbed system (illustrated in Figure 7.1), we train a traditional VAE on the training data and use the encoder f to extract features and the decoder f' to recover samples. Given a pair of samples from different classes, $s_1 \in C_1$ and $s_2 \in C_2$, a VAE can extract their features $f(s_1)$ and $f(s_2)$ in the space $L2$. The feature difference $f_\Delta(s_1, s_2)$ in the space $L2'$ can be calculated as $f_\Delta(s_1, s_2) = f(s_1) - f(s_2)$, an element-wise subtraction between two feature vectors. The space $L2'$ can be thought of as the complement of the space $L2$, whence the name $L2'$.

C2C-VAE is based on the class-to-class assumption that the feature differences (in space $L2'$) from one class to another class follow a consistent pattern. This pattern can be represented as an embedding vector (or a distribution of embeddings) in another latent space $L3$. The C2C-VAE is itself another variational autoencoder with an encoder g that encodes a feature difference f_Δ in $L2'$ to an embedding $g(f_\Delta)$ in $L3$ and a decoder g' that decodes an embedding $g(f_\Delta)$ back to a feature difference f_Δ. Note that both the VAE encoder f and the C2C-VAE encoder g are *variational encoders* that encode an input to a distribution of embeddings. This is to ensure regularity in the latent space $L2$ and $L3$. For simplicity, the encoder f (or g) can be thought of as extracting one feature (or a feature difference embedding) from an input.

Because there exist multiple pairs of classes and each pair $C_i - C_j$ has its own unique pattern, a C2C-VAE either learns only one pattern, reflecting a specific pair of classes $C_i - C_j$, or learns multiple patterns by conditioning its encoder and decoder with extra parameters indicating C_i and C_j. In our tests, we take the later approach. Therefore, the C2C-VAE presented here is actually a *conditional* variational autoencoder (Sohn et al., 2015).

7.3.1 Training a C2C-VAE

A C2C-VAE is trained using the following procedure:

- Train a traditional VAE with encoder f and decoder f'.
- Assemble training pairs: Randomly collect 10000 pairs of samples during every training epoch. For each sample pair s_i (of class C_i) and s_j (of class C_j), extract their features $f(s_i)$ and

$f(s_j)$, calculate their feature difference $f_\Delta(s_i, s_j) = f(s_i) - f(s_j)$.

- Train C2C-VAE with the vector $< f_\Delta, C_i, C_j >$: The encoder g (conditioned on the class pairs) learns to encode the input to embedding $g(< f_\Delta, C_i, C_j >)$. The decoder g' (also conditioned on the class pairs) learns to decode the embedding back to $f'_\Delta = g'(g(< f_\Delta, C_i, C_j >), < C_i, C_j >)$. Both g and g' are trained to minimize the reconstruction loss and the KL-divergence between the prior distribution (in this case, a Gaussian distribution) and the distribution of embeddings $g(< f_\Delta, C_i, C_j >)$. The loss function for C2C-VAE is:

$$\begin{aligned} loss =& ||g'(g(< f_\Delta, C_i, C_j >), < C_i, C_j >) - f_\Delta||^2 \\ &+ KL[g(< f_\Delta, C_i, C_j >), N(0,1)] \end{aligned}$$

Just as a VAE can generate new samples of the original space $L1$. A C2C-VAE can generate new feature differences in $L2'$, which in turn can be used to modify features in $L2$. The modified features can then be recovered as new samples in $L1$. More specifically, A C2C-VAE can be used to generate a new sample s_j of a target class C_j by adapting an existing sample (called as the source sample) using the following procedure:

- Choose a source sample s_i of class C_i in the space $L1$. Get its feature $f(s_i)$ in the space $L2$.
- Sample an embedding $g(< f_\Delta, C_i, C_j >)$ from the Gaussian distribution $N(0,1)$ in space $L3$. Decode this embedding to get a feature difference $f'_\Delta = g'(g(< f_\Delta, C_i, C_j >), < C_i, C_j >)$ in the space $L2'$.
- Apply the feature difference f'_Δ in $L2'$ to $f(s_i)$ in $L2$, to get the target sample feature $f(s_j) = f(s_i) - f'_\Delta$ in $L2$.
- Decode the target sample feature $f'(f(s_j))$ to the sample space $L1$

In this procedure, the C2C-VAE touches each of Boden's three proposed aspects of creativity. It is *exploring* the space $L3$ of difference patterns, *combining* a feature difference with another feature in $L2$, and *transforming* the sample in $L1$.

In Boden's original definition, the boundary between exploratory and transformational creativity is blurred. For this reason, Wiggins (2006) refines Boden's original framework by identifying two rule sets defining the conceptual space (in our previous terminology, this is $L1$, perhaps even $L2$.): the rule set $\mathscr{R}$ that constrains the space and the rule set $\mathscr{T}$ that traverses the space. Transformational creativity can emerge from either transforming $\mathscr{R}$, leading to a new conceptual space, or $\mathscr{T}$, leading to new traversal in the same conceptual space. Under Wiggins' definition, C2C-VAE is applying $\mathscr{T}$-transformation, where samples in the original conceptual space are modified by difference patterns sampled by C2C-VAE.

From the perspective Boden's three criteria of creativity, we hypothesize that C2C-VAE can generate realistic feature differences leading to valuable (within-category) samples, thanks to the regularity offered by both VAE and C2C-VAE. C2C-VAE provides three means for achieving novelty: 1) sampling in the space $L3$, allowing variation in the feature difference; 2) diversifying the source sample and adapting from different samples or classes; 3) sampling in the space $L2$, allowing variation in the feature of the source sample. The current implementation focuses on the first means for creativity.

7.4 Evaluating Creativity for One-Shot Classification Domains

We consider automated evaluation of creativity of generated samples in a classification domain in which a generator can learn to generate samples from data and a classifier can learn to classify samples. We propose a creativity evaluation approach named GOF/TOM (for "generate on few, test on many"). The approach is applicable to multiple forms of generators (e.g. VAE, GAN). Although we describe it in a classification task domain (as our focus is evaluation of C2C-VAE), its approach could be applied to regression or other task domains as well.

Given a data set, a class is chosen as the target class. Data samples of that class are called target samples. Some or all target samples are removed from the training set, creating a zero/one/few-shot setting (for simplicity, we will ignore the differences between the three settings and refer them as the one-shot setting), where the target data is not available to the generator in its entirety. Then the generator is trained on the trimmed training set. Meanwhile, a separate model (e.g. classifier) is trained on the untrimmed data set. Because the model sees target samples unknown to the generator, we call it the *oracle*. There can exist multiple oracles serving different purposes, as in our experimental evaluation.

After training of both the generator and the oracle, the generator is used to generate new target samples. The oracle can facilitate the evaluation of the generated sample on the three aspects of creativity (Boden, 1991). We assume that the oracle is implemented using deep learning or other techniques enabling extraction of features for the assessment of novelty and/or surprise.

Value

The oracle classifies the generated samples. We consider the generator able to generate valuable samples if it can consistently generate samples of the target class. The accuracy of the generator is the percentage of the generated samples falling into the intended class. High accuracy means highly valuable generator.

Novelty

The oracle can extract features of the samples into a latent space. The latent space needs to be smooth so that similar samples are close together and different samples are distant from each other. The generated samples are novel if their extracted features show variety. For our experiment, we use the activation of the embedding layer of an VAE as the feature values extracted for each sample. The variance of the features is used as the measure of the variety of the samples generated.

Surprise

A generator achieves *surprising* results if it can consistently generate samples of the target class that are unexpected. This means that even though the oracle classifies a sample as the target class, the sample is significantly different from existing samples (untrimmed data set). A distance measure is needed to measure the distance from a generated sample to existing samples. Existing measures of surprise are surveyed in Franceschelli and Musolesi (2021). Surprise is beyond the scope of this chapter and the capability for C2C-VAE to generate surprising samples is left for future research.

7.4.1 Uniqueness and Benefits

Our evaluation method differs from those mentioned earlier in a few ways: The untrimmed data set is not the same as the *inspiring set* (Ritchie, 2007) because the data contained are not necessarily creative; The generator learns from the trimmed data set and its knowledge of the target class is deliberately limited. Even if the untrimmed data set is inspiring, its trimmed version may not be; Instead of using a reference set as in Gervás (2011), the oracle classifier is used to evaluate the samples; Different from methods (Gervás, 2011; Morris et al., 2012) that use some (portion of) inspiring set for the evaluation of generated samples. Unlike the many evaluations such as in Norton et al. (2010) and Morris et al. (2012), the oracle classifier does not require user rating data or other human assessment of artistry or creativity. It requires only classification labels, which are more widely available.

Methods (Gervás, 2011; Morris et al., 2012) that use some (portion of) inspiring set for the evaluation integrate evaluation within the creative system. The creative system filters generated samples by evaluation. However, GOF/TOM estimates is envisioned as a tool for after-the-fact evaluation of the system's performance.

As observed by (Franceschelli and Musolesi, 2021), there is an ongoing debate between human evaluation versus machine evaluation. GOF/TOM blurs the boundary between the two. It learns about the task domain from the classification labels set by human (very often not for the purpose of evaluating creativity) through an oracle and relays the evaluation to the machine.

7.4.2 Caveats

In GOF/TOM, the generator is trained in a one-shot setting and then its generated samples are evaluated by an oracle trained using the untrimmed data set. There are three implications: 1) If the task domain is truly one-shot, then the construction of the oracle is impossible, due to the lack of additional data. 2) GOF/TOM may be less suitable for generators with weaker one-shot learning capability. Therefore, GOF/TOM should be used for *representing* the creativity of a generator *capable of* few-shot learning.

3) The method assesses value based on classifications by the oracle, and assesses novelty based on features generated by the oracle; such features could potentially be used for assessing surprise as well. Thus the usefulness all three measures is limited by how well the oracle has learned from the training data.

