

# Twin-Systems to Explain Artificial Neural Networks using Case-Based Reasoning: Comparative Tests of Feature-Weighting Methods in ANN-CBR Twins for XAI

Eoin M. Kenny<sup>1,2\*</sup> and Mark T. Keane<sup>1,2,3</sup>

<sup>1</sup>School of Computer Science, University College Dublin, Dublin, Ireland

<sup>2</sup>Insight Centre for Data Analytics, Dublin, Ireland

<sup>3</sup>VistaMilk SFI Research Centre, Dublin, Ireland  
{eoin.kenny, mark.keane}@insight-centre.org

## Abstract

In this paper, *twin-systems* are described to address the eXplainable artificial intelligence (XAI) problem, where a black box model is mapped to a white box “twin” that is more interpretable, with both systems using the same dataset. The framework is instantiated by twinning an artificial neural network (ANN; black box) with a case-based reasoning system (CBR; white box), and mapping the feature weights from the former to the latter to find cases that explain the ANN’s outputs. Using a novel evaluation method, the effectiveness of this twin-system approach is demonstrated by showing that nearest neighbor cases can be found to match the ANN predictions for benchmark datasets. Several feature-weighting methods are competitively tested in two experiments, including our novel, contributions-based method (called COLE) that is found to perform best. The tests consider the “twinning” of traditional multilayer perceptron (MLP) networks and convolutional neural networks (CNN) with CBR systems. For the CNNs trained on image data, qualitative evidence shows that cases provide plausible explanations for the CNN’s classifications.

## 1 Introduction

The recent explosion in the use of AI for prediction and decision making in our daily lives has raised questions about the ability of these systems to explain what they do [Ribeiro *et al.*, 2016]; this problem is especially evident in artificial neural networks (ANN). Although the problem of eXplainable AI (XAI) is not new [Clancey, 1983], there is a renewed urgency to solving this problem because of the emerging ubiquity of these systems and new data regulations that encourage AI systems to explain their decisions (e.g., GDPR). This paper proposes the *twin-system* approach to address the XAI problem, in which a less interpretable system is mapped to one generally considered more interpretable, with both using the same dataset. We demonstrate this twinning approach has potential to help explain the decisions of “black box” AI systems. This

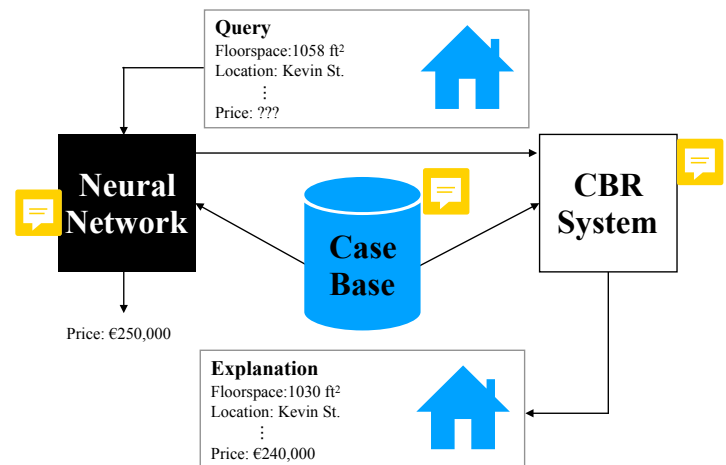


Figure 1: Twin-System Example: A query to an ANN produces an accurate, but unexplained prediction for a house price. The ANN is twinned with a CBR system, allowing the latter to retrieve a nearest neighbor case, using the ANN’s feature weights, to explain the prediction.

twin-system approach has some commonality with white box testing [Pei *et al.*, 2017], however the twin-system idea differs in its mapping of the ANN into a separate “twin” model that is more interpretable.

**ANN-CBR twins.** Twin-systems are based on the premise that if meaningful mappings can be computed from a black box model to its white box “twin”, then we can understand and explain the former using the latter. Whilst there are many options for how different AI techniques could be twinned, this paper explores the twinning of opaque ANN systems with arguably more transparent case-based reasoning (CBR) systems (i.e., ANN-CBR twins). CBR systems have qualitatively different interpretable properties to ANNs, as they employ an intuitive *reasoning from precedent or example* method based on similarity between cases, where these cases can provide useful explanations for predictions [Cunningham *et al.*, 2003]. In an ANN-CBR twinning both systems work from an identical dataset, and the ANN is analyzed to find the feature weights that contributed to the outputs produced, these are then used in the CBR system to retrieve cases which

\*Contact Author


can explain the ANN’s outputs. Assessing the quality of extracted ANN feature weights has historically been a difficult process [Shin *et al.*, 2000; Olden *et al.*, 2004], but here we use a novel evaluation method that quantitatively measures predictive agreement between the twinned systems (see Section 3.1).

**Explanation.** In AI, as in Psychology and Philosophy, there is no agreed definition for the notion of “explanation” (see e.g., [Sørmo *et al.*, 2005]), and a wider discussion of this is not possible here. However, often a distinction is made between explanations that make the system more *transparent* (by directly describing the reasoning/decision process of the system that led to its outputs) or that *justify* the system (i.e., by evidentially providing some basis for the system’s outputs). The twin-system approach proposed here, is arguably, much closer to *justification* than *transparency* (e.g., [Lipton, 2018] calls it “*post-hoc* explanation by example”).

