On Combining Knowledge-Engineered and Network-Extracted Features for Retrieval

Zachary Wilkerson, David Leake and David J. Crandall

Luddy School of Informatics, Computing, and Engineering, Indiana University
Bloomington IN 47408, USA
{zachwilk,leake,djcran}@indiana.edu

Abstract. The quality of case retrieval in case-based reasoning (CBR) systems depends on assigning appropriate case indices. Defining feature vocabularies for indexing is an important knowledge acquisition problem for CBR, often addressed by hand. The manual process may result in high-quality vocabularies, but at considerable effort and expense, and it may be difficult for non-symbolic input such as images. Recently, the ability of deep learning (DL) to identify important features has made it appealing for learning to assign case features. However, such methods may miss features apparent to knowledge engineers. This paper presents a case study on methods for combining benefits of both engineered and DL-generated features. It considers case-based classification of cases described by both symbolic features and images. It evaluates the power of both types of features individually, examines how quality of engineered feature information affects their combined benefit, and tests network methods to generate weights for their combination. Experimental results show that in the test domain under suitable circumstances, the combined approach can outperform either method individually.

Keywords: Case-Based Reasoning, Deep Learning, Indexing, Hybrid Systems, Knowledge Containers, Integrated Systems

1 Introduction

The performance of CBR systems depends critically on retrieving the right cases. This depends on the indices used to organize and retrieve cases (e.g., [10, 13, 21]), which in turn depend on the vocabulary of features from which indices can be constructed. The feature vocabulary may be generated through a knowledge acquisition process in which experts analyze a domain (e.g., [17]). However, relying on manual feature acquisition can be problematic. First, especially in instances where the domain is poorly understood, or in non-symbolic domains (e.g., classifying images), it may be hard to identify the right set of features for a feature vocabulary. Second, developing feature vocabularies may be highly expensive—and the expense may need to be repeated as vocabularies lose their appropriateness over time due to concept drift. Third, the situation assessment process required

to characterize inputs in terms of the vocabulary may be difficult, resulting in partial, erroneous, or noisy case descriptions.

Some of the previous problems can be alleviated by applying machine learning (ML) to feature selection and similarity assessment. For example, learning techniques may be used to identify features to consider [6] or assign feature weightings [4]. Recently, substantial effort has focused on the potential of DL approaches to generate features and feature weightings. For example, convolutional neural networks (CNNs) have been used to extract feature information from images [23] and tri-axis sensors [20]. In that work, rather than relying on human-engineered features and situation assessment, the CBR system imports feature information from a network and uses it as the sole feature source during retrieval.

Such methods facilitate feature generation and enable features to be tuned as data changes. However, they are not guaranteed to capture the deep relationships that may be contained in expert-generated features. Thus each approach has benefits and drawbacks. In domains where a set of knowledge-engineered (KE) features exists, it is natural to consider combining human-engineered and network-learned (NL) features extracted using ML techniques. This paper presents a new method for extracting NL features and a case study on combining symbolic KE features with features extracted from CNNs for a classification task. It addresses how the benefit of combining such features varies with symbolic feature quality. As the effectiveness of retrieval depends strongly on feature weightings (e.g., [1]), it also studies how feature weight learning can be applied when merging the two sets of features, and its benefit. Results show that in the test domain, which combines symbolic and image information, the combined approach can outperform either method individually. This performance increase can be augmented with certain weight-learning strategies, though results also suggest that the benefit may be primarily for low-dimensional spaces, so new strategies may be necessary to accommodate feature-dense spaces created by NL techniques.

2 Convolutional Neural Networks for Classification

As a reference for the architecture described later in this paper, we begin with a brief description of convolutional neural networks for image classification. A CNN for image classification begins with alternating convolution and pooling layers that identify common shapes, contrasts, etc. present in similar regions of images with the same class during training; these extracted features then are "flattened" into a single layer and passed through a dense multilayer perceptron (MLP) section connected into the final output layer. A graphic representation of this process is shown in Figure 1. CNNs may be applied broadly to multi-dimensional data (e.g., image data [23] or sensor data that tracks movement in three dimensions [20]), where their architecture enables processing and condensing of complex data into features based on data relationships. Such features then may be extracted from the CNN's internal structure and applied to the feature set in a case-based reasoner.