A concern for any automated evaluation of creativity is whether it truly captures the important characteristics. We believe that the use of accuracy as a proxy for value, and variance as a proxy for novelty, is reasonable in the domains used for the evaluation. However, these measures may miss important aspects of creativity for some domains. More work is needed on the measures to apply.

7.5 Evaluation

We carried out experiments on two data sets, using the GOF/TOM approach to evaluate the creativity of C2C-VAE. The first data set is the MNIST dataset of hand-written digits from 0 to 9. The second is the fashion-MNIST dataset of 10 classes of clothing and accessories. Each of MNIST and fashion-MNIST is provided with predetermined splits for training (60,000 samples) and testing (10,000 samples) data sets; these were used for training and validation. Each sample is a 28x28 grayscale image, associated with a label from 10 classes. The oracle classifier and the VAE are both trained with the full data set. The two data sets are chosen because a traditional VAE can extract features from them.

In all experiments, each class is successively chosen as the target class. For comparison with C2C-VAE, we also trained a conditional VAE (CVAE). The CVAE is trained to generate a sample conditioned on an additional parameter controlling the class of the sample generated. During testing, both C2C-VAE and CVAE generate samples of the target class.

7.5.1 System Design

The oracle for value is a resnet18 (not pretrained) network, of which the first layer is replaced with a convolutional layer with $(in\_channels = 1, out\_channels = 64, kernel\_size = (7,7), stride = (2,2), padding = (3,3), bias = False)$, and the last layer is replaced with a linear layer with 10 outputs for classifications.

The VAE follows a standard design. The encoder of the VAE is composite of two consecutive convolutional layers $(out\_channels = c, kernel\_size = 4, stride = 2, padding = 1)$ and $(out\_channels = c * 2, kernel\_size = 4, stride = 2, padding = 1)$, where $c = 64$. A linear layer for mean and another linear layer for log variance follow the convolutional layers and extract a distribution of features from the output of the convolutional layers. A feature is a vector of dimension 32. The decoder of the VAE reverses the design of the encoder: It consists of a linear layer and two consecutive convolutional layers. The input and output dimensions are the reverse of their corresponding encoder layers, other parameters being equal. The VAE is trained with standard reconstruction loss and KL-divergence loss: $Loss_{vae} = loss_{recon} + KL\text{-}divergence$

The VAE's encoder f serves as a feature extractor for the C2C-VAE and also as the oracle for novelty. The resnet18 classifier can also extract features but the feature space is not as smooth as that of the VAE.

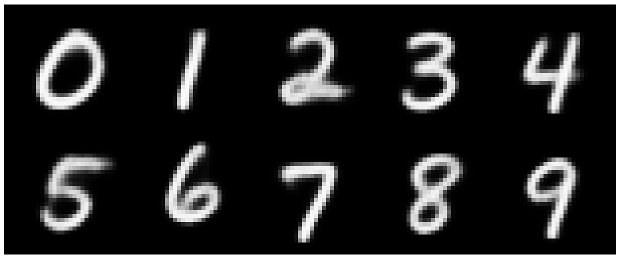
The C2C-VAE has its own encoder and decoder. Given a pair of samples, their features are extracted by the VAE and their class labels are one-hot encoded. The encoder takes the feature difference and the two class labels as input. The input is passed to a fully connected RELU layer with ($out\_features = 32$). A linear layer for mean and another linear layer for log variance follow and extract a distribution of embeddings, which are of 10 dimensions. The decoder takes an embedding and the two class labels as input. The input is passed to two consecutive linear layers to recover a feature difference similar to the original.

The CVAE has a very similar architecture to the VAE, except it is modified to be conditioned on the class label. Specifically, the encoder and the decoder use the same convolutional layers, but their linear layers that interact with features also take in the class labels as extra inputs.

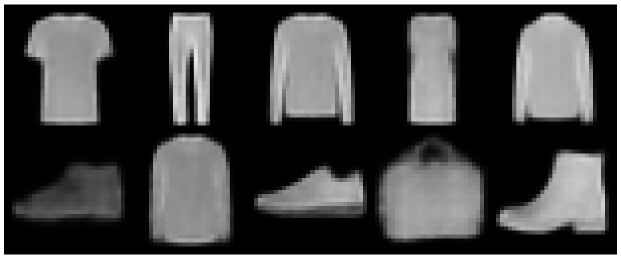
The C2C-VAE can only generate a new sample by adapting a source sample. We choose an average sample s_{avg} (see Figure 7.2) from each class C (other than the target class) as the source sample by the following procedure:

- Select all n samples s_1 - s_n of the class C;
- Calculate the average of their features $avg(\Sigma f(s_i)) = (\Sigma_{i=0}^{n} f(s_i))/n$;
- Use the decoder f' to recover the average sample $f'(avg(\Sigma f(s_i)))$.

In addition to the reconstruction loss, both the C2C-VAE and the CVAE are learning to minimize a KL-divergence loss with a Gaussian distribution ($\mu = 0, \sigma = 1$). This means that they are both trained to project their corresponding input to the Gaussian distribution. Although their inputs and embeddings carry different meanings (The CVAE takes input from $L1$ while the C2C-VAE takes input from $L2'$), we note that their embedding distributions are intended to be the same Gaussian. This also means that we could compare their performance with regard to the



(a) MNIST



(b) Fashion-MNIST

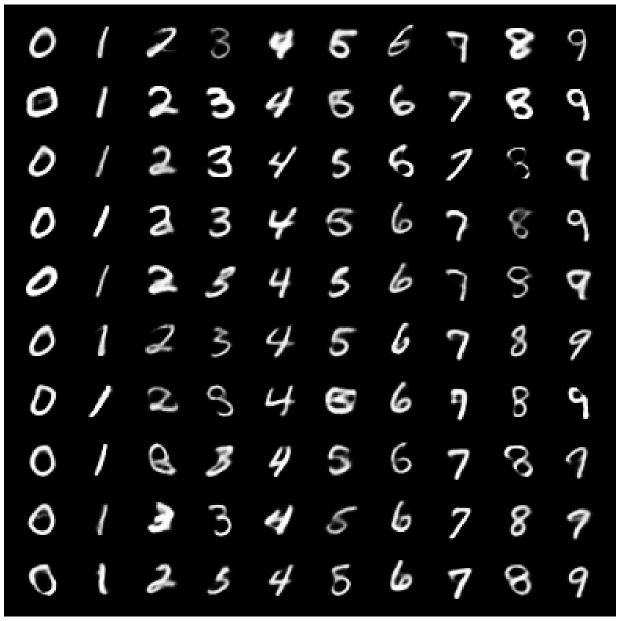
Figure 7.2: Average Samples Constructed by the VAE

standard deviation *std* of the Gaussian. The experimenter can make either model produce more or less various samples by tuning *std*, controlling the distribution from which the model is sampling from. The higher *std* is, the wider the distribution becomes, and the generated samples lose value (accuracy) but gain variety. Note that the C2C-VAE can also introduce additional novelty by altering the source sample, but this comes at a further cost of stability of the results (discussed in the Discussion and Future Work section).

7.5.2 Comparison between the C2C-VAE and the CVAE

Under the Normal Setting

Before examining the models under the GOF/TOM setting, we present some results under the normal setting to provide a backdrop for comparison. Under the normal setting, all models are trained on the untrimmed data set. In all figures of generated samples, column j represents the samples generated for class C_j. In all the figures of generated samples by the C2C-VAE, unless otherwise specified, the (i, j), where $i \neq j$, sample is a sample generated by choosing the average sample of class C_i, sampling a feature difference from class i to class j, and applying this feature difference to the chosen sample; Additionally, the (i, i) sample (on the diagonal) is an average sample of class C_i. In all the figures of samples by the CVAE, column j represents samples generated by random sampling in class C_j.


(a) C2C-VAE

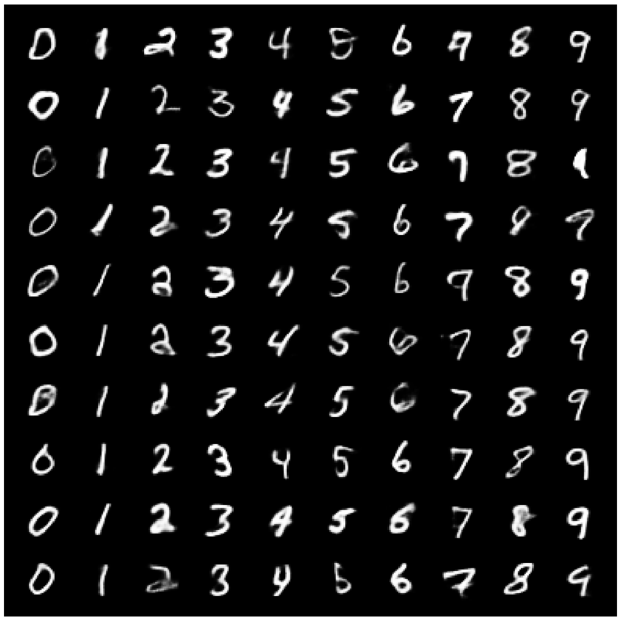

(b) CVAE

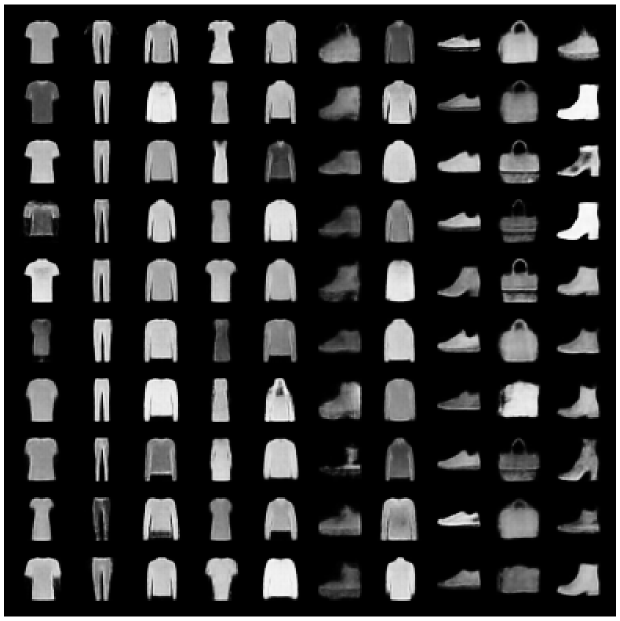
Figure 7.3: MNIST samples generated by the models trained under normal setting. $std = 1$. Both models demonstrate valuable and various samples.

