Examining the Impact of Network Architecture on Extracted Feature Quality for CBR

David Leake[0000−0002−8666−3416], Zachary Wilkerson, Vibhas Vats, Karan Acharya, and David Crandall

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

Abstract. Classification accuracy for case-based classifiers depends critically on the features used for case retrieval. Feature extraction from deep learning classifier models has proven a useful method for generating case-based classifier features, especially for domains in which manual feature engineering is costly or difficult. Previous work has explored how the quality of extracted features is influenced by structural choices such as the number of features extracted and the location/depth of extraction. This paper investigates how feature quality is influenced by another factor: the choice of the network model itself. We consider a selection of deep learning models for a computer vision classification task and test the accuracy of a case-based classifier using features extracted from them, both as the sole feature source and in combination with a supplementary set of knowledge-engineered features. Results suggest that feature quality reflects a trade-off between model complexity and training data requirements and provide lessons for the selection of deep learning architectures for feature extraction to support case-based classification.

Keywords: Case-Based Reasoning, Deep Learning, Feature Learning, Hybrid Systems, Indexing, Integrated Systems, Retrieval

1 Introduction

The accuracy of case-based classification depends on retrieving useful cases from the system's case base. In turn, retrieval efficacy depends on the quality of indices used. Traditionally, indices have been generated based on knowledge-engineered features supplied by domain experts, with feature values determined based on a combination of problem input and a situation assessment process performed by the case-based reasoning (CBR) system [9, 16, 27]. Indices based on knowledge-engineered features may capture key domain properties and are inherently interpretable, facilitating explanation of retrieval. However, knowledge engineering can be expensive, and there exist numerous domains for which hand-coded features are difficult to identify or only partially capture the domain. For example, in image processing domains, it is difficult to formulate effective feature vocabularies by hand.

Classifiers using deep learning (DL) have achieved impressive accuracy in hard-to-characterize domains such as computer vision (e.g., [6]) and their ability to learn effectively from raw data makes them promising for a wide variety of domains. However, DL models require considerable training data to achieve such accuracy, limiting their applicability for data-sparse domains. Furthermore, DL systems are "black-box" models without natural human-understandable justifications for their reasoning, limiting their application for domains in which high system trust is imperative. Considerable research seeks to mitigate this shortcoming through post-hoc explanation [11], but Rudin illustrates the limitations of post-hoc methods and shows that critical tasks may demand inherently interpretable methods [25].

CBR systems can learn from single examples and can leverage retrieval and case adaptation knowledge to operate effectively in data-sparse domains, and they can explain their decisions by presenting cases [17]. Consequently, methods that blend the complimentary strengths offered by DL and CBR approaches are appealing and are receiving much attention in CBR. Some methods integrate CBR concepts directly into DL models [3, 7, 20], some pair networks with CBR to explain DL-based decisions [13], and others apply network learning for tasks such as similarity assessment [23] and adaptation learning [21, 34] or coordinate network learning for both [19].

This paper begins with an overview of our DL-CBR hybrid approach applying DL systems as feature extractors for CBR retrieval. The results of such feature extractors still depend on the availability of sufficient training data for network learning. However, the ability of CBR to exploit additional knowledge containers [24] in concert with the DL features, including knowledge in the case base, case representation vocabulary, knowledge-engineered indices, and similarity and adaptation knowledge, reduces this dependence for the system as a whole and can result in superior performance [32]. Previous CBR research on extracting features from deep neural networks has taken the network to use as a given, exploring methods for extraction from that architecture. However, DL research studies multiple alternative architectures and parameterizations and has shown them to have strong impact on performance [14]. This raises two important questions for extracting features from DL networks to use for CBR: (1) which network architectures are most suitable as substrates for generating CBR features, and (2) how network parameters and training strategies may be fine-tuned to best support the CBR system. This paper presents, to our knowledge, the first attempt to address these questions.

Specifically, we outline an experimental approach for exploring the impact of DL architecture of the feature extraction model on feature quality for a selection of DL models. Experimental results show that more complex or recently developed DL models (e.g., that have higher task performance than previous models used to study CBR feature extraction) do not necessarily generate more useful features. The model architecture and parameters do have a significant impact on feature quality, but performance also depends on the balance between model complexity and the number of training examples. In this context,

the paper identifies some DL architectures that were better able to maintain performance with less data. In addition, it highlights the benefit of combining learned and knowledge-engineered features for overcoming some limitations of DL features learned from small data sets, and we have explored avenues for optimizing DL-based feature extraction for use in conjunction with knowledge-engineered features to increase transparency of the retrieval process [18, 32]. We begin by discussing related work, then present our general approach followed by the candidate network architectures, and close with evaluation, next steps, and conclusions.

