Applying Class-to-Class Siamese Networks to Explain Classifications with Supportive and Contrastive Cases

Xiaomeng Ye, David Leake, William Huibregtse, Mehmet Dalkilic

Luddy School of Informatics, Computing, and Engineering
Indiana University, Bloomington IN 47408, USA
{xiaye, leake, whuibreg, dalkilic}@iu.edu

Abstract. Case-based classification is normally based on similarity between a query and class members in the case base. This paper proposes a difference-based approach, *class-to-class siamese network* (C2C-SN) classification, in which classification is based on learning patterns of both similarity and difference between classes. A C2C-SN learns patterns from one class C_i to another class C_j. The network can then be used, given two cases, to determine whether their similarity and difference conform to the learned patterns. If they do, it provides evidence for their belonging to the corresponding classes. We demonstrate the use of C2C-SNs for classification, explanation, and prototypical case finding. We demonstrate that C2C-SN classification can achieve good accuracy for case pairs, with the benefit of one-shot learning inherited from siamese networks.

Key words: Case-based reasoning, Classification, Inter-class pattern, Class-to-class, Difference measure, Prototypical cases, Siamese network, Similarity

1 Introduction

The success of neural networks in deep learning has underlined both their capabilities and limitations. As they are applied in safety-critical task domains, such as for autonomous and semi-autonomous vehicles, there has been much effort to exploit their capabilities while providing explainability (e.g., [8]), as well as interest in harnessing their capabilities while requiring fewer training examples. This has prompted interest in the case-based reasoning (CBR) [16] community in integrations of network methods with CBR (e.g., [10]). For example, case-based reasoning, paired with a "black box" system, can provide explanations based on similar cases [12].

Case-based classification is normally based on similarity between a query and an already-classified case—not on the differences between the query and cases from other classes. Interestingly, learning the difference between classes has been considered an essential part of human cognition [22]. For example, in counseling psychology, both schizophrenia and delusional disorder are psychoses. To learn

to classify instances of the two illnesses, a counselor may put the cases of the two classes side-by-side and focus on the differences: schizophrenia causes functional impairment, while delusional disorder may not; furthermore, schizophrenia is associate with hallucination, while delusional disorder may cause "non-bizarre" delusions—that is, perceptions that might ordinarily occur.

Early research on case-based interpretation focused extensively on both similarities and differences, through processes such as "compare and contrast" [1] and the use of differences for indexing, to replace a retrieved case with a nearby alternative [2]. Automated classification methods for learning similarity have made impressive progress (e.g., [19]), including the use of deep learning—in particular, applying siamese networks [3]—to learn similarity [18]. However, such work generally focuses on capturing similarity, with difference captured implicitly, rather than explicitly focusing on difference considerations.

This paper presents an approach that uses *inter-class patterns*, the patterns of similarity and difference between two classes, in case-based classification. Specifically, it applies a class-to-class (C2C) approach for learning to distinguish inputs that belong to different classes, implemented with siamese networks, to create what we call a class-to-class siamese network (C2C-SN).

The proposed C2C-SN method is a knowledge-light approach that can make and explain classification decisions using inter-class patterns. We demonstrate that it can support classification tasks, one-shot learning, an enriched form of explanation by cases, and prototype finding. For explanation, as in standard case-based reasoning (CBR) systems, a system using C2C-SN can explain classifications by retrieving a case similar to the query ("The patient Q has delusional disorder because a similar patient A also has delusional disorder"). However, a C2C-SN can also offer explanations for negative conclusions by providing a constrastive argument ("The patient Q does not have schizophrenia. Although both Q and schizophrenia patient B have delusions, Q's delusion is far less bizarre").

The paper is organized as follows. First, we present background, briefly describing the class-to-class approach to classification, siamese networks, and explanation by presentation of cases. We next describe the C2C-SN approach and an evaluation of its performance for classification and one-shot learning. We then illustrate its value for providing explanations for classifications and for generating prototypical cases, which can in turn be used for classification. We close with conclusions and future directions.

2 Background

2.1 The Class-to-class (C2C) Approach

The C2C approach is based on the assumption that there exist consistent similarity and difference patterns between different classes. Such inter-class patterns can be learned and reused for various purposes.

The C2C approach was initially tested as a feature weighting method (C2C weighting) for a k-nearest neighbors classification algorithm. In general, the traditional weighting methods assume that similar cases share similar (non)important

features [17, 23]. C2C weighting adds another assumption: that cases of different classes differ from each other, with respect to certain features, in a consistent manner. Unlike traditional weighting methods, which focus on finding the pattern of features within a class, C2C weighting aims to learn the patterns between pairs of classes and to apply these patterns as an additional information source for classification [24, 25]. C2C weighting can be used in classification, case retrieval, and explanation. However, C2C weighting has limitations as well, such as poor classification accuracy when the inter-class patterns involve hidden relations between features. The approach presented here does not use C2C weighting.