**Paper outline.** This paper explores ANN-CBR twinning as a solution to the XAI problem, by comparatively testing different feature-weighting schemes. In Section 2, previous approaches to twinning are reviewed. Section 3 describes the feature-weighting techniques tested (including our novel method COLE) and the new evaluation methodology used. Sections 4 and 5 report two experiments testing MLP-CBR and CNN-CBR twins, respectively, on benchmark datasets. Section 6 concludes and considers future directions for this approach (see code at <https://github.com/EoinKenny/Twin-Systems>).


## 2 Prior Work

### 2.1 Artificial Neural Networks (ANN)

Biologically inspired, ANNs deal well with both structured and unstructured data [Shin *et al.*, 2000; Chen *et al.*, 2018]. A division is often made between traditional multilayer perceptron networks (MLPs), and the more recent deep learning models [such as, convolutional neural networks (CNNs)]. However, the non-linear nature of both types make them difficult to interpret and poor at explaining their outputs [Olden and Jackson, 2002; Selvaraju *et al.*, 2017]. MLPs trained on tabular data have explicit input features, but their *contribution* to outputs is opaque. Statistical methods based on sensitivity analysis [Bai *et al.*, 2011] have dominated attempts to explain how these systems operate, but we demonstrate here that recent techniques developed specifically for ANNs are more useful in twin-systems. CNNs can be even more opaque with regards to their features and respective contributions to classifications. Attempts to improve the interpretability of CNNs have largely focused on visualizing what specific neurons in the network have learned (see e.g., [Zeiler and Fergus, 2014; Erhan *et al.*, 2009]). However, arguably, such visualizations still have to be contextualized. 

### 2.2 Case-Based Reasoning (CBR)

CBR is an AI technique that has been applied in a wide variety of domains for many different tasks (e.g., diagnosis, design, classification, recommendation, and law) [De Mantaras *et al.*, 2005; Bench-Capon, 2017]. CBR’s intuitive

method involves *reasoning from example or precedent* via four main processing steps: *retrieval*, *reuse*, *revise* and *re-train*. Accordingly, CBR systems can explain their outputs using cases [Cunningham *et al.*, 2003]. At its simplest, when a query case is presented, the most similar cases to it are retrieved before being used directly (or adapted) to make a prediction. Typically, the *retrieval* step finds cases by matching the features of the query against the case base using *k*-nearest neighbor (*k*-NN). Retrieval accuracy can heavily depend on the weights given to these features, which reflect their importance in the domain. For example, a CBR system predicting house prices might weight the *location* feature over *floorspace*, because *location* is much more important for predicting the house price (see Fig. 1). So, the extraction of accurate feature weights is a key step in twin-systems because they determine the cases retrieved 

### 2.3 Twin-Systems

Hybrid systems typically combine several AI techniques to meet a task requirement [Medsker, 2012]. For example, Reategui *et al.* [1997] combined an ANN and CBR system for medical diagnosis, where the ANN generated hypotheses to direct the search for cases in a CBR system (for a recent example using deep learning see Amin *et al.* [2018]). However, these systems are not twin-systems in our sense, as the motivation for combining techniques is to build a processing pipeline to achieve some task, where each technique is typically insufficient on its own. Furthermore, the primary motivation is not explanation. In contrast, the twin-system idea uses two models alongside one another where the purpose for using both techniques is, specifically, *explanation*. In effect, twin-systems are a distinct, special case of hybrid systems (see Keane and Kenny [2019a], [2019b] for a survey and definitions).

#### Previous Twin-Systems and MLPs

Examples of hybrid systems using MLPs exist that meet the twin-system definition (see [Keane and Kenny, 2019a] for a review). Shin *et al.* [2000] built MLP-CBR twins to test the accuracy of different feature-weighting schemes, mapping between the MLP and CBR systems to retrieve explanatory cases using *k*-NN. They found that a weighted *k*-NN worked better than an unweighted one, provided good explanatory cases, and was more resilient in noisy feature spaces. This work mainly explored *global* weighting methods, which assume that the input space is isotropic, and use a single ubiquitous feature-weight vector for each domain.

In contrast, Park *et al.* [2004] examined *local* weighting by accounting for varying feature weights across the instance space, but their solution cannot be applied to find *post-hoc* explanations for standard MLPs. Nugent and Cunningham [2005] used local-weighting methods by perturbing a query to build a localized dataset, fitting a linear model to this new data to approximate the MLP function locally, and then using its coefficients to weight the CBR’s *k*-NN search. The present paper builds on these previous efforts by considering the weighting of the entire training set *and* the individual weighting of query instances (see Section 3).

### Recent Twin-Systems and Deep Learning

Beyond this earlier work, we have found few recent papers that could be called twin-systems in deep learning. Here, visual explanations are typically used such as activity maximization [Erhan *et al.*, 2009], deconvolution [Zeiler and Fergus, 2014], activation maps [Zhou *et al.*, 2015], and saliency maps [Bach *et al.*, 2015]. However, recently  $k$ -NN has been combined with CNNs using nearest neighbor approaches [Papernot and McDaniel, 2018; Sani *et al.*, 2017], but they rely on neuron activations to fit their  $k$ -NN “twins”, an approach which we show is not always applicable (see Section 5).