2 Related Work

CBR retrieval performance depends significantly on indexing quality, which in turn depends on the feature vocabulary used for similarity assessment. Traditionally, features are generated through knowledge engineering, reflecting domain expertise [9, 16, 27]. However, manually developing features in this way can be expensive, and feature sets may incompletely or inaccurately capture domain properties when the domain is poorly understood. Initially, this issue was addressed through symbolic learning methods (e.g., [2, 4, 5, 8, 10]). However, the ability of DL systems to learn features makes integrating DL and CBR systems appealing to address this problem. Previous research has investigated integration approaches such as injecting CBR knowledge into the DL model directly using prototypes to facilitate more interpretable feature generation [3, 7, 20], twinning CBR systems with DL systems to retrieve explanatory cases [13], using a series of networks to classify problems hierarchically into subclasses until a single case is found [22], and using DL-extracted features for similarity calculations for retrieval [26, 30, 31].

2.1 Extracting DL Features for CBR

Previous studies have combined DL index extraction with CBR systems to develop hybrid systems, sometimes even enabling performance superior to end-to-end DL classification (e.g., [26, 31]). In such implementations, feature vectors are extracted prior to the output layer of a convolutional neural network (CNN) for use in the CBR similarity calculation for retrieval. Turner et al. apply this process for classification of novel images [30, 31]. Their approach trains and uses a CNN image classifier for end-to-end classification, while simultaneously associating each image with its corresponding extracted feature vector in a separate case base. Based on clusters that arise in the case base, the system can assign a relative classification for images for which the CNN has a low classification confidence and for classes it has not seen before. Sani et al. take this approach a step further, extracting features in the same way but using the CBR system as the classifier [26]; this facilitates a degree of explanation via presentation of the nearest-neighbor case that is not present in the end-to-end CNN model.

Both approaches assume that the CNN is the only source of similarity knowledge via feature extraction. In contrast, our previous work [32] uses knowledge-engineered features in concert with extracted features for similarity calculations, resulting in an increase in retrieval accuracy when knowledge-engineered features accurately–but incompletely–capture the domain. We have also explored relationships between both feature extraction location/depth in the CNN and number of features extracted and feature quality [18]. Results of that work suggested that extraction of features before the output layer of the network may result in the highest quality features. That work also introduced a novel multi-net architecture to minimize the number of extracted features required for strong performance.

3 A General Architecture for DL-CBR Integration

3.1 The Structure of Convolutional Neural Networks

Because our work explores DL-CBR performance for computer vision, we illustrate our feature extraction model in the context of CNNs, which are well-suited to vision tasks. CNNs process raw input data into refined features that can be used to classify the original image. At a high level, a CNN begins with a set of convolution and pooling steps designed to condense the multi-dimensional pixel data into numeric features. Each convolution layer consists of filters that are applied iteratively across regions of the input image; this process can be represented by:

$$O_{xy} = \sum_{i=-k}^{k} \sum_{j=-l}^{l} F_{ij}(I_{(x-i)(y-j)}) \tag{1}$$

In this equation, O_{xy} is a single value in the output feature map, I is the input image (and so indexing into I references a single pixel), and F is the convolution filter of size $(2k+1) \times (2l+1)$. The filter may be conceptualized as a matrix of weights that are applied to pixels based on their location in the image, with the results for all filter weights then summed to create the output value. Following the convolution layers are pooling layers, which further reduce the resolution of the post-convolution feature maps by selecting representative regional values.

After the last pooling layer, the remaining features are flattened into a linear feature vector. This vector passes through multiple densely-connected layers (that together resemble a multilayer perceptron network), with the final outputs being used as inputs to the output layer for classification. We note that not all DL computer vision models conform to all aspects of this outline, but the models discussed in this paper each apply this methodology to varying degrees; for each one we will discuss deviations. Additionally, while we present a proof-of-concept implementation for CNN-based feature extraction for classification, we believe that other DL models from which linear feature vectors may be extracted could leverage our approach, and that it could apply for regression as well.