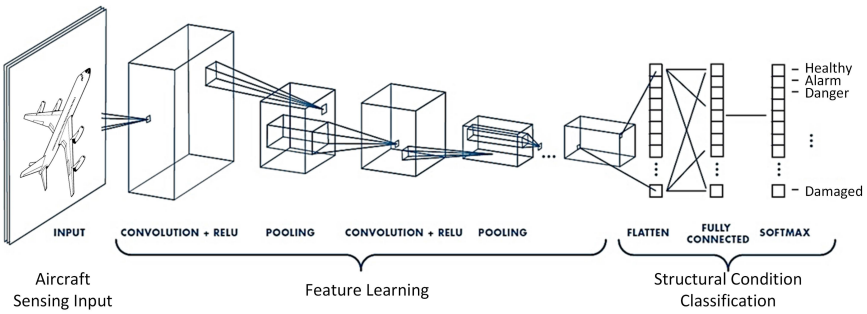

Fig. 1. Procedural diagram for a CNN in an aircraft sensor domain. Figure by Iuliana Tabian, Hailing Fu, and Zahra Sharif Khodaei is licensed under CC BY 4.0 [22].

3 Related Work

There has been much CBR research on feature learning using symbolic learning methods. Recently, there has been much interest in combining CBR and DL (e.g., [8, 14, 16, 19]) Much of this work focuses on CBR-DL hybrids in which DL components provide capabilities such as feature extraction to a CBR system.

Feature Learning A range of symbolic methods have been used to refine features/indices for CBR, often using knowledge-rich techniques. One example of feature learning strategies involves hybridizing with model-based learning to inform feature selection [3]. Bhatta and Goel apply a model-based system to select indices based on features simulated in the model. Barletta and Mark [2] propose explanation-based indexing. Cox and Ram [5] and Fox and Leake [6] apply introspective reasoning to refine features as expectation failures are encountered. Such methods rely on rich knowledge but can do powerful feature learning.

More recent research focuses on applying neural networks to directly infer similarity information from raw input data. Such methods do not require domain knowledge within the system (however, the dependence of network architecture on input structure makes many such methods domain-specific). Sani et al. present a system for human activity recognition that extracts features from a sensor and then uses a CNN to interpret the input data, which is represented in three dimensions [20]. The generated features are then compared against known wave form cases to infer the type, duration, etc. of activity that generated the sensory input data. Other approaches go a level of abstraction higher and look at the similarity functions themselves. Grace et al. [7] propose a hybrid system for creating plausible, yet unexpected, recipe designs. Their system applies DL techniques to infer 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. [15] use neural networks to learn similarity measures and also analyze different types of similarity metrics in depth. Outside of CBR, Kraska et al. [11] explore feature learning

using linear models and neural networks to aggregate and discriminate between features, showing performance benefits over traditional structures like B-trees, hashmaps, and bloom filters; they also propose combinations of ML techniques or multi-dimensional indices as potential means to greater efficiency.

Inductive feature learning is especially applicable in 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. [23] apply this to novel object recognition. A CNN architecture classifies inputs that correlate with known classes with high confidence; when encountering "new" 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 that do not have known classification labels by loosely classifying them in terms of one another. Turner et al. extract features for their CBR system from between the convolution/pooling and dense layers of the CNN; we take a different approach by extracting features just before the output layer (details in Section 4).

Learning Weights Many strategies exist to dynamically generate feature weights for case-based classification. Wettschereck et al. [24] present a survey of methods including hill-climbers, which modify feature weights according to a gradient to gradually maximize classification accuracy; genetic algorithms, which evaluate weights based on fitness as measured relative to the similarity calculation; and conditional probability models, which define weights based on the probability that a given class has the feature in question, among other wrapper and filter models. However, this is also a ripe domain for neural networks, which provide numerous opportunities to analyze relative importance of input features. In particular, Kenny and Keane [9] analyze multiple methods involving generating weights by taking advantage of neural network properties. One method generates weights by perturbing input elements individually and tracking the corresponding change in accuracy; this method builds on the assumption that feature importance correlates with the magnitude of accuracy change. We explore a version of this approach in Section 5, with slight modification for our test domain.