Since the above approaches are often seen as compromising interpretability for accuracy, recent attempts have built interpretability into the network itself using *prototypes* and CBR ideas. Li *et al.* [2017] used an encoder to represent features in the latent space and judged nearby cases via distance metrics. However, a decoder is required for visualizing *prototypes* as explanations, and although this solution performed well on MNIST, it reportedly fails in more complex domains, such as CIFAR-10 [Chen *et al.*, 2018]. Chen *et al.* [2018] expanded this work by allowing their prototypes to represent *parts* of images, that were projected onto representations in training data for visualization and explanation. These are promising approaches, but at present are difficult to train, achieve lower accuracy than standard models, and require additional hyperparameters (e.g., number and size of prototypes). In Expt. 2, we develop CNN-CBR twins in which higher computational costs are avoided, and accuracy is maintained, while still finding plausible explanations (see Section 5).

### 3 Techniques for Extracting Feature Weights

In the above review, we saw that *global* and *local* techniques are distinguished for extracting feature weights from an ANN. Global techniques use a single weight-vector ubiquitously, whereas local techniques generate instance-specific weight-vectors. In Expt. 1, we explore MLP-CBR twin-systems by competitively testing seven different weighting methods: three global and four local techniques (see Section 4). These methods represent the best techniques identified in the literature. However, they have not all been comparatively tested before.

#### Global Weighting Methods

All global weighting methods use the following formula when calculating distances in  $k$ -NN searches:

$$Distance(x, q) = \sqrt{\sum_{f=1}^n |w_f| Difference(x_f, q_f)^2} \quad (1)$$

where  $|w_f|$  is the absolute weight derived for feature  $f$ ,  $x_f$  is the feature  $f$  in training instance  $x$ ,  $q_f$  is the feature  $f$  in the query instance  $q$ , and  $n$  is the total number of features. The three global methods are:

**Sensitivity (SENS).** Used by Shin *et al.* [2000], this works by setting a feature’s input to zero and measuring the difference between the network’s new and old predictions to gauge

the feature’s importance. This technique has been found to be very promising in their research.

$$S_i = \frac{(\sum_L \frac{|P^0 - P^i|}{P^0})}{n} \quad (2)$$

where  $S_i$  is the sensitivity (feature weight) of input  $i$ ,  $P^0$  is the original prediction of the network,  $P^i$  is the prediction after said input is removed,  $L$  is the set of training data and  $n$  the number of training set instances.

**Perturbation (PTB).** A popular method for extracting feature weights from MLPs [de Oña and Garrido, 2014], the method works by perturbing input features to observe the effect on the output. Previous work shows the optimal perturbation range to be perhaps 20% [Bai *et al.*, 2011], which we use. This technique has not been tested in MLP-CBR systems, we propose the following formulation to do so:

$$W_i = \frac{\sum_{j=1}^n \delta(L_j, \sigma)}{2n} \quad (3)$$

where  $W_i$  represents the global weight of feature  $i$ ,  $\sigma$  the perturbation range,  $n$  the number of training set instances,  $L$  the training data, and  $\delta$  a function returning the absolute summed change in two predictions with a positive and negative perturbation respectively to the feature  $i$  in instance  $L_j$ .

**Connection Weights (CW).** Advanced by Olden and Jackson [2002], as a possible improvement on Garson’s Algorithm [Garson, 1991] (which considers the absolute values of a network’s weights to derive feature weighting), this considers if the network’s weights are positive or negative:

$$R_{ij} = \sum_{k=1}^H W_{ik} \cdot W_{kj} \quad (4)$$

where  $R_{ij}$  is the relative importance of the variable  $x_i$  with respect to the output neuron  $j$ ,  $H$  is the number of neurons in the hidden layer,  $W_{ik}$  is the connection weight between the input neuron  $i$  and the hidden neuron  $k$ , and  $W_{kj}$  is the weight between the hidden neuron  $k$  and the output neuron  $j$ .

#### Local Weighting Methods

Of the four local methods, the first uses (1) but computes the weight vector individually for each test instance. The last three use a standard  $k$ -NN algorithm, but it is fit to a different dataset of “contributions” to weight the search. The four methods examined are described below.

**Local Linear Model (LLM).** This is a name we coin for Nugent and Cunningham’s [2005] method for finding feature weights from an MLP to inform case retrieval for explanation. This method perturbs the features of a query randomly to generate a new local dataset, the MLP is then used to generate labels for this new data. Using the new dataset a linear model is then built which works to approximate the MLP function locally around the query. The linear model’s coefficients are then used to weigh features in the  $k$ -NN search. Recently, Ribeiro *et al.* [2016] generalized a similar approach in Local Interpretable Model-Agnostic Explanations (LIME), whose code library is used here to implement LLM. As LIME is model agnostic, it can be applied to any classifier, so this method could be used with other twinning options, other than ANN-CBR twins (e.g., SVM-CBR twin-systems).