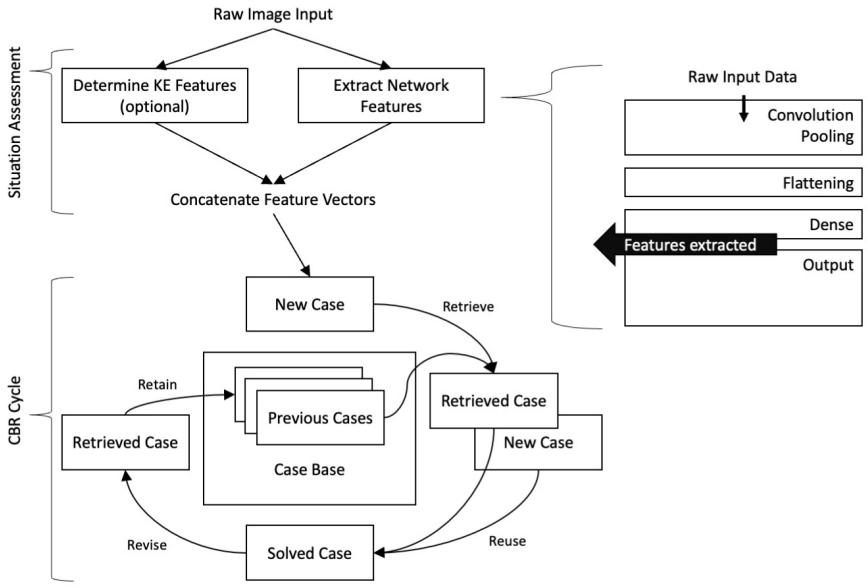

Fig. 1. Feature extraction dataflow between the CBR cycle (after Aamodt and Plaza [1]; bottom left) and the DL process (right). The figure illustrates a CNN structure for feature extraction but may be generalized to other DL models.

3.2 Extracting CBR Features from DL Models

CBR systems using nearest-neighbor similarity calculations for retrieval commonly characterize cases with linear feature vectors. This conveniently parallels the flattened feature vectors that are generated later in the CNN's data flow, and could also generalize well for other architectures such as artificial neural networks and multilayer perceptron models that employ linear layers for processed features. We have developed methods for extracting features for CBR retrieval from a CNN by removing the CNN's output layer from consideration post-training, and then extracting the outputs of the preceding layer for a target image (Figure 1) [32, 18]. These features may be augmented by concatenating them with a vector of knowledge-engineered features if available [32], with the final feature vector being associated with the image's ground truth class, as the solution part, to create a case.

It is possible to extract features from elsewhere in the CNN model, such as immediately after flattening (in which case all subsequent layers are removed from consideration), providing different feature information [18]. However, our previous work suggests that extracting after the densely-connected layers leads to the highest feature quality. This appears to apply especially for the more complex DL models for computer vision to be described in the following section. Such models follow a less linear conceptualization of data flow than typical CNN models (described in detail in Section 4). Because their layers may be intercon-

nected or promote parallel data flow pathways within the network architecture, there is less clear-cut conceptual justification for extracting features elsewhere than before the output layer.

4 Four Candidate Network Architectures to Compare for CBR Feature Extraction

Our previous research focused on applying the classic AlexNet model [15] for feature extraction [18, 32], because of its simplicity and compatibility with nearest-neighbor CBR retrieval. Other, more complex models have been shown to produce more accurate end-to-end classifications. Here we investigate whether basing extraction on such models may enable extracting features that are more useful for a CBR system. Specifically, we compare extraction from AlexNet with VGGNet [28], Inception V3 [29], and DenseNet [12], chosen as influential models designed to improve upon AlexNet classification performance in DL literature. For example, AlexNet's arrangement of densely-connected layers in the latter half of the model is convenient for extracting linear feature vectors for use in CBR similarity calculations, and VGGNet builds upon this with additional parameters/structural optimizations for training and feature refinement [28].

Inception V3 addresses a key shortcoming of CNNs like AlexNet and VGGNet, for which the size of the features to which they are sensitive is dependent on the size of the convolution filters used, which must be chosen as a parameter in advance. Inception takes a different approach by using modules that contain differently-sized convolution filters in parallel [29]. This enables features of varying granularity to be captured and concatenated together to be processed by the rest of the model. Furthermore, Inception is used as a feature extractor model for CBR by Turner et al. [31], giving particular interest to assessment of its properties compared to alternatives.

DenseNet addresses the possibility that—because fine-grained feature generation depends on a combination of atomic features from earlier in the network—training steps for earlier layers should be dependent on the outputs from later layers [12]. DenseNet compartmentalizes the typical CNN architecture in a series of "blocks" repeated throughout the model; each block connects to each other block, resulting in both a sequential flow of information reminiscent of AlexNet and VGGNet and an interconnected behavior through which various blocks influence one another during training.

As Inception and DenseNet have a less obvious layered structure than AlexNet and VGGNet, they are less naturally suited for feature extraction anywhere but at the end of the network. Consequently, for them our feature extraction approach extracts features from after the global average pooling (GAP) layer, which is positioned similarly to the densely-connected layers in AlexNet and VGGNet models. In addition, based on promising preliminary results from using VGGNet (potentially due to the presence of densely-connected layers), we investigate using a flattened layer followed by densely-connected layers instead of GAP for feature extraction for Inception and DenseNet (Figure 2).

