Chapter 18

Enhancing Case-Based Reasoning with Neural Networks

David Leake, Indiana University

Zachary Wilkerson, Indiana University

Xiaomeng Ye, Berry College

David J. Crandall, Indiana University

18.1. Introduction

Neuro-symbolic systems enable leveraging strengths of both neural and symbolic approaches [1]. Case-based reasoning (CBR) is a knowledge-based reasoning and learning methodology that exploits a memory of prior cases—records of prior instances or experiences—by adapting their lessons to solve new problems [2, 3, 4, 5]. The integration of CBR with neural systems is appealing because of their complementary characteristics. Neural models have achieved impressive performance for tasks such as categorization, but reasoning and synthesis of structured solutions (e.g., for planning and design) remain a challenge, as does incorporating existing knowledge into network processes. Training network models—and retraining them for additional data—can be costly. In addition, network models are opaque, functioning as "black boxes." Much research has addressed explanation of black box systems [6] but there are strong arguments for benefits of applying inherently interpretable models, especially as AI is increasingly used in critical domains [7]. Case-based reasoning can be effective with few examples, is amenable to reasoning and synthesis tasks, and supports addition of expert knowledge; it also provides inertia-free learning and is naturally interpretable [8]. However, its success depends on knowledge—not only case knowledge, but also the indexing, similarity and case adaptation knowledge required to retrieve and apply cases—which may be difficult to obtain.

This chapter presents research on combining neural network methods with CBR, illustrating how neural system components can reduce the knowledge engineering burden for CBR and improve system performance. It also describes how the use of neural components can benefit CBR system performance by enabling optimization of retrieval and adaptation components in context of each other [9].

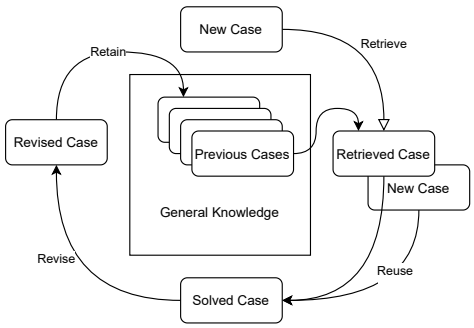

Figure 1. The CBR cycle, based on Aamodt and Plaza [14]

Specifically, the chapter presents three research strands on neuro-symbolic integrations to support CBR, addressing: (1) extracting features for case retrieval from deep neural networks to use in concert with expert-generated features, (2) refining a CBR approach to case adaptation knowledge learning, the "case difference heuristic" approach, by applying neural network learning, and (3) harmonizing retrieval learning with case adaptation learning, in order to retrieve cases that are adaptable to address new problems. It summarizes some strengths, weaknesses and tradeoffs of these approaches, and points to future challenges for neuro-CBR integrations.

18.2. Case-Based Reasoning: Motivations and Method

Case-based reasoning (CBR) is a methodology for reasoning and learning by retrieving and adapting the lessons of prior episodes, and storing the result as a new case for future use. One of the inspirations for studying CBR comes from observations of human reasoning [3, 10, 11, 12]. Human experts—and others—are reminded of past experiences as they encounter new problems; the sharing of "war stories" is a common way experts transmit knowledge. In addition, CBR is an appealing methodology for knowledge-based AI systems for pragmatic reasons, such as the relative ease of case acquisition—both because cases may be easier to elicit than rules [8] and because, in some domains, cases may be captured routinely as a byproduct of other processes. CBR models have been developed for many knowledge-rich tasks and have been widely applied (e.g., [13]). Another benefit of case-based models is their interpretability (e.g., [7, 8]): The results of a CBR system can be explained to users in terms of the prior cases on which they are based.

The CBR Cycle: The CBR process is a cycle in which problems are solved by steps often described as *retrieve, reuse, revise*, and *retain* [14]. Given a new problem, the most relevant prior case is retrieved, its solution is reused—matched to the new situation—and then revised—adapted to fit the new situation—and finally, retained—stored as a new case learned by the system. The process is illustrated in Figure 1.

The CBR Knowledge Containers: The CBR process uses multiple forms of knowledge, identified by Richter as the CBR knowledge containers [5, pp. 34–37]: representational vocabulary, case knowledge, similarity knowledge, and case adaptation knowledge. The

knowledge in the containers can overlap, in the sense that placing knowledge in one can decrease the need for knowledge in another. For example, increasing the case base size can decrease the need for adaptation knowledge, if the additional cases enable retrieving cases more similar to incoming problems (which reduces the need for adaptation). The ability to choose where to place knowledge provides flexibility for knowledge acquisition from humans and by automated learning methods.

Complimentary Strengths of CBR and Neural Networks: CBR can function successfully with very limited data, and facilitates integrating multiple forms of expert knowledge. CBR is a lazy learning method with inexpensive learning: CBR systems learn by simply storing new cases, without generalization until (and only if) needed to process a new problem. However, CBR is not a panacea for knowledge acquisition [15]. Feature engineering and acquisition of similarity knowledge may be difficult, and acquiring case adaptation knowledge is a longstanding challenge for CBR [8, 16, 17]. Neural network models, in contrast to CBR, do not easily exploit prior knowledge. They carry out eager learning on large data sets, generalizing at training time, with expensive training and retraining for new data. However, they can achieve high performance without feature engineering (in the case of deep networks) and with minimal prior knowledge requirements.

The contrasting properties of CBR and network models make it appealing to combine network methods and CBR. This pairing can involve either using networks to support CBR, or CBR to support networks [18]. For example, Das et. [19] present a neuro-symbolic CBR question-answering approach that uses a neural retriever to retrieve similar cases. Their system outperforms the state of the art on sample data sets while enabling the use of new human-labeled examples without retraining. Kenny and Keane propose pairing neural networks with CBR in twin systems, in which indices extracted from networks are used to retrieve cases to explain network results [20]. When sufficient data is available, both similarity knowledge and case adaptation knowledge can be learned by network models, and the trained networks can themselves be used as components within a CBR system. In the resulting hybrid system, the case retrieval and adaptation processes are more opaque than symbolic retrieval and adaptation methods. However, the resulting systems still have a level of inherent interpretability, via case presentation, that is not present in end-to-end network-based models.

18.3. Learning CBR Similarity Criteria and Retrieval Features

18.3.1. Background on CBR Similarity Learning

Similarity assessment for CBR may involve complex processes, including inference and assessment of structural similarity for complex cases (e.g., [17]). For many problems, however, it is sufficient to characterize problems in terms of feature vectors and define similarity based on a weighted distance function. For the remainder of this paper we will assume feature-vector representations. Similarity learning in this context has long been a research focus for CBR (e.g., [21]). Considerable recent work on neural network similarity for CBR has applied siamese networks [22], which learn an embedding for data points, which can then be used to calculate similarity based on the embedding space. [23, 24, 25]. Mathisen, Bach, and Aamodt propose an extension that learns how to use the embedding to calculate distances [26]. The focus of our work is instead on learning

to extract features and feature weights from input cases, to apply a (possible knowledge-based) similarity function to that feature information.

18.3.2. Background on CBR Feature Learning

Much CBR research on feature learning applies symbolic learning methods (some of these are ed in Mantaras et al. [17]). Recently, there has been much interest in combining CBR with neural network components that perform feature extraction to support case retrieval in an external CBR system (e.g., [27, 28, 29, 30]).

Sani et al. [31] present a system for human activity recognition that extracts features from a sensor and then uses a convolutional neural network (CNN) to interpret the input data, which is represented in three dimensions. The generated features are then compared against known wave form cases to infer the type, duration, etc. of the activity that generated the sensory input data. Other approaches go a level of abstraction higher and consider the similarity functions themselves. Grace et al. [32] propose a hybrid system for creating plausible yet unexpected, recipe designs. Their system applies deep learning (DL) techniques to learn relationships between cases in a case base; this provides additional knowledge that can be patterned to expectations when attempting to address the parameters of a presented goal. Mathisen et al. [26] use neural networks to learn similarity measures.