4 Bridging Engineered and Network-Extracted Features

This paper focuses on two aspects of bridging engineered features with features generated by DL:

1. Extracting features from DL models to use in concert with KE features
2. Using neural networks to learn feature weights for both KE features and network-learned features

It investigates these in the context of a case study of case-based classification. We propose a general model integrating three major components as shown in

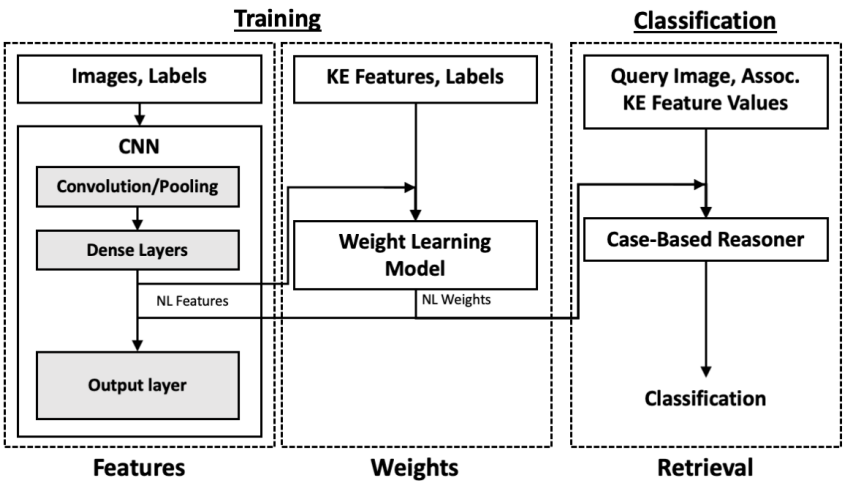

Fig. 2. Illustration of data flow through our model.

Figure 2. In the model, features are learned by a CNN from training data, feature weightings for the new features and existing knowledge engineered features are learned by another network, and the features and weights are used in a case-based classifier. Specifically, components are:

1. A CNN that extracts features from input data (e.g., images) to be used for case-based classification
2. A neural network that generates weights for both learned and knowledge engineered features, for the classifier similarity calculation.
3. A case-based classifier that uses a combination of engineered features and features from (1), weighted according to (2), for case retrieval.

CNN Architecture Our feature extraction CNN derives closely from the AlexNet architecture [12]. AlexNet is a foundational CNN architecture for image classification that employs a batch-normalized interleaving of five convolution and three pooling layers that are flattened into a network of two fully-connected dense layers that feed into the output layer. Our method deviates from other approaches on extracting CBR features from a CNN [20, 23] by extracting features from the dense layer preceding the output layer in the CNN, rather than before the dense layers. The rationale for this approach is as follows. An output node's activation in a neural network is determined by a weighted sum of the outputs from the previous layer. Thus, extracting features immediately after the final convolution layer neglects intermediate layers' modifications to the feature set ultimately used to perform classification, motivating extracting features from later in the CNN structure. Also, we remove the bias node from the CNN output layer. This ensures that NL features are not skewed during training, because a

bias node would factor into the weighted sum used for prediction but would not be extracted as a feature.

Sequential Architecture and Weight Generation Approaches To generate network learned weights, we apply a sequential architecture (i.e., successive fully-connected layers) mapping inputs corresponding with each feature directly to the classifying output layer.

1. **Directly extracted weights:** After training, local feature weights are generated for each case in the case base. For each feature, the local feature weight is the normalized absolute value of the weight of the link leading into the output node corresponding to that case's class (for later similarity calculations, only magnitude is important). This produces a localized set of feature weights for the cases that are unique on a per-class basis. Both linear and RELU activation functions were considered for the output layer before applying softmax to select a class prediction, with comparative results reported in Section 5.
2. **Weights from Perturbation:** Calculating weight values based on the shift in prediction accuracy as feature inputs (KE features only, NL features only, or both combined into a single input set) are perturbed individually, according to the following equation derived from Kenny and Keane [9]:

$$w_i = \frac{\Delta acc(f_i, \sigma) + \Delta acc(f_i, -\sigma)}{2} \tag{1}$$

   Here weight w_i is the average change in prediction accuracy that results from perturbing feature f_i by $\pm\sigma$. In contrast to extracting weights from the network directly, this generates a global set of feature weights applied to all cases, regardless of class.