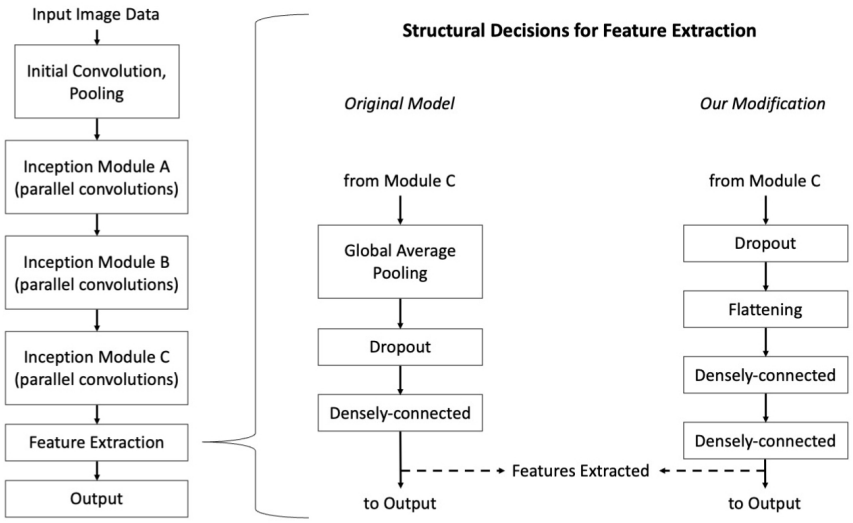

Fig. 2. High-level organization of Inception V3 [29] (left), and comparison of original layer organization post-convolution with our modification, applying densely-connected layers more directly rather than using GAP (right). All features are extracted immediately before the output layer.

Once the neural model is trained for end-to-end classification, features are extracted by removing the output layer from consideration post-training—for AlexNet and VGGNet—or by removing the final few layers from the network from consideration to expose the GAP layer outputs—for Inception V3 and DenseNet—and then extracting the outputs of the preceding layer (Figure 1). These features are provided for use by the CBR classifier.

5 Evaluation

We test how using each of the four DL models for feature extraction affects feature quality (and by extension, CBR classification accuracy) for various potential scenarios. The aim of the evaluation is twofold: (1) to better understand the characteristics of feature extraction from each model, and (2) to provide information to help CBR practitioners to select suitable DL architectures for feature extraction for their tasks. For all tests, we use CBR classification accuracy as a proxy for feature quality.

One scenario concerns the use of DL features in concert with knowledge-engineered features. Our previous research found that using extracted features from AlexNet with knowledge-engineered features could produce a net increase in classification accuracy compared to either alone [32]. When performing feature extraction in domains with existing retrieval knowledge, it would be desirable to

select DL models supporting this property. Additionally, it may be useful to identify models whose extracted feature quality suffers minimally from overfitting for comparatively small-data domains to which CBR systems may be applied.

Specifically, we investigate the following hypotheses:

1. **When training on small data sets, quality of extracted features may reflect the DL model overfitting.** To test the suitability of feature extraction for the sizes of data sets commonly used in CBR, we evaluate the models after training on comparatively small training sets. Because small data sets may result in DL models overfitting, we expect that the quality of extracted features will reflect that.
2. **Using a combination of extracted and knowledge-engineered features will lead to higher classification accuracy than with extracted features alone.** We expect that—similarly to our previous study using AlexNet [32]—high-quality knowledge-engineered features will augment the retrieval power of the extracted feature vectors.
3. **More complex DL feature extractors will generate better features.** More recent models with superior end-to-end classification performance (exemplified in our study by VGGNet, Inception, and DenseNet) will generate higher-quality features than older/simpler models (e.g., AlexNet).
4. **Models with densely-connected layers will generate higher-quality extracted features.** DL models that employ densely-connected layers prior to feature extraction (e.g., VGGNet and modified Inception and DenseNet structures) will generate higher-quality features (e.g., than post-GAP extraction), reflecting the hypothesis that densely-connected layers combine atomic features into more complex indices. This could apply to AlexNet as well, though we expect it will still generate lower-quality features due to a reduced number of available parameters.

5.1 Testbed Case-Based Classifier

We explore the performance of case-based classifiers using either extracted features alone or in concert with knowledge-engineered features. As this research focuses only on feature quality for CBR retrieval, we use a simple case-based classifier with no adaptation component. It performs 1-NN classification. Similarity calculations use unweighted Euclidean distance, with numeric analogs for nominal feature values provided in the data set (see Section 5.2).

5.2 Data Set Considerations and Model Training

Model training and evaluation use the Animals with Attributes 2 (AwA2) data set [33]. In particular, network models are trained, and the full system is tested, on image data in the AwA2 data set, and knowledge-engineered features are simulated by perturbing the per-class supplementary features from AwA2 as in Wilkerson et al. [32]. The training data set for each experimental iteration (30 iterations total) is defined from scratch, containing 500 images randomly selected

from and evenly distributed among AwA2's 50 classes. Each DL model is trained for 50 epochs, and the case base is instantiated by extracting features for each training image to create cases as in Section 3.2, and then storing each result.