Figures 7.3 and 7.4 illustrate that both C2C-VAE and C-VAE produce both valuable and varied samples. Because the models have seen various samples of the target class during training, the variety here is not equivalent to novelty (but it will be in GOF/TOM evaluation).

Figure 7.5 illustrating the tradeoff between value (measured by the accuracy of samples generated judged by the oracle for value) and variety (measured by the variance of the features extracted by the oracle for novelty). Under the normal setting, the two models share similar tradeoffs.

Under the GOF/TOM Setting

Our specific implementation of the GOF/TOM setting trims all samples of a target class except one average sample. We choose the average sample to better represent target class. Both CVAE and C2C-VAE are trained with the trimmed data set. During each training epoch, 10% of the training batch is this one-shot sample while 90% is randomly chosen from other samples (CVAE


(a) C2C-VAE

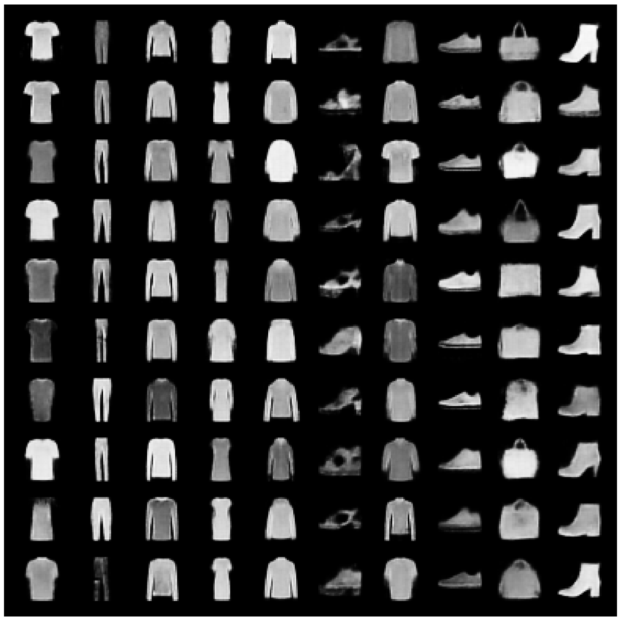

(b) CVAE

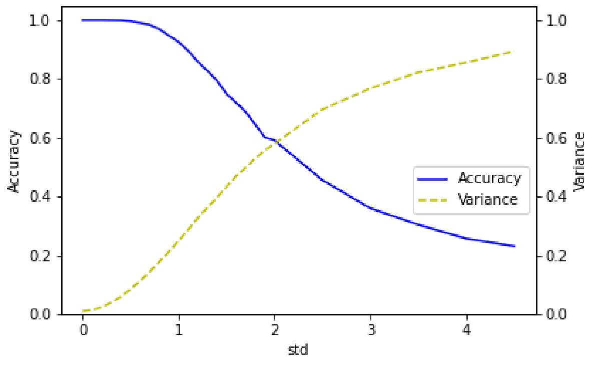
Figure 7.4: Fashion-MNIST samples generated by the models trained under normal setting. $std = 1$. Both models demonstrate valuable and various samples.

trains on the batch directly while C2C-VAE trains on pairs from the batch). This design counters the imbalanced classes caused by trimming. Therefore the CVAE learns about the target class from only the average sample, while the C2C-VAE learns from pairs of this sample and samples of other classes.

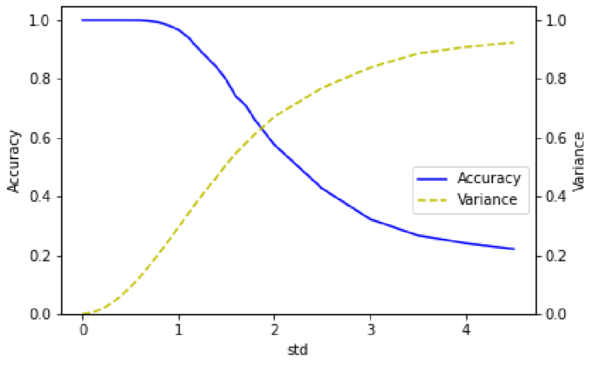
In contrast to the figures of generated samples under the normal setting, the figures presented in this section follow an additional rule: Column j is generated by models which are trained under the one-shot setting, where all samples except the average sample of class j are removed during training. Because the models have only seen a single sample of the target class during training, any variety of generated samples of the target class is equivalent to novelty.

When $std = 1$, the C2C-VAE generates novel—thus creative—samples while the CVAE can only generate similar samples, as shown in Figures 7.6 and 7.7.

Figures 7.8 and 7.9 illustrate the tradeoff between value (measured by the accuracy of samples



(a) C2C-VAE



(b) CVAE

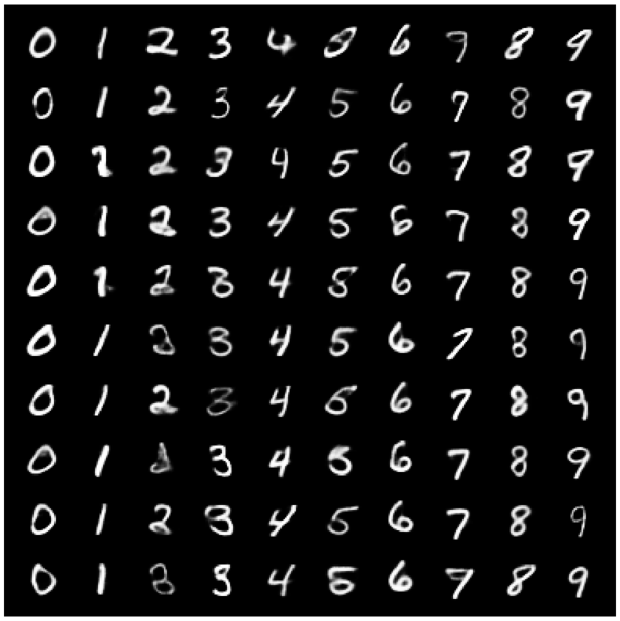
Figure 7.5: Under the normal setting in MNIST, both C2C-VAE and VAE trade off accuracy and variance as *std* is adjusted. Results are similar to those in the normal setting in fashion-MNIST

generated judged by the oracle for value) and novelty (measured by the variance of the features extracted by the oracle for novelty). C2C-VAE exhibits an accuracy and variance tradeoff as *std* is tuned. For a given level of *variance*, CVAE needs a bigger *std* change at a bigger cost of *accuracy* than C2C-VAE. For example, CVAE needs $std = 4$ to gain the same variance as C2C-VAE for $std = 1$ in MNIST, and $std = 4.5$ to gain the same variance as C2C-VAE ($std = 1$) in fashion-MNIST. When *std* is so high, the quality of the images is very poor, as shown in Figure 7.10.

7.6 Discussion and Future Work

7.6.1 Conditions for Success of the C2C-VAE Method

The applicability of the C2C-VAE method depends on three conditions: (1) There exists a VAE that can extract features of samples, (2) there exists a C2C-VAE that can extract embeddings of feature differences of two classes, and (3) the generated feature differences can be applied to a chosen sample. Condition (1) depends on properties of VAEs and is beyond the scope here. However, a large body of work supports the existence of effective VAEs for many domains.


(a) C2C-VAE

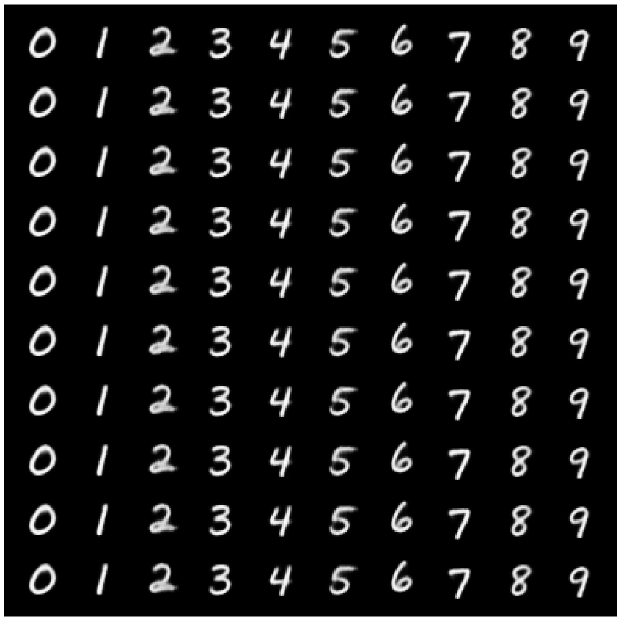
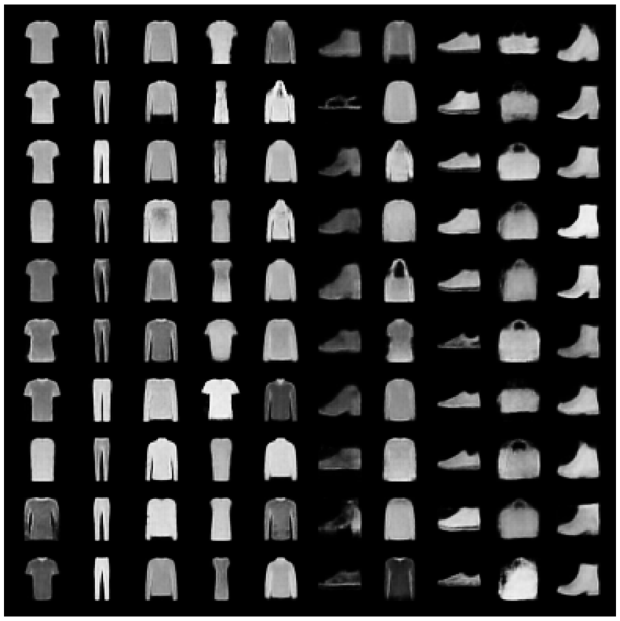

(b) CVAE

Figure 7.6: MNIST samples generated by the models trained under one-shot setting. $std = 1$. Intuitively, C2C-VAE generates more creative samples.