2.2 Siamese Networks

Siamese networks (SN) were introduced in the 1990's by Bromely *et al.* [3]. A siamese network consists of a pair of identical networks, each receiving different input vectors, but joined together at a distance measure layer, which outputs a result value. The twin networks share the same weights and configuration and, therefore, perform identical feature extraction on each of the two inputs. At the distance layer, the distance between the extracted features is computed and transformed to a value between 0 and 1 using a sigmoid function [4]. A siamese network can be used for classification, similarity assessment, as well as feature extraction in CBR [18].

In contrast to a neural network that learns to directly classify input cases into classes, a siamese network learns a similarity function between cases. While a neural network for classification needs many samples from every class, a siamese network may even require only a single instance of a new class to achieve one-shot learning ability [13].

An important benefit of siamese networks for learning from limited data is that training is based on pairs of cases, rather than only single cases [13]. If there are n cases in a case base, a neural network for classification can train on n input cases and their expected classifications, while a SN can train on $n \times n$ pairs of input cases and their expected similarities (which, in the absence of other information, can correspond to 0 if they belong to different classes and 1 if they belong to the same class). When given a single case from a new class, a neural network for classification can only train once for the new class, while a siamese network can train on $n + 1$ pairs of cases involving the new case, by pairing the new case with n old cases and itself. This enables much more rapid training.

2.3 Explanation by Cases in CBR

From the early days of CBR, the ability to explain the outputs of CBR systems by presenting the cases on which they are based has been an important benefit of case-based reasoning [14]. The value of such explanations has received experimental support [6]. A recent focus is explaining black-box systems such as neural networks by "twinning" them with CBR systems [11]. In the twin system, the artificial neural network (ANN) component is expected to produce

high-quality predictions while the CBR component provides explanations for the ANN's outputs. The explainability provided by the CBR system is post-hoc, meaning that the CBR system provides explanation of the ANN's output after the ANN makes the prediction. Displaying the conclusion along with the retrieved case is expected to boost the user's confidence compared to simply displaying the solution or displaying a rule used in finding the solution [6].

For explaining classifications based on cases, multiple approaches have been advanced for providing convincing evidence. Doyle et al. [7] suggest that cases between the query and the class boundary are more convincing support than the nearest neighbor of the query. By using a metric based on explanation utility rather than on similarity, their CBR system retrieves cases to best explain the class prediction. To aid users of a design feasibility assessment system in assessing the severity of design problems, Leake et al. [15] use *bracketing cases* (the most similar cases and without the problem) to illustrate the limits of the problem. Nugent et al. [20] illustrate an example of *a fortiori* arguments: A child pleading to her parent to see the movie Harry Potter will use the example of a much younger child who has seen the movie, instead of the example of a similar-age child, based on the assumption that "the older you are, the more likely you are allowed to see the movie." The authors frame this in terms of the concept of nearest unlike neighbor (NUN), the nearest neighbor of a different class. If the difference between NUN and the query case is large, this contrastive evidence suggests that the query is far from the class boundary and the prediction is thus convincing.

3 Class-to-class Siamese Networks

To combine C2C weighting with siamese networks for case-based classification, we propose a new network approach, the *class-to-class siamese network* (C2C-SN). The general structure of a C2C-SN is shown in Figure 1. The network is trained by pairs of cases to extract features that can be used to characterize a pattern between two specific classes. The twin networks extract features from the input cases. The difference between the extracted features is passed to a neural network learning the inter-class pattern, which outputs a number between 0 and 1 indicating the extent to which the extracted feature difference matches with the target pattern.

A premise of the approach is that, because the pattern between every pair of classes is unique, so is the feature extraction procedure for this pattern. For example, considering classifying psychotic diagnoses such as schizophrenia, delusional disorder, and schizotypal personality disorder: The difference between schizophrenia and delusional disorder may be focused on functional impairment; while the difference between schizophrenia and schizotypal personality disorder may be on delusions and illusions.

In the following, a C2C-SN learning the pattern from a class C_i to a class C_j will be denoted as a $C_i - C_j$ SN. When $i = j$, the corresponding siamese network $C_i - C_i$ SN learns the similarity pattern within the class C_i.