For hybrid models, determining how to divide data into training and testing sets involves additional issues because of having to accommodate differences between the normal evaluation processes for CBR and DL. CBR evaluation is traditionally performed using leave-one-out testing, and so the differentiation between training and testing cases is temporary and implicit, rather than permanent and explicit, as it is for DL models. Leave-one-out testing can be seen as testing a CBR system on its training data using k-fold cross-validation where k is the number of cases/training examples. This raises the question of whether it is most appropriate to evaluate a DL-CBR hybrid classifier on its own training data or an independent test set. For evaluation on the training data, the CBR system will still classify "novel" examples via leave-one-out testing, but any example during testing will correspond to a training example from the DL model's perspective [18, 32]. This assumption could be desirable in applications where CBR system performance with extracted features is the principal focus. That is, especially for domains for which novel training examples occur infrequently, and/or for which the DL model may be easily retrained to consider novel examples, evaluating the DL-CBR model on training data approximates "ideal" or "upper bound" feature extraction performance for the DL feature extractor, moving evaluation focus to how the CBR system leverages extracted features.

By contrast, testing on an independent test set provides a stricter criterion for the DL system, assuming that DL overfitting may limit the applicability of a hybrid system evaluated only on training data. In the independent approach, evaluation occurs on a set of novel images, and the CBR classifier is evaluated based on classification accuracy for dynamically-generated cases for these images. We present results for both evaluation approaches in this paper.

6 Results and Discussion

Results presented in this section concern experiments both with and without supplementary knowledge-engineered features, as well as evaluated both on the training data and on an independent testing set of 500 images selected in the same way as the training data (Figures 3, 4, and 5).

These data suggest several broad trends. At the outset, we see ample evidence of DL model overfitting on the small training sets used, though some do still generalize relatively well. Using knowledge-engineered features in concert with extracted features generally can increase overall system accuracy, though the phenomenon is actually quite nuanced, as we discuss in Section 6.2. Whether including knowledge-engineered features or not, there appears to be no one "best" model; instead classification accuracy appears to fluctuate based on the conditions of case base instantiation and which of the two evaluation criteria is considered more important. Finally, replacing GAP with densely-connected structures does not appear to improve the quality of extracted features; in fact, the oppo-

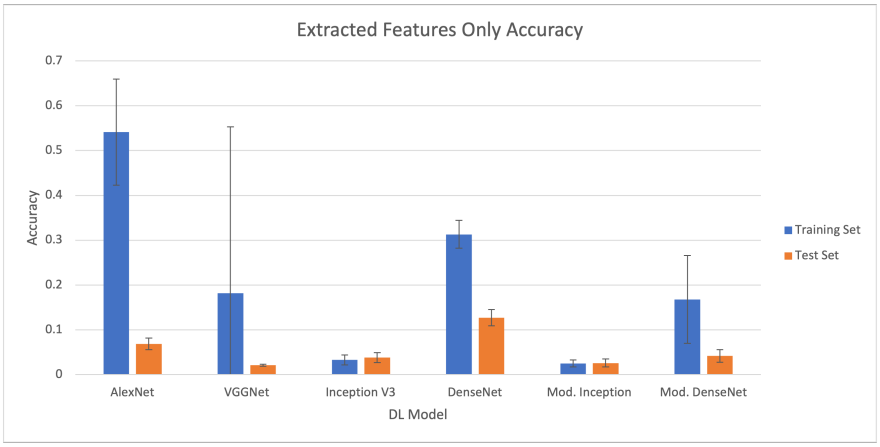

Fig. 3. CBR classification accuracy values when using extracted features alone, evaluated using leave-one-out testing on the training set or on an independently-selected testing set. All error bars represent one standard deviation.

site appears to be true with a few exceptions. We explore these conclusions in greater detail in the sections below.

6.1 Using Extracted Features Only and Model Overfitting

When performing CBR retrieval using extracted features only (Figure 3), we note significant differences in performance, both between feature extraction models and between evaluation approaches. Notably, the Inception model posts accuracy values on par with random guessing; VGGNet has a higher average accuracy value on the training set, but its significant standard deviation suggests inconsistency in model learning that often results in poor performance similar to that of Inception. In these instances, it seems that VGGNet and Inception are not learning well, likely due to an unsuitable ratio of trainable parameters to training examples. By contrast, AlexNet and DenseNet appear to generate features that facilitate higher classification accuracy values. One caveat of this observation is that given DenseNet's architecture, if the same number of features were extracted as for other models (2048), then the necessary structural change could have influenced the entire architecture through the interdependence of DenseNet's blocks; so, only 1024 features were extracted. As shown in Leake et al.[18], the number of features extracted can significantly impact CBR classification accuracy, so it is possible that the unmodified DenseNet accuracy values are slightly inflated.