5 Evaluation

Our evaluation addresses the following questions:

1. How is classification accuracy affected by degradation of reliability of input (KE features)?
2. How does using NL features in concert with KE features affect classification accuracy?
3. How do CBR retrieval weights based on NL weights influence classification accuracy for different combinations of NL and KE features?

5.1 Test Domain and Testbed System

Test Domain: As a test domain including both engineered features and non-symbolic information, we selected the Animals with Attributes 2 data set (AwA2)

[25]. This data set, designed for one-shot learning, includes over 37000 images across 50 animal classes; each class also has an associated feature vector of 85 features corresponding to 85 symbolic descriptions (e.g., herbivorous, desert habitat, quadrupedal, etc.). Each feature is assigned a continuous value in $[-1.0, 100.0]$. Because all instances of a class are assigned the same feature vector, with no variance, these feature vectors yield "perfect" classification accuracy when used for retrieval. To simulate imperfect situation assessment assigning symbolic feature values and/or symbolic feature characterizations that are not 100% predictive, we use perturbation. This is defined by a multiplier x that is applied to each feature value individually; x is defined as a random integer on the interval [1, n] with 50% probability or its inverse with 50% probability. We consider values of n on the interval [1,10).

Testbed System: As case adaptation is beyond the scope of our work, the testbed case-based classifier has no adaptation component. The classifier retrieves the nearest neighbor (i.e., 1-NN) using a weighted Euclidean distance metric for similarity calculations, using either local feature weights (for directly extracted weights) or global weights (for weights extracted by perturbations).

Properties of the chosen data set were reflected in parameter choices for the networks. The CNN architecture was modified to use 1024 nodes in the dense layers (rather than the traditional 4096) to concentrate extracted information into fewer features in an effort to make comparisons between KE and NL features more one-to-one. However, this was only partially possible, since smaller layers increase training time and decrease accuracy, to the point where epoch training steps do not converge. Even though we found a one-to-one comparison impossible as a result, the number of nodes was still used as it did not appear to negatively impact classification accuracy. The output layers of both the CNN and sequential architecture contained 50 nodes based on the number of AwA2 classes, and the input layer of the sequential architecture contained one node for each feature. Specifically, this translated to 85 nodes when considering only KE features, 1024 when considering only NL features, and 1109 when considering both feature sets in tandem. Lastly, we found that $\sigma = 0.8$ led to the highest retrieval accuracy in preliminary tests when generating weights using perturbation, likely due to the lack of variance in the KE feature set.

5.2 Preliminary Experiments to Set Network Parameters

Both NL features and NL weights depend on training the networks from which they are generated. We first determined the number of epochs to use, to balance the trade-off between predictive accuracy and low training time. For the CNN, models are trained on ten randomly-selected images from each of the fifty classes in the AwA2 data set, for a total of 500 images. Sequential architecture models are trained on the 1024 NL features generated by the CNN and/or the 85 KE features. All epoch training evaluations are performed for a number of epochs on the interval [10, 100] in increments of ten, with higher-resolution tests conducted

Epochs	Train Accuracy	Test Accuracy	Epochs	Train Accuracy	Test Accuracy
10	0.176 ± 0.028	0.045 ± 0.008	60	1.0	0.085 ± 0.015
20	0.765 ± 0.063	0.064 ± 0.009	70	1.0	0.083 ± 0.014
30	0.974 ± 0.010	0.076 ± 0.014	80	1.0	0.088 ± 0.014
40	0.997 ± 0.003	0.076 ± 0.012	90	1.0	0.088 ± 0.013
50	1.000 ± 0.001	0.081 ± 0.010	100	1.0	0.092 ± 0.013

Table 1. Comparing classification accuracy values ($\pm$ one standard deviation) for the sequential architecture for a given number of epochs, evaluated using the training set and an independent testing set.

for feature-dense spaces (i.e., requiring fewer then ten training epochs). Evaluations are performed thirty separate times and averaged to compute a sample mean and its standard deviation.