Neural-network-based feature learning is especially appealing for domains such as image recognition, for which CNNs have been used to extract feature data from complex inputs to inform case-based reasoning systems. Turner et al. [33] apply this to novel object recognition. In their work, a CNN architecture classifies inputs that correlate with known classes with high confidence; when encountering novel inputs with a correspondingly lower confidence, the image features are extracted from the CNN to be used in similarity calculations to group the new input with other similar images. As a result, the combined system can be sensitive to images without known classification labels by loosely classifying them in terms of other cases. The Turner et al. model extracts features for their CBR system from between the convolution/pooling and dense layers of the CNN. Our model applies a different approach by extracting features just before the output layer, as explained in Section 18.4.

Outside the focus of this chapter, Kraska et al. apply linear models and neural networks to feature aggregation and discrimination [34]. As an alternative to explicit feature extraction, Das et al. apply network methods directly to retrieval [19]; Kim et al. explore methods for selecting features to maximize interpretability by looking for features that create clustering "gaps" between data points [35], and they also integrate CBR principles directly into network models to create inherently interpretable networks [36]. Such networks leverage prototypes, which may represent whole classes/concepts [37, 38] or elements that correlate with a class or subset of classes [39], to focus network learning along potentially more human-understandable channels by using similarity information across a field of prototypes for classification.

18.4. Combining Expert and Network-Learned Features

One of our research focuses concerns the integration of expert-generated and neural network-generated feature information to guide case similarity assessment and retrieval.

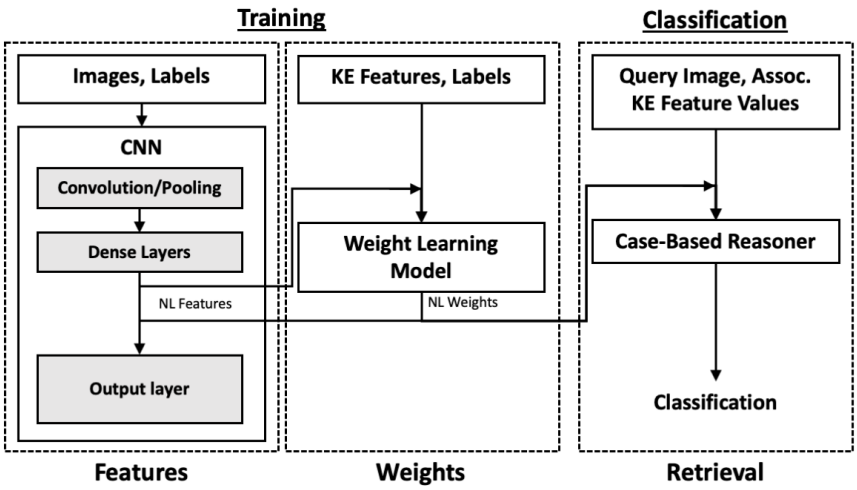

Figure 2. Illustration of data flow for integrating knowledge-engineered and network-learned features.

Traditional retrieval in CBR depends on case indexing, using atomic features and possibly more complex indexing structures to characterize cases. Knowledge acquisition and feature engineering (e.g., [40, 41, 42]) can provide high quality indices, but may be expensive or infeasible. For example, even domain experts may not be capable of providing comprehensive feature vocabularies for poorly-understood domains or for tasks such as image recognition. Given that DL systems can learn effectively from raw data, automated indexing using features extracted from a DL system seems a natural supplement or alternative to feature engineering. The research described in this section explores using features extracted from a trained convolutional neural network in concert with engineered features. Section 18.5 presents work on alternative feature extraction strategies.

18.4.1. Extracting Features from Convolutional Neural Networks for Classification

The proposed DL-CBR hybrid model, illustrated in Figure 2, trains a CNN from scratch for a classification task and then uses the trained network to extract features from input data (e.g., images) to be used as indices for retrieval by a case-based classification system [43]. Given a new input, another sequential network generates weights for both learned and knowledge engineered features for the case-based classifier similarity calculation. The case-based classifier then uses a combination of engineered features and extracted features, weighted according to the generated weights, for case retrieval.

18.4.1.1. Testbed Extraction Model

The structure of the CNN used for feature extraction in our research derives closely from the AlexNet architecture [44]. However, the bias node is removed from the CNN output layer to avoid feature skew during training, because a bias node would factor into the weighted sum used for prediction by the CNN but would not be extracted as a feature itself. The proposed method deviates from other feature extraction approaches (e.g., [31, 33, 45]) by extracting outputs from the dense layers of the CNN as features,

rather than extracting the inputs to the dense layers. This choice is motivated by the potential for dense layers to modify and/or combine features from previous layers into more complex values that may be for useful for CBR indices. Section 18.5 examines this design choice. After extraction, features derived from the CNN are combined with knowledge-engineered features from the raw data set.

18.4.2. Evaluation

Our evaluation of feature extraction concerned how use of network features can make CBR more robust to flaws in knowledge-engineered features. Specifically, it addressed *how classification accuracy of the CBR system is affected by degradation of reliability of knowledge-engineered (KE) features*, and *how using network-learned (NL) features in concert with KE features affects classification accuracy, as quality of KE features changes.* For reasons of space, the following sections only briefly summarize key points. Full quantitative results and experimental procedure (including explorations for weight learning strategies when using NL features) are described in Wilkerson et al. [43].

Experiments used a case-based classifier with no adaptation component that retrieves the nearest neighbor (1-NN) using a weighted Euclidean distance metric for similarity calculations. The test domain was the Animals with Attributes 2 data set (AwA2) [46], which includes both non-symbolic information and engineered features. In AwA2 all instances of a class are assigned the same KE feature vector, so these feature vectors yield perfect classification accuracy when used for retrieval. Consequently, in our experiments, these features are perturbed to simulate imperfect situation assessment assigning symbolic feature values and/or a feature vocabulary that is not 100% predictive. This artificial variance among feature values is used in our tests to simulate different degrees of reliability in the knowledge-engineered feature information.

How using KE and NL Features in Concert Affects Accuracy: Retrieval accuracy values for various perturbations of KE features in concert with NL features are compared against the values for each feature set individually. Each experiment is performed using unweighted leave-one-out testing on a case base of 500 cases.

Predictably, retrieval accuracy decreases as the perturbation magnitude increases, because a higher degree of noise is present in the KE feature set. Results of combined tests suggest that classification accuracy improves when using both KE and NL features together versus either feature set individually, supporting the assertion that features extracted from the CNN model capture aspects of the feature space not adequately covered by KE features alone. Arguably, this is further supported by the fact that the accuracy increase persists even when KE features are perturbed by large magnitudes, though this outcome may also be due to the relatively large number of NL features (1024) compared to KE features (85). As a result, the trend may be influenced by the dominating NL features, and/or accuracy increases may be partially attributable to the existence of more features in general.

18.4.3. Summary on Combining Network-Generated and Expert Features

This section has illustrated a DL-CBR hybrid system operating at a "middle ground" between the powerful yet opaque learning capability of neural networks and the more easily-explained but knowledge-intensive structure of traditional case-based reasoners.

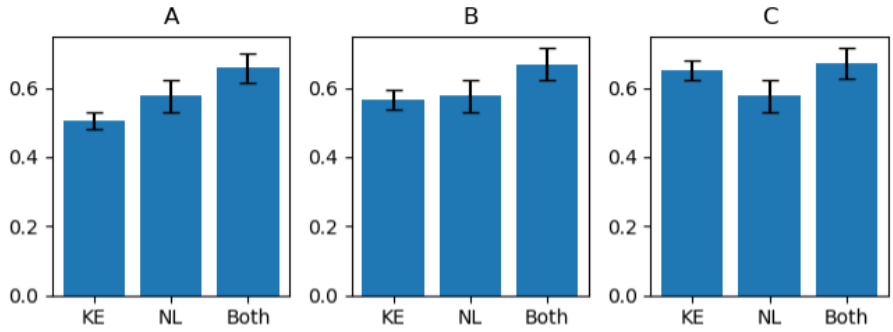

Figure 3. Comparison of classification accuracy values for different feature perturbations; left to right, the degree of KE feature perturbation decreases. Error bars represent one standard deviation relative to thirty iterations.