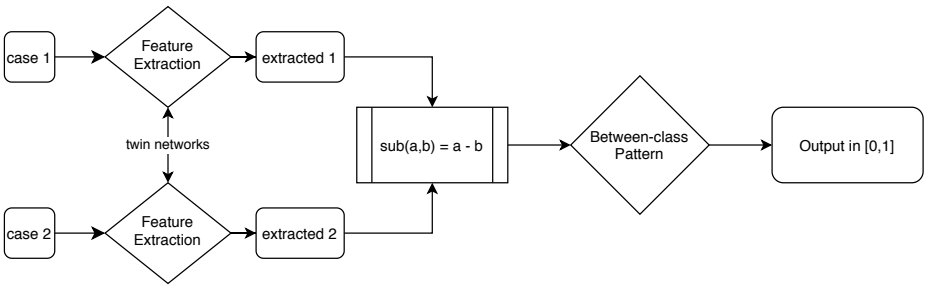


Fig. 1: The Structure of a C2C-SN

Each $C_i - C_j$ SN can be trained by back propagation. The features of a pair of cases $(case_1, case_2)$ are the input. A pair of cases is a positive pair (with output label 1) if the $case_1$ is an element of C_i and $case_2$ is an element of C_j. All other pairs of cases are negative pairs (with output label 0). If there are m classes, then there can be a family of up to m^2 C2C-SNs as there are m^2 pairs of classes. If the inter-class patterns are symmetric, then the number of patterns and networks is reduced by half.

3.1 Benefits of the C2C-SN Approach

A C2C-SN combines benefits from both the C2C approach and neural network learning. The C2C approach enables explaining membership in a class C_i by the fact that that the input case is different from C_j cases in a way that existing C_i cases are different from C_j cases, as shown in their C2C weighting [24]. In other words, C2C-SN can offer a supportive/contrastive explanation by providing a case of the same class or a different class.

The use of a siamese network provides several benefits beyond prior work on the C2C approach: (1) Hidden Features and Relationships: Prior work on C2C weighting assigns weightings to surface features to reflect inter-class patterns. Use of the network in C2C-SN enables learning patterns in both surface features and implicit features. (2) Flexible Patterns: A major flaw of C2C weighting is that one weighting can only capture one pattern. If there exist multiple patterns between two classes, multiple weightings are needed and training convergence is more difficult. The ability of networks to represent rich concepts enables the C2C-SN approach to capture complex relationships between two classes with one network. (3) Difference Direction and Magnitude: C2C weighting learns the direction of a inter-class pattern. In addition to the direction, C2C-SN also learns the strength of an inter-class pattern. (4) Lastly, inherited from siamese networks, a C2C-SN has one-shot learning ability.

3.2 Building a C2C-SN

Given a working SN for a task domain, a C2C-SN learning the $C_i - C_j$ pattern can be generated by converting the existing SN with following steps.

Assemble the network:

1. Reuse the upper layers of the network: If the SN performs well (as a classification tool or a similarity measure), then the upper layers are powerful enough to extract hidden features for the task domain. The same configuration (layers and connections) for the upper layers of the SN can be used in a C2C-SN, however the trained weights and biases will be different.
2. Reconfigure the lower layers of the network: A SN's lower layers are used to calculate the distance between two extracted features, while a C2C-SN's lower layers are used to learn the pattern between the two extracted features. Therefore the lower layers of the SN need to be replaced. Because the inter-class pattern might be hidden, a dense network is recommended.

Train the network:

1. Assemble and relabel training/testing pairs: Collect pairs of cases for training and testing. If the first case of the pair is of class C_i and the second case is of class C_j, then the pair label is 1, otherwise 0.
2. Retrain: Train the network weightings using back-propagation.

4 Experiments

This section illustrates and tests the performance of a C2C-SN for classification, one-shot learning, explanation, and prototypical case finding. Most experiments were conducted on the MNIST dataset. The dataset contains 60,000 training cases and 10,000 testing cases. Each case is a 28×28 image of a handwritten numerical digit, with each digit considered a class, providing ten classes labeled C_0 through C_9 for digits 0 through 9. Each digit appears in roughly the same number of cases.

The standard SNs and C2C-SNs were trained and tested on pairs of cases. Training pairs were assembled from the training set and testing pairs from the testing set.

We modified an existing SN implementation for classification in MNIST [5] to build C2C-SNs. In its original form, the upper layers first extract features from two cases, and the lower layer is a distance layer that computes the Euclidean distance between extracted feature vectors. The feature extraction layers are optimized by contrastive loss. This SN is referred to as the standard SN.

We reused the same initial configuration for the upper layers because the standard SN proved to be capable of extracting feature vectors for the classification task. The lower layers, however, are replaced with a subtraction layer, calculating the element-wise difference between two hidden vectors, followed by four fully connected ReLU (rectified linear unit) layers of 128 nodes, and a final output layer with a single node using sigmoid activation.

For a C2C-SN learning the $C_i - C_j$ pattern, we assembled $C_i - C_j$ pairs as the positive examples (labeled 1), and $C_x - C_y (x \neq i$ **or** $y \neq j)$ pairs as negative examples (labeled 0). Note that a $C_i - C_y$ pair and a $C_x - C_j$ pair are both negative examples. Last, we retrained the network using contrastive loss [9].