Tuning Results From these procedures, we chose the following parameter settings. For learning NL features, the CNN model is trained for 50 epochs. The sequential architecture is trained for 80 epochs when learning weights for KE features only, 5 epochs when learning weights for NL features or both NL and KE features combined, and 50 epochs when learning weights using feature perturbation. These decisions reflect values that maximize classification accuracy on the training set while also minimizing the number of epochs. Further research could investigate finer tuning parameters, such as learning rate and early stopping.

We note that for our modified AlexNet architecture, training appears to hit a point of diminishing returns after fifty epochs. This pattern holds for prediction both on the training set and on an independent testing set of 500 new images (Table 1). Furthermore, the accuracy on the testing set is significantly lower, suggesting that a general set of NL features is difficult to learn from the training set, and/or that the model overfits to the training set. However, considering that the model is designed to learn features that discriminate between cases of different classes, overfitting relative to a given case base may be acceptable so long as network training can efficiently be redone as new cases are added.

5.3 How Retrieval Accuracy Changes with KE Feature Degradation

Experiment Overview We explore relationships between retrieval accuracy and perturbation of the KE feature set. Specifically, retrieval accuracy is evaluated for leave-one-out experiments that are unweighted or weighted using linear or RELU activation functions for the sequential architecture to facilitate NL weight generation; each experiment is conducted for thirty iterations per value of n to establish a sample mean and standard deviation.

Sensitivity of Retrieval Accuracy to Feature Quality Results for these experiments are shown in Table 2. Predictably, retrieval accuracy decreases as

n	Unweighted	Linear	RELU
9	0.440 ± 0.028	0.235 ± 0.045	0.238 ± 0.041
8	0.458 ± 0.030	0.251 ± 0.044	0.253 ± 0.040
7	0.506 ± 0.025	0.266 ± 0.047	0.287 ± 0.035
6	0.567 ± 0.028	0.291 ± 0.056	0.313 ± 0.033
5	0.651 ± 0.028	0.363 ± 0.053	0.358 ± 0.057
4	0.785 ± 0.022	0.481 ± 0.059	0.449 ± 0.060
3	0.931 ± 0.014	0.625 ± 0.068	0.642 ± 0.046
2	0.999 ± 0.002	0.941 ± 0.035	0.924 ± 0.028
1	1.0	1.0	1.0

Table 2. Comparing classification accuracy values ($\pm$ one standard deviation) across the various perturbation levels. Results are shown for unweighted features and features weighted using linear and RELU output activation functions for the sequential network.

the perturbation magnitude increases, because a higher degree of noise is present in the KE feature set. Given this relationship, it is interesting to consider the possibility of using retrieval accuracy with KE features alone as proxy for the comprehensiveness/completeness of the KE feature set (and by extension, under what conditions its combination with a NL feature set might provide the greatest benefit). More research is required to provide a finer-grained assessment. We observe that the linear and RELU activation strategies appear less performant than the unweighted strategy; we explore this result more deeply in Section 5.5.

5.4 How using KE and NL Features in Concert Affects Accuracy

Experiment Overview We evaluate classification accuracy using KE and NL features in tandem by considering various perturbations of KE features in concert with the NL feature set against retrieval accuracy using each set individually. Each experiment is performed using unweighted leave-one-out testing on the case base of 500 cases using uniform feature weights for thirty iterations to establish a sample mean and standard deviation.

Benefits of Combining Features As shown in Figure 3, there frequently exists an interesting–if not always statistically significant–increase in classification accuracy when considering a combination of KE and NL features over either feature set's individual classification accuracy. This accuracy increase is most evident and most significant when both feature sets considered individually lead to similar classification accuracy values (i.e., when the perturbation magnitude is such that their accuracy values are similar) and when classification accuracy using NL features alone is higher than when using KE features alone.

