Déjà vu from the SVM Era: Example-based Explanations with Outlier Detection

Penny Chong[1,2][0000−0001−8198−7369], Yuval Elovici[2,3][0000−0002−9641−128X], and Alexander Binder[1,2][0000−0001−9605−6209]

[1] ISTD Pillar, Singapore University of Technology and Design, Singapore
[2] ST Engineering-SUTD Cyber Security Laboratory, Singapore University of Technology and Design, Singapore
[3] Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel
penny_chong@mymail.sutd.edu.sg

Abstract. Understanding the features that contributed to a prediction is important for high-stake tasks. In this work, we revisit the idea of a student network to provide an example-based explanation for its prediction in two forms: i) identify top-k most relevant prototype examples and ii) show evidence of similarity between the prediction sample and each of the top-k prototypes. We compare the prediction performance and the explanation performance for the second type of explanation with the teacher network. In addition, we evaluate the outlier detection performance of the network. We show that using prototype-based students beyond similarity kernels deliver meaningful explanations and promising outlier detection results, without compromising on classification accuracy.

Keywords: Prototypes · Explainability · LRP · Outlier Detection

1 Introduction

Deep neural networks (DNNs) are widely used in various fields as they show superior performance. Explainable AI, aiming at transparency for predictions [21,26,4] has gained research attention in the recent years.

In this work, we revisit explainability by examples. Since humans are known to learn fast from a few examples, it is therefore suitable to explain a prediction on a test sample, by providing a set of similar examples. Such example-based explanation from a model can be simply achieved using a similarity search over an available dataset, based on a metric defined by feature maps obtained from the trained model. For this, one computes a feature map for the test sample to be explained, and then searches an available dataset in the same feature map space for the nearest candidates to the test sample. This yields indeed an informative visualization of the feature embedding which the model has learned, yet the most similar examples found do not participate in the prediction of the test sample. Even if the similar examples share the same prediction with the test

sample, it is not obvious as to what quantifiable extent the model shares the same reasoning to predict the same label. An alternative to achieve such explainability by examples is to employ kernel-based predictors. They compute a weighted sum of similarities between the test and the training samples, and the impact of each training sample on the prediction of the test sample is naturally quantifiable. Neural networks perform very well these days without the need for kernel-based setups, yet for the goal of example-based explanation, one may consider to train a prototype-based student architecture from an arbitrarily structured teacher network.

In this paper, we consider the question of how well do prototype-based student networks perform as compared to standard CNN-based teacher. We are interested to quantify the performance in three aspects: prediction accuracy, the quality of explanations of the student network as compared to the teacher, and finally the aspect of measuring how well the prototype-based network can be used to quantify the outlierness of samples. The first two are the natural tradeoffs to be considered when training any surrogate. Looking beyond scientific benchmarks, a deployment of a predictor in high-stake settings requires the capability to flag inputs which are either anomalous or poorly represented by the training set for additional validation.

Prototype-based approaches deliver this property naturally by the sequence of sorted similarities or distances between the test sample and the prototypes. The idea of prototype sampling is inspired by the dual kernel Support Vector Machines (SVM) which will be touched upon in Section 3.1. During training, the student network will use the soft-labels obtained from the teacher network, which is a typical convolutional neural network (CNN). We use the trained similarities to identify outliers although they are not trained for this task.

The following summarizes our contribution in this work:

(1) We propose a student network based on prototypes directly sampled from the training set unlike learned prototypes. This guarantees that the provided similar examples come from the true data distribution.
(2) We propose and analyze various generic head architectures which compute prototype-based predictions from CNN feature maps. We point out that this provides two layers of explainability, firstly by identifying the top-k nearest prototype examples, and secondly, by showing the pixel-wise evidence of similarity between each of these examples and the test sample.
(3) We compare for the first time the performance of pixel-wise explanations for the teacher against the student architectures defined by their heads.
(4) We revisit a method to quantify the degree of outlierness for a sample by computing the top-k prototype similarity scores and evaluate the outlier detection capabilities for the proposed heads.

2 Related Work

This work draws from the literature on prototype learning. One of the earliest works in this field is the k-nearest neighbor (k-NN) classifier [1]. In the recent

years, researchers have proposed to combine DNNs with prototype based classifiers. A recent work [24] proposed a framework known as the convolutional prototype learning (CPL). In the framework, the prototype vectors and the weights of the CNN are trained jointly with prototype loss to encourage the learning of intra-class compact and inter-class separable representations. Subsequently, [13,15] proposed interpretable prototype networks to provide explanation based on similar prototypes for image and sequence classification tasks, respectively. The former used a decoder to map the prototype vectors back to the feature space, while the latter project each prototype to its closest embedding from the training set during training. Another work [6] introduced a prototypical part network (ProtoPNet) that is able to point to the parts of the input it looks at, and identify the prototypical cases that are similar to those parts. These explanations are used during classification and not created posthoc. Several other works use prototype-based predictors for few-shot learning [22,14].