Experimentation suggests that features extracted from a CNN model can usefully augment existing knowledge-engineered features for CBR retrieval, especially when such KE features are inconsistent or incomplete across classes in the domain. Additional topics for research include further examination of where in the network features should be extracted (addressed in the following section), how to tune the number of features to extract and how to weight the combined features in the CBR system. Because using large numbers of network features could hamper explanation of system similarity judgments, we have begun research on methods for feature pruning, as described in Section 18.5.4.1.

18.5. Where in a CNN to Extract Features for CBR

The previous section illustrates the value of using convolutional neural networks to extract features from inputs for case retrieval. The illustrated method, as well as most current approaches for feature extraction for CBR, are based on plausible assumptions about where in a CNN to extract features for maximal usefulness to CBR. To test those underlying assumptions, we compared three extraction methods for image processing: extracting image features after the convolution layer (Turner et al. [33, 45], Sani et al. [31]); after the densely-connected layers (the authors' previously described work [43]); and after the densely-connected layers using multiple networks. Results show that the latter two approaches substantially increase case retrieval accuracy in example-sparse domains, to which case-based reasoning systems are commonly applied.

18.5.1. Candidate Feature Extraction Strategies

When extracting features from a neural network model for a CBR system, approaches typically extract feature vectors after the convolution and pooling steps, before the vectors are further processed by later densely-connected layers in the CNN (e.g., [31, 33, 45]). Convolution and pooling theoretically capture salient features from raw data (e.g., shapes, edges, etc. from images), and because such features are traditionally conceptualized as atomic elements of an image, it is appealing to map CNN feature vectors post-convolution to features for CBR similarity assessment.

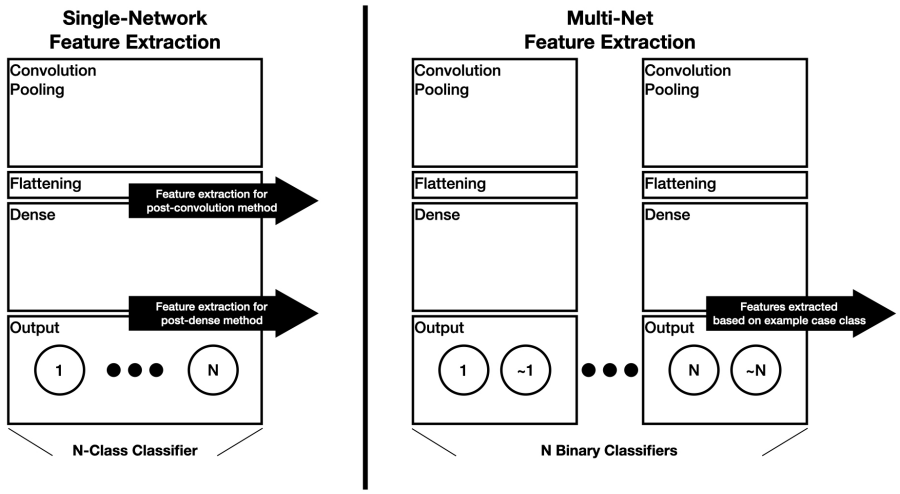

Figure 4. Feature extraction locations for post-convolution, post-dense (both left), and multi-net (right) feature extraction methods.

By contrast, if convolution and pooling theoretically highlight atomic features of the image, then the "mixing and matching" that occurs in densely-connected layers could generate richer feature representations through the recombination of primitive elements into more complex structures. As applied to CBR, this parallels the idea of atomic features and richer, more complex indices. Indeed, extracting feature vectors from after the densely-connected layers has led to success in generating counterfactual explanations [47], so it is also appealing to apply such an approach to classification as well.

With these fundamental extraction ideas in mind, we proposed three methods to compare these different feature extraction methods: extracting features after the convolution layers, extracting them after the dense layers, and extracting them after the dense layers from multiple CNN systems specialized to different classes (Figure 4). The latter model is motivated by the goal of minimizing the number of features extracted while still preserving CNN convergence [43].

Notes on Multi-Net Feature Extraction In CBR retrieval, certain features may become more applicable for similarity calculations depending on the class of the case to which the query is compared. This concept traditionally manifests through feature weighting, but features themselves may also be capable of reflecting this variable applicability for similarity. To this end, the multi-net feature extraction approach generates localized features, as opposed to a uniform set of features used across the entire case base.

As opposed to training one n-class CNN model from which features are extracted, multi-net leverages n binary CNN classifiers to distinguish between examples of a unique class and examples that do not have that class. This results in a unique extracted feature set for every ground truth class, from which a retrieval query case's feature set is selected based on the class of the candidate case to which it is compared (i.e., as if query and candidate are both members of the same class).

18.5.2. Evaluation

We evaluated feature quality across the three extraction methods, using retrieval accuracy as proxy for feature quality. We compare retrieval accuracy of the three extraction models and of an end-to-end CNN classifier. The end-to-end classifier was trained using all training examples, versus leave-one-out testing for the CBR model, so the end-to-end classifier can be informally considered an "upper bound" for network accuracy with complete information. Accuracy values are calculated for a spectrum of extracted feature set sizes. The following paragraphs highlight the principal findings only; full procedure, results, and discussion can be found in Leake et al. [48].

The CBR component used for these experiments is similar to that described in the previous section [43], with no adaptation component and performing retrieval using 1-NN based on an unweighted Euclidean distance similarity metric. The CNN component is identical, with only variations in the number of neurons in the feature extraction regions. The Places data set [49] is used for training and evaluation image data.

18.5.3. Extraction Location Influence on Feature Quality

Figure 5 shows retrieval accuracy versus number of features extracted for the evaluated methods [48]. In general, the post-dense extraction method outperforms post-convolution extraction, with multi-net extraction leading to the highest retrieval accuracy overall. This suggests that extracting features after the dense layers leads to superior feature quality; it appears that the feature aggregation/recombination that happens in the densely connected layers is useful for CBR retrieval indexing. Multi-net achieves superior performance at the expense of additional training time, reflecting an accuracy-training time trade-off in these retrieval methods.

Traditionally, CBR addresses relative feature importance through feature weighting, but neural network index generation makes it feasible to adjust the feature space itself, as illustrated by the multi-net design. The multi-net design strategy creates local models that recast query case features into a feature vocabulary relevant to the class of the candidate prior case being considered at that point in retrieval. Thus, the traditional CBR feature extraction question of "what features are present in the query?" becomes "what features related to class x are present in the query?"

18.5.4. Future Avenues for Maximizing Feature Quality

Beyond feature extraction location, promising avenues for improving extracted feature quality include exploring different DL models for feature extraction and revisiting feature weighting approaches.

Exploring different DL models for feature extraction: Both studies described previously use AlexNet as the base feature extraction model, while other computer vision algorithms and DL architectures have been developed that may also be applicable in a DL-CBR hybrid system. Other CNN architectures (e.g., VGGNet [50] or similar approaches) and structures (e.g., InceptionNet [51], etc.) could potentially facilitate higher-quality extracted features, because they are considered more accurate when applied for end-to-end image classification. Moving in the direction of newer CNN implementations, we have begun a comparative study of architectures including VGGNet, which further refines