Table 1: Pair Accuracies for the MNIST Dataset

i=	0	1	2	3	4	5	6	7	8
$C_i - C_i$ SN	0.991	0.994	0.982	0.981	0.985	0.985	0.987	0.978	0.978
$C_i - C_{i+1}$ SN	0.992	0.987	0.982	0.981	0.984	0.985	0.980	0.978	0.969

4.1 Classification Accuracy of Pairs

Pair accuracy is defined as the percentage of correctly identified labels for positive and negative pairs. To illustrate the classification performance, we tested the classification accuracies of multiple C2C-SNs. The $C_i - C_i$ SNs illustrate the capability of C2C-SNs that learn the similarity patterns within each class, while the $C_i - C_{i+1}$ SNs illustrate the capability of C2C-SNs that learn inter-class patterns.

For both training and testing, 5,000 positive and 5,000 negatives pairs were used. For a $C_i - C_j$ SN, all of the 5,000 positive pairs were $C_i - C_j$ pairs. Of negative pairs, 35% pairs were $C_i - C_y$ pairs, 35% were $C_x - C_j$ pairs, and the remaining 30% were $C_x - C_y$ pairs, where $x \neq i$ and $y \neq j$. The breakdown of negative pairs is intended to emphasize pairs that partially mismatch. Table 1 shows the performance of the best validation run chosen among 20 epochs.

In comparison, the original implementation of the standard SN (from which we derived our C2C-SNs) achieved a pair accuracy of 97.2% after 20 epochs, each epoch with 60,000 positive pairs and 60,000 negative pairs [5]. Note that the meanings of accuracy are different for a standard SN and a C2C-SN, therefore they are not directly comparable: (1) For the standard SN, a positive pair is a pair of cases in the same class C_i, where i is unspecified (2) For a $C_i - C_j$ SN, a positive pair is of $C_i - C_j$, where i and j are determined.

4.2 One-shot Learning

We tested the one-shot learning ability of C2C-SNs in comparison with SNs. One-shot learning ability is the ability to learn when a minimum number of training cases for a certain class are presented.

In this experiment, we restricted the number of C_5 cases, n_5, in the training set, and compared the performance of the standard SN, the $C_5 - C_5$ SN, and the $C_5 - C_6$ SN. The number of C_5 training cases n_5 is set to 1, 10, 100, and 1,000, for four different experiments.

To ensure fairness and consistency in the comparison, each network was trained with 1,000 positive pairs and 1,000 negative pairs. The pairs were different for different networks. For the standard SN and the $C_5 - C_5$ SN: the positive pairs were 1,000 $C_5 - C_5$ pairs (which may include repeated pairs because C_5 cases are limited); the negative pairs were 350 $C_5 - C_j (j \neq 5)$ pairs, 350 $C_i - C_5 (i \neq 5)$ pairs, and 300 $C_i - C_j (i \neq 5$ and $j \neq 5)$ pairs. For the $C_5 - C_6$ SN: the positive pairs were 1,000 $C_5 - C_6$ pairs; the negative pairs were 350

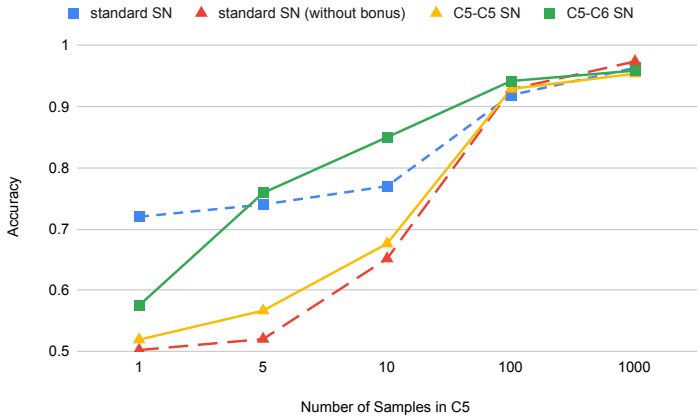


Fig. 2: The accuracies of networks under different one-shot learning constraint

$C_5 - C_j (j \neq 6)$ pairs, 350 $C_i - C_6 (i \neq 5)$ pairs, and 300 $C_i - C_j (i \neq 5$ and $j \neq 6)$ pairs.

Additionally, the standard SN received 100 $C_i - C_i$ training pairs for every i except when $i = 5$. The extra training pairs were available only for the standard SN but not for C2C-SNs, because the standard SN learns a distance measure across all classes. We tested the performance of the standard SN with and without the extra training pairs.