In addition, it appears that each model overfits on the relatively small training data sets used. In particular, AlexNet generates features that appear to characterize the training data well but generalize poorly on novel data; by contrast, DenseNet appears to suffer least from overfitting. This supports our hy-

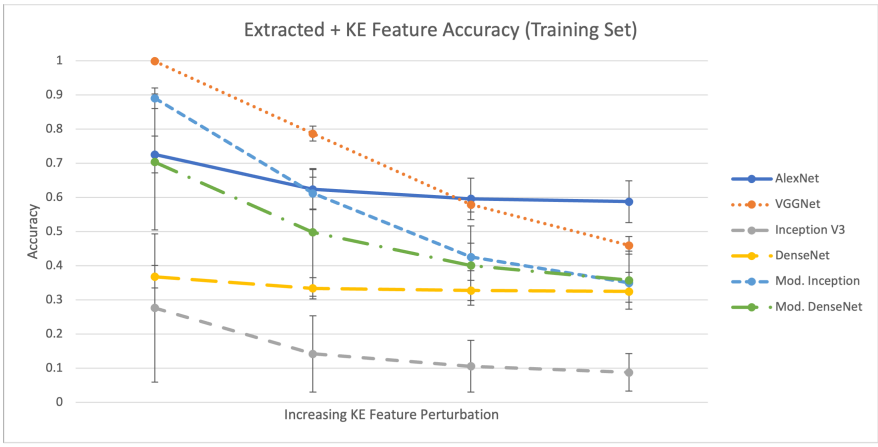

Fig. 4. CBR classification accuracy values when using extracted features and simulated knowledge-engineered (KE) features in concert, evaluated using leave-one-out testing on the training set. All error bars represent one standard deviation.

pothesis that small training set size does lead to overfitting that is reflected in extracted feature quality, and we conclude that DenseNet–while far from an ideal performer–is the most resilient model given minimal training data, as well as that having more training data would lead to improvements in feature quality and generalization across the board.

6.2 Potentially Inflated Accuracy Values with Knowledge-Engineered Features

At first glance, it appears that when knowledge-engineered features are concatenated onto the extracted feature vectors for retrieval (Figures 4 and 5), VGGNet and the modified Inception model are highly accurate, especially for minimally perturbed knowledge-engineered feature values. However, these methods have pronounced accuracy loss for higher degrees of perturbation. For evaluation on the training set, AlexNet outperforms either of these models as a feature extractor for higher perturbation values. Additionally, the accuracy values for both models are very similar when comparing accuracy on the independent test set versus on the training set, despite being dramatically lower for other models.

We hypothesize that these phenomena further point to VGGNet and Inception learning poorly from training based on the limited training set size. Specifically, in the absence of effective learning, randomly-initialized feature values in the network remain essentially random; when they are extracted and concatenated with knowledge-engineered feature values, the resulting feature sets have large subsets that are essentially equidistant from the corresponding feature subsets for all other cases due to their mutual near-randomness. In that instance, only the distances that correspond to knowledge-engineered features are sig-

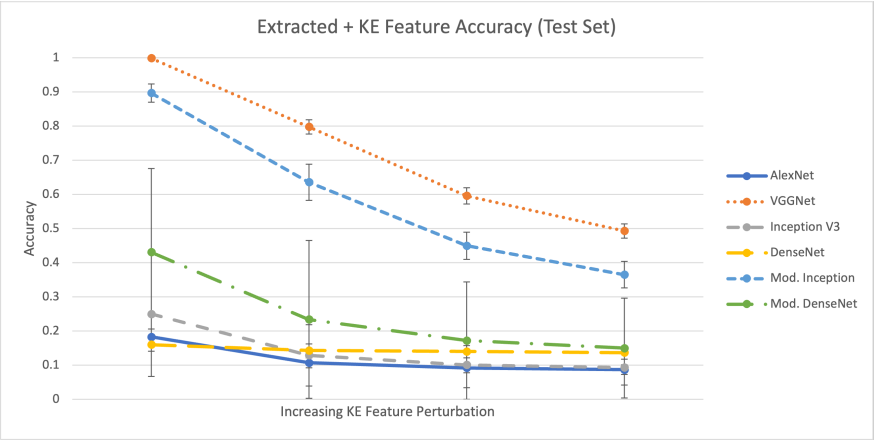

Fig. 5. CBR classification accuracy values when using extracted features and simulated knowledge-engineered (KE) features in concert, evaluated on an independently-selected testing set. All error bars represent one standard deviation.

nificant. This explains both the consistency between the two testing strategies and–as in Wilkerson et al. [32]–high accuracy values for smaller perturbations.