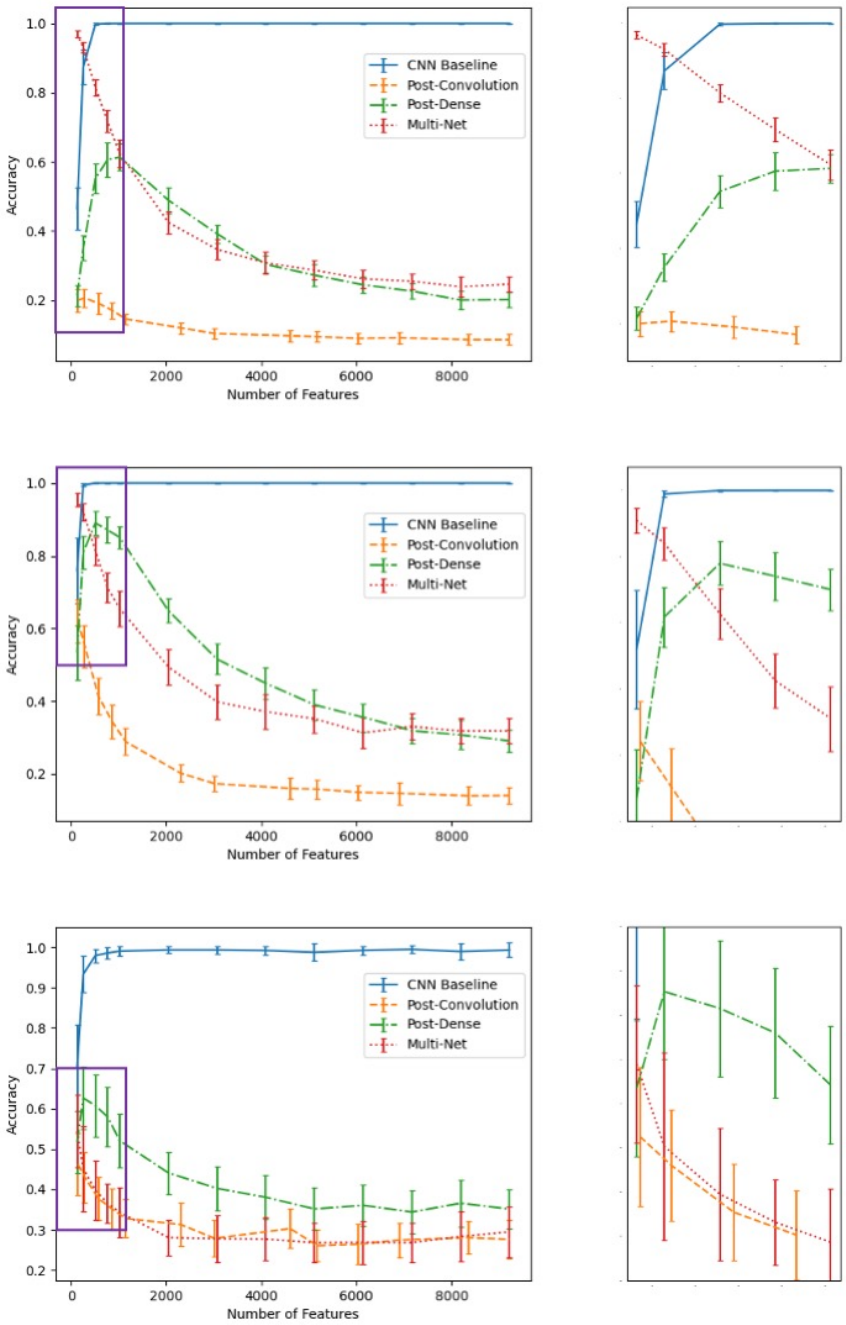

Figure 5. Accuracy versus number of features for a 50-class case (top), 25-class case (middle), and 10-class case (bottom), using 500 training examples overall. Error bars represent one standard deviation, and regions at right magnify bordered sub-regions at left.

the AlexNet architecture; InceptionNet, which applies convolution operations of various sizes in parallel; and DenseNet [52], which focuses on block-based feature processing rather than layer-based processing.

Revisiting feature weighting approaches: Wilkerson et al. [43] tested the effects of sample feature weighting approaches from the literature, but found none to be effective at increasing retrieval accuracy for those tests. We conjecture that the targeted algorithms, which focus on weights of individual features, could not effectively handle the large feature sets extracted from the networks. However, studies on number of features and feature extraction locations provide fuller context on feature requirements for convergence; as a result, it may be possible to reduce feature space dimensionality such that weight learning algorithms become more applicable.

18.5.4.1. Feature Pruning for Dimensionality Reduction

In addition to methods described above, it may be possible to further reduce the dimensionality of an extracted feature set using post-extraction feature pruning. This would maintain the CNN's capability for convergence while still reducing the size of a feature set to better suit the CBR system (and to facilitate explanations of similarity as well). Weight learning (e.g., perturbation methods, as in [47]) may then be applied. We have begun to explore feature pruning strategies [53].

18.5.5. Summary on CBR Feature Extraction from Networks

The previous section illustrates the effects of feature extraction location and number of features extracted on feature quality for a DL-CBR hybrid system. The results suggest that extracting from after the densely-connected layers in a CNN generates the most useful features, and there appears to be a quantifiable "sweet spot" for the number of features extracted, such that the CNN has enough parameters to converge but that performance is not impaired by the "curse of dimensionality." These results highlight DL model structure/parameters as critically important to CBR retrieval performance when using extracted features.

18.6. Network-Based Case Adaptation

The "reasoning" in case based reasoning is primarily case adaptation—the process of adjusting a prior solution to fit a new problem. Because CBR is often applied to domains for which only partial domain knowledge is available, acquiring case adaptation knowledge is a classic challenge for CBR [8, 16]. Such knowledge is often encoded in the form of rules, whose effectiveness may depend on the quality of the domain theory characterizing a domain. Early case-based reasoning research invested extensive effort to develop case adaptation knowledge for certain domains and to identify general adaptation methods (e.g., [54]). However, the difficulty of generating case adaptation knowledge limited the application of full CBR systems, resulting instead in emphasis on case-based aiding systems which retrieve cases to present to the user without adaptation [55]. Later work recognized the potential of machine learning methods to capture case adaptation knowledge. Such adaptation methods included the generation of rules by decision tree

learning [56, 57], and the use of case-based reasoning for the adaptation process itself [58, 59, 60, 61]

The most influential CBR adaptation rule learning approach is the *case difference heuristic* (CDH) approach [16]. This knowledge-light approach learns adaptation knowledge from cases in the case base and has been widely studied, especially for regression tasks (e.g. [16, 62, 63, 64, 65, 66, 67]). Given a pair of cases, the CDH approach calculates the difference between their problem descriptions (generally represented as feature vectors) and the difference between their solutions (generally numerical values for regression tasks). A rule is generated for the problem difference and solution difference. The rule encodes that when an input problem and retrieved case have a problem difference similar to the one from which the rule was generated, the solution for the retrieved case should be adjusted by the previously observed solution difference. For example, in the real estate price prediction domain, an adaptation rule might be generated from two similar apartments, one a two-bedroom and the other a three-bedroom, to adjust the price given an additional bedroom. Normally, human knowledge determines the design of how the adjustment should be done, which can involve additive, multiplicative, or other changes (e.g., for the real estate example, price might be adjusted by a fixed or percent increment). The process of CDH relies on the assumption that the old and new contexts will be sufficiently similar for the rule to apply to future cases.

In their seminal work of network-based case adaptation, Liao, Liu and Chao [68] applied deep learning to learn differences to assign to the solution of a retrieved case for regression problems. Their method presents the problem difference of two cases to a network which has been trained on pairs to output solution differences. Rather than providing the network only with the problem *difference*, as in their work, our approach provides the neural model with both the difference and the retrieved problem itself, as the context in which the adaptation is performed, enabling the network to adjust adaptations for that context. Craw, Wiratunga, and Rowe [69] showed that with careful tuning of feature weights, superior performance can be achieved by taking more context into account for the case difference heuristic. We are investigating the use of network methods to avoid the tuning step when adding context to the case-difference heuristic approach.

18.6.1. An Illustration of Network-Based Adaptation Performance

Liao, Liu and Chao [68] tested neural network adaptation for the NACA 0012 airfoil dataset [70] from the UCI repository [71], demonstrating that neural networks can learn adaptations for a CDH approach in that domain. To generate a fine-grained assessment of CDH compared to alternatives we compared five different systems: a k-NN system with $k = 1$, which can be seen as a CBR system with no case adaptation, a k-NN system with $k = 3$, which can be viewed as a CBR system with very simple adaptation (by averaging solutions of 3 different cases), a CBR system using adaptation rules generated using the case difference heuristic ("normal CDH"), inspired by Craw, Wiratunga and Rowe [69], a CBR system using a neural network to learn rules from CDH and carry out adaptation ("network CDH"), and, as a further baseline for comparison, a neural network system that solves the regression problem directly. The following subsections present a synopsis of design and results for this study; full details are available in Leake, Ye, and Crandall [72].