After training, each SN was tested with 1,000 positive pairs and 1,000 negative pairs assembled from the testing dataset, in which the number of C_5 cases was not restricted. We recorded the highest validation accuracy across 20 epochs of training and testing. The experiment was run 10 times and the average was used as the final results. The results are shown in Figure 2. We observe:

- The standard SN benefits strongly from the extra training pairs. A distance measure trained for all classes rather than only for C_5, giving additional data, benefits performance on C_5 as well. This contributes to the standard SN being superior when a minimum number of C_5 cases is available.
- The $C_5 - C_6$ SN performs better than the $C_5 - C_5$ SN. The positive examples for the $C_5 - C_5$ SN are pairs of few C_5 cases. Both sides of the $C_5 - C_5$ pairs have few cases and the knowledge available to exploit is minimal. On the other hand, the positive examples for the $C_5 - C_6$ SN are $C_5 - C_6$ pairs. While C_5 cases are limited, C_6 cases are abundant, leading to more variety of training pairs and thus more knowledge to exploit.
- The $C_5 - C_6$ SN has the fastest accuracy growth throughout the multiple experiments. As more C_5 cases are available, the first case of $C_5 - C_6$ pairs is no longer restricted to a single case, and the $C_5 - C_6$ pattern becomes easier to learn. In Figure 2, when $n_5 = 5$ and $n_5 = 10$, the $C_5 - C_6$ SN achieves superior accuracy, even in comparison to the standard SN with bonus training pairs.

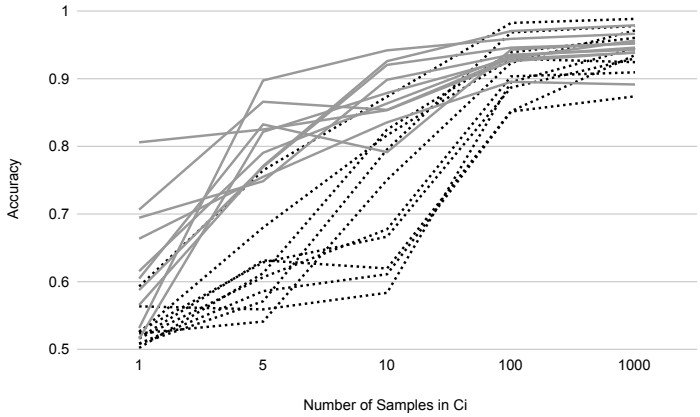


Fig. 3: The accuracy of C2C-SNs under different one-shot learning constraints. Dashed lines are $C_i - C_i$ SNs and solid lines are $C_i - C_{i+1}$ SNs

A second experiment was conducted using the same settings but for 10 $C_i - C_i$ SNs and and 10 $C_i - C_{i+1}$ SNs where $0 \leq i \leq 9$. The highest validation accuracy across 10 training epochs was recorded for each SN. As shown in Figure 3, $C_i - C_{i+1}$ SNs generally outperformed $C_i - C_i$ SNs in one-shot learning settings, and they eventually converged to similar accuracy when more cases were available.

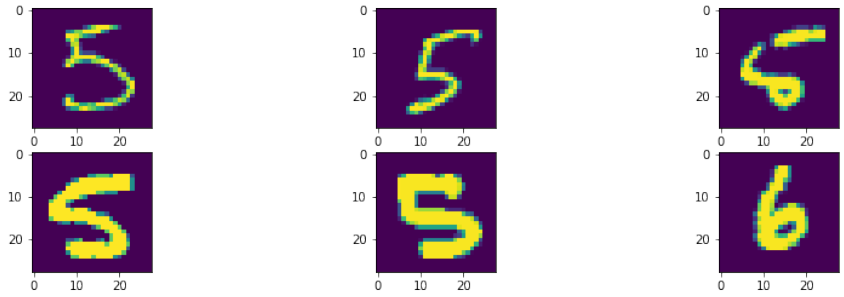
4.3 Explanation By Cases

Here we illustrate the ability of C2C-SNs to support explanation. We start with showing conventional CBR explanation (by presenting a similar case), followed by explaining contrastively (by presenting a relevant different case from a non-target class). Finally, we demonstrate its ability to find a prototypical case.

Explanation by a Similar Case in the Target Class: Given a query q, we can pair it with C_i cases to form $q - C_i$ pairs for each i, then apply a $C_i - C_i$ SN to the pairs. The highest activation achieved by a $q - C_i$ pair indicates q is of class C_i. The second case of the $q - C_i$ pair is a similar case of class C_i, thus offering an explanation by a similar case in the target class. This follows the usage standard SNs for classification.

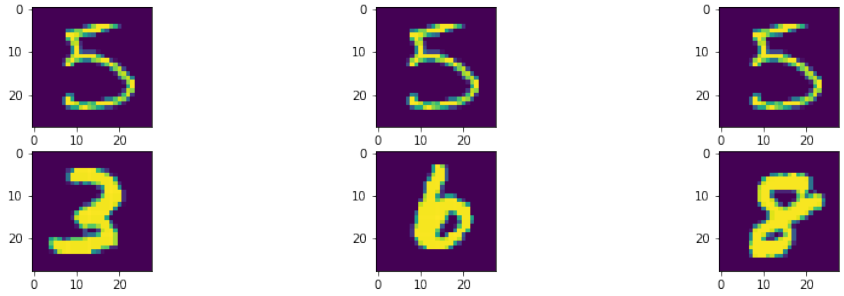
Figure 4a and Figure 4b show examples of explanation by a similar case. Figure 4c shows a misclassification where a badly written digit 5 was misclassified as 6. In this experiment, when the activation threshold was lowered to 0.9, there were 4,998 instances of digit 5s and 5,824 instances of digit 6s achieving the activation threshold. This shows that although classification by C2C-SN is not perfect, it has the potential to indicate the query case as an outlier which is difficult to classify.