In contrast to above, we select prototypes directly from the training set and do not use learnable prototype vectors [24,13,15,6]. In terms of outlier detection, we are interested to revisit the idea of quantifying outliers using statistics from the set of top-k similarity scores. For this, k-NN-based distance methods have been frequently considered in the form of summed distances [2], local outlier factors [5,12] or kernel-based statistics [19] and are still of recent interest [10] due to favorable theoretical properties. Unlike these works, we are interested in the performance of similarities which do not define a kernel, but rather emerged due to attention-based operations on feature maps and we omit to compute density estimates of nearest neighbors of a test sample. Overall, our idea of providing an example-based explanation that also shows the region of similarity between the example and the prediction sample is related to [6]. The authors in [6] explicitly focused on a patch-wise representation of prototypes, whereas we are interested in the evaluation of several similarity heads that use the whole feature map and the resulting outlier performance.

3 Methodology

In this section, we introduce our proposed network and the learning objectives, followed by explainability and the outlier detection method.

3.1 Relation to Kernel SVM

Our proposed architecture is inspired by the dual formulation of SVMs [8] which is required to solve the optimization problem

$\max_{\alpha_i \geq 0} \sum_i^N \alpha_i - \frac{1}{2} \sum_j^N \sum_k^N \alpha_j \alpha_k y_j y_k K(\mathbf{x_j}, \mathbf{x_k})$ subject to the constraints $0 \leq \alpha_i \leq C, \forall i$ and $\sum_i^N \alpha_i y_i = 0$. For a binary problem, the dual classifier takes the form of $f(\mathbf{x}) = \sum_i^N \alpha_i y_i K(\mathbf{x_i}, \mathbf{x}) + b$, where $K(\mathbf{x_i}, \mathbf{x})$ is a positive definite kernel. The solution is based on the set of similarities $\{K(\mathbf{x_i}, \mathbf{x}) \mid 1 \leq i \leq N\}$ between the training samples and the test sample $\mathbf{x}$ which can be used to compute statistics to quantify outliers [19].

3.2 Proposed Network

Algorithm 1 Prototype Sampling and Replacement Algorithm

Let $S = D \cup P$ be the original train set with $n(S) = N + K$. D is the new training set and $P = P_1 \cup ... \cup P_M$ is the set of all prototypes. P_m is the set of prototypes from class m and M is the number of classes. $\forall m,\ n(P_m) = L$ and $\sum_m n(P_m) = L \times M = K$.
while $e \leq max\_epoch$ **do**
  $count_k = 0, \forall k$ where $1 \leq k \leq K$
  for phase in [train, val] **do**
    while $i \leq iteration$ **do** ▷ *loop through mini-batches*
      if phase is train **then**
        Update network parameters based on D.
      else
        for misclassified val sample **do**
          $m'_l = arg\ min(W_{m,m'_1}, W_{m,m'_2}, ..., W_{m,m'_L})$
          ▷ W_{m,m'_l} *is the linear layer weight at row* m *and col* m'_l.
          ▷ *Groundtruth class is* m.
          ▷ $m'_l \in [1, K]$ *is the* l*-th prototype from class* m.
          $count_{m'_l} \leftarrow count_{m'_l} + 1$
        end for
      end if
    end while
  end for
  $P \leftarrow P \setminus P_{replace} \cup D_{random},\ where\ n(P_{replace}) = n(D_{random}) = p$ ▷ *Replace top-p prototypes based on the count metric with random samples from* D.
  $D \leftarrow D \setminus D_{random} \cup P_{replace}$
end while

We propose a prototype sampling method where K prototypes are sampled at each epoch from the training set. The sampling method is detailed in Algorithm 1. In the algorithm, we consider the set of prototypes from the same ground truth class as the misclassified validation sample and increase the count metric for the prototype that has the minimum weight in the linear layer. These prototypes are chosen to be replaced at the end of each epoch in an attempt to construct a better set of prototypes particularly for the ground truth classes that are often misclassified. By selecting prototypes directly from the training set, we ensure an explanation relative to examples present in the training distribution.

We explore various head architectures for the prototype student network to determine their suitability for explanations and identifying outlier samples.

They fall into three classes: similarity after spatial pooling, similarity without spatial pooling, and, attention-guided similarity without spatial pooling. Figures 1-3 show the three investigated head architectures. Let $f(\cdot\,; \mathcal{W}) : \mathbb{R}^{C_0 \times H_0 \times W_0} \rightarrow \mathbb{R}^{C \times H \times W}$ be a CNN without spatial pooling layers with a set of learnable weights

$\mathcal{W}$. For simplicity, we omit the weights $\mathcal{W}$ and denote the feature representation of x as $f(x)$.

Head I is based on a complete spatial average pooling of feature maps $f(\cdot)$, resulting in a vector $g(\cdot)$ that consists of only C feature channels. This is followed by a cosine similarity over the feature channels $g(\cdot)$, as seen in Figure 1:

$$s(x_i, p_k) = \frac{g(x_i)^\top g(p_k)}{\|g(x_i)\| \|g(p_k)\|}. \tag{1}$$

$s(x_i, p_k)$ falls in the range of $[-1, 1]$ where higher value indicates higher similarity between the input sample x_i and the prototype p_k. The linear layer $h(x_i, \{p_1, p_2, ..., p_K\}) = \sum_{k=1}^{K} w_k ReLU(s(x_i, p_k)) + b$ then predicts the output y_i. It can be observed that the linear layer takes similar form as the dual kernel SVM described in Section 3.1. The prediction for the dual kernel SVM depends on the support vectors while our formulation depends on the set of prototypes. The linear layers in Head II and III as shown in Figure 2 and 3, respectively, will share the same formulation.