**Contributions Oriented Local Explanations (COLE: C-LPR, C-IG, C-DeepLIFT).** Our method COLE is based on the premise that the *contributions* of features in a model’s classification represent the most sensible basis to inform case-based explanations. Consider a linear model  $f(\vec{x})$  with  $n$  input features and a weight vector  $\vec{w} \in \mathbb{R}^n$ . The contribution vector  $\vec{c}$  of an instance  $\vec{x}$  is:

$$\vec{c} = \langle x_1.w_1, x_2.w_2 \dots x_n.w_n \rangle \quad (5)$$

where  $x_i.w_i$  calculates  $c_i$  (i.e., the contribution of feature  $i$ ). Intuitively, it makes more sense to find explanatory cases using  $\vec{c}$ , rather than  $\vec{x}$  or  $\vec{w}$ , because  $\vec{c}$  more closely represents the final classification logic of  $f(\vec{x})$  (as the final output is typically influenced by  $\sum_{i=1}^n c_i + Bias$ ). In  $f(\vec{x})$ , the weights  $\vec{w}$  are known, but in an MLP determining  $\vec{w}$  is non-trivial, and it is likely to mutate at every area of the input space. To deal with this issue, we used saliency map techniques, to heuristically estimate  $\vec{c}$  directly (avoiding the need for  $\vec{w}$ ), as they have proven useful in deep learning [Lundberg and Lee, 2017]. These saliency map methods backpropagate contributions (usually from the output to input layer) and, hence, can provide feature weights for finding explanatory cases using COLE in the context of MLP-CBR twins. We used three saliency map techniques (see Algorithm 1): *Layer-wise Relevance Propagation* (LRP) [Bach et al., 2015], *Integrated Gradients* (IG) [Sundararajan et al., 2017] and *DeepLIFT* [Shrikumar et al., 2017], which we denote as C-LPR, C-IG and C-DeepLIFT to show they are COLE variants of these methods. Specifically, we used Epsilon-LRP ( $\epsilon$ -LRP), a black reference point for IG, and a reference activation of zero for DeepLIFT. In Expt. 1 the contribution scores for all the training and testing data were generated by attributing scores for the output classification neuron to the input-layer features (see Algorithm 1); these were then used in a  $k$ -NN classifier for the CBR system. The training and testing instances are all individually weighted to represent the MLP locally, enabling the weighted  $k$ -NN (i.e., the CBR system) to find explanatory cases based on how discriminating the MLP finds certain features.

### 3.1 Twin-Systems: Evaluation Methodology

In the present experiments, we want to find the best feature-weighting method for MLP-CBR twin-systems. To meet this aim, we developed a novel evaluation methodology that assesses predictive agreement between the twinned systems. This “agreement” score is calculated by dividing the number of agreed predictions on testing data from both models by the number of test instances. As such, 0.0 and 1.0 represent zero and perfect agreement in the twin-system, respectively. Note, this aim is different to determining whether the system can deliver good explanatory cases (but, see Expt. 2 for some qualitative evidence). However, knowing the best feature-weighting scheme clearly will enable better testing of such explanatory capabilities in the future.

To evaluate the weighting, a situation is setup where an increase in agreement between the systems will correlate with an increase the fidelity of the extracted weighting. This is done by making  $k$ -NN closely imitate the MLP decision boundary, and then applying the extracted weighting from the

---

#### Algorithm 1 COLE weighting

---

**Input:** *train*, training data; *test*, testing data; *MLP*, MLP model;  $\delta(\vec{x}, MLP)$ , function (e.g., DeepLIFT) returning a 2D array  $A \in \mathbb{R}^{m,n}$  of contributions for all input features to all classification outputs of *MLP* for instance  $\vec{x}$  where  $m$  is the number of neurons in the last layer, and  $n$  is the number of features in the input layer.

**Output:** *trainC*, *testC*, the contributions of input features to the training and testing data classifications of *MLP*, respectively.

```

1: trainC, testC = emptyArray, emptyArray
2: for  $i = 0$ ; to  $\text{len}(\text{train})$  do
3:    $\text{idx} = 0$  // For Sigmoid Output
4:   if MLP.outputLayer == SoftMax then
5:      $\text{idx} = \text{MLP.predictClass}(\text{train}[i])$ 
6:   end if
7:   trainC.append( $\delta(\text{train}[i], \text{MLP})[\text{idx}]$ )
8: end for
9: for  $i = 0$ ; to  $\text{len}(\text{test})$  do
10:   $\text{idx} = 0$ 
11:  if MLP.outputLayer == SoftMax then
12:     $\text{idx} = \text{MLP.predictClass}(\text{test}[i])$ 
13:  end if
14:  testC.append( $\delta(\text{test}[i], \text{MLP})[\text{idx}]$ )
15: end for
16: return trainC, testC

```

---

MLP to the  $k$ -NN to complete the system mapping. In this situation a high agreement score will signify feature-weights of high quality. Formally, the evaluation has the following steps:

1. Datasets with a large number of features are chosen such that an unweighted  $k$ -NN (henceforth  $k$ -NN\*) is found to have relatively low accuracy on testing data.
2. An MLP trained on the same data is found to have superior accuracy on the testing data (by finding good weights in the feature space).
3.  $k$ -NN\* is made to closely imitate the MLP decision boundary by temporarily removing outlier cases (defined here as training cases the MLP miss-classifies).
4. Feature weighting is extracted from the MLP and used in the weighted  $k$ -NN. So, a high agreement score between the weighted  $k$ -NN and MLP signifies that the extracted MLP feature-weighting is of high fidelity.

In this situation, provided the weighting is credible, similar cases will “cluster” together in a weighted  $k$ -NN and it will closely mimic the MLP’s classifications (which our results confirm), and nearest neighbors will represent cases with similar discriminating features used in a classification. In this evaluation method, outliers were only removed to allow a “clean” test of agreement and, by extension, the different feature-weighting methods<sup>1</sup>.