The design of the network CDH system builds on the model of Liao, Liu and Chao [68], but differs in two respects. First, as previously stated, in addition to taking as input

the problem differences, it takes as input the problem of the retrieved case, which provides context for the adaptation. Second, in addition to the CDH rule generation being trained on pairs of similar cases, rule generation is trained on pairs of random cases, enabling generation of rules applicable to larger differences (cf. Jalali and Leake [73]). Our experimental procedure differs from theirs in testing on data sets for which we restrict the available training cases so that the test query is always novel with respect to cases already seen by the system.

Implementations: The neural network system is implemented as a feedforward neural network of four fully connected layers. Depending on the task domain for which performance is being tested, there is minor variation in the number of neurons per layer. The system is trained until the validation error converges. The same structure is used for the adaptation network.

The CBR system with normal CDH follows the example from Craw, Wiratunga, and Rowe [69]. To generate adaptations, a pair of cases is compared to produce an adaptation example, with one case arbitrarily chosen as source case and the other as a target case. The problem description of the source case is used as a context, indicating that a problem difference in such a context can lead to such a solution difference. This system is denoted as "**CBR + normal CDH**", with subprocesses as follows:

- Case retrieval: Training cases are stored in a case base. Given a query, the case retrieval process finds the most similar case from the case base using 1-NN.
- Case adaptation: During training, adaptation examples are assembled from pairs of training cases and stored in an adaptation case base CB_a, distinct from the case base used to store cases for solved problems. During testing, the problem difference between the query and the retrieved case is calculated. The problem description of the retrieved case is used as the context. Then a 1-nearest neighbor algorithm retrieves the most similar adaptation example. This solution difference is added to the retrieved solution to produce the final solution.

The second system, the CBR system using CDH assisted by a neural network, is denoted as "**CBR + network CDH**", and is based on both Craw, Wiratunga, and Rowe [69] and Liao, Liu and Chao [74]. The system is implemented as follows:

- Case retrieval: Training cases are stored in a case base. Given a query, the case retrieval process finds the most similar case from the case base using 1-NN. This is the same as in CBR + normal CDH.
- Case adaptation: During training, pairs of training cases are used to train an adaptation neural network NN_a to produce a solution difference given a problem difference and a context. During testing, the problem difference between the query and the retrieved case is calculated. The problem description of the retrieved case is used as the context. The problem difference and context are provided as input to NN_a, which generates a solution difference. This solution difference is added to the retrieved solution to produce the final solution.

We note that the experiments use a minimalistic design for all three systems. A CBR system can take many forms involving design choices for aspects such as retrieval, adaptation, case maintenance, user feedback and intervention, etc.; similarly, a neural network

Table 1. Average MSE of systems for different values of *ncr* on the Airfoil dataset

	Number of cases removed (ncr)				
	100	200	300	400	500
3-NN	1.083	1.229	1.387	1.600	1.742
1-NN	1.374	1.698	1.845	2.184	2.403
CBR + network CDH	0.484	0.693	0.824	1.016	**1.168**
NN	**0.409**	**0.549**	**0.749**	**0.864**	1.267
CBR + normal CDH	1.175	1.893	1.919	2.522	2.593

system can vary by using different layers, numbers of neurons, activation functions, and connectivity. The CBR + network CDH and CBR + normal CDH systems are trained on the same adaptation examples, and the CBR + network CDH and NN systems use the same neural network structure for their networks. Our choices of models are based on the goal of a simple yet fair comparison, for which all models are given the same case base and similar computational characteristics.

All experiments are done under a constrained setting previously used by Leake and Ye [75], and described there. In this setting, the training phase is done **after** a test case is chosen. The top *ncr* (standing for number of cases removed) cases similar to the test case are temporarily removed from the training cases, so the systems are only allowed to train on not-too-similar cases. This procedure is repeated for each test case. In other words, the systems are retrained with a reduced training set so that the test case is "forced" to be novel with respect to cases previously provided to the system.

Experiment on Airfoil Data Set: For comparison with previous results [74], we performed the above experiment for the airfoil self-noise data set. In this data set, a problem description X is a vector of 5 numeric attributes describing wind and the airfoil blade section, and a solution description y is the sound level of the noise generated by the airfoil blade. The data set contains 1503 cases, 10% of which are used as the test cases. We use neighboring pairs of each case and its neighbor and 5000 random pairs to train CDH algorithms. The parameter *ncr* is chosen from the range of $\{100, 200, 300, 400, 500\}$.

The results are shown in Table 1. As *ncr* increases, all systems suffer to some extent because the queries become harder to solve. The system with the best result for each *ncr* is highlighted. We note that 3-NN consistently outperforms 1-NN, presumably because multiple retrievals decrease the influence of a potentially misleading nearest case. CBR + network CDH consistently outperforms 1-NN and 3-NN, which is expected because of the ability to do better adaptation. CBR + normal CDH performs poorly through all experiments. Given the better performance of CBR + network CDH, we hypothesize that the poor performance of normal CDH is due to inability to reliably select the right adaptation. A similar effect was observed by Craw, Wiratunga, and Rowe [69], where a suitable technique was needed to retrieve the best adaptation example. The NN system consistently outperforms the other systems, and the CBR system ranks second, except when $ncr = 500$ and the CBR + network CDH ranks first.

In this data set, there are plenty of samples for values in each dimension, and many cases share the same attributes. In such a setting the NN system can learn to solve novel queries. When enough cases are removed to impair the NN system, the adaptation knowledge and overall performance of CBR + network CDH are also impaired.

Table 2. Average MSE of systems for different values of *ncr* on the Car Dataset

	Number of cases removed (ncr)					
	0	1	2	10	50	100
3-NN	0.106	0.216	0.560	1.623	1.477	1.768
1-NN	0.065	0.040	0.497	1.677	1.527	2.039
CBR + network CDH	**0.029**	**0.030**	**0.049**	**0.257**	**0.237**	**0.256**
NN	0.035	0.080	0.108	0.413	0.544	0.560
CBR + normal CDH	0.076	0.067	0.489	1.672	1.487	1.973

Experiment on Car Features and MSRP Data Set The next experiment is carried out on the Car Features and MSRP Data Set from Kaggle [76]. A problem description contains fifteen numeric features of a car such as engine horse power, and nominal features such as make and model. A solution description is the Manufacturer Suggested Retail Price (MSRP) of the car. The cleaned data set contains 6629 cases, each with 1009 dimensions. The high dimensionality is due to the variety of one-hot encoded values in nominal attributes. 10% of the cases are used as test queries. We use neighboring pairs of each case and its neighbor and 10000 random pairs to train CDH algorithms. The parameter *ncr* is chosen from the range of $\{0,1,2,10,50,100\}$. Differently from previous experiments, we evaluate systems when $ncr = 0$. Due to the time cost of the testing procedure, when *ncr* is nonzero we only test 50 random queries per experiment.

Table 2 reports the text results. The best systems have comparable performance when $ncr = 0$ or $ncr = 1$. The CBR system substantially outperforms all other systems when $ncr \geq 2$. Due to the high dimensionality, removing cases heavily impacts the quality of the nearest neighbor retrieval, as shown by the k-NN systems when $ncr \geq 2$. Without similar cases, the NN system cannot learn effectively even if a minimal number of cases is removed, as shown by the NN system when $ncr \geq 2$. Nonetheless, the CBR system performs well for novel queries in such a high dimensional data set. Learning by the NN system may be less suited to this novelty, while the adaptation knowledge learned by the CBR + network CDH system from pairs of cases is less affected. Finally, we note that CBR + network CDH is essentially adapting the results of 1-NN. By comparing the two rows, we notice that often 1-NN performs poorly but the adaptation process often successfully estimates a correct result. This illustrates the benefit to accuracy of including the adaptation step.