Condition (2) can fail if the feature differences between two classes do not conform to a single pattern. When two classes each have wide distributions, the difference between the two distribution can have very high variation, decreasing effectiveness of the C2C approach (Ye, 2018b). For example, drawings in the Quick, Draw! dataset vary considerably within classes. For example, a cat or dog may be drawn with a head only, or with a body, or with limbs and a tail. The difference between a cat without a body and a dog without a abody is not the same as the difference between a cat with a body and a dog with body.

Even if condition (2) holds, C2C-VAE can be sensitive to the choice of source sample, causing the failure of condition (3). C2C-VAE can generate creative samples by generating from different source samples, while at the risk of generating bad samples (see Figure 7.11).

In the procedure for generating new samples using C2C-VAE, the choice of a source sample s_i and the choice of a feature difference embedding and its subsequently induced feature difference


(a) C2C-VAE

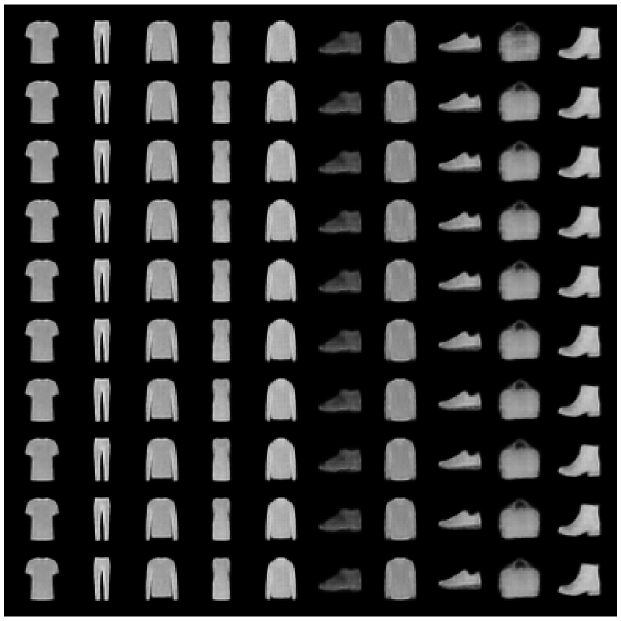
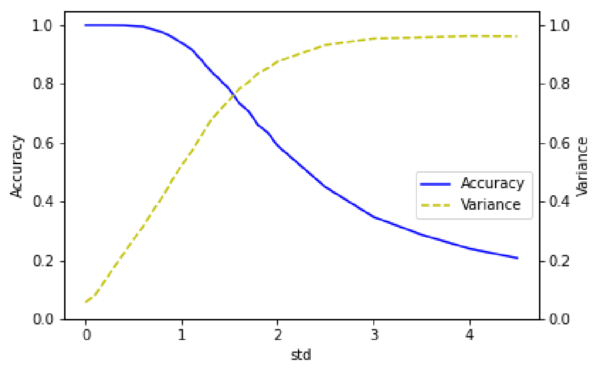

(b) CVAE

Figure 7.7: Fashion-MNIST samples generated by the models trained under one-shot setting. $std =$ 1. Intuitively, C2C-VAE generates more creative samples.

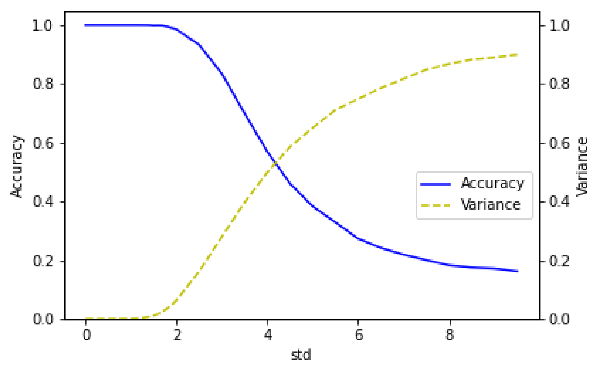
f'_{Δ} are currently two independent choices. There could (and perhaps even should) exist some dependency between the two choices. As a future direction, both conditions (2) and (3) may be resolved by conditioning the C2C-VAE on the source sample.

7.6.2 Relationship to CycleGAN

Although C2C-VAE and CycleGAN are completely different techniques, both share many foundational assumptions. CycleGAN assumes there is a pattern between two classes and trains translation functions (generators) on all possible pairs between two classes to learn this pattern. The reconstruction loss of C2C-VAE (that the feature difference can be recreated) corresponds to the cycle-consistency loss of CycleGAN (that the sample can be recovered). The reconstruction loss of the VAE which C2C-VAE depends upon for feature extraction (a realistic sample can be reconstructed from a feature) corresponds to the adversarial loss of CycleGAN (the recovered sample is



(a) C2C-VAE



(b) CVAE

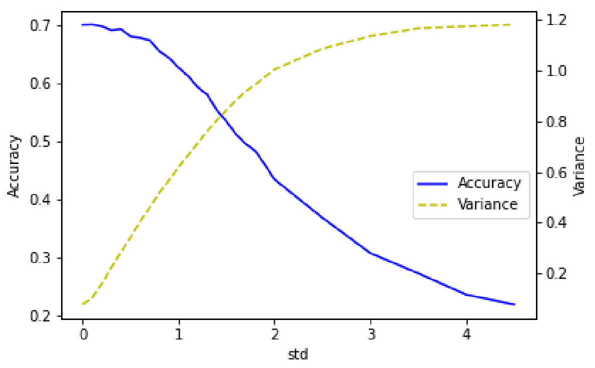
Figure 7.8: Under the one-shot setting in MNIST, C2C-VAE can trade off accuracy and variance when *std* is tuned. For a given level of *variance*, CVAE needs a bigger *std* change at a bigger cost of *accuracy* than C2C-VAE.

realistic). The two model are similar in their foundations, but C2C-VAE works with the space $L2'$ while CycleGAN works with the space $L1$ (and arguably $L2$).

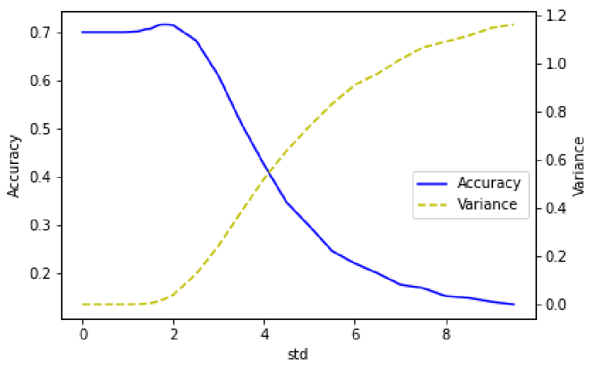
GOF/TOM can benefit from GAN. In its current design, the oracle for value often classifies a generated sample confidently with high activation score even if it is of poor quality by human perception. As a future direction, the oracle might be integrated with a discriminator of GAN to better distinguish poor samples.

7.6.3 Relationship with CDH-Style Adaptation

As discussed in the introduction, C2C-VAE is inspired by CDH-style adaptation. The difference is that CBR adaptation in general is adapting a case's problem description toward a target case's problem description while modifying its solution correspondingly. The source case and the target problem is known before the adaptation, which seeks the target solution by adapting the source case. CBR adaptation generates a new solution description. On the other hand, C2C-VAE does not assume the existence of a target case or a target problem. Instead it is sampling an adaptation



(a) C2C-VAE



(b) CVAE

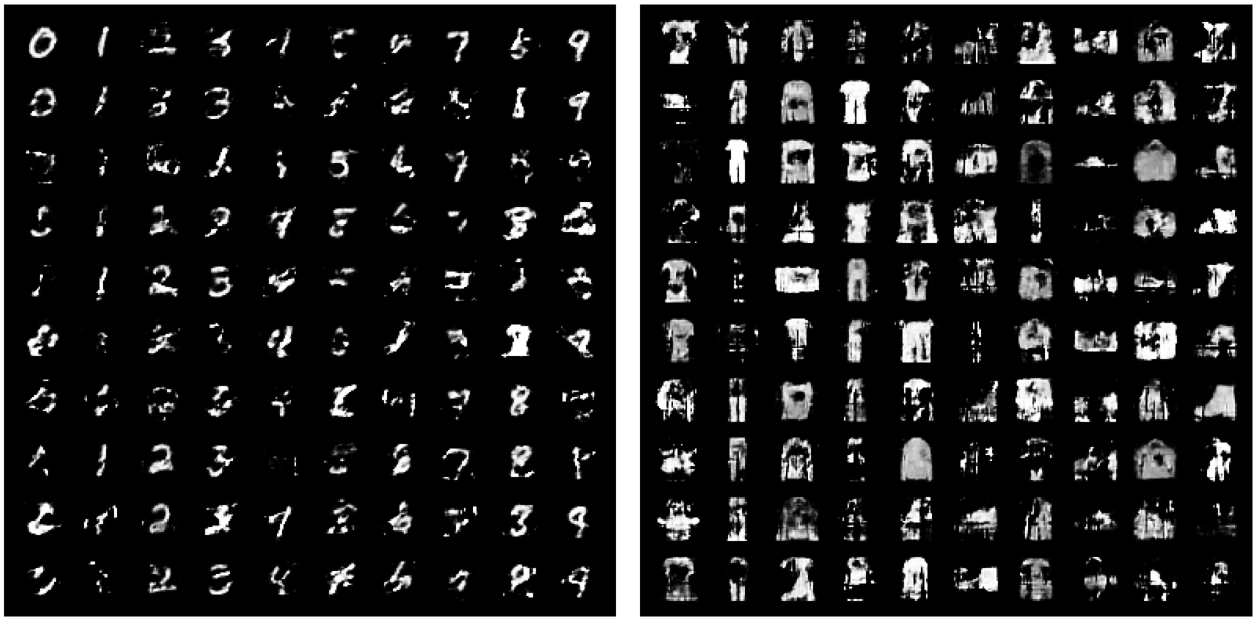
Figure 7.9: Under the one-shot setting in fashion-MNIST, C2C-VAE can trade off accuracy and variance when *std* is tuned. For a given level of *variance*, CVAE needs a bigger *std* change at a bigger cost of *accuracy* than C2C-VAE.

from one class label to another, or from one solution to another in CBR terminology. C2C-VAE aims to generate a new sample (with its new problem and solution descriptions).