---

<sup>1</sup>Both experiments were run with and without the outlier data and, as expected, while overall agreement was worse with the outliers left in, there was no appreciable change to the rank ordering of feature-weighting techniques.

Dataset	Task	$n$	$k$	$k$ -NN*	PTB	SENS	CW	C-DeepLIFT	C-IG	C-LRP	LLM
Connect-4	Multi.	126	5	0.812	0.861	0.849	N/A	1.000	1.000	1.000	0.834
Thyroid Disease	Multi.	21	5	0.988	0.974	0.985	N/A	1.000	1.000	0.983	0.983
Nursery	Multi.	27	1	0.790	0.943	0.966	N/A	1.000	1.000	0.999	0.878
Adult Census	Binary	16	1	0.987	0.991	0.982	0.989	0.991	0.989	0.990	0.990
Default of Credit.	Binary	24	1	0.971	0.987	0.985	0.983	0.990	0.989	0.989	0.985
Bank Marketing	Binary	127	10	0.936	0.939	0.964	0.934	0.967	0.964	0.960	0.934
Breast Cancer	Binary	30	2	0.895	0.947	0.877	0.912	0.983	0.965	0.930	0.930
Mean Agreement				0.911	0.949	0.944	0.954	0.990	0.987	0.979	0.933

Table 1: Agreement scores in Expt. 1 for MLP-CBR twins across seven datasets, comparing unweighted  $k$ -NN\* (the control) with seven variants reflecting the different feature-weighting schemes ( $n$  is the number of features used).

## 4 Experiment 1: MLP-CBR Twins

In this experiment, we measure the agreement for MLP-CBR twins using seven different feature-weighting methods for  $k$ -NN compared to a baseline unweighted  $k$ -NN ( $k$ -NN\*). The aim being to discover the feature-weighting method that best represents the MLP function for a given domain. In these tests, we are only interested in mapping one system to the other, irrespective of the data split, so cross-validation was not used. The splits ranged from 10-33% for testing data and were selected to challenge the unweighted  $k$ -NN\* baseline. Values for  $k$  were chosen to give the best general performance across all weighting techniques, whilst trying to keep it as low as possible.

### 4.1 Method: Architecture, Datasets and Procedure

All MLPs used had a single input, hidden, and output layer. The hidden layer used ReLU activation functions and the output used Sigmoid and SoftMax for binary and multi-class classification, respectively. Seven popular datasets were selected and modified as described above (comparable results were found for five regression datasets, that are not reported here). Thus, the MLP and CBR systems were tested on the same datasets using eight different  $k$ -NN variants ( $k$ -NN\* and the seven weighting methods described above). The Connection Weights (CW) method was an exception as it was not developed originally for SoftMax outputs, so its average agreement is over four datasets rather than seven (see Table 1). The measure is the agreement in predictions between the model-pairs (see Section 3.1). The CBR system is a retrieval-only system (there is no adaptation) that simply uses  $k$ -NN to find cases from which predictions are derived.

### 4.2 Results and Discussion

Table 1 summarizes the agreement scores for all tests in Expt. 1. Overall, a feature-weighted  $k$ -NN does better than an unweighted  $k$ -NN\* (Mean Agreement = 0.91). Moreover, the local weighting methods (C-DeepLIFT, C-IG, C-LRP, and LLM) do better on average (Mean Agreement = 0.93-0.99) than global methods (PTB, SENS, CW; Mean Agreement = 0.94-0.95). Within the local methods, the contribution-based methods (i.e., C-DeepLIFT, C-IG, C-LRP) generally do better than the non-contribution LLM method, the former often achieving perfect agreement with the MLP, with C-DeepLIFT being the best of all these methods (Mean Agreement = 0.99).

However, LLM is model agnostic (whereas the other local methods are not), meaning it may have wider applicability in other twin-systems. Finally, it should be noted that the multi-class classification problems achieve higher mean agreement scores than binary ones when using the contribution methods. This is possibly due to additional weight parameters in the SoftMax layer allowing for greater discretization of the classes, as the saliency map techniques used to realize COLE here utilize these weight values when calculating contribution scores.

## 5 Experiment 2: CNN-CBR Twins

This experiment tests the prediction agreement for a deep learning model (a CNN) twinned with a CBR system using a control,  $k$ -NN\*, and a weighted  $k$ -NN computed from the best contributions-based method found in Expt. 1 (i.e., C-DeepLIFT; see Section 4.2).

### 5.1 Method: Architecture, Datasets and Procedure

All CNN models used had three convolutional layers with two max pooling layers following the first two convolutions. The models then contained three fully-connected (FC) layers, the last of these being a SoftMax output. Consequently, the first FC layer represents the latent features extracted by the CNN, analogous to the input features of the MLPs tested in Expt. 1. The exception to this is the final test which used the *InceptionResNetV2* CNN. This model was used for feature extraction, so a standard three-layer MLP could be trained on these extracted features to discriminate the classes.