Note that the $C_5 - C_5$ pairs were not selected based on their similarity, but based on the extent of which the pattern matches an average pattern between



(a) (b) (c) 5 misclassified as 6

Fig. 4: The images of different 5s (top), and their paired cases (bottom) achieving highest activation in $C_i - C_i$ SNs



(a) 5 classified by 3 (b) 5 classified by 6 (c) 5 classified by 8

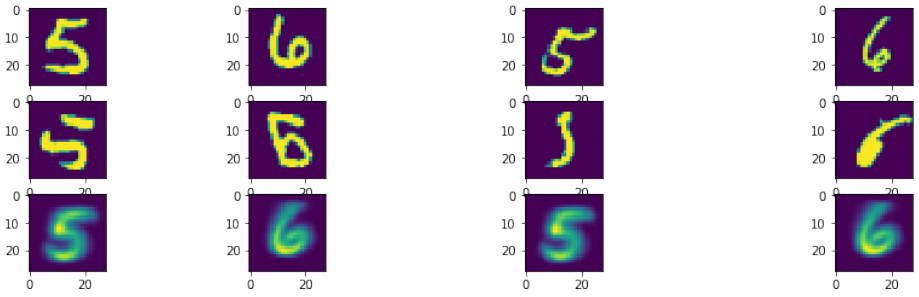
Fig. 5: The images of a single 5 (top), and its paired cases (bottom) achieving highest activation in $C_i - C_j$ SNs

two digit 5s. The second cases of $C_5 - C_5$ pairs in Figure 4a and 4b were not necessarily the most similar cases in terms of surface features.

Explanation by a Different Case in the Non-target Class Given a query q, for each i and j ($i \neq j$), we can pair q with C_j cases to form $q - C_j$ pairs, and then use a $C_i - C_j$ SN on the pairs. The highest activation of a $C_i - C_j$ SN achieved by a $q - C_j$ pair suggests q is of class C_i. In this scenario, the second case of the $q - C_j$ pair is not a similar case, because it is of class C_j instead of class C_i. Each $q - C_j$ pair provides an explanation with a contrastive argument.

In this experiment, the top row of Figure 5 shows the query q, a digit 5. The bottom row of Figure 5 shows the paired cases in digit 3s, 6s, and 8s achieving the highest activation.

The second cases of the high-activation $q - C_j$ pairs are often not the most standard C_j cases, but rather the C_j cases that magnify the difference between C_i and C_j *when they are compared to the query* q. For example: (1) In Figure 5a,



(a) $C_5 - C_6$ (b) $C_5 - C_5$ (c) $C_6 - C_6$

Fig. 6: The most and least prototypical cases found from C2C-SNs. Top: prototypical cases; Middle: least prototypical cases; Bottom: average of cases

the digit 3 has a large upper curve in the top right portion; (2) In Figure 5b, the digit 6 has no horizontal bar in the top portion and no sharp turn in the top left portion; (3) In Figure 5c, the digit 8 is large in the upper portion but small in the lower portion. These features are observable in $C_j (j = 3, 6, 8)$ but not usual in C_5. These features are also not present in the specific digit 5 in the pairs. Therefore the $C_5 - C_j$ pairs exemplify the $C_5 - C_j$ patterns.

4.4 Finding Prototypical Cases

A prototypical case is a case that best represents a class. Traditional machine learning methods find a prototypical case by clustering algorithms that find the center, or the average of cases of a class.

A $C_i - C_i$ SN can find a prototypical case a_i in C_i by finding the case with the highest average score in all $C_i - C_i$ pairs. A prototypical case of class C_i thus represents the center of the intra-class pattern of class C_i. In addition, a $C_i - C_j$ SN can find a prototypical case in C_i by finding the case b_i with the highest average score in all $C_i - C_j$ pairs. The prototypical case b_i represents the inter-class pattern of $C_i - C_j$, instead of the intra-class pattern of class C_i.

A $C_i - C_j$ SN can also be used to find the least prototypical case, the case achieving the lowest average score in all $C_i - C_j$ pairs.

Figure 6 illustrates the prototypical cases found in multiple C2C-SNs. Figure 6a shows the most and the least prototypical 5s and 6s in the $C_5 - C_6$ pattern. Figure 6b and 6c show the most and least prototypical 5s and 6s respectively in the $C_5 - C_5$ pattern and the $C_6 - C_6$ pattern. We observe:

- A prototypical case in a intra-class pattern is not necessarily close to the average case, as shown in Figure 6b and 6c.
- The least prototypical cases are outliers that do not conform to a C2C pattern. In the $C_5 - C_6$ pattern, intuitively, distinctive features are the upper left of the digit being a sharp bend (for digit 5s) or a curve (for digit 6s).