If C2C-VAE is improved to generate new sample while being conditioned on the input source samples, as discussed in Section 7.6.1, then it can be potentially used as an adaptation method to generate counterfactual/semi-factual samples by modifying samples toward a target class. By controlling the extent of the adaptation, the generated sample may be near the boundary between the source class and the target class. If the generated sample is still of the same class as the source class, then it is a semi-factual sample; if it is of the target class, then it becomes a counterfactual sample. This procedure is inspired by Keane and Smyth (2020).

7.7 Conclusion: Creativity from Inter-Class Patterns

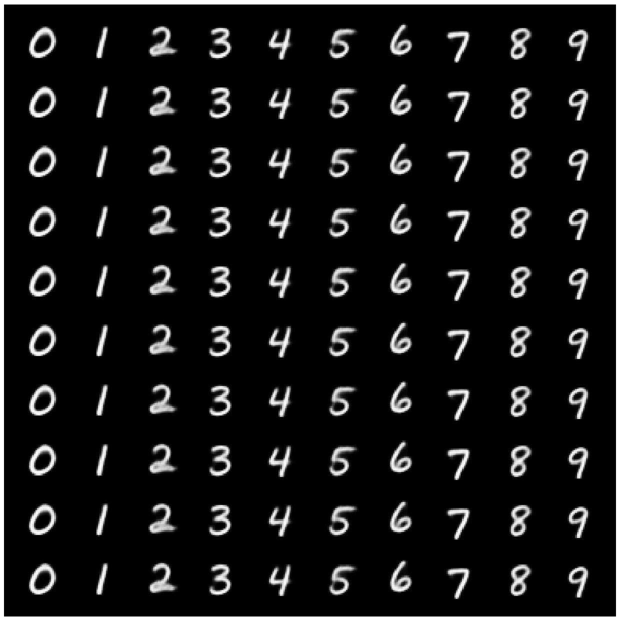
Network-based models provide exciting mechanisms for modeling creativity in AI systems. Existing work on generative methods for creativity can be seen as oriented primarily towards the conceptual space of samples, while C2C-VAE exploits the relationship between samples. If existing approaches



(a) MNIST ($std = 4$) (b) Fashion-MNIST ($std = 4.5$)

Figure 7.10: To gain variance, CVAE requires greatly increases *std*, sacrificing quality of generated samples

look at the foreground of conceptual space $L2$, C2C-VAE looks at the background $L2'$, in order to bring that background to bear as additional information to facilitate creative sample generation in one-shot settings. The presented experiments support that for a creative image generation task, C2C-VAE can achieve high novelty—variance in generated samples—while maintaining accuracy.


(a) Generated by Modifying Average Samples

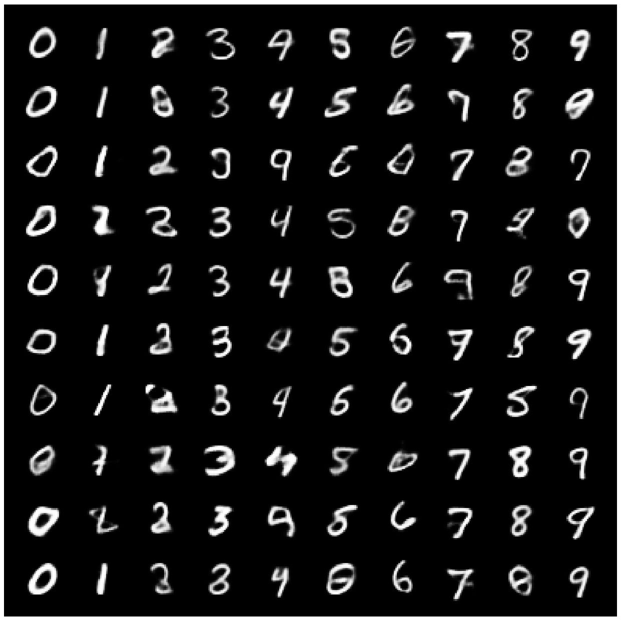

(b) Generated by Modifying Random Samples

Figure 7.11: If the source samples are randomly selected, C2C-VAE might generate bad samples. Here $std = 0$, so variance is solely due to the choice of source samples.

Chapter 8

Conclusions and Future Directions

8.1 Summary of Results

This thesis proposes two major methods in strengthening the case adaptation process of CBR: Robust adaptation and machine learning driven adaptation. It also examines the implications of these adaptation methods in the bigger scope of CBR.

The first method, robust adaptation (ROAD), uses heuristics to guide generation of multi-step adaptation paths in rule-based adaptation and reset the adaptation path when the path is deemed unreliable. Evaluations support the benefit of the approach, and especially the benefit of resetting.

Introspection about the paths generated by robust adaptation can also lead to improvement in case retrieval (RISA) and case adaptation (CARS). By learning from path generation failures, RISA improves the similarity measure to better align it with adaptability. Experimental results support its value for retrieving more adaptable cases. By favoring pairs of rules that have participated in successful adaptation paths, CARS compresses the adaptation rule set while retaining useful rules. Experimental results support its value for increasing adaptation efficiency and that deletion of least reliable rules can improve accuracy, subject to an efficiency/coverage tradeoff.

The second method, machine learning driven adaptation, uses a neural network (NN-CDH) to learn adaptation knowledge. The proposed method can learn to predict a solution difference based on problem difference and adaptation context, for both regression and classification task domains. Experiments with both real and artificial data sets show that NN-CDH achieves comparable performance to its counterpart, the traditional neural network. Moreover, a well-trained retrieval process, like SN retrieval, may outperform both neural network and NN-CDH, and NN-CDH can worsen the retrieval result. This illustrates the need to harmonize retrieval and adaptation methods.

Alternating optimization (AO) is used to address the synchronization issue between case retrieval and adaptation. Experiments show that with AO, the retrieval stage generally provides a good initial case, and adaptation further modifies the solution to be closer to the correct solution. A CBR system trained under the AO scheme shows reliable behavior while the same system trained under a traditional independent training scheme or an adaptation-guided retrieval training scheme suffers asynchronization or codependence effects.

As most of this thesis focuses on building adaptation algorithms to better solve a target query, the last chapter takes a turn and proposes an creative sample generation method (C2C-VAE) inspired by case adaptation. It also introduces a general for evaluating creativity of sample generators (GOF/TOM — "Generated On Few, Tested On Many.") A generator is trained in a zero, one, or few-shot setting where samples of a target class are trimmed from the training set. Meanwhile, an oracle is trained on the untrimmed dataset to evaluate the generated samples. Because the generator has limited examples of the target class, its ability to generate satisfactory unseen samples of the target class can be used as measure for its creativity. By learning and applying the difference patterns between existing samples, C2C-VAE can adapt old samples into novel samples. Experiments support that for a creative image generation task, C2C-VAE can achieve higher novelty—variance in generated samples—while maintaining accuracy.

8.2 Future Directions

The chapters offer many future directions in their corresponding research projects. This section recapitulates the most noteworthy ones.

8.2.1 Interaction between Adaptation Rules

In the current implementation of CARS, the concepts of reliability and compatibility are mixed. A single score represents the compatibility of a rule with all other rules. It would be possible to distinguish reliability and compatibility and implement a more fine-grained version of CARS.

For example, a rule might be incompatible with a set of rules but perfectly compatible with another set. A matrix can be built where every rule is represented by a row and a column and an entry in the matrix represent the compatibility between the rule of its row and the rule of its column. the ROAD system in its building of adaptation paths may search only compatible rules and prioritize reliable rules. This can be especially helpful in a domain where certain rules can be compatible with few rules but highly reliable when used in a path.

8.2.2 Extending AO to the Full CBR Cycle

Chapter 6 discusses the interaction between case retrieval and adaptation and uses AO to find a balance between the two processes. This holistic view can be carried over to the full CBR cycle, or all 4 "REs". For example, retrieval and reuse could optimize to provide a confidence score for the proposed solution and alarm the revision process when probable; Revision training could optimize to focus on exceptions that cannot be solved by adaptation; Retention training could optimize to retain cases for which changing adaptation capabilities are most accurate (or most efficient), or could adjust indexing as the similarity criteria for retrieval change.

8.2.3 Neural-Symbolic CBR

From rule-based adaptation (Chapters 2, 3, and 4) to machine learning driven adaptation (Chapters 5 and 6), this thesis finds that although case adaptation integrating with machine learning techniques gains the benefits of higher efficiency and accuracy, it also loses certain benefits from the symbolic side. For example, the artificial data set in Chapter 5 can be solved by two simple symbolic rules but is difficult for both the traditional neural network and machine learning driven CBR.