When using a CNN for image classification, the pixel space is transformed into a convolutional output before being used for classification. Formally, let the transformation be  $X \rightarrow C \in \mathbb{R}^{(h,w,d)}$  where the image tensor  $X$  is transformed into a new representation  $C$  of the image, where  $h$ ,  $w$ , and  $d$  represent the height, width and number of feature maps, respectively. After this a CNN will typically consist of a FC or global average pooling (GAP) layer. Either way, let the next layer be a vector representation  $\vec{x} \in \mathbb{R}^n$  of the image, where  $n$  is the number of latent features. Provided there are no more layers before the SoftMax output, computing contribution scores can be trivially done by multiplying the feature activations generated for a specific instance in  $\vec{x} \in \mathbb{R}^n$  by the weight connecting them to the predicted class output node. In this situation, we can examine the activation map of the



Dataset	$n$	$k$	$k$ -NN*	C-DeepLIFT
MNIST	128	1	0.9962	0.9999
MNIST-Fashion	128	1	0.9517	0.9992
CIFAR-10	128	1	0.8107	0.9972
CIFAR-10	1000	1	0.7832	0.9999
CIFAR-10*	1000	1	0.7048	0.9838
Mean Agreement			0.84932	0.996

Table 2: Agreement scores in Expt. 2 for CNN-CBR twins using three datasets across five tests, comparing unweighted  $k$ -NN\* (control) and C-DeepLIFT ( $n$  is again the number of features; CIFAR-10\* uses transfer learning implemented with *InceptionResNetV2*).

class and it’s discriminating features [Zhou *et al.*, 2015] (see Fig. 2b).

However, when using additional FC layers as we do here, it is necessary to follow the process outlined in Expt. 1 where contribution scores are generated using (for example) DeepLIFT by attributing scores from the output classification neuron to the feature layer  $\vec{x} \in \mathbb{R}^n$ . In Expt. 2, the control  $k$ -NN\* used neuron activations from  $\vec{x}$  to retrieve cases and the weighted  $k$ -NN used contributions from  $\vec{x}$  with C-DeepLIFT. Selvaraju *et al.* [2017] proposed a solution to interpretability with multiple FC layers by generalizing the class activation maps (discussed by Zhou *et al.* [2015]) into more CNN architectures, such as those with several FC layers. However, this solution does not focus on the use of CBR or twin-systems to explain outputs.

Three popular image datasets were selected and modified as described earlier (see Section 3.1). We compared fitting  $k$ -NN with activations in the latent feature layer  $\vec{x} \in \mathbb{R}^n$  as a control (i.e.,  $k$ -NN\*), and the contributions derived by C-DeepLIFT for the same layer. The first four tests used a CNN trained locally and the last used a pre-trained (on ImageNet) CNN *InceptionResNetV2* for prior feature extraction to train a standard MLP. This final model was chosen because it is among the most complex, publicly available CNN models (572 layers), and allows tests using image transfer learning. The agreement measure used is the same as Expt. 1. Finally, as before, the CBR systems were retrieval-only systems using variants of  $k$ -NN.

## 5.2 Results and Discussion

Table 2 summarizes the agreement scores for all the tests made in Expt. 2. Overall, the  $k$ -NN using the contribution scores (i.e., C-DeepLIFT) did markedly better than the baseline  $k$ -NN\*. Furthermore, the mean agreement for C-DeepLIFT in CNN-CBR twins was very high (0.996 out of 1.0). Notably, the agreement scores decrease in  $k$ -NN\* as complexity of the task increases, particularly in *InceptionResNetV2*, which is a model designed to discriminate between a large number of classes with large number of features; hence it is not surprising that more intricate weighting details are required for mapping accurately between the systems. However, if the CNN parameters had been fine-tuned to CIFAR-10, it is likely agreement would be closer to 1.0 for both  $k$ -NN\* and C-DeepLIFT. Lastly, echoing Expt. 1, the SoftMax outputs here achieve very high mean agreement, suggesting

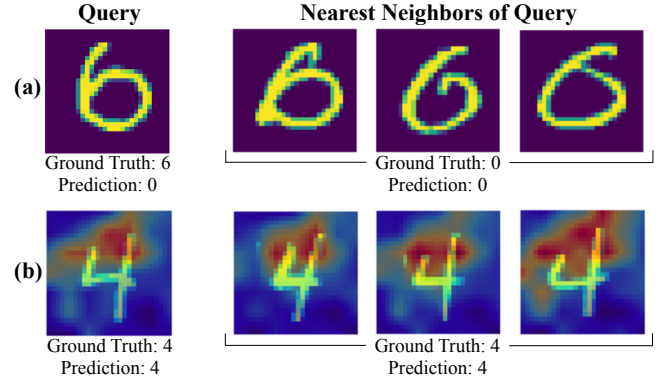


Figure 2: (a) shows the MNIST CNN from Expt. 2 miss-classifies a query of “6” as “0”. Nearest neighbors from the CBR-twin show the model was trained on data labelled as a “0” but looking like “6”. (b) shows a correctly classified “4” and its nearest neighbors, FAMs highlight the most discriminating feature used during classification. Red signifies areas of high activation for the feature.

that they may always be the optimal choice, even in a binary classification domain when Sigmoid is typically used.

## 5.3 Qualitative Evidence

The focus here is on finding the best feature-weighting methods for ANN-CBR twin-systems rather than assessing the explanatory value of retrieved cases for users. However, CNNs are often “visually explained” when they are applied to image datasets, mainly because people have sufficient expertise to evaluate the image-predictions made. Fig. 2 shows some of the explanatory cases retrieved from queries to the CNN-CBR twin-system from Expt. 2, trained on the MNIST dataset, using C-DeepLIFT to weight the  $k$ -NN search.