Concerning the other models, there is not a clear front-runner in terms of performance; the modified DenseNet appears to generate features that facilitate reasonably high-accuracy classification, but high standard deviation values make the significance unclear. That said, it appears that including knowledge-engineered features with extracted features results in accuracy increases for all models, supporting our corresponding hypothesis and our earlier work [32].

6.3 Best Approaches Overall for Feature Quality

These results suggest that there may be no "catchall" model for generating high-quality features for CBR retrieval. Our hypothesis that more advanced DL models generate higher-quality features is generally not supported for small training set sizes that we have used for our research to date; models that have large numbers of trainable parameters actually perform poorly because of large training data requirements. It may be the case that complex DL generates better CBR features given large training sets, but because of practical data limitations for many CBR domains, this might not be actionable.

Encouraging preliminary results shaped our hypothesis that densely-connected layers help generate useful features and led to the inclusion of modified Inception and DenseNet models that incorporated densely-connected layers during experimentation. However, it appears that densely-connected layers do not improve feature quality unilaterally. It is possible that the appended densely-connected layers do provide some benefit for DenseNet in the form of increased compatibility with knowledge-engineered features (Figures 4 and 5), but this is somewhat

unclear due to the different numbers of features extracted as discussed previously, as well as the high accuracy variability illustrated in the large standard deviation values. This leaves open the possibility that DL models may be modified/parameterized to align with CBR needs, but such modifications likely will need to be made on a per-model basis.

In sum, some models (e.g., VGGNet and Inception) clearly do not facilitate useful feature extraction under the tested circumstances, and models such as AlexNet and DenseNet show some promise–though they are prone to overfitting given minimal training data, as evidenced by their lower accuracy on the test set. The inclusion/appending of densely-connected layers is not an indicator of useful feature generation in general; careful parameterization considerations and interplay between the DL and CBR models still appear to be the dominant factor for architecture selection and modification. That said, using both knowledge-engineered and extracted features in concert still appears to improve CBR classification accuracy in general, with the caveat that when comparing results, the accuracy effects of high-quality knowledge-engineered features can mask deficiencies in model learning.

7 Next Steps: Other Models, Transfer Learning, and Integrated Training

This research explores a necessarily incomplete subset of DL models. With the speed of new discoveries and development of new implementations, novel DL architectures will continue to be available for further testing. Models such as MLP mixer and transformer architectures appear especially promising for feature extraction. They represent additional approaches for feature extraction for computer vision, and they are also useful in other broad domains (e.g., transformers for natural language processing); they are also promising candidates for linear feature vector extraction from similarly-shaped layers, potentially facilitating application of our DL-CBR methodology to other domains.

It may be rewarding to focus on addressing the relationship between model complexity/modeling power and larger training data requirements. This may manifest as evaluation of DL-CBR feature quality given different training set sizes (e.g., to quantify how CBR classification may reduce training data needs for the hybrid system); alternatively, using pretrained models appears promising for minimizing training data requirements. That is, a pretrained DL model may be specialized to the data set in question via transfer learning and then leveraged for feature extraction for CBR retrieval. In this way, the model may generate more useful features for the CBR system without having to significantly increase the size of the training set. Indeed, we are beginning to address this, and preliminary results in this direction appear extremely promising, even using the VGGNet and Inception architectures, which this study finds to be difficult to train in data-light scenarios (Table 1).

Finally, a tighter coupling of DL and CBR systems during model training may increase extracted feature quality as well. Specifically, rather than training

Extractor Model	Accuracy	St. Dev.
VGGNet	0.890	0.021
Inception	0.713	0.026

Table 1. Preliminary accuracy values using pretrained models for feature extraction, evaluating CBR classification accuracy on the training set (500 training examples–10 for each of the 50 classes) using extracted features only. Models are trained for 25 epochs, and ten trials are conducted for each model to determine standard deviation values. Accuracy values illustrate the potential for pretraining to increase system accuracy.

the DL model end-to-end independently before evaluating the CBR classifier using features extracted from it, it might be more appropriate to use the CBR system's classification loss to supplement or replace the end-to-end DL loss. As a result, weight updates in the DL system are sensitized to the needs of the CBR system, ideally resulting in higher-quality extracted features for CBR retrieval.

8 Conclusions

We presented a comparative analysis of feature quality for case retrieval features extracted from several DL models with the goal of maximizing classification accuracy, aimed at illuminating which DL models are most suitable to use as a basis for feature extraction in different scenarios. It is clear that this is a complex issue for which no all-purpose solution exists; model selection is highly dependent on the balance between DL model complexity and available training data, and the novelty of potential test data versus training data is important as well. This study supports prior observations about the effectiveness of using knowledge-engineered and network-extracted features in concert [32], and about the influence of DL structure on feature quality [18].