As the recent rise of interest in neural symbolic research (such as IBM Neuro-Symbolic AI Workshop 2022) and combining CBR with deep learning (such as the CBRDL 2021 workshop), we envision that a balanced blend of domain knowledge, brought to bear by symbolic AI methods, with network-learned methods, has the potential to achieve a performance edge.

Moreover, as presented in Chapter 6, the different processes and knowledge containers of CBR interact with each other and contribute to the CBR system in a holistic manner. It is promising, although undoubtedly hard, to advance all aspects of CBR toward neuro-symbolic methods so that they can interact over both symbolic knowledge and statistically driven generalizations.

8.2.4 Conditioning the Adaptation of C2C-VAE on the Source Sample

In the procedure for generating new samples using C2C-VAE, the choice of a source sample s_i and the choice of a feature difference embedding and its subsequently induced feature difference f'_Δ are currently two independent choices. If the later choice is conditioned on the former, then C2C-VAE can potentially generate a most reasonable adaptation given a source sample and a target class.

C2C-VAE can then be used as an adaptation method to generate counterfactual/semi-factual cases by modifying cases toward a target class. By controlling the extent of the adaptation, the generated sample may be near the boundary between the source class and the target class. If the generated sample is still of the same class as the source class, then it is a semi-factual sample; if it is of the target class, then it becomes a counterfactual sample.

Bibliography

Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–52.

Aha, D. W. and Goldstone, R. L. (1992). Concept learning and flexible weighting. In *In Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pages 534–539. Erlbaum.

Badra, F., Cordier, A., and Lieber, J. (2009). Opportunistic adaptation knowledge discovery. In *Case-Based Reasoning Research and Development, ICCBR 2009*, pages 60–74, Berlin. Springer.

Bezdek, J. and Hathaway, R. (2002a). Some notes on alternating optimization. In Pal, N. R. and Sugeno, M., editors, *Advances in Soft Computing — AFSS 2002*, pages 288–300, Berlin. Springer.

Bezdek, J. C. and Hathaway, R. J. (2002b). Some notes on alternating optimization. In Pal, N. R. and Sugeno, M., editors, *Advances in Soft Computing — AFSS 2002*, pages 288–300, Berlin, Heidelberg. Springer Berlin Heidelberg.

Boden, M. A. (1991). *The Creative Mind: Myths and Mechanisms.* Basic Books, Inc., USA.

Bonzano, A., Cunningham, P., and Smyth, B. (1997). Using introspective learning to improve retrieval in CBR: A case study in air traffic control. In Leake, D. and Plaza, E., editors, *Proceedings of the 2nd International Conference on Case-Based Reasoning (ICCBR-97)*, volume 1266 of *LNAI*, pages 291–302, Berlin. Springer.

Broad, T., Berns, S., Colton, S., and Grierson, M. (2021). Active divergence with generative deep learning - A survey and taxonomy. *CoRR*, abs/2107.05599.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1993). Signature verification using a ”siamese” time delay neural network. In *Proceedings of the 6th International Conference* *on Neural Information Processing Systems*, NIPS'93, pages 737–744, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Corchado, J. and Lees, B. (2001). *Adaptation of Cases for Case Based Forecasting with Neural Network Support*, pages 293–319. Springer London, London.

Craw, S., Jarmulak, J., and Rowe, R. (2001). Learning and applying case-based adaptation knowledge. In Aha, D. and Watson, I., editors, *Proceedings of the Fourth International Conference on Case-Based Reasoning*, pages 131–145, Berlin. Springer Verlag.

Craw, S., Wiratunga, N., and Rowe, R. (2006a). Learning adaptation knowledge to improve case-based reasoning. *Artificial Intelligence*, 170:1175–1192.

Craw, S., Wiratunga, N., and Rowe, R. C. (2006b). Learning adaptation knowledge to improve case-based reasoning. *Artificial Intelligence*, 170(16):1175 – 1192.

D'Aquin, M., Lieber, J., and Napoli, A. (2006). Adaptation knowledge acquisition: a case study for case-based decision support in oncology. *Computational Intelligence*, 22(3/4):161–176.

Díaz-Agudo, B., Gervás, P., and González-Ca͂lero, P. A. (2003). Adaptation guided retrieval based on formal concept analysis. In *Proceedings of the 5th international conference on case-based reasoning: research and development*, pages 131–145, Berlin. Springer.

Draper, S. (2010). Creativity. `https://www.psy.gla.ac.uk/~steve/best/creative.html`. Accessed: 2022-05-08.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

D'Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A., and Szathmary, L. (2007). Case base mining for adaptation knowledge acquisition. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 750–755, San Mateo. Morgan Kaufmann.

Elgammal, A. M., Liu, B., Elhoseiny, M., and Mazzone, M. (2017). CAN: creative adversarial networks, generating "art" by learning about styles and deviating from style norms. *CoRR*, abs/1706.07068.

Elhoseiny, M. and Elfeki, M. (2019). Creativity inspired zero-shot learning. *CoRR*, abs/1904.01109.

F. Zhang, M. Ha, X. Wang, and X. Li (2004). Case adaptation using estimators of neural network. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, volume 4, pages 2162–2166 vol.4.

Fox, S. and Leake, D. (2001). Introspective reasoning for index refinement in case-based reasoning. *The Journal of Experimental and Theoretical Artificial Intelligence*, 13(1):63–88.

Franceschelli, G. and Musolesi, M. (2021). Creativity and machine learning: A survey. *CoRR*, abs/2104.02726.

Friedman, J. H. (1994). Flexible metric nearest neighbor classification. Technical report, Stanford University.

Fuchs, B., Lieber, J., Mille, A., and Napoli, A. (2014a). Differential adaptation: An operational approach to adaptation for solving numerical problems with CBR. *Knowledge-Based Systems*, 68:103–114.

Fuchs, B., Lieber, J., Mille, A., and Napoli, A. (2014b). Differential adaptation: An operational approach to adaptation for solving numerical problems with CBR. *Knowledge-Based Systems*.

Gervás, P. (2011). Dynamic inspiring sets for sustained novelty in poetry generation. In Ventura, D., Gervás, P., Harrell, D. F., Maher, M. L., Pease, A., and Wiggins, G. A., editors, *Proceedings of the Second International Conference on Computational Creativity, ICCC 2011, Mexico City, Mexico, April 27-29, 2011*, pages 111–116. computationalcreativity.net.

Ha, D. and Eck, D. (2017). A neural representation of sketch drawings. *CoRR*, abs/1704.03477.

Hammond, K. (1989). Chef. In Riesbeck, C. and Schank, R., editors, *Inside Case-Based Reasoning*, chapter 6, pages 165–212. Lawrence Erlbaum.

Hanney, K. and Keane, M. (1996). Learning adaptation rules from a case-base. In *Proceedings of the Third European Workshop on Case-Based Reasoning*, pages 179–192, Berlin. Springer.

Hanney, K., Keane, M., Smyth, B., and Cunningham, P. (1995). What kind of adaptation do CBR systems need? a review of current practice. In *Proceedings of the Fall Symposium on Adaptation of Knowledge for Reuse*. AAAI.

Jalali, V. and Leake, D. (2013). Extending case adaptation with automatically-generated ensembles of adaptation rules. In *Case-Based Reasoning Research and Development, ICCBR 2013*, pages 188–202, Berlin. Springer.

Jalali, V. and Leake, D. (2014). On retention of adaptation rules. In *Case-Based Reasoning Research and Development, ICCBR 2014*, Berlin. Springer.

Jalali, V. and Leake, D. (2015). Enhancing case-based regression with automatically-generated ensembles of adaptations. *Journal of Intelligent Information Systems*, pages 1–22.

Jalali, V. and Leake, D. (2016). Enhancing case-based regression with automatically-generated ensembles of adaptations. *Journal of Intelligent Information Systems*, 46(2):237–258.

Jalali, V., Leake, D., and Forouzandehmehr, N. (2017a). Learning and applying case adaptation rules for classification: An ensemble approach. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4874–4878. International Joint Conferences on Artificial Intelligence.

Jalali, V., Leake, D., and Forouzandehmehr, N. (2017b). Learning and applying case adaptation rules for classification: An ensemble approach. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4874–4878.

Jarmulak, J., Craw, S., and Rowe, R. (2001). Using case-base data to learn adaptation knowledge for design. In *Proceedings of the 17th international joint conference on Artificial intelligence - Volume 2*, IJCAI'01, pages 1011–1016, San Francisco. Morgan Kaufmann.

Kaggle (2017). Automobile dataset. data retrieved from Kaggle, `https://www.kaggle.com/toramky/automobile-dataset`.

Karras, T., Laine, S., and Aila, T. (2018). A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948.

Keane, M. T. and Smyth, B. (2020). Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable ai (xai). In *Case-Based Reasoning Research and Development: 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8–12, 2020, Proceedings*, page 163–178, Berlin, Heidelberg. Springer-Verlag.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kolodner, J. (1993). *Case-Based Reasoning.* Morgan Kaufmann, San Mateo, CA.

Kolodner, J. L. (1992). An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1):3–34.

Leake, D., editor (1996). *Case-Based Reasoning: Experiences, Lessons, and Future Directions.* AAAI Press/MIT Press, Menlo Park, CA.

Leake, D. and Dial, S. (2008). Using case provenance to propagate feedback to cases and adaptations. In *Proceedings of the Ninth European Conference on Case-Based Reasoning*, pages 255–268. Springer.

Leake, D., Kinley, A., and Wilson, D. (1996). Linking adaptation and similarity learning. In

Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society, pages 591–596, Mahwah, NJ. Lawrence Erlbaum.

Leake, D., Kinley, A., and Wilson, D. (1997). Learning to integrate multiple knowledge sources for case-based reasoning. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 246–251. Morgan Kaufmann.