The existence of this "accuracy bump" has multiple potential causes. For one, general accuracy trends and standard deviation patterns appear to be dominated by the NL features; this is unsurprising given that many more NL features (1024)

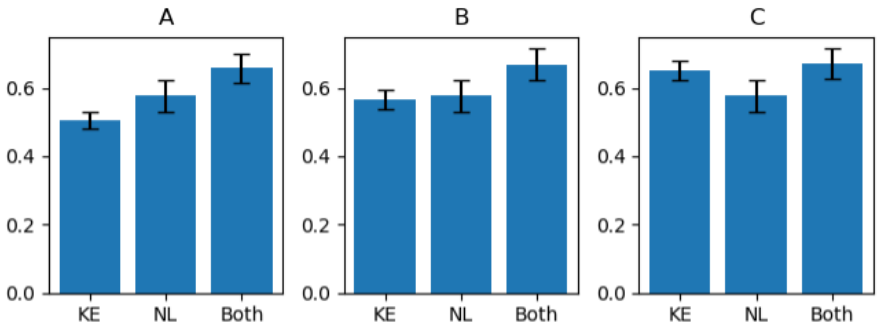

Fig. 3. Comparison of classification accuracy values for different feature perturbations ($n = 7$ A; $n = 6$ B; $n = 5$ C). Error bars represent one standard deviation relative to thirty iterations.

are considered than KE features (85). While it can be argued that this blunts the significance of the observed trend, it is important to provide further context. In particular, the modified AlexNet CNN produces novel features that capture aspects of the feature space not adequately represented in the KE feature set. That is, even though this trend may at least partially be attributed simply to the existence of more features, the NL features must also be significant/helpful in order to produce an increase in accuracy. The real question becomes whether the increase in accuracy when considering the union of the feature sets comes strictly from the existence of new features or from new interplay between the two feature spaces that creates a whole greater than the sum of its parts. This proved difficult to measure directly given the chosen domain. In preliminary tests a CNN having only 85 nodes per dense layer never converged (i.e., it could never outperform a random baseline).

Implications for Hybrid Systems These data suggest an interesting potential implication. Specifically, if this accuracy increase can be at least partially attributed to the nature of the two sets of features in a hybrid system (rather than simply an influx of new features alone), such a result could highlight direct hybridization of KE and NL features as a new avenue for accuracy improvement for CBR retrieval. That is, in the presence of additional environmental information (represented by the images in the AwA2 domain), a neural network may be able to generate features that are both novel when compared against the KE feature set and especially useful in concert with the KE feature set. This is naturally difficult to verify due to the well-documented inexplainability of neural network features, but future work focusing on detailed feature relationships and/or correlations, while likely computationally costly, might be able to identify useful correspondences between the feature sets for exactly this purpose.

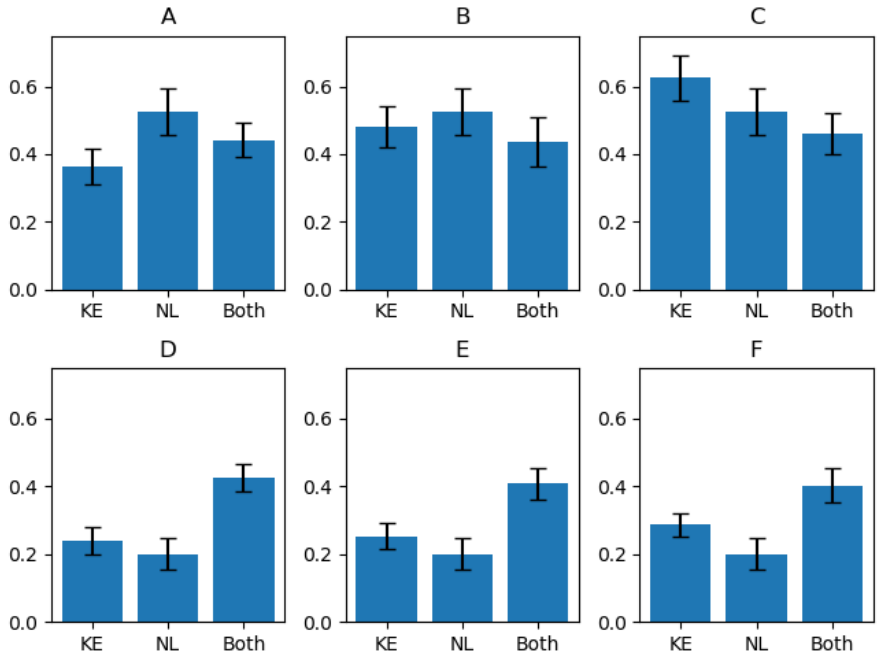

Fig. 4. Comparison of classification accuracy values for different feature perturbations ($n = 9$ D; $n = 8$ E; $n = 7$ F; $n = 5$ A; $n = 4$ B; $n = 3$ C). Error bars represent one standard deviation in experiments using a linear activation function (top) or RELU activation function (bottom) to generate NL weights.