In addition to the avenues for future work discussed in Section 7, it would be useful to investigate additional domains involving image classification and supplementary knowledge-engineered features, feature weighting methods applied to extracted features (building on Wilkerson et al. [32]), and applications for DL-based feature extraction in CBR adaptation.

9 Acknowledgments

This work was funded by the US Department of Defense (Contract W52P1J2093009), and by the Department of the Navy, Office of Naval Research (Award N00014-19-1-2655).

References

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

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. Barnett, A.J., Schwartz, F.R., Tao, C., Chen, C., Yinhao, R., Lo, J.Y., Rudin, C.: Interpretable mammographic image classification using case-based reasoning and deep learning. In: IJCAI Workshops 2021 (2021)
4. 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)
5. Bonzano, A., Cunningham, P., Smyth, B.: Using introspective learning to improve retrieval in CBR: A case study in air traffic control. In: Case-Based Reasoning Research and Development: Proceedings of the Second International Conference on Case-Based Reasoning, ICCBR-97. pp. 291–302. Springer, Berlin (1997)
6. Chai, J., Zeng, H., Li, A., Ngai, E.W.: Deep learning in computer vision: A critical review of emerging techniques and application scenarios. Machine Learning with Applications **6**, 100134 (2021)
7. Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., Su, J.K.: This looks like that: Deep learning for interpretable image recognition. In: Advances in Neural Information Processing Systems 32, pp. 8930–8941. Curran (2019)
8. Cox, M., Ram, A.: Introspective multistrategy learning: On the construction of learning strategies. Artificial Intelligence **112**(1-2), 1–55 (1999)
9. Domeshek, E.: Indexing stories as social advice. In: Proceedings of the Ninth National Conference on Artificial Intelligence. pp. 16–21. AAAI Press, Menlo Park, CA (1991)
10. 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)
11. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Computing Surveys **51**(5), 1–42 (2018)
12. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks (2016), https://arxiv.org/abs/1608.06993
13. 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)
14. Khan, A., Sohail, A., Zahoora, U., Qureshi, A.S.: A survey of the recent architectures of deep convolutional neural networks. Artificial Intelligence Review **53**, 5455 – 5516 (2019)
15. 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)
16. Leake, D.: An indexing vocabulary for case-based explanation. In: Proceedings of the Ninth National Conference on Artificial Intelligence. pp. 10–15. AAAI Press, Menlo Park, CA (July 1991)
17. 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), http://www.cs.indiana.edu/~leake/papers/a-96-01.html
18. 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 (2022)

19. Leake, D., Ye, X.: Harmonizing case retrieval and adaptation with alternating optimization. In: Case-Based Reasoning Research and Development - ICCBR 2021. pp. 125–139. Springer, Cham (2021)
20. Li, O., Liu, H., Chen, C., Rudin, C.: Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In: Proceedings of the thirty-second AAAI conference on artificial intelligence (2017)
21. 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). pp. 106–109 (2018)
22. Main, J., Dillon, T.S.: A hybrid case-based reasoner for footwear design. In: Case-Based Reasoning Research and Development. pp. 497–509. Springer, Berlin (1999)
23. Mathisen, B.M., Aamodt, A., Bach, K., Langseth, H.: Learning similarity measures from data. Progress in Artificial Intelligence (10 2019)
24. Richter, M.: Introduction. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.D., Wess, S. (eds.) CBR Technology: From Foundations to Applications, chap. 1, pp. 1–15. Springer, Berlin (1998)
25. Rudin, C.: Please stop explaining black box models for high stakes decisions. Nature Machine Intelligence **1**, 206–215 (2019)
26. 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)
27. 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)
28. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014). https://doi.org/10.48550/ARXIV.1409.1556, https://arxiv.org/abs/1409.1556
29. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision (2015), https://arxiv.org/abs/1512.00567
30. 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. Springer, Cham (2018)
31. Turner, J.T., Floyd, M.W., Gupta, K.M., Oates, T.: NOD-CC: A hybrid CBR-CNN architecture for novel object discovery. In: Case-Based Reasoning Research and Development, ICCBR 2019. pp. 373–387. Springer, Cham (2019)
32. 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. pp. 248–262. Springer, Cham (2021)
33. Xian, Y., Lampert, C.H., Schiele, B., Akata, Z.: Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI) **40**(8), 1–14 (2018)
34. Ye, X., Leake, D., Crandall: Case adaptation with neural networks: Capabilities and limitations. In: Case-Based Reasoning Research and Development, ICCBR 2022. pp. 143–158. Springer, Cham (2022)