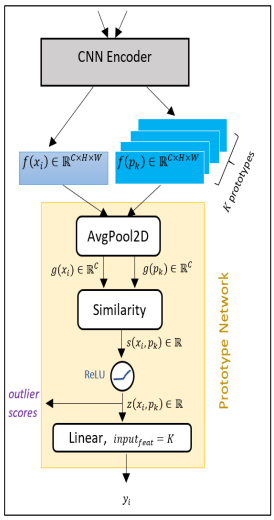

Fig. 1: Head I : Similarity using features after spatial pooling.

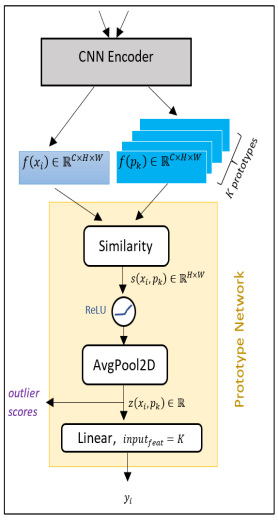

Fig. 2: Head II : Similarity using features before spatial pooling.

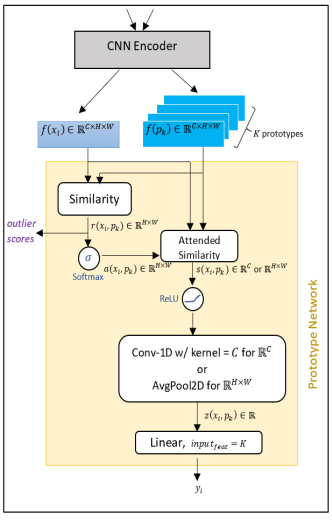

Fig. 3: Head III : Similarity w/attention using features before spatial pooling.

Head II in Figure 2, uses the features before spatial pooling for the computation of similarity scores. Therefore, the similarity layer computes $s(x_i, p_k) \in \mathbb{R}^{H\times W}$, i.e. one similarity score for each spatial position (h, w). We explore two types of similarity function: A) similarity between the input and prototype at the same spatial position (h, w), and B) the maximum similarity between input

and prototype over a set of spatial location pairs. We refer to the former as Head II-A and the latter as Head II-B. The similarity operation for Head II-A (Eq.(2)) and Head II-B (Eq.(3)) at the spatial location (h, w) is formally defined as follows.

$$s_{h,w}(x_i, p_k) = \sum_c \frac{f_{x_i}[c,h,w]}{\sqrt{\sum_j f_{x_i}[j,h,w]^2}} \cdot \frac{f_{p_k}[c,h,w]}{\sqrt{\sum_j f_{p_k}[j,h,w]^2}}, \tag{2}$$

$$s_{h,w}(x_i, p_k) = \max_{h',w'} \left(\sum_c \frac{f_{x_i}[c,h,w]}{\sqrt{\sum_j f_{x_i}[j,h,w]^2}} \cdot \frac{f_{p_k}[c,h',w']}{\sqrt{\sum_j f_{p_k}[j,h',w']^2}} \right), \tag{3}$$

where $f_{x_i}[c,h,w]$ and $f_{p_k}[c,h,w]$ are the features c at spatial location (h, w) of $f(x_i)$ and $f(p_k)$, respectively. The $max(\cdot)$ operation in Eq. (3) ensures that the similarity at the spatial location (h, w) is computed only between the location (h, w) in x_i and the most relevant location (h', w') in p_k, i.e. the pair that gives locally the highest cosine similarity score. In the remaining parts of the paper, $\frac{f[c,h,w]}{\sqrt{\sum_j f[j,h,w]^2}}$ will be denoted as $\hat{f}[c,h,w]$.

Note that head II-B as a maximum over inner products does not define a kernel anymore, as the maximum of two positive definite matrices is not guaranteed to be positive definite. Here we evaluate the performance of a non-kernel-based symmetric similarity.

Head III employs a common attention mechanism to compute attended features. For Head III, shown in Figure 3, the similarity scores are passed into a $softmax$ layer to obtain an attention map $a(x_i, p_k) \in \mathbb{R}^{H \times W}$ that will be used in the attended similarity layer together with the unnormalized features $f(x_i)$ and $f(p_k)$. Head III-A and Head III-B are the attention-augmented counterparts of Head II-A and Head II-B, respectively. The operation in the attended similarity layer for Head III-A (Eq. (4)) and Head III-B (Eq. (5)) is defined below.

$$s_c(x_i, p_k) = \sum_{h,w} a_{h,w}(x_i, p_k) \cdot f_{x_i}[c,h,w] \cdot f_{p_k}[c,h,w], \tag{4}$$

$$s_c(x_i, p_k) = \sum_{h,w} a_{h,w}(x_i, p_k) \cdot f_{x_i}[c,h,w] \cdot f_{p_k}[c,h_{\max},w_{\max}], \tag{5}$$

where $a_{h,w}(x_i, p_k) = softmax(r(x_i, p_k))_{h,w}$ is the attention score at spatial location (h, w), $r_{h,w}(x_i, p_k)$ for Head III-A is the similarity score from Eq. (2), and $r_{h,w}(x_i, p_k)$ for Head III-B is the similarity score from Eq. (3). The indexes $h_{\max}$ and $w_{\max}$ in Eq. (5) are the selected h' and w' indexes used in the computation of $r_{h,w}(x_i, p_k)$. The output from the similarity attended layer is passed into a convolutional-1D layer with kernel size C to compute a weighted sum of the C features. Since $C \gg H \times W$, we avoid using a simple averaging over the C features for better explainability. The convolutional-1D layer has a minimum weight clipping at zero to ensure the outputs are always non-negative in order to apply the explainability method described in Section 3.4. Finally, the use of different attention maps for features $f(x_i)$ and $f(p_k)$ is also investigated. The

following defines the operation for Head III-C (Eq. (6)) and Head III-D (Eq. (7)).