Fig. 2a shows a miss-classification by the CNN, where an image of a “6” is categorized as a “0”, along with the three nearest neighbors found to explain this prediction. Note these are all cases of a “0”, but resemble a “6”, making the miss-classification of the CNN reasonable. Fig. 2b shows a correct classification of a “4” by the CNN, along with the three explanatory, nearest-neighbor cases. These images also have a feature activation map (FAM)<sup>2</sup> showing the most discriminating feature across the cases. Interestingly, these FAMs show that the presence of a gap across the top of the number “4” is the primary feature making it a “4”. Note, the *nearest-unlike-neighbor* found is the class “9” (using the CBR twin), showing the importance of the “link” in discriminating it as a “4”. Fig. 3 shows some results for CNN-CBR twins on the CIFAR-10 dataset. So, twin-systems potentially provide justification, scrutability and transparency for the operations of the CNN (as defined by [Tintarev and Masthoff, 2007]).

<sup>2</sup>These FAMs resemble the class activation maps of Zhou *et al.* [2015]. However, here FAMs show the most discriminating feature for a query rather than the whole class. These are computed by multiplying the activations of the convolutional output  $C \in \mathbb{R}^{(h,w,d)}$  by the weights  $W \in \mathbb{R}^{(h,w,d)}$  connecting them to the feature contributing most to the classification. The resulting matrix  $M \in \mathbb{R}^{(h,w,d)}$  is then summed along  $d$  into  $M \in \mathbb{R}^{(h,w)}$  and up-sampled to the original image size before being superimposed.

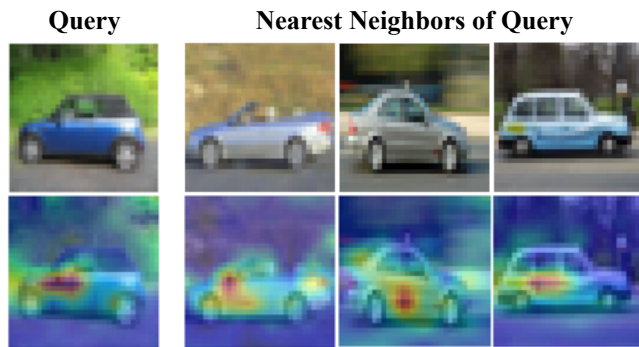


Figure 3: A correctly classified query of a car from the CNN trained locally with  $n = 1000$  in Expt. 2. Again the nearest neighbors are shown above, with a FAM of the most discriminating feature used in the classification of the query shown below, which reassuringly remains comparably consistent across the four cases regardless of the car’s orientation.

## 5.4 Computational Costs

To answer concerns about the computational costs of these methods, it should be said that Algorithm 1 took  $\sim 40$  seconds to run Expt. 1 with Connect-4 (67,557 training and testing instances with  $n = 126$ ), whilst in Expt. 2 it took  $\sim 6613$  seconds for CIFAR-10 (60,000 training and testing instances with  $n = 1000$ ). The experiments used a MacBook Pro; processor: 2.9 GHz Intel Core i5; memory: 16 GB 2133 MHz LPDDR3. Hence, the costs of deriving the weights for the  $k$ -NN are quite modest. Moreover, this initial computation is only required once, subsequent queries took  $< 30$  seconds to derive a single query’s weighting and  $< 0.05$  seconds to find explanatory cases for both tabular and image data.

## 6 Conclusions and Future Work

This paper offers four novel contributions. First, a description of the twin-system framework for XAI where opaque AI black boxes are explained by interpretable white boxes. Second, a promising demonstration of this framework, twinning ANNs and CBR systems for case-based explanation. Third, the identification of the best feature-weighting techniques for such ANN-CBR twin-systems (testing both MLP-CBR and CNN-CBR twins) using a novel evaluation method and showing that our contribution-based method, COLE, does best (using DeepLIFT). Finally, qualitative evidence shows that CNN-CBR twins provide very plausible visual explanations on benchmark image datasets (regardless of the number of FC layers in the model, a point of importance in image transfer learning [Zhang *et al.*, 2018]). For future research, user studies assessing the explanatory value of the cases retrieved are planned, alongside generalizing COLE to be applicable to any machine learning model beyond just ANNs (e.g., for SVM-CBR twin-systems).

## Acknowledgments

This publication has emanated from research conducted with the financial support of (i) Science Foundation Ireland (SFI)

to the Insight Centre for Data Analytics under Grant Number 12/RC/2289 and (ii) SFI and the Department of Agriculture, Food and Marine on behalf of the Government of Ireland to the VistaMilk SFI Research Centre under Grant Number 16/RC/3835.