Figure 6a shows that the least prototypical 5 and 6 lack the corresponding features.

– The prototypical cases for the same class found from different C2C-SNs are not necessarily alike, because they are representing prototypes in different patterns. The same applies to the least prototypical cases.

To further illustrate the last point above, Figure 7 shows the prototypical cases in every class found from all C2C patterns. For reasons of efficiency, 1,000 samples from every class were used to assemble the case pairs. An entry on ith row and jth column shows the prototypical C_i case in the $C_i - C_j$ pattern. The diagonals are intra-class prototypes while the rest are inter-class prototypes.

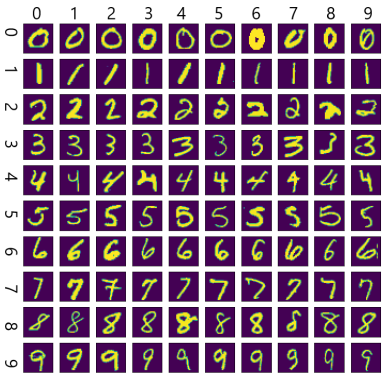


Fig. 7: The prototype matrix. A entry at (i,j) indicates the prototypical C_i case in the $C_i - C_j$ pattern

4.5 Using Prototypical Cases in Classification of a Case

A $C_i - C_j$ SN has two prototypical cases, one for C_i and one for C_j. Instead of pairing the query with each case, the CBR system can classify a query case by pairing the query with prototypical cases of C2C-SNs and finding the highest activation pairs.

For reasons of efficiency, a prototypical case for C_i in the $C_i - C_j$ pattern is found among pairs made from n C_i cases and n C_j cases. When $n = 300$, the number of pairs for training one C2C-SN is $n^2 = 90,000$. A query q is paired with the prototypical C_i case of the $C_i - C_j$ pattern, for every i and j. The $C_i - q$ pair with the highest activation is used for classification. The C2C-SN achieved an accuracy of 92.6%.

In Section 4.3, a digit 5 misclassified as a digit 6 is shown in Figure 4c. This pair actually achieved the highest activation possible, 1.0, in the $C_6 - C_6$ SN.

Table 2: Pair Accuracy for the Fashion MNIST Dataset

i=	0	1	2	3	4	5	6	7	8
$C_i - C_i$ SN	0.931	0.976	0.894	0.954	0.895	0.958	0.839	0.979	0.975
$C_i - C_{i+1}$ SN	0.954	0.934	0.922	0.923	0.943	0.799	0.903	0.976	0.971

Even though the correct classification pairs achieved high scores, a certain misclassification pair achieved the maximum score. To remediate this issue, instead of finding the maximum activation pair, we changed the algorithm to have the pairs vote, where each $C_i - q$ pair for the $C_i - C_j$ pattern with activation > 0.5 counts as one vote for the class C_j. This algorithm improved the accuracy to 94.23%. In addition, 98.37% of the test cases' true labels were within the top two votes.

Note that the classification accuracy for cases is different from the pair accuracy from Section 4.1. In comparison, the standard SN performing case classification by finding the highest activation pair achieves an accuracy of 96.9%; A neural network using the same structure of the upper layers of the C2C-SN and a final classification layer achieved an accuracy of 98.31%.

Last, we note that we built C2C-SN on a simple SN implementation with only dense layers. Preprocessing techniques such as deskewing, noise removal, blurring, and other layers like convolutional layers and pooling layers may be easily applied and could further improve performance. However, such refinements are not the focus of this paper.

5 Additional Results and Future Directions

To assess the performances of the models on a second dataset, experiments were conducted on the Fashion MNIST dataset, which contains images of 10 types of clothing [21]. On this dataset the standard SN achieves pair accuracy of 91.6% and case accuracy of 87.4%. The C2C-SNs pair accuracies are shown in Table 2. Using prototype voting the C2C-SNs achieve case accuracy of 84.2%, and 95.0% within top two votes.

The C2C-SN approach achieves good accuracy in the classification of pairs, and offers a new perspective for explanations and prototypical cases. However, its classification accuracy for cases does not equal existing techniques. It would be interesting to explore ensemble methods to unify all C2C-SNs for the purpose of case classification.

One future direction concerns applying C2C-SNs for outlier detection. When the C2C-SNs agree with each other, the prediction is of a single class. However, when C2C-SNs disagree, the result is a set of votes for multiple classes. Such disagreements may suggest outliers. Moreover, the votes may be used as attributes to describe unseen classes for zero-shot learning.