$$s_c(x_i, p_k) = \sum_{h,w} (a_{h,w}(x_i, p_k) \cdot f_{x_i}[c,h,w]) \cdot (a'_{h,w}(x_i, p_k) \cdot f_{p_k}[c,h,w]), \quad (6)$$

$$s_{h,w}(x_i, p_k) = \sum_{c} (a_{h,w}(x_i, p_k) \cdot f_{x_i}[c,h,w]) \cdot (a'_{h,w}(x_i, p_k) \cdot f_{p_k}[c,h,w]), \quad (7)$$

where $a'_{h,w}(x_i, p_k) = softmax(r'(x_i, p_k))_{h,w}$ is the attention score at spatial location (h, w) and $r'_{h,w}(x_i, p_k) = \max_{h',w'}(\sum_c \hat{f}_{x_i}[c,h',w'] \cdot \hat{f}_{p_k}[c,h,w])$. The attended similarity layer for Head III-C sums over the spatial dimension H, W, while Head III-D sums over the feature dimension C.

3.3 Learning Objectives

We minimize the following objective function for our student network.

$$\mathcal{L}_{final} = \mathcal{L}_{CE} + \lambda_1 \mathcal{L}_{CE\_Soft} + \lambda_2 \mathcal{L}_{Dist}, \quad (8)$$

where $\mathcal{L}_{CE}$ is the standard cross-entropy loss, $\mathcal{L}_{CE\_Soft}$ is the cross-entropy loss with soft-labels obtained from the teacher network, and $\mathcal{L}_{Dist}$ is the squared $L2$-norm which encourages the input sample x_i to be close to the set of prototypes of the same class but further away from the other classes.

We define the training set at epoch t to be $D_t = \{(x_i, y_i) \mid 1 \leq i \leq N\}$ and the set of prototypes at epoch t to be $P_t = \{(p_k, \tilde{y}_k) \mid 1 \leq k \leq K\}$. y_i and $\tilde{y}_k$ are the groundtruth class labels for input x_i and prototype p_k, respectively. For Head I architecture, the objective function $\mathcal{L}_{Dist}$ is defined as

$$\mathcal{L}_{Dist} = \frac{1}{NK} \sum_{i=1}^{N} \sum_{k=1}^{K} (\ \|\hat{g}(x_i) - \hat{g}(p_k)\|^2\ )^{\alpha_{i,k}}\ , \quad (9)$$

where $\alpha_{i,k} = 1$, if $y_i = \tilde{y}_k$ (same class), otherwise $\alpha_{i,k} = -1$ (different class). For Head II-A and Head III-A architectures, the objective function $\mathcal{L}_{Dist}$ is defined as

$$\mathcal{L}_{Dist} = \frac{1}{NK} \sum_{i=1}^{N} \sum_{k=1}^{K} \left[\frac{1}{HW} \sum_{h=1,w=1}^{H,W} \sum_{c=1}^{C} (\hat{f}_{x_i}[c,h,w] - \hat{f}_{p_k}[c,h,w])^2 \right]^{\alpha_{i,k}}. \quad (10)$$

The term inside the the square bracket is simply the average of all $L2$-norms computed for each spatial position (h, w). Similarly, the objective function $\mathcal{L}_{Dist}$ for Head II-B and Head III-B is defined as

$$\mathcal{L}_{Dist} = \frac{1}{NK} \sum_{i,k}^{N,K} \left[\frac{1}{HW} \sum_{h,w,c}^{H,W,C} (\hat{f}_{x_i}[c,h,w] - \hat{f}_{p_k}[c,h_{\max},w_{\max}])^2 \right]^{\alpha_{i,k}}, \quad (11)$$

and $\mathcal{L}_{Dist}$ for Head III-C and Head III-D is expressed as $\mathcal{L}_{Dist} = \mathcal{L}_{Dist_x} + \mathcal{L}_{Dist_p}$, where $\mathcal{L}_{Dist_x}$ is from Eq. (11) and $\mathcal{L}_{Dist_p}$ can be defined in a similar manner.