## References

- [Amin *et al.*, 2018] Kareem Amin, Stelios Kapetanakis, Klaus-Dieter Althoff, Andreas Dengel, and Miltos Petridis. Answering with cases: A CBR approach to deep learning. In *International Conference on Case-Based Reasoning*, pages 15–27. Springer, 2018.
- [Bach *et al.*, 2015] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.
- [Bai *et al.*, 2011] Runbo Bai, Hailei Jia, and Pingzhou Cao. Factor Sensitivity Analysis with Neural Network Simulation based on Perturbation System. *Journal of Computers*, 6(7), July 2011.
- [Bench-Capon, 2017] Trevor JM Bench-Capon. Hypo’s legacy: introduction to the virtual special issue. *Artificial Intelligence and Law*, 25(2):205–250, 2017.
- [Chen *et al.*, 2018] Chaofan Chen, Oscar Li, Alina Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: deep learning for interpretable image recognition. *arXiv preprint arXiv:1806.10574*, 2018.
- [Clancey, 1983] William J. Clancey. The epistemology of a rule-based expert system —a framework for explanation. *Artificial Intelligence*, 20(3):215–251, May 1983.
- [Cunningham *et al.*, 2003] Pádraig Cunningham, Dónal Doyle, and John Loughrey. An Evaluation of the Usefulness of Case-Based Explanation. In *Case-Based Reasoning Research and Development*, volume 2689, pages 122–130. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.
- [De Mantaras *et al.*, 2005] Ramon Lopez De Mantaras, David McSherry, Derek Bridge, David Leake, Barry Smyth, Susan Craw, Boi Faltings, Mary Lou Maher, Michael T Cox, Kenneth Forbus, et al. Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, 20(3):215–240, 2005.
- [de Oña and Garrido, 2014] Juan de Oña and Concepción Garrido. Extracting the contribution of independent variables in neural network models: a new approach to handle instability. *Neural Computing and Applications*, 25(3-4):859–869, September 2014.
- [Erhan *et al.*, 2009] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pascal Vincent, and P O Box. Visualizing Higher-Layer Features of a Deep Network. page 14, 2009.
- [Garson, 1991] G David Garson. Interpreting neural-network connection weights. *AI expert*, 6(4):46–51, 1991.

- [Keane and Kenny, 2019a] Mark T. Keane and Eoin M. Kenny. How case-based reasoning explains neural networks: A theoretical analysis of XAI using post-hoc explanation-by-example from a survey of ANN-CBR twin-systems. *arXiv preprint arXiv:1905.07186*, 2019a.
- [Keane and Kenny, 2019b] Mark T. Keane and Eoin M. Kenny. The twin-system approach as one generic solution for XAI: An overview of ANN-CBR twins for explaining deep learning. *arXiv preprint arXiv:1905.08069*, 2019b.
- [Li et al., 2017] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep Learning for Case-Based Reasoning through Prototypes: A Neural Network that Explains Its Predictions. *arXiv:1710.04806 [cs, stat]*, October 2017. arXiv: 1710.04806.
- [Lipton, 2018] Zach C. Lipton. The mythos of model interpretability. *Queue*, 16(3):30, 2018.
- [Lundberg and Lee, 2017] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.
- [Medsker, 2012] Larry R Medsker. *Hybrid intelligent systems*. Springer Science & Business Media, 2012.
- [Nugent and Cunningham, 2005] Conor Nugent and Pádraig Cunningham. A Case-Based Explanation System for Black-Box Systems. *Artificial Intelligence Review*, 24(2):163–178, October 2005.
- [Olden and Jackson, 2002] Julian D Olden and Donald A Jackson. Illuminating the “black box”: a randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling*, 154(1-2):135–150, August 2002.
- [Olden et al., 2004] Julian D Olden, Michael K Joy, and Russell G Death. An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. *Ecological Modelling*, 178(3-4):389–397, November 2004.
- [Papernot and McDaniel, 2018] Nicolas Papernot and Patrick McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.
- [Park et al., 2004] Jae Heon Park, Kwang Hyuk Im, Chung-Kwan Shin, and Sang Chan Park. MBNR: Case-Based Reasoning with Local Feature Weighting by Neural Network. *Applied Intelligence*, 21(3):265–276, November 2004.
- [Pei et al., 2017] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18. ACM, 2017.
- [Reategui et al., 1997] E.B Reategui, J.A Campbell, and B.F Leao. Combining a neural network with case-based reasoning in a diagnostic system. *Artificial Intelligence in Medicine*, 9(1):5–27, January 1997.
- [Ribeiro et al., 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.
- [Sani et al., 2017] Sadiq Sani, Nirmalie Wiratunga, and Stewart Massie. Learning deep features for k-NN-based human activity recognition. In *Proceedings of the ICCBR-17 Workshop*. ICCBR (Organisers), 2017.
- [Selvaraju et al., 2017] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.
- [Shin et al., 2000] Chung-Kwan Shin, Ui Tak Yun, Huy Kang Kim, and Sang Chan Park. A hybrid approach of neural network and memory-based learning to data mining. *IEEE Transactions on Neural Networks*, 11(3):637–646, 2000.
- [Shrikumar et al., 2017] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3145–3153. JMLR. org, 2017.
- [Sundararajan et al., 2017] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org, 2017.
- [Sørmo et al., 2005] Frode Sørmo, Jörg Cassens, and Agnar Aamodt. Explanation in Case-Based Reasoning: Perspectives and goals. *Artificial Intelligence Review*, 24(2):109–143, October 2005.
- [Tintarev and Masthoff, 2007] Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *2007 IEEE 23rd International Conference on Data Engineering Workshop*, pages 801–810, Istanbul, Turkey, April 2007. IEEE.
- [Zeiler and Fergus, 2014] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.
- [Zhang et al., 2018] Chen-Lin Zhang, Jian-Hao Luo, Xiu-Shen Wei, and Jianxin Wu. In defense of fully connected layers in visual representation transfer. In *Advances in Multimedia Information Processing – PCM 2017*, volume 10736, pages 807–817. Springer International Publishing, Cham, 2018.
- [Zhou et al., 2015] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. 2015.