6 Conclusion

Traditional classification methods focus on learning and reasoning from information about the hidden patterns within a class, in the context of all classes, or rely on similarities between individual cases. Similarity is a well studied topic, with difference often simply defined as the complement of a similarity measure. However, differences between classes can be exploited in novel ways. This paper has argued for the potential of learning about inter-class patterns for classification. In service of this goal, it has shown how a standard siamese network design can be converted into a C2C-SN, by replacing the lower layers and re-purposing the network towards learning inter-class patterns. Experiments illustrated how the C2C-SN approach provides a novel method for classification, one-shot learning, explanation by contrastive cases, and finding prototypes.

Acknowledgment: This material is based upon work supported in part by the Department of the Navy, Office of Naval Research under award number N00014-19-1-2655.

References

1. Ashley, K., Rissland, E.: Compare and contrast, a test of expertise. In: Proceedings of the Sixth Annual National Conference on Artificial Intelligence. pp. 273–284. AAAI, Morgan Kaufmann, San Mateo, CA (1987)
2. Bareiss, R.: Exemplar-Based Knowledge Acquisition: A Unified Approach to Concept Representation, Classification, and Learning. Academic Press, San Diego (1989)
3. Bromley, J., Bentz, J.W., Bottou, L., Guyon, .I., LeCun, Y., Moore, C., Sackinger, E..u., Shah, R.: Signature verification using a siamese time delay neural network. International Journal of Pattern Recognition and Artificial Intelligence 7(04), 669–688 (1993)
4. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R.: Signature verification using a "siamese" time delay neural network. In: Proceedings of the 6th International Conference on Neural Information Processing Systems. pp. 737–744. NIPS'93, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
5. Chollet, F., et al.: Mnist siamese (2015), code retrieved from keras.io, `https://keras.io/examples/mnist_siamese/`
6. Cunningham, P., Doyle, D., Loughrey, J.: An evaluation of the usefulness of case-based explanation. In: Case-Based Reasoning Research and Development, ICCBR-03. pp. 122–130. Springer-Verlag, Berlin (2003)
7. Doyle, D., Cunningham, P., Bridge, D., Rahman, Y.: Explanation oriented retrieval. In: Funk, P., González Calero, P.A. (eds.) Advances in Case-Based Reasoning. pp. 157–168. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
8. Gunning, D., Aha, D.W.: DARPA's explainable artificial intelligence program. AI Magazine 40(2), 44–58 (Summer 2019)
9. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2. p. 17351742. CVPR 06, IEEE Computer Society, USA (2006)

10. Kapetanakis, S., Martin, K., Wijekoon, A., Amin, K., Massie, S. (eds.): Proceedings of the ICCBR-19 Case Based Reasoning and Deep Learning Workshop CBRDL-19 (2019)
11. Keane, M., Kenny, E.: How case based reasoning explained neural networks: An XAI survey of post-hoc explanation-by-example in ANN-CBR twins. In: Case-Based Reasoning Research and Development, ICCBR 2019. Springer, Berlin (2019), in press
12. Keane, M.T., Kenny, E.M.: How case based reasoning explained neural networks: An XAI survey of post-hoc explanation-by-example in ANN-CBR twins. CoRR abs/1905.07186 (2019)
13. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML 2015 Deep Learning Workshop (2015)
14. Leake, D.: CBR in context: The present and future. In: Leake, D. (ed.) Case-Based Reasoning: Experiences, Lessons, and Future Directions, pp. 3–30. AAAI Press, Menlo Park, CA (1996)
15. Leake, D., Birnbaum, L., Hammond, K., Marlow, C., Yang, H.: An integrated interface for proactive, experience-based design support. In: Proceedings of the 2001 International Conference on Intelligent User Interfaces. pp. 101–108 (2001)
16. 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., Watson, I.: Retrieval, reuse, revision, and retention in CBR. Knowledge Engineering Review 20(3) (2005)
17. Marchiori, E.: Class dependent feature weighting and k-nearest neighbor classification. In: Ngom, A., Formenti, E., Hao, J.K., Zhao, X.M., van Laarhoven, T. (eds.) Pattern Recognition in Bioinformatics: 8th IAPR International Conference, PRIB 2013, pp. 69–78. Springer, Berlin (2013)
18. Martin, K., Wiratunga, N., Sani, S., Massie, S., Clos, J.: A convolutional siamese network for developing similarity knowledge in the selfback dataset. In: ICCBR (Workshops) (2017)
19. Mathisen, B.M., Aamodt, A., Bach, K., Langseth, H.: Learning similarity measures from data. Progress in Artificial Intelligence (Oct 2019)
20. Nugent, C., Doyle, D., Cunningham, P.: Gaining insight through case-based explanation. J. Intell. Inf. Syst. 32, 267–295 (06 2009)
21. Research, Z.: Fashion mnist (2020), data retrieved from Kaggle, `https://www.kaggle.com/zalando-research/fashionmnist`
22. Tversky, A.: Features of similarity. Psychological Review 84(4), 327–352 (1977)
23. 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 11(1-5), 273–314 (February 1997)
24. Ye, X.: 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. pp. 389–394 (2018)
25. Ye, X.: 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. pp. 353–358 (2019)