Leake, D., Kinley, A., and Wilson, D. (2011). Enhancing case adaptation with introspective reasoning. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*. Morgan Kaufmann.

Leake, D. and Schack, B. (2018). Exploration vs. exploitation in case-base maintenance: Leveraging competence-based deletion with ghost cases. In *Case-Based Reasoning Research and Development, ICCBR 2018*, pages 202–218, Berlin. Springer.

Leake, D., Smyth, B., Wilson, D., and Yang, Q., editors (2001). *Maintaining Case-Based Reasoning Systems*. Blackwell. Special issue of *Computational Intelligence*, 17(2), 2001.

Leake, D. and Whitehead, M. (2007). Case provenance: The value of remembering case sources. In *Case-Based Reasoning Research and Development: Proceedings of the Seventh International Conference on Case-Based Reasoning, ICCBR-07*, pages 194–208, Berlin. Springer-Verlag.

Leake, D. and Wilson, D. (1998). Categorizing case-base maintenance: Dimensions and directions. In Cunningham, P., Smyth, B., and Keane, M., editors, *Proceedings of the Fourth European Workshop on Case-Based Reasoning*, pages 196–207, Berlin. Springer Verlag.

Leake, D. and Ye, X. (2019a). On combining case adaptation rules. In Bach, K. and Marling, C., editors, *Case-Based Reasoning Research and Development*, pages 204–218, Cham. Springer International Publishing.

Leake, D. and Ye, X. (2019b). On combining case adaptation rules. In *Case-Based Reasoning Research and Development, ICCBR 2019*, pages 204–218. Springer.

Leake, D. and Ye, X. (2020). Learning to improve efficiency for adaptation paths. In *Case-Based Reasoning Research and Development: 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8–12, 2020, Proceedings*, page 325–340, Berlin, Heidelberg. Springer-Verlag.

Leake, D. and Ye, X. (2021a). Harmonizing case retrieval and adaptation with alternating optimization. In *Case-Based Reasoning Research and Development, ICCBR 2019*, pages 125–139. Springer.

Leake, D. and Ye, X. (2021b). Harmonizing case retrieval and adaptation with alternating optimization. In Sánchez-Ruiz, A. A. and Floyd, M. W., editors, *Case-Based Reasoning Research and Development*, pages 125–139, Cham. Springer International Publishing.

Leake, D., Ye, X., and Crandall, D. (2021a). Supporting case-based reasoning with neural networks: An illustration for case adaptation. In *AAAI Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE)*.

Leake, D., Ye, X., and Crandall, D. (2021b). Supporting case-based reasoning with neural networks: An illustration for case adaptation. In *Proceedings of AAAI Spring Symposium AAAI-MAKE 2021: Combining Machine Learning and Knowledge Engineering*. https://www.aaai-make.info/program.

Leake, D. B. and Crandall, D. J. (2020). On bringing case-based reasoning methodology to deep learning. In *ICCBR*.

Leake, D. B., Kinley, A., and Wilson, D. (1995). Learning to improve case adaptation by introspective reasoning and cbr. In Veloso, M. and Aamodt, A., editors, *Case-Based Reasoning Research and Development*, pages 229–240, Berlin, Heidelberg. Springer Berlin Heidelberg.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database. `http://yann.lecun.com/exdb/mnist/`.

Li, H., Hu, D., Hao, T., Wenyin, L., and Chen, X. (2007). Adaptation rule learning for case-based reasoning. In *Semantics, Knowledge and Grid, Third International Conference on*, pages 44–49.

Liao, C., Liu, A., and Chao, Y. (2018). A machine learning approach to case adaptation. In *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 106–109.

López de Mántaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M., Cox, M., Forbus, K., Keane, M., Aamodt, A., and Watson, I. (2005). Retrieval, reuse, revision, and retention in CBR. *Knowledge Engineering Review*, 20(3).

Martin, K., Wiratunga, N., Sani, S., Massie, S., and Clos, J. (2017). A convolutional siamese network for developing similarity knowledge in the selfback dataset. In *ICCBR*.

Mathisen, B. M., Aamodt, A., Bach, K., and Langseth, H. (2019). Learning similarity measures from data. *Progress in Artificial Intelligence*.

McDonnell, N. and Cunningham, P. (2006). A knowledge-light approach to regression using case-based reasoning. In *Proceedings of the 8th European conference on Case-Based Reasoning*, ECCBR'06, pages 91–105, Berlin. Springer.

Minor, M., Bergmann, R., and Gorg, S. (2014). Case-based adaptation of workflows. *Information Systems*, 40:142–152.

Morris, R. G., Burton, S. H., Bodily, P., and Ventura, D. (2012). Soup over bean of pure joy: Culinary ruminations of an artificial chef. In *ICCC*.

Nobari, A. H., Rashad, M. F., and Ahmed, F. (2021). Creativegan: Editing generative adversarial networks for creative design synthesis. *CoRR*, abs/2103.06242.

Norton, D., Heath, D., and Ventura, D. (2010). Establishing appreciation in a creative system. In *ICCC 2010*.

Nouaouria, N. and Boukadoum, M. (2010). Case retrieval with combined adaptability and similarity criteria: application to case retrieval nets. In *Proceedings of the ninth international conference on case-based reasoning*, pages 242–256, Berlin. Springer Verlag.

Nugent, C. and Cunningham, P. (2005). A case-based recommender for black-box systems. *Artificial Intelligence Review*, 24(2):163–178.

NVIDIA (2021). AI artist Helena Sarin. `https://www.nvidia.com/en-us/research/ai-art-gallery/artists/helena-sarin/`. Accessed: 2022-02-17.

Pan, Z., Yu, W., Yi, X., Khan, A., Yuan, F., and Zheng, Y. (2019). Recent progress on generative adversarial networks (gans): A survey. *IEEE Access*, 7:36322–36333.

Patterson, D., Rooney, N., and Galushka, M. (2002). A regression based adaptation strategy for case-based reasoning. page 87–92, USA. American Association for Artificial Intelligence.

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, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Petrovic, S., Khussainova, G., and Jagannathan, R. (2016). Knowledge-light adaptation approaches in case-based reasoning for radiotherapy treatment planning. *Artificial Intelligence in Medicine*, 68:17–28.

Policastro, C. A., Carvalho, A. C., and Delbem, A. C. (2006). Automatic knowledge learning and case adaptation with a hybrid committee approach. *Journal of Applied Logic*, 4(1):26–38.

Policastro, C. A., Carvalho, A. C. P. L. F., and Delbem, A. C. B. (2003). Hybrid approaches for case retrieval and adaptation. In *KI 2003: Advances in Artificial Intelligence*, pages 297–311, Berlin. Springer.

Richter, M. (1998). Introduction. In Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., and Wess, S., editors, *CBR Technology: From Foundations to Applications*, chapter 1, pages 1–15. Springer, Berlin.

Riesbeck, C. and Schank, R. (1989). *Inside Case-Based Reasoning.* Lawrence Erlbaum, Hillsdale, NJ.

Ritchie, G. (2007). Some empirical criteria for attributing creativity to a computer program. *Minds and Machines*, 17(1):67–99.

Roberts, A., Engel, J., Raffel, C., Simon, I., and Hawthorne, C. (2018a). Musicvae: Creating a palette for musical scores with machine learning. `https://magenta.tensorflow.org/music-vae`. Accessed: 2022-02-17.

Roberts, A., Engel, J. H., Raffel, C., Hawthorne, C., and Eck, D. (2018b). A hierarchical latent vector model for learning long-term structure in music. *CoRR*, abs/1803.05428.

Schank, R. (1982). *Dynamic Memory: A Theory of Learning in Computers and People.* Cambridge University Press, Cambridge, England.

Shiu, S., Yeung, D., Sun, C., and Wang, X. (2001). Transferring case knowledge to adaptation knowledge: An approach for case-base maintenance. *Computational Intelligence*, 17(2):295–314.

Smiti, A. and Elouedi, Z. (2011). Overview of maintenance for case based reasoning systems. *International Journal of Computer Applications*, 32:49–57.

Smyth, B. and Cunningham, P. (1993). Complexity of adaptation in real-world case-based reasoning systems. *The Irish Journal of Psychology*, 14(3):476–477.

Smyth, B. and Keane, M. (1998). Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. *Artificial Intelligence*, 102(2):249–293.

Sohn, K., Lee, H., and Yan, X. (2015). Learning structured output representation using deep conditional generative models. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Toivonen, H. and Gross, O. (2015). Data mining and machine learning in computational creativity. *WIREs Data Mining and Knowledge Discovery*, 5(6):265–275.

Varshney, L. R., Pinel, F., Varshney, K. R., Bhattacharjya, D., Schörgendorfer, A., and Chee, Y. (2013). A big data approach to computational creativity. *CoRR*, abs/1311.1213.

Veloso, M. (1994). *Planning and Learning by Analogical Reasoning.* Springer Verlag, Berlin.

Wettschereck, D., Aha, D., and Mohri, T. (1997). A review and empirical evaluation of feature-weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11(1-5):273–314.

Wiggins, G. A. (2006). A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems*, 19(7):449–458. Creative Systems.

Wiggins, G. A. (2021). Creativity and consciousness: Framing, fiction and fraud. In de Silva Garza, A. G., Veale, T., Aguilar, W., and y Pérez, R. P., editors, *Proceedings of the Twelfth International Conference on Computational Creativity, México City, México (Virtual), September 14-18, 2021*, pages 182–191. Association for Computational Creativity (ACC).

Wilson, D. and Leake, D. (2001). Maintaining case-based reasoners: Dimensions and directions. *Computational Intelligence*, 17(2):196–213.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747.

Xiong, N. and Funk, P. (2006). Building similarity metrics reflecting utility in case-based reasoning. *Journal of Intelligent and Fuzzy Systems*, 17(4):407–416.

Ye, X. (2018a). The enemy of my enemy is my friend: Class-to-class weighting in k-nearest neighbors algorithm. In *Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018*, pages 389–394.