5.5 How Learned Weights Further Influence Retrieval Accuracy

Experiment Overview For these experiments, features are weighted based on the strategies described for NL weights in the model section. Classification accuracy values for combined NL and KE features are evaluated against using each set of features individually, based on leave-one-out experiments repeated thirty times to establish a sample mean and standard deviation.

On Feature Weights and the "Curse of Dimensionality" While previous research appears to achieve reasonable success generating weights by perturbing KE features [9], such methods may not be applicable to feature-dense spaces. Specifically, when generating NL weights using feature perturbation on NL features, we observe that perturbing a single feature seldom changes the overall classification accuracy of the model, even when considering large values of σ. Therefore, many of the generated weights are at or near zero, crippling similarity assessment. It is possible that perturbing features in batches or using more complex neural network models might address some of these shortcomings; however, we suspect that existing weighting algorithms are significantly

less effective in high dimensional spaces created by generating NL features. Alternative weight generation algorithms may be applicable here (e.g., [1]), but research in additional domains is needed.

Weighting can Augment Retrieval Benefits In terms of overall retrieval accuracy, our initial results on using NL weights drawn directly from a network model in concert with combined NL and KE weights appear disappointing (Figure 4). However, trend behavior in these experiments is interesting. First, we note that accuracy values for the linear activation function are consistently at least as high as those for the RELU activation function. This is reasonable given that RELU would likely favor large-magnitude negative correlation weights less strongly than positive correlation weights. Curiously, however, the linear activation accuracy values suggest that combining KE and NL features produces a harmful effect. Contrast this with the RELU activation function, where combining KE and NL features produces the most significant relative accuracy improvement across all tests (Figure 4). So why did the weighting methods attempted not increase classification accuracy overall? This could be a result of the lack of variation in the raw KE features, so weighting provides little benefit for features that exist due to random perturbations; alternatively, this could simply be a symptom of the simplicity of the weighting algorithms investigated. However, the dramatic relative improvement in retrieval accuracy when generating weights using an RELU activation function suggests that deeper investigation into interplay between KE and NL features with NL weights is worthwhile.

6 Ramifications for Explainability

The previous experiments support the accuracy benefits of combining knowledge engineered and neural network features, especially for domains where additional features may be extracted from supplementary/environmental information. Unfortunately, while such features may be powerful and have the potential to capture aspects of the case base that humans cannot, this comes at a cost for explainability of retrieval. As the network-based features may be difficult to explain, it may be equally difficult to assess similarity judgments when they are based on network features.

Such a loss might not always be important. In a domain for which humans can assess similarity directly from the retrieved case, no explanation may be needed. In domains for which the combination of features results in substantial accuracy gains, the loss of explainability might be considered less important than gains in accuracy. However, the accuracy-explainability trade-off merits future research, and potential ways to mitigate it, such as integrating aides to interpreting feature assessments (e.g., CBR-LIME [18]) would be an interesting area for future research.

7 Conclusions

This paper presents results from a case study on methods for supplementing existing knowledge-engineered features with features learned from data with deep learning, with feature weightings for both learned by a neural network. The paper illustrates circumstances under which combining network-learned features with knowledge engineered features can produce classification accuracy values greater than either of the feature sets considered individually. It also points to challenges in weight generation for high-dimensional spaces, as may arise from learning large features sets from deep learning, and considers strategies to alleviate this difficulty.

These conclusions suggest numerous avenues for future work. First, testing across additional domains and network architectures and baselines is an essential next step. Also important exploring the tuning conditions under which combining KE and NL features produces maximum benefit, or under which the CNN generates features that are especially useful for retrieval. Investigating weighting strategies that perform better in feature-dense spaces is an another important step. Finally, an interesting question outside of the learning methods is how the inclusion of NL features and NL weights affects the explainability of the CBR model that applies them and how explanation issues might be addressed.

8 Acknowledgments

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).