18.6.2. Extending Neural Network CDH

Our initial architectures for NN-CDH were designed for adaptation of numeric attributes; adapting nominal attributes was an open challenge, made harder because of the lack of standard methods for expressing difference between nominal values and nominal adaptations in network architectures. To this end, Ye, Leake and Jalali [77] modified NN-CDH into an approach called C-NN-CDH which predicts a target solution (class label) based on a source case and a target problem of a query, without calculating the problem difference involving nominal features. However, this leads to a second issue. Neural network adaptation architectures such as C-NN-CDH take a query and retrieved case as input and generate a solution. This raises a question about whether the networks such as C-NN-

CDH truly learn case adaptation——in principle, they could learn to ignore the retrieved case and simply generate a solution based on the query.

In response, Ye, Leake and Crandall [62] introduced a simple method to express the difference between two one-hot encoded nominal values as a vector, and extended the NN-CDH adaptation approach [72] to predict a solution difference given a problem difference, where the differences can involve both numeric and nominal attributes. The new model is referred as the unified NN-CDH approach (simply referred to as "NN-CDH" for short in the rest of the text) as it unifies earlier work to handle both classification and regression tasks. In contrast to the previous C-NN-CDH, the unified NN-CDH guarantees learning adaptation knowledge from the difference between a pair of cases.

18.6.3. Evaluation of the Unified Model

Using the unified model we addressed two questions: Does NN-CDH consistently improve the result of retrieval, and how does a CBR system using NN-CDH perform when compared to a neural network of equivalent computational power? To address these questions, we compared the predictive accuracy of six methods: 1) a baseline neural network, 2) k-nearest neighbor (k-NN), 3) a CBR system whose retrieval is either 1-NN retrieval or retrieval with similarity based on a siamese network (SN retrieval) and whose adaptation is either a rule-based CDH or an NN-CDH. The rule-based CDH is a baseline adaptation method in which the case pairs are stored in an adaptation case base (as for normal CDH in Section 18.6.1). 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.

Network structure and implementations: The test 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 the baseline neural network takes in a problem description while NN-CDH takes in adaptation context and problem difference. This is to ensure fair comparison of the two models when they share similar computational power. The structure of the NN-CDH is detailed in Ye, Leake and Crandall [62]. 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 for regression and of categorical cross entropy for classification. Such designs are widely used for neural networks for regression and classification.

Both k-NN and 1-NN are default implementations from the scikit-learn package [78]. 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 for similarity measure, the siamese network first extracts features from two cases, and then estimates their similarity based on the extracted features. The feature extraction network is composed of three dense layers (of dimension 32, 32 and 16) with dropout layers (dropout rate = 0.1) between. For harder problems, the number of

neurons in each layer is multiplied by 4. Features extracted from pairs of cases are then passed to a 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 a Rectified Linear Unit (ReLU) activation function for regression or a sigmoid activation function for classification. The siamese network is trained with the mean average error loss for regression or the contrastive loss for classification.

Data sets: Experiments were conducted on five standard regression data sets (Airfoil, Car, Student Performance, Yacht, Energy Efficiency) and five classification data sets (Credit, Balance, Car, Yeast, Seeds), as well as on a set of artificial data. They also compared the extended NN-CDH with other models. Some trends of results paralleled those obtained in previous tests [72, 77], providing additional support for those trends as general characteristics of neural network adaptation.

18.6.4. When NN-CDH is Beneficial

Table 3 shows experimental results on regression data sets (additional results can be found in Ye et al. [62]). Each data set is tested with 10-fold cross validation, and on three different settings. The novel setting (X) simulates when problem descriptions of training cases are not similar to the query problem. The novel setting (Y) simulates when the solutions of training cases are not similar to the target 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). Both rule CDH and NN-CDH adapt a case provided by the retrieval method. If, for example, retrieval error is 6.94% and NN-CDH accuracy is 5.71%, it means that adaptation is improving the result of retrieval.

The experimental results suggest that NN-CDH is most useful when (1) retrieval is relatively good, in the sense of providing a starting point harmonized to learned adaptation knowledge (see [59, 79], and Section 18.7) and (2) queries are relatively novel, so adaptation is needed, and yet not too novel, which could require adaptation capabilities beyond the learned adaptation knowledge.

The results illustrated that multiple factors can affect performance. First, in some cases, good case retrieval alone was sufficient to surpass the full CBR process (with case adaptation) or the neural network. Second, counterintuitively, learned adaptation may actually worsen the retrieval result if the two processes are not harmonized (an example is row 2 in Table 3); we address this issue further and present an approach to addressing it in the following section. Third, directly integrating neural networks into a CBR system may lead to results comparable to the counterpart neural network, but when CBR is implemented without expert knowledge to exploit, it is not a "magic bullet" for superior performance. As discussed in the introduction, its appropriateness depends on overall needs and circumstances, such as the availability of knowledge to integrate into the system, the need to handle small data sets, and the need for interpretability.

18.7. Tuning CBR with Alternating Optimization of Network-Based Components

CBR research on learning for the case retrieval and adaptation steps generally focuses on one or the other, learning for each step separately with the natural assumption that

Table 3. Comparison of Model Error Rates (%) on Regression Data Sets [62]

	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**

strengthening any component of a CBR system will strengthen the system as a whole. However, the similarity/retrieval and case adaptation knowledge containers are intimately connected. For example, Smyth and Keane [79] show that adaptation-guided retrieval (AGR), basing retrieval directly on adaptability, can increase system efficiency, and Leake, Kinley, and Wilson [59] 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.

The choice of whether to strengthen retrieval of adaptation knowledge may also affect explainability. For example, a CBR system with strong adaptation might be capable of successfully adapting cases that are distant from an input problem, making retrieval of the closest cases less important. However, if system results are to be explained to a user based on retrieved cases (*e.g.*, [80]), presenting distant cases might be less compelling than presenting nearby ones. Therefore it may be preferable to learn to retrieve closer cases rather than to strengthen adaptation, even if either provides equivalent solution quality. On the other hand, if explaining by similarity is not important, it may be best to retrieve the most adaptable cases, no matter how dissimilar they appear to the user, for 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.

Because neural network based retrieval and adaptation have training procedures that use gradient descent to iteratively minimize loss functions, alternating optimization (AO) [81] can be applied to harmonize them for a desired balance. We tested the AO approach on five regression tasks, 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 based on a neural network based case difference heuristic (NN-CDH) approach for adaptation learning [72]. Experiments compared prediction error for three training schemes. The first is independent training of the two stages, where the retrieval and adaptation are trained using separate loss functions. The second is training adaptation first and then training retrieval, where both loss functions of retrieval and adaptation are based on the accuracy of the solution after adaptation. In other words, the adaptation process fully guides the retrieval and therefore the second scheme is referred to as AGR training. The third is alternating optimization, where the loss function of the retrieval is a weighted combination of the retrieval loss and adaptation loss. AO alternates between training retrieval and adaptation in multiple iterations, with minor changes per iteration.

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 while training both stages, the retrieval stage generally provides a good initial case, and adaptation further modifies the solution to be closer to the correct solution. Full details and results are presented in Leake and Ye [82].

18.8. Conclusion and Next Steps

Case-based reasoning is an interpretable reasoning and learning method able to exploit domain knowledge and to reason effectively from small data sets. However, its application may be hampered by difficulties of characterizing case features and obtaining knowledge to adapt cases. This chapter has illustrated how integrations with network approaches can help alleviate these issues. For feature generation, we have illustrated the benefits and limitations with examples from experiments both on using features from CNNs to supplement knowledge engineered features, and on using them on their own. We have also presented work aimed at better understanding where in networks to extract features and at increasing feature quality by local feature generation using multiple networks. We are now investigating how alternative network architectures affect the quality of extracted features.

For adaptation knowledge generation, we have presented research on neural network adaptation based on the case difference heuristic approach. Our approach trains a case adaptation network using both problem differences and problem context to learn adaptations for both numerical and categorical solutions. We have illustrated the benefits and limitations of this approach as well, showing that the approach is useful for increasing the ability of the system to handle novel queries.

A next step will be to extend the CDH approach by exploiting the strength of deep learning to generate feature descriptions. Rather than relying on a network to learn the solution difference for a rule to apply to a given solution difference, we intend first to use a deep network to derive the features to use to represent problems and solutions, and apply the case difference heuristic to learn adaptation rules for that new representation. This approach will use machine learning to refine both the vocabulary and adaptation knowledge of the CBR system.