3.4 Explanation for Top-k Nearest Prototype Examples

We provide two forms of explainability: (1) prediction explanation with top-k nearest prototype examples and (2) the pixel evidence of similarity between the prediction sample and the prototypes. In this work, we use the Layer-wise Relevance Propagation (LRP) algorithm [16,4] for explainability. The LRP algorithm is an explanation method where the prediction score $h(x)$ is backpropagated to the input space x. For the convolutional-2D layers, we used the following LRP-$\alpha\beta$ rule: $R_d^{(l)} = \sum_j \left(\alpha \, \frac{(a_d w_{dj})^+}{\sum_{0,d}(a_d w_{dj})^+} - \beta \, \frac{(a_d w_{dj})^-}{\sum_{0,d}(a_d w_{dj})^-} \right) R_j^{(l+1)}$, where $\alpha - \beta = 1$, $\alpha > 0$, and $\beta \geq 0$. For the other layers, including the similarity (in Head I and II) and the attended similarity (in Head III) layers, we use the following LRP-ϵ rule: $R_d^{(l)} = \sum_j \frac{a_d w_{dj}}{\epsilon + \sum_{0,d} a_d w_{dj}} \, R_j^{(l+1)}$. The similarity/attended similarity layer is treated like a linear layer which allows the easy application of the LRP-ϵ rule in the standard manner without resorting to more complex LRP methods such as BiLRP [9]. For Head III architectures, the relevance scores flow through the attended similarity layer branch only through the features, while the attention weights are treated as constants.

From the last linear layer to the similarity/attended similarity layer, we compute the relevance scores $R(x, p_k)$ for each pair of (x, p_k). To compute the LRP heatmap for x based on the similarity between x and p_k, the relevance score $R_{(\cdot)}^{(l)}(x, p_k)$ at the similarity/attended similarity layer l is backpropagated to the input space of x. The computation of LRP heatmap for p_k also follows the same manner. For each layer l' between the input x and the similarity/attended similarity layer, the relevance map for explaining x differs from the relevance map for explaining a prototype p_k due to the different forward pass values, i.e. $f(x) \neq f(p_k)$, resulting in a different redistribution of relevance across feature map neurons.

3.5 Quantifying the Degree of Outlierness

Our proposed architectures have the capability to quantify the degree of outlierness for each input sample x using the corresponding prototype similarity scores $U = \{u_k \mid 1 \leq k \leq K\}$. For all Head I and Head II architectures, the u_k score is assigned as $u_k = z(x, p_k)$, where $z(x, p_k)$ is the input to the linear layer as shown in Figure 1 and 2. For Head III-A and Head III-B architectures, the u_k score is computed as $u_k = \frac{1}{HW} \sum_{h=1,w=1}^{H,W} r_{h,w}(x, p_k)$. Finally, for the Head III-C and Head III-D architectures, the u_k score is computed as $u_k = \max_{h,w}(r_{h,w}(x, p_k))$. It is to be noted in this case that the substitution of the $r_{h,w}(x, p_k)$ with $r'_{h,w}(x, p_k)$ will produce the same u_k score (refer to Section 3.2 for the formulation of $r'_{h,w}(x, p_k)$).

With the placement of ReLU layers in the CNN, the elements of U lie in the range of $[0, 1]$, where a higher value indicates a larger similarity between the input sample x and the prototype p_k. From set U, top-K' highest scores are selected. For sample x, the outlier score $o(x)$ is defined as $o = 1 - \frac{1}{K'} \sum_{k=1}^{K'} u_k$. Note that the u_k are in general no inner products, and thus we cannot compute

true metric distances and follow rather the idea in [19]. A larger value indicates a larger anomality.

4 Experimental Settings

In our experiments, we used the LSUN [25] dataset consisting of 10 classes with the image size of 128×128 pixels and the PatchCamelyon [3,23] dataset consisting of 2 classes with the original image size of 96×96 pixels. For the LSUN dataset, we subsampled $10,000$ samples per class from the given training set and used the $80:20$ split for training and validation. The given validation set was used as the test set. For the heatmap perturbation experiments, we subsampled only 100 samples for each class, from our LSUN test set, and 500 samples for each class, from our PatchCamelyon test set. We set $\alpha = 1.7$, $\beta = 0.7$, and $\epsilon = 1e-10$ for the LRP-$\alpha\beta$ and LRP-ϵ rules. Three different types of outlier setups are used in our experiments. Setup A consists of our LSUN test set which is labelled as the normal sample set and the test set from Oxford Flowers 102 [17] dataset which is labelled as the outlier sample set. For Setup B, we created a synthetic outlier counterpart for each test sample in our test set (LSUN/PatchCamelyon) by drawing random *strokes* of thickness M on the original image. More details regarding the creation of *strokes* anomalies can be found in the work [7]. In Setup C, we created synthetic outlier counterpart for each test sample by manipulating the color of the image, by increasing the minimum saturation and value components, and then randomly rotating the hue component. Due to the large number of test samples in the PatchCamelyon dataset, 1500 samples per class were sampled from the test set for the outlier detection experiments. For each class, we used 10 and 20 prototypes for the LSUN and PatchCamelyon experiments, respectively.

We used a ResNet-50 [11] pretrained on ImageNet as the CNN encoder and the SGD optimizer with momentum 0.9 and weight decay of $1e-4$. The learning rates of $1e-3$ and $1e-4$ were used in the prototype network and the CNN encoder, respectively. A decaying learning rate with a step size of 10 epochs and $\gamma = 0.1$ was also adopted. For the learning objectives, we set $\lambda_1 = \lambda_2 = 1$. We use PyTorch [18].