References

1. Aha, D., Kibler, D., Albert, M.: Instance-based learning algorithms. Machine Learning **6**(1), 37–66 (1991)
2. Barletta, R., Mark, W.: Explanation-based indexing of cases. In: Kolodner, J. (ed.) Proceedings of a Workshop on Case-Based Reasoning. pp. 50–60. DARPA, Morgan Kaufmann, Palo Alto (1988)
3. Bhatta, S., Goel, A.: Model-based learning of structural indices to design cases. In: Proceedings of the IJCAI-93 Workshop on Reuse of Design. pp. A1–A13. IJCAI, Chambery, France (1993)
4. Bonzano, A., Cunningham, P., Smyth, B.: Using introspective learning to improve retrieval in CBR: A case study in air traffic control. In: Proceedings of the Second International Conference on Case-Based Reasoning (ICCBR-97). pp. 291–302. Springer, Berlin (1997)
5. Cox, M., Ram, A.: Introspective multistrategy learning: On the construction of learning strategies. Artificial Intelligence **112**(1-2), 1–55 (1999)
6. Fox, S., Leake, D.: Introspective reasoning for index refinement in case-based reasoning. The Journal of Experimental and Theoretical Artificial Intelligence **13**(1), 63–88 (2001)

7. Grace, K., Maher, M.L., Wilson, D.C., Najjar, N.: Combining CBR and deep learning to generate surprising recipe designs. In: Case-Based Reasoning Research and Development, ICCBR 2016. Springer, Berlin (2016)
8. 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 (09 2019)
9. Kenny, E.M., Keane, M.T.: 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)
10. Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann, San Mateo, CA (1993)
11. Kraska, T., Beutel, A., Chi, E.H., Dean, J., Polyzotis, N.: The case for learned index structures. In: Sensors. pp. 489–504 (2019)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems. vol. 1, pp. 1097–1105 (2012)
13. 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)
14. Martin, K., Wiratunga, N., Sani, S., Massie, S., Clos, J.: A convolutional siamese network for developing similarity knowledge in the selfBACK dataset. In: Sanchez-Ruiz, A.A., Kofod-Petersen, A. (eds.) Proceedings of the ICCBR 2017 Workshop on Case-Based Reasoning and Deep Learning. pp. 85–94. CEUR Workshop Proceedings (2017)
15. Mathisen, B.M., Aamodt, A., Bach, K., Langseth, H.: Learning similarity measures from data. Progress in Artificial Intelligence (10 2019)
16. Nasiri, S., Helsper, J.F., Jung, M., Fathi, M.: 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 (07 2018)
17. Osgood, R., Bareiss, R.: Automated index generation for constructing large-scale conversational hypermedia systems. In: Proceedings of the Eleventh National Conference on Artificial Intelligence. pp. 309–314. AAAI, Washington, DC (1993)
18. Recio-Garcia, J.A., Diaz-Agudo, B., Pino-Castilla, V.: CBR-LIME: A case-based reasoning approach to provide specific local interpretable model-agnostic explanations. In: Case-Based Reasoning Research and Development: 28th International Conference, ICCBR 2020. Springer (2020)
19. Samakovitis, G., Petridis, M., Lansley, M., Polatidis, N., Kapetanakis, S., Amin, K.: 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. pp. 39–48 (2019)
20. 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), Trondheim, Norway, June 26-28, 2017. CEUR Workshop Proceedings, vol. 2028, pp. 95–103. CEUR-WS.org (2017)
21. Schank, R., Brand, M., Burke, R., Domeshek, E., Edelson, D., Ferguson, W., Freed, M., Jona, M., Krulwich, B., Ohmayo, E., Osgood, R., Pryor, L.: Towards a general

content theory of indices. In: Proceedings of the 1990 AAAI spring symposium on Case-Based Reasoning. AAAI Press, Menlo Park, CA (1990)
22. Tabian, I., Fu, H., Khodaei, Z.S.: A convolutional neural network for impact detection and characterization of complex composite structures. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI). vol. 19 (2018)
23. Turner, J.T., Floyd, M.W., Gupta, K.M., Aha, D.W.: Novel object discovery using case-based reasoning and convolutional neural networks. In: Case-Based Reasoning Research and Development, ICCBR 2018. pp. 399–414 (2018)
24. 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 (1997)
25. Xian, Y., Lampert, C.H., Schiele, B., Akata, Z.: 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, pp. 1–14 (2018)