The integration of neural network components into CBR can provide an additional benefit, in facilitating the adjustment of the components in concert, using alternating optimization. This realizes a longstanding goal of CBR of optimizing overall performance by selecting the cases most suitable for adaptation, given the system's case adaptation knowledge, as well as providing capabilities to improve CBR system interpretability. In future work we plan to extend the optimization process to guide choices for case retention and indexing.

Acknowledgments

This chapter is based on material excerpted and adapted from conference publications [9, 43, 48, 62, 72, 75] which include full details on algorithms, experimental designs, and results. We acknowledge support from the Department of the Navy, Office of Naval Research (Award N00014-19-1-2655), and the US Department of Defense (Contract W52P1J2093009), and helpful discussions with the Indiana University Deep CBR group.

References

[1] Sarker MK, Zhou L, Eberhart A, et al. Neuro-symbolic artificial intelligence: Current trends. CoRR. 2021;abs/2105.05330. Available from: `https://arxiv.org/abs/2105.05330`.
[2] Riesbeck C, Schank R. Inside case-based reasoning. Hillsdale, NJ: Lawrence Erlbaum; 1989.
[3] Kolodner J. Case-based reasoning. San Mateo, CA: Morgan Kaufmann; 1993.
[4] Leake D, editor. Case-based reasoning: Experiences, lessons, and future directions. Menlo Park, CA: AAAI Press/MIT Press; 1996.
[5] Richter M, Weber R. Case-based reasoning: A textbook. Berlin: Springer; 2013.
[6] Guidotti R, Monreale A, Ruggieri S, et al. A survey of methods for explaining black box models. ACM Computing Surveys. 2018;51(5):1–42.
[7] Rudin C. Please stop explaining black box models for high stakes decisions. Nature Machine Intelligence. 2019;1:206–215.
[8] Leake D. CBR in context: The present and future. In: Leake D, editor. Case-based reasoning: Experiences, lessons, and future directions. Menlo Park, CA: AAAI Press; 1996. p. 3–30.
[9] Leake D, Ye X. Harmonizing case retrieval and adaptation with alternating optimization. In: Case-Based Reasoning Research and Development, ICCBR 2021; Cham. Springer; 2021. p. 125–139.
[10] Schank R, Riesbeck C, Kass A, editors. Inside case-based explanation. Hillsdale, NJ: Lawrence Erlbaum; 1994.
[11] Leake D. Cognition as case-based reasoning. In: Bechtel W, Graham G, editors. A companion to cognitive science. Oxford: Blackwell; 1998. p. 465–476.
[12] Schank R. Dynamic memory: A theory of learning in computers and people. Cambridge, England: Cambridge University Press; 1982.
[13] Cheetham W, Watson I. Fielded applications of case-based reasoning. The Knowledge Engineering Review. 2005;20:321–323.
[14] Aamodt A, Plaza E. Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications. 1994;7(1):39–52.

[15] Simoudis E, Ford K, Cañas A. Knowledge acquisition in case-based reasoning: "...and then a miracle happens". In: Proceedings of the 1992 Florida AI Research Symposium; FLAIRS; 1992.
[16] Hanney K, Keane M. Learning adaptation rules from a case-base. In: Proceedings of the Third European Workshop on Case-Based Reasoning; Berlin. Springer; 1996. p. 179–192.
[17] López de Mántaras R, McSherry D, Bridge D, et al. Retrieval, reuse, revision, and retention in CBR. Knowledge Engineering Review. 2005;20(3).
[18] Leake D, Crandall DJ. On bringing case-based reasoning methodology to deep learning. In: Case-Based Reasoning Research and Development - ICCBR 2020. Springer; 2020. p. 343–348.
[19] Das R, Zaheer M, Thai D, et al. Case-based reasoning for natural language queries over knowledge bases. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics; 2021. p. 9594–9611.
[20] Kenny EM, Keane MT. Twin-systems to explain artificial neural networks using case-based reasoning: Comparative tests of feature-weighting methods in ANN-CBR twins for XAI. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence; 2019.
[21] Wettschereck D, Aha D, Mohri T. A review and empirical evaluation of feature-weighting methods for a class of lazy learning algorithms. Artificial Intelligence Review. 1997;11(1-5):273–314.
[22] Bromley J, Bentz JW, Bottou L, et al. Signature verification using a "Siamese" time delay neural network. International Journal of Pattern Recognition and Artificial Intelligence. 1993;7(04):669–688.
[23] Amin K, Lancaster G, Kapetanakis S, et al. Advanced similarity measures using word embeddings and siamese networks in CBR. In: Bi Y, Bhatia R, Kapoor S, editors. Intelligent Systems and Applications; Cham. Springer; 2020. p. 449–462.
[24] Hoffmann M, Malburg L, Klein P, et al. Using siamese graph neural networks for similarity-based retrieval in process-oriented case-based reasoning. In: Case-Based Reasoning Research and Development - ICCBR 2020. Springer; 2020. p. 229–244.
[25] Martin K, Wiratunga N, Sani S, et al. A convolutonal siamese network for developing similarity knowledge in the Selfback dataset. In: Proceedings of the International Conference on Case-Based Reasoning Workshops, CEUR Workshop Proceedings, ICCBR; 2017. p. 85–94.
[26] Mathisen BM, Aamodt A, Bach K, et al. Learning similarity measures from data. Progress in Artificial Intelligence. 2019;9:129–143.
[27] Hegdal S, Kofod-Petersen A. A CBR-ANN hybrid for dynamic environments. In: Proceedings of the ICCBR 2019 Workshop on Case-Based Reasoning and Deep Learning; 2019. p. 18–28. Workshops Proceedings for the Twenty-seventh International Conference on Case-Based Reasoning (ICCBR 2019), CEUR Workshop Proceedings, https://ceur-ws.org/Vol-2567/.
[28] Martin K, Wiratunga N, Sani S, et al. A convolutional siamese network for developing similarity knowledge in the selfBACK dataset. In: Proceedings of the ICCBR 2017 Workshop on Case-Based Reasoning and Deep Learning; 2017. p. 85–94. Proceedings of ICCBR 2017 Workshops (CAW, CBRDL, PO-CBR), Doctoral Consortium, and Competitions. http://ceur-ws.org/Vol-2028/.
[29] Nasiri S, Helsper JF, Jung M, et al. Enriching a CBR recommender system by classification of skin lesions using deep neural networks. In: Proceedings of the ICCBR 2018 Workshop on Case-Based Reasoning and Deep Learning; 2018.
[30] Samakovitis G, Petridis M, Lansley M, et al. Seen the villains: Detecting social engineering attacks using case-based reasoning and deep learning. In: Proceedings of the ICCBR 2019 Workshop on Case-Based Reasoning and Deep Learning; 2019. p. 39–48.
[31] Sani S, Wiratunga N, Massie S. Learning deep features for kNN-based human activity recognition. In: Proceedings of ICCBR 2017 Workshops (CAW, CBRDL, PO-CBR), Doctoral Consortium, and Competitions co-located with the 25th International Conference on Case-Based Reasoning (ICCBR 2017); (CEUR Workshop Proceedings; Vol. 2028). CEUR-WS.org; 2017. p. 95–03.
[32] Grace K, Maher ML, Wilson DC, et al. Combining CBR and deep learning to generate surprising recipe designs. In: Case-Based Reasoning Research and Development, ICCBR 2016; Berlin. Springer; 2016. p. 154–169.
[33] Turner JT, Floyd MW, Gupta KM, et al. Novel object discovery using case-based reasoning and convolutional neural networks. In: Case-Based Reasoning Research and Development, ICCBR 2018; 2018. p. 399–414.
[34] Kraska T, Beutel A, Chi EH, et al. The case for learned index structures. In: Proceedings of the 2018 International Conference on Management of Data, SIGMOD 2018; 2018. p. 489–504.
[35] Kim B, Shah J, Doshi-Velez F. Mind the gap: A generative approach to interpretable feature selection