5 Results and Discussion

5.1 Comparison of Prediction Performance

Based on Table 1, it can be observed that all the proposed head architectures except Head III-A, perform better than the teacher network on the LSUN dataset. The classification performance of Head III-A is only slightly below the teacher network. This implies that our student architectures do not need to compromise on classification accuracy to provide the two forms of explainability (refer to the beginning of Section 3.4). Among the proposed architectures, Head III-C

and Head III-D architectures achieved the best performance among the proposed networks. However, they are computationally more expensive to train due to the additional attention map used and they do not give the best heatmaps explanation which will be discussed later with reference to Figure 4. Thus, we choose only the best variant from each type of Head architecture (Head I, Head II-B, and Head III-B) for the experiments on the PatchCamelyon dataset. Head II-B is chosen as it is the counterpart of Head III-B. In Table 2, our proposed networks also outperform the teacher network on the PatchCamelyon dataset.

Table 1: Classification performance on LSUN test set.

Model	**Acc. (%)**
Teacher (*baseline*)	80.1
Head I	82.0
Head II-A	82.1
Head II-B	82.1
Head III-A	79.1
Head III-B	81.2
Head III-C	**83.1**
Head III-D	**83.1**

Table 2: Classification performance on PatchCamelyon test set.

Model	**Acc. (%)**	**AUC**
Teacher (*baseline*)	80.5	0.9142
Head I	81.5	0.9061
Head II-B	**81.8**	0.8991
Head III-B	81.3	**0.9307**

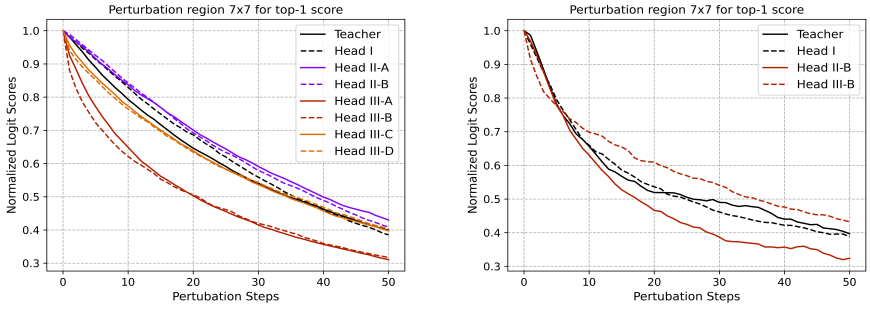

Fig. 4: Heatmap quality evaluation on LSUN (left) and PatchCamelyon (right) datasets. The perturbation for the student will use the LRP score computed between each test sample and its top-1 prototype.

5.2 Comparison of Pixel-wise Explanation Performance

In Figure 4, we assess the quality of the LRP heatmap explanation for each network by performing a series of region perturbations in the input space of

the test sample. The perturbation will start from the most relevant region first (based on the LRP scores) to the least relevant region. This will lead to a sharp decrease in the logit score of the predicted class as the relevant features in the input sample are gradually removed in a decreasing order of importance. A steeper decrease indicates better heatmap quality. For more information on this method, refer to [20].

The most important result we can observe in Figure 4 is that using prototype-based students does not hurt the explanation sensitivity as it results in a similar local explanation sensitivity as compared to the teacher. The Head III-B architecture has the best heatmap quality, followed by Head III-A architecture for the LSUN dataset. Although Head III-C and Head III-D architectures achieved the best performance in Table 1, they do not provide the best explanation. In general, the attention modules in Head III contribute to better explanation qualities for the networks compared to Head I and II. For the PatchCamelyon dataset, the plot suggests better heatmap quality for Head III-B architecture in the first 5 perturbation steps only, and Head II-B architecture for the remaining perturbations. This is not surprising, as there is a lower diversity of objects in a patch in PatchCamelyon dataset compared to LSUN dataset, resulting in a lesser need for attention weighting. The former consists of a small number of cell types, stromal tissue and background.

Due to lack of space, we show only LRP heatmaps from Head III-B architecture in Figure 5 and 6. For each input sample, we compute one heatmap for the sample and for its top-1 nearest prototype from each class. The score u_k from Section 3.5 determines the nearest prototype. For the LSUN dataset, we show the top-1 nearest prototype only for the groundtruth class and 4 selected classes due to lack of space. The positive and negative evidence in the LRP heatmap is represented by red and blue pixels, respectively.

In the left subplot of Figure 5, heatmaps for mostly all classes are dimmed out due to their low similarity u_k score, except class 3 and 4 (*classroom* and *conference room*). This is semantically plausible.

In the lower right subplot of Figure 5 explaining the prototypes, we observe that regions with tables, chairs, and sofa are lit up in the heatmaps. It means these prototype regions are similar to the region that lit up in the input heatmap, which is shown in the upper right subplot (sofa and coffee table).

For the PatchCamelyon dataset in Figure 6, we show from the left to the right, heatmaps for the predicted class for a true negative, a true positive and a false negative. The true negative prediction highlights positive scores on dense clusters of lymphocytes (the dark regular nuclei). Not shown due to lack of space, we confirmed that the closest prototype is an image of tissue-invading lymphocytes. For the false negative, the heatmap focuses on structures resembling cancer nuclei, which however are outside of the 32×32 window which defines the label for a 96×96 patch of the PatchCamelyon dataset. Thus, the heatmap is plausible given that the label considers only the 32×32 center.