Ye, X. (2018b). The enemy of my enemy is my friend: Class-to-class weighting in k-nearest neighbors algorithm. In *FLAIRS Conference*.

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

Ye, X., Leake, D., Jalali, V., and Crandall, D. J. (2021a). Learning adaptations for case-based classification: A neural network approach. In Sánchez-Ruiz, A. A. and Floyd, M. W., editors, *Case-Based Reasoning Research and Development*, pages 279–293, Cham. Springer International Publishing.

Ye, X., Leake, D., Jalali, V., and Crandall, D. J. (2021b). Learning adaptations for case-based classification: A neural network approach. In *International Conference on Case-Based Reasoning*, pages 279–293. Springer.

Ye, X., Zhao, Z., Leake, D., Wang, X., and Crandall, D. J. (2021c). Applying the case difference heuristic to learn adaptations from deep network features. *CoRR*, abs/2107.07095.

Zhu, J., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593.

Xiaomeng Ye

Curriculum Vitae

✉ *xiaye@iu.edu*
My Webpage

Research Interest

Case-based Reasoning	Case Retrieval, Case Adaptation, Case Base Maintenance, Similarity Metric
Machine Learning	Feature Weighting, Explainable AI, Difference Pattern
Cross-discipline	Cognitive Science, Counseling Psychology

Education

2015–present **PhD, Computer Science**, *Indiana University Bloomington*, Indiana, USA.
Minor in Cognitive Science

2011–2014 **B.A., Computer Science & Math**, *The College of Wooster*, Ohio, USA.
Magna Cum Laude

2010–2011 **Computer Science & German**, *Beloit College*, Wisconsin, USA.

Publications

Journal Articles

2020 Nicholas A. Bowman, Nayoung Jang, D. Martin Kivlighan, Nancy Schneider, and Xiaomeng Ye. The Impact of a Goal-Setting Intervention for Engineering Students on Academic Probation. *Research in Higher Education*, volume 61, pages 142–166, February 2020.

2018 III D. Martin Kivlighan, Marie C. Adams, Kuo Deng, Xiaomeng Ye, and Elizabeth J. Menninga. A social network analysis of international collaboration in counseling psychology. *The Counseling Psychologist*, volume 46, pages 274–295, 2018.

Communicated Journal Article

In Conference Proceedings

2021 Xiaomeng Ye, Ziwei Zhao, David Leake, Xizi Wang, and David Crandall. Applying the case difference heuristic to learn adaptations from deep network features. In *IJCAI-21 Workshop on Deep Learning, Case-Based Reasoning, and AutoML: Present and Future Synergies*, 2021. In press.

2021 Xiaomeng Ye, David Leake, Vahid Jalali, and David Crandall. Learning adaptations for case-based classification: A neural network approach. In *Case-Based Reasoning Research and Development, ICCBR 2021.* Springer, 2021. In press.

2021 David Leake, Xiaomeng Ye, and David Crandall. Supporting case-based reasoning with neural networks: An illustration for case adaptation. In *AAAI-MAKE 2021: Combining Machine Learning and Knowledge Engineering*, 2021.

2021 David Leake and Xiaomeng Ye. Harmonizing case retrieval and adaptation with alternating optimization. In *Case-Based Reasoning Research and Development, ICCBR 2021.* Springer, 2021. **Best paper award**.

2020 Xiaomeng Ye, David Leake, William Huibregtse, and Mehmet Dalkilic. Applying class-to-class siamese networks to explain classifications with supportive and contrastive cases. In *International Conference on Case-Based Reasoning*, pages 245–260. Springer, 2020.

2020 David Leake and Xiaomeng Ye. Learning to improve efficiency for adaptation paths. In Ian Watson and Rosina Weber, editors, *Case-Based Reasoning Research and Development*, pages 325–340, Cham, 2020. Springer International Publishing.

2019 Xiaomeng Ye. C2C trace retrieval: Fast classification using class-to-class weighting. In *Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference, FLAIRS 2019*, pages 353–358, 2019.

2019 David Leake and Xiaomeng Ye. On combining case adaptation rules. In Kerstin Bach and Cindy Marling, editors, *Case-Based Reasoning Research and Development*, pages 204–218, Cham, 2019. Springer International Publishing.

2018 Xiaomeng Ye. The enemy of my enemy is my friend: Class-to-class weighting in k-nearest neighbors algorithm. In *Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018*, pages 389–394, 2018.

Research Experience

May, 2020 – present **_Research Assistant in DL-CBR Group_**, *Indiana University Bloomington*, IN.
Involved in multiple research projects: (1) Survey of combining DL and CBR; (2) Using neural network to carry out case adaptation in regression and classification tasks.
Currently working on: (1) Using ML to predict and explain gentrification; (2) Learn difference pattern in data using variational autoencoder; (3) Harmonizing all four stages of CBR using Alternating Optimization.

Advisor : **Dr. David Leake**, *Professor of Computer Science*, *The Luddy School of Informatics, Computing, and Engineering*, Indiana University
Dr. David Crandall, *Professor of Computer Science*, *The Luddy School of Informatics, Computing, and Engineering, Indiana University*

2020 **Gratitude Journal App**.
Developed a mobile app for both android and iOS, for gratitude support group.
Collected user journals on a secured server.

Summer 2017 **_Research Intern_**, *Knexus Research Corporation*, National Harbor, MD.
Worked on an Information Extraction task using technology including Apache UIMA, link-grammar, POS tagging, C45 decision tree.
Built a machine learning application, which learns from annotated training examples. Trained model can extract information such as assignee, due date, action, start date from an unstructured text (eg. Email).

2014 **Undergraduate Thesis: _Evolving Meaningful Lambda Calculus Functions using Genetic Programming_**.
Applied genetic programming to evolve multiple mathematical operators for Church numerals.
Year-long independent study finished with a graduation thesis.
Best Poster in Math & Sciences in **_Independent Study Symposium Poster Contest_**

Teaching Experience

2017-2021 **_Research Mentor_ for Undergraduate Research Opportunities in Computing**, *Indiana University Bloomington*, Bloomington, IN.
Mentored six teams of 1-2 novice researchers in topics of their choice, including: instance-based reasoning, siamese network, case-based reasoning, forgery detection, and features in ML.
Each mentorship is a semester-long project with weekly meeting, literature review, brainstorming, coding review, and project presentation.

2015-2020 ***Associate Instructor***, *Indiana University Bloomington*, Bloomington, IN.

Taught in classrooms of various sizes ($< 50^a$ or $> 100^A$) and levels (undergraduate^b or graduate^B). Designed course material and homework ^{C}. Hosted labs and office hours. Graded students' projects and homework. Attended regular meetings with professors and fellow instructors. Courses taught include:

- Discrete Structures for Computer Science ^{Ab}, Fall 2015-Fall 2016
- Introduction to Algorithm Design and Analysis ^{ab}, Spring 2017
- Applied Machine Learning ^{AB}, Fall 2017
- Introduction to Data Analysis and Mining ^{AB}, Spring 2018
- Elements of Artificial Intelligence ^{AB}, Fall 2018
- Introduction to Computers and Programming ^{AbC}, Spring 2019-Fall 2019
- Computer Vision ^{AB}, Spring 2020

Fall 2012-2014 ***Teaching Assistant* for Computer Science Department**, *The College of Wooster*, Wooster, OH.

Facilitated the teaching of classes including Java, C++, Python, and Alice.
Attended labs and hosted office hours.
Graded students' labs and projects. Analyzed and reported results to professors.

Spring 2012-2013 ***Grader* for Mathematics Department**, *The College of Wooster*, Wooster, OH.
Graded students' homeworks. Analyzed and reported results to professors.

Industrial Experience

2014-2015 ***Software Engineer***, *Cureo*, Wooster, OH.
Worked on a business-oriented network platform in a startup setting.

Spring 2014 ***Software Engineering Intern***, *Westfield Insurance*, Westfield Center, OH.
Worked on the front-end of a testing software for QA usage.
Programmed with JSF framework, Javascript, SQL.

Summer 2013 ***Game Developer Intern***, *Dreamwork.cn*, Chengdu, Sichuan, China.
Worked on an iOS massive multiplayer online role playing game.
Worked in cooperation with other programmers, artwork designers, and game designers.
Built game logic, interfaces, animations, mini-map, visual effects and gadgets.

Winter 2012 ***Software Engineering Intern***, *Cureo*, Wooster, OH.
Rehired and continued my work during Summer 2012.

Summer 2012 ***Software Engineering Intern***, *Cureo*, Wooster, OH.
Worked on a business-oriented network platform in a startup setting.
Learned and coded with Javascript, HTML, CSS, JQuery, ASP.NET, C#, SQL.
Built 600 unit tests, implemented both front-end webpages and back-end database in a MVC website project

Fellowships & Awards

2021 Best Paper Award in 29th ***International Conference on Case-Based Reasoning* (ICCBR 2021)**, Salamanca, Spain.

2018 Video Competition in 26th ***International Conference on Case-Based Reasoning* (ICCBR 2018)**, Stockholm, Sweden.

2018 Student Travel Grant for 26th ***International Conference on Case-Based Reasoning* (ICCBR 2018)**, Stockholm, Sweden.

2014 Best Poster in Math & Sciences in ***Independent Study Symposium Poster Contest***, The College of Wooster, Wooster, OH

Computer skills

Programming Languages Python, PyTorch, keras, R, C, C++, JAVA

Web Technologies	HTML 5, PHP, JSP, Javascript
Database	SQL, MySQL, Apache, Neo4j

Services

Summer 2022	**Reviewer, The Fifth International Conference on Intelligence Science (ICIS2022)**, *Xi'an*, China.
Summer 2021	**Organizer, IJCAI 2021 DeepCBR workshop**.
2019	**Volunteer, Middle-way House**, Bloomington, IN.