and extraction. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2; Cambridge. MIT Press; 2015. p. 2260–2268; NIPS'15.
[36] Kim B, Rudin C, Shah J. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2; Cambridge, MA, USA. MIT Press; 2014. p. 1952–1960; NIPS'14.
[37] Li O, Liu H, Chen C, et al. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. CoRR. 2017;Available from: https://arxiv.org/abs/1710.04806.
[38] Barnett AJ, Schwartz FR, Tao C, et al. Interpretable mammographic image classification using case-based reasoning and deep learning. In: Proceedings of IJCAI-21 Workshop on Deep Learning, Case-Based Reasoning, and AutoML; 2021. Available from: https://arxiv.org/pdf/2107.05605.
[39] Chen C, Li O, Tao D, et al. This looks like that: Deep learning for interpretable image recognition. In: Advances in neural information processing systems 32. Curran; 2019. p. 8930–8941.
[40] Domeshek E. Do the right things: A component theory for indexing stories as social advice [dissertation]. The Institute for the Learning Sciences, Northwestern University; 1992.
[41] Leake D. An indexing vocabulary for case-based explanation. In: Proceedings of the Ninth National Conference on Artificial Intelligence; July; Menlo Park, CA. AAAI Press; 1991. p. 10–15.
[42] Schank R, Osgood R, Brand M, et al. A content theory of memory indexing. Institute for the Learning Sciences, Northwestern University; 1990. 1.
[43] Wilkerson Z, Leake D, Crandall D. On combining knowledge-engineered and network-extracted features for retrieval. In: Case-Based Reasoning Research and Development, ICCBR 2021; 2021. p. 248–262.
[44] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems; Vol. 1; 2012. p. 1097–1105.
[45] Turner JT, Floyd MW, Gupta KM, et al. NOD-CC: A hybrid CBR-CNN architecture for novel object discovery. In: Case-Based Reasoning Research and Development, ICCBR 2019; 2019. p. 373–387.
[46] Xian Y, Lampert CH, Schiele B, et al. Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI); Vol. 40; 2018. p. 1–14.
[47] Kenny EM, Keane MT. On generating plausible counterfactual and semi-factual explanations for deep learning. In: Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21). AAAI; 2021. p. 11575–11585.
[48] Leake D, Wilkerson Z, Crandall D. Extracting case indices from convolutional neural networks: A comparative study. In: Case-Based Reasoning Research and Development, ICCBR 2022. Springer; 2022.
[49] Zhou B, Lapedriza A, Xiao J, et al. Learning deep features for scene recognition using places database. In: Advances in Neural Information Processing Systems 27 (NIPS); 2014. p. 487–495.
[50] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. CoRR. 2014;Available from: https://arxiv.org/abs/1409.1556.
[51] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision. CoRR. 2015;Available from: https://arxiv.org/abs/1512.00567.
[52] Huang G, Liu Z, van der Maaten L, et al. Densely connected convolutional networks. CoRR. 2016; Available from: https://arxiv.org/abs/1608.06993.
[53] Wilkerson Z, Leake D, Crandall D. Leveraging SHAP and CBR for dimensionality reduction on the psychology prediction dataset. In: ICCBR XCBR'22: 4th Workshop on XCBR: Case-based Reasoning for the Explanation of Intelligent Systems at ICCBR-2022; 2022.
[54] Hammond K. Case-based planning: Viewing planning as a memory task. San Diego: Academic Press; 1989.
[55] Kolodner J. Improving human decision making through case-based decision aiding. AI Magazine. 1991 Summer;12(2):52–68.
[56] Craw S. Introspective learning to build case-based reasoning (CBR) knowledge containers. In: Machine learning and data mining in pattern recognition. Vol. 2734. Springer; 2003. p. 1–6.
[57] Shiu S, Yeung D, Sun C, et al. Transferring case knowledge to adaptation knowledge: An approach for case-base maintenance. Computational Intelligence. 2001;17(2):295–314.
[58] Craw S, Jarmulak J, Rowe R. Learning and applying case-based adaptation knowledge. In: Case-Based Reasoning Research and Development - ICCBR 2001; Berlin. Springer; 2001. p. 131–145.
[59] Leake D, Kinley A, Wilson D. Learning to integrate multiple knowledge sources for case-based reasoning. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. Morgan

Kaufmann; 1997. p. 246–251.
[60] Leake D, Powell J. Mining large-scale knowledge sources for case adaptation knowledge. In: Weber R, Richter M, editors. Proceedings of the Seventh International Conference on Case-Based Reasoning; Berlin. Springer Verlag; 2007. p. 209–223.
[61] Minor M, Bergmann R, Gorg S. Case-based adaptation of workflows. Information Systems. 2014; 40:142–152.
[62] Ye X, Leake D, Crandall D. Case adaptation with neural networks: Capabilities and limitations. In: Case-Based Reasoning Research and Development- ICCBR - 2022; Cham. Springer; 2022. p. 143–158.
[63] Jalali V, Leake D. Extending case adaptation with automatically-generated ensembles of adaptation rules. In: Case-Based Reasoning Research and Development, ICCBR 2013; Berlin. Springer; 2013. p. 188–202.
[64] McDonnell N, Cunningham P. A knowledge-light approach to regression using case-based reasoning. In: Proceedings of the 8th European conference on Case-Based Reasoning; Berlin. Springer; 2006. p. 91–105; ECCBR'06.
[65] McSherry D. An adaptation heuristic for case-based estimation. In: Proceedings of the Fourth European Workshop on Advances in Case-Based Reasoning; London, UK, UK. Springer-Verlag; 1998. p. 184–195; EWCBR '98.
[66] Wilke W, Vollrath I, Althoff KD, et al. A framework for learning adaptation knowledge based on knowledge light approaches. In: Proceedings of the Fifth German Workshop on Case-Based Reasoning; 1997. p. 235–242.
[67] D'Aquin M, Badra F, Lafrogne S, et al. Case base mining for adaptation knowledge acquisition. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07); San Mateo. Morgan Kaufmann; 2007. p. 750–755.
[68] Liao C, Liu A, Chao Y. A machine learning approach to case adaptation. In: 2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE); 2018. p. 106–109.
[69] Craw S, Wiratunga N, Rowe R. Learning adaptation knowledge to improve case-based reasoning. Artificial Intelligence. 2006;170:1175–1192.
[70] Brooks T, Pope D, Marcolini M. Airfoil self-noise and prediction. National Aeronautics and Space Administration, Office of Management, Scientific and Technical Information Division; 1989. (NASA reference publication; 1218).
[71] Dua D, Graff C. UCI machine learning repository ; 2019. Available from: `http://archive.ics.uci.edu/ml`.
[72] Leake D, Ye X, Crandall D. 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; 2021. Https://www.aaai-make.info/program.
[73] Jalali V, Leake D. Enhancing case-based regression with automatically-generated ensembles of adaptations. Journal of Intelligent Information Systems. 2015;:1–22.
[74] Liao C, Liu A, Chao Y. A machine learning approach to case adaptation. In: 2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE); 2018. p. 106–109.
[75] Leake D, Ye X. On combining case adaptation rules. In: Case-Based Reasoning Research and Development - ICCBR 2019. Springer; 2019. p. 204–218.
[76] Kaggle. Car features and MSRP ; 2016. Data retrieved from Kaggle,`https://www.kaggle.com/CooperUnion/cardataset`.
[77] Ye X, Leake D, Jalali V, et al. Learning adaptations for case-based classification: A neural network approach. In: Case-based reasoning research and development, ICCBR 2021; Springer; 2021. p. 279–293.
[78] Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research. 2011;12:2825–2830.
[79] Smyth B, Keane M. Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. Artificial Intelligence. 1998;102(2):249–293.
[80] Nugent C, Cunningham P. A case-based explanation system for black-box systems. Artificial Intelligence Review. 2005;24(2):163–178.
[81] Bezdek J, Hathaway R. Some notes on alternating optimization. In: Pal NR, Sugeno M, editors. Advances in Soft Computing — AFSS 2002; Berlin. Springer; 2002. p. 288–300.
[82] Leake D, Ye X. Harmonizing case retrieval and adaptation with alternating optimization. In: Case-Based Reasoning Research and Development - ICCBR 2021; Cham. Springer; 2021. p. 125–139.