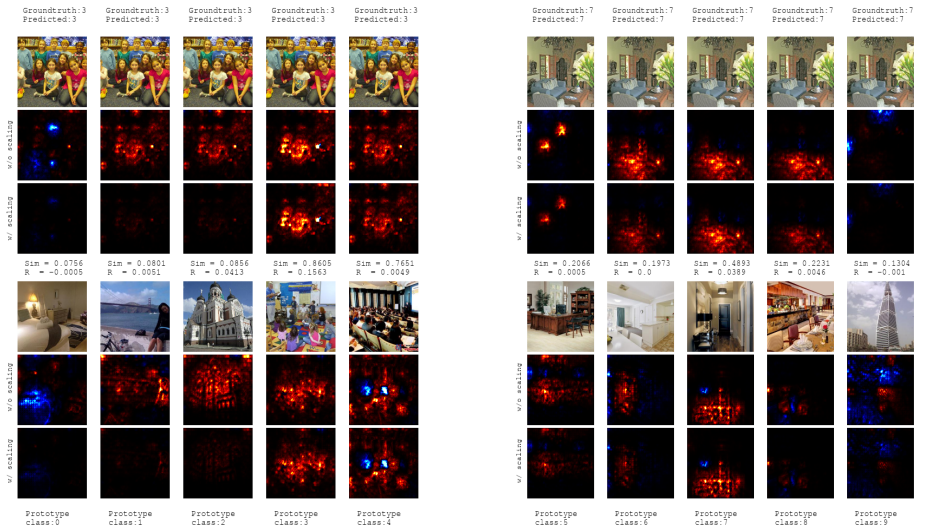

Fig. 5: LRP heatmaps from Head III-B on the LSUN dataset for the prediction *classroom* (left) and *living room* (right). Subplots in the upper part show heatmaps explaining the test sample, while the lower part show heatmaps explaining each shown prototype. In all subplots, each column corresponds to heatmaps with respect to the top-1 prototype of a selected class. The heatmaps in the second row are unscaled, while heatmaps in the third row are scaled such that the pair (x, p_k) with lower u_k score will appear dimmer.

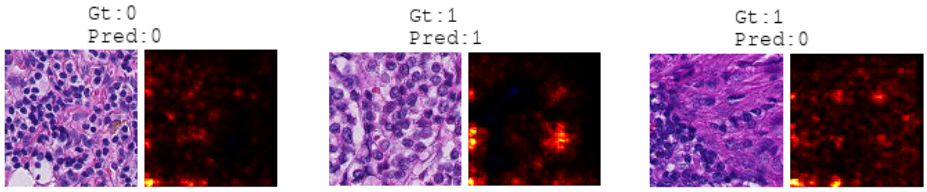

Fig. 6: LRP heatmaps from Head III-B on the PatchCamelyon dataset. The left, center, and right subplots are for a true negative (tumor absent), true positive (tumor present), and false negative predictions, respectively.

5.3 Evaluation of Outlier Detection Performance

Besides providing two forms of explainability, our proposed architectures are also equipped with outlier detection capability. The networks are able to quantify the degree of outlierness o, for a given input sample x using the same network trained on LSUN and PatchCamelyon datasets. The three outlier setups are described in Section 4. Table 3 shows the outlier results for the networks trained with the LSUN dataset when all samples (normal and outlier) are used in the evaluation.

Table 3: Reported AUC using the outlier score o for LSUN networks.

Model	Setup A Flowers			Setup B LSUN strokes_5			Setup C LSUN altered color		
	Top-1	*Top-20*	*All proto.*	*Top-1*	*Top-20*	*All proto.*	*Top-1*	*Top-20*	*All proto.*
Head I	0.976	**0.991**	0.604	0.689	0.660	0.416	0.716	0.793	0.585
Head II-A	0.951	0.968	0.850	0.744	0.766	0.625	0.739	0.816	0.763
Head II-B	0.977	0.987	0.958	0.747	**0.778**	0.653	0.766	**0.840**	0.833
Head III-A	0.978	0.985	0.917	0.772	0.752	0.616	0.678	0.687	0.594
Head III-B	0.934	0.951	0.809	0.725	0.756	0.626	0.654	0.717	0.640
Head III-C	0.977	0.964	0.523	0.710	0.574	0.379	0.709	0.637	0.488
Head III-D	0.969	0.946	0.447	0.707	0.582	0.364	0.719	0.648	0.464

Table 4: Reported AUC using the outlier score o for PatchCamelyon networks.

Model	Setup B PCam strokes_5			Setup C PCam altered color		
	Top-1	*Top-20*	*All proto.*	*Top-1*	*Top-20*	*All proto.*
Head I	0.626	0.595	0.545	0.494	0.430	**0.892**
Head II-B	**0.677**	0.619	0.442	0.432	0.407	0.531
Head III-B	0.617	0.606	0.569	0.494	0.566	0.720

We refrained from the common usage of CIFAR-10 or MNIST classes, which are known to cluster very easily, and offer little potential for aggregating similarities spatially due to low spatial resolution.

Based on Table 3, the Head I architecture using the top-20 u_k scores achieved the best AUC on the setup with the flower samples as outliers. This makes sense, as the texture of the flowers dataset is different from LSUN scenes across the whole image, and using early spatial pooling does not affect detection performance. Similarly, for PatchCamelyon, the altered color setup turns every region of an image into abnormal colors. Thus, the spatial pooling layer does not hurt and offers the lowest complexity with least potential for overfitting. In contrast, for the more complex sceneries of LSUN the spatial matching Heads II have slightly better performance over Head I, but the attention Heads III overfit.

For the top-20 u_k scores, the Head II-B architecture achieved the best performance for the other setups, namely LSUN/strokes and LSUN/colors. The outlier performance for Head III-B architecture that gives the best explanation quality is on par with the best performing network for both flowers and *strokes* outlier setups, but a larger performance difference is observed for the altered color setup. Generally, the Setup A which uses flowers as outliers and all samples from the given LSUN test set are easier to be detected due to the significant difference in the image statistics. Thus, the AUC metric reported for Setup A is notably better than the other two setups. The other two setups are considered more difficult as they involved the manipulation of the samples from the original test set, and thus are harder to be detected as they share texture statistics between normal and outliers. These results motivate our usage of more complex

outlier definitions in hindsight as compared to the common usage of CIFAR-10 or MNIST classes.

We reported results on the PatchCamelyon dataset only for the more difficult setups. In Table 4, Head II-B also achieved the best AUC score on Setup B but with top-1 score, and Head I demonstrated superior performance for Setup C when using all prototype scores.

6 Conclusion

We explored various head architectures to provide two forms of explainability. Our student network also ensures that the generated explanations are meaningful and is capable of quantifying the degree of outlierness for a sample. However, there is a trade-off between the quality of the generated explanations and the outlier detection performance. The network architecture that generates the best LRP explanation does not have the best outlier detection capability. For future work, we will explore the application of the proposed network for other tasks.

References

1. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician **46**(3), 175–185 (1992)
2. Angiulli, F., Pizzuti, C.: Fast outlier detection in high dimensional spaces. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) Principles of Data Mining and Knowledge Discovery. pp. 15–27. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
3. Bejnordi, B.E., Veta, M., Van Diest, P.J., Van Ginneken, B., Karssemeijer, N., Litjens, G., Van Der Laak, J.A., Hermsen, M., Manson, Q.F., Balkenhol, M., et al.: Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. Jama **318**(22), 2199–2210 (2017)
4. Binder, A., Bach, S., Montavon, G., Müller, K.R., Samek, W.: Layer-wise relevance propagation for deep neural network architectures. In: Information Science and Applications (ICISA) 2016, pp. 913–922. Springer (2016)
5. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: Identifying density-based local outliers. In: ACM SIGMOD. p. 93–104 (2000). https://doi.org/10.1145/342009.335388
6. Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., Su, J.K.: This looks like that: deep learning for interpretable image recognition. In: NeurIPS. pp. 8930–8941 (2019)
7. Chong, P., Ruff, L., Kloft, M., Binder, A.: Simple and effective prevention of mode collapse in deep one-class classification. In: IJCNN 2020. pp. 1–9 (2020)
8. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**(3), 273–297 (1995)
9. Eberle, O., Büttner, J., Kräutli, F., Müller, K.R., Valleriani, M., Montavon, G.: Building and interpreting deep similarity models. arXiv preprint arXiv:2003.05431 (2020)
10. Gu, X., Akoglu, L., Rinaldo, A.: Statistical analysis of nearest neighbor methods for anomaly detection. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) NeurIPS, pp. 10923–10933 (2019), `http://papers.nips.cc/paper/`

9274-statistical-analysis-of-nearest-neighbor-methods-for-anomaly-detection.pdf
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
12. Latecki, L.J., Lazarevic, A., Pokrajac, D.: Outlier detection with kernel density functions. In: Perner, P. (ed.) MLDM 2007. Lecture Notes in Computer Science, vol. 4571, pp. 61–75. Springer (2007). https://doi.org/10.1007/978-3-540-73499-4_6
13. Li, O., Liu, H., Chen, C., Rudin, C.: Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. arXiv preprint arXiv:1710.04806 (2017)
14. Lu, W., Pang, C., Zhang, B.: Attentive prototype few-shot learning with capsule network-based embedding
15. Ming, Y., Xu, P., Qu, H., Ren, L.: Interpretable and steerable sequence learning via prototypes. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 903–913 (2019)
16. Montavon, G., Binder, A., Lapuschkin, S., Samek, W., Müller, K.R.: Layer-wise relevance propagation: an overview. In: Explainable AI: interpreting, explaining and visualizing deep learning, pp. 193–209. Springer (2019)
17. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing. pp. 722–729. IEEE (2008)
18. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
19. Ramirez-Padron, R., Foregger, D., Manuel, J., Georgiopoulos, M., Mederos, B.: Similarity kernels for nearest neighbor-based outlier detection. In: International Conference on Advances in Intelligent Data Analysis. p. 159–170. Springer-Verlag, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13062-5_16
20. Samek, W., Binder, A., Montavon, G., Lapuschkin, S., Müller, K.R.: Evaluating the visualization of what a deep neural network has learned. IEEE TNNLS **28**(11), 2660–2673 (2016)
21. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
22. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Advances in neural information processing systems. pp. 4077–4087 (2017)
23. Veeling, B.S., Linmans, J., Winkens, J., Cohen, T., Welling, M.: Rotation equivariant cnns for digital pathology. In: MICCAI. pp. 210–218. Springer (2018)
24. Yang, H.M., Zhang, X.Y., Yin, F., Liu, C.L.: Robust classification with convolutional prototype learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3474–3482 (2018)
25. Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., Xiao, J.: Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365 (2015)
26. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision. pp. 818–833. Springer (2014)