DRC: Discrete Representation Classifier with Salient Features via Fixed-prototype

Qinglei Li, Qi Wang*, *Member*, Yongbin Qin, Xinyu Dong, Xingcai Wu, Shiming Chen, *Member*, Wu Liu, *Member*, Yong-Jin Liu, *Member*, Jiebo Luo, *Fellow, IEEE*

***Abstract*—Image classification models including convolutional neural networks (CNN) and vision transformers (ViT) commonly employ a fully connected (FC) layer as the classifier. However, the fully connected nature of FC brings large amounts of weight parameters, limits the efficiency of inference, tends to over-fit the training data, and struggles to learn distinct class weights. To solve these problems, we propose a discrete representation classifier (DRC), a generic parameter-free classifier that offers efficiency, robustness, and more discriminative categorization. Specifically, the DRC discards numerous unimportant features and focuses solely on the salient features which are reinforced during training and presented in short discrete form during inference. Unlike the way of learning pseudo-prototypes (weights) from data laden with complex patterns and noises in FC, the DRC introducing discriminative fixed-prototypes which are almost uniformly distributed across the high-dimensional feature space, thus helps the model to learn more distinct boundaries between categories. Further leveraging the advantage of DRC's focus on salient features, we propose Salient-CAM, which is able to locate the most important region in image without the need for weighting feature maps. The experiments demonstrate that simply replacing the model's classifier from FC to DRC can lead to a significant acceleration in the whole model's inference and a more robust classification. Additionally, the proposed Salient-CAM exhibits excellent object localization ability in complex natural scenes.**

***Index Terms*—Classification, Fully connected layer, Inference, CAM, Discrete representation, Prototypes.**

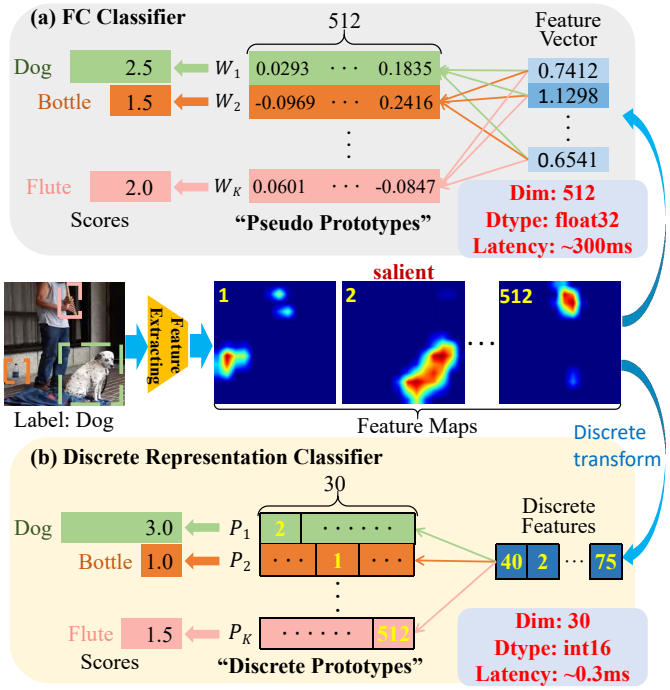


Fig. 1: The difference between DRC and FC in classification implementation. The above FC classifier requires receiving complete floating-point features and performing linear calculations with the learned class weights. In contrast, the DRC accepts light discrete features and performs computations with fixed discrete prototypes that exhibit better discriminability, achieving more efficient and accurate classification. The figure displays the inference latency of the classifiers in a cold start state, demonstrating the higher inference efficiency of DRC.

I. Introduction

IMAGE classification [19] is one of the fundamental tasks in computer vision, which has many downstream tasks such as object detection [6], semantic segmentation [1]. Moreover, image classification plays a crucial role in numerous practical applications, including plant disease detection [8], [9], autonomous driving systems [17], and medical image analysis [18]. Therefore, research on image classification has a significant impact on the entire field of computer vision, driving advancements and progresses.

Most of the commonly used image classification models are based on convolutional neural networks (CNNs) [13] and transformers [36], typically ending with a Fully Connected (FC) layer as the default classifier. In the development of deep learning, FC is first proposed in AlexNet [19], which serves as a bridge between high-level feature representations learned by convolutional layers and final category predictions. Subsequent neural networks like VGGNet [30], GoogLeNet [32], ResNet [13], and ViT [36] employ more complex structures and continue to incorporate FC for prediction. However, the fully connected nature of the FC layer, which links every input feature to every output class, results in a high computational burden. This architectural choice significantly increases the number of parameters and computational complexity, leading to delays and inefficiencies that are particularly problematic in

Q. Li, Q. Wang, Y. Qin, X. Dong and X. Wu are with the State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China.

S. Chen is with the Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China.

W. Liu is with the College of Information Science and Technology, University of Science and Technology of China, Anhui 230026, China.

Y. Liu is with the College of Computer Science and Technology, Tsinghua University, Beijing 100084, China.

J. Luo is with the Department of Computer Science, University of Rochester, Rochester, NY 14627, USA.

real-time applications or on devices with limited computational resources. The vast parameters in FC make the model more prone to over-fitting [2]. Furthermore, the learning process within the FC classifier can be viewed as prototype learning. In this process, each weight vector independently corresponds to a category, which we refer to as pseudo-prototypes, as shown in Fig. 1 (a). It is difficult for FC to learn enough discriminative pseudo-prototypes on a large number of generally noisy data [18], hence resulting in fuzzy and inaccurate class boundaries. To cope with the issues of vast parameters in FC layer, researchers begin adopting pruning strategies [39], [43] and quantization techniques [10] to simplify the FC. For instance, Hubara et al. [16] utilize quantized neural networks that convert the weights and activations of a model from a floating point to a discrete low-bit representation. These approaches enable the reduction of FC layer size and accelerate computation [10]. But in fact, the fine-tuning procedures are often required to ensure distinguishability between pseudo-prototypes. Overall, these approaches optimize FC but still do not get rid of the limitation of using FC as a classifier.

To achieve a novel classifier that does not require the full connection of features to classes, which often leads to high computational costs, we propose a discrete representation that converts full-dimensional floating-point features into low-dimensional integer features, thus lowering computational costs. We consider striking a balance between simplifying features and preserving the discriminative power of features. On the one hand, by integrating the concept of salient features [46], we design a concise representation to emphasizes the most important features in a discretized form, as shown in Fig. 1 (b). This approach effectively reduces dimensionality and complexity, leading to enhanced computational efficiency. On the other hand, motivated by the perspective of learning "pseudo prototypes" in FC classifier, we redefine the classification process to initialize fixed discrete prototypes of each category, as shown in Fig. 1. The advantage of the fixed discrete prototype is that it facilitates the convergence of samples to regions with distinct category boundaries. For example, in Fig. 1 (b), an input image can be transformed into a 30-dim (int16) discrete feature representation, and a more accurate classification is rapidly achieved by calculating similarity with fixed discrete prototypes.

In this paper, we introduce a discrete representation classifier (DRC) with salient features via a fixed-prototype, including training and inference phases. During the inference process, we utilize solely the salient features, which significantly enhances the speed and efficiency of the inference. Meanwhile, the training process guides the model to learn binary-like features, helping to ensure that the information loss during discrete inference is minimal. In the training phase, the prototype of each category is randomly assigned in the feature space in binary form, holding a fixed and substantial distance between categories, ensuring distinct separation. Then, we design an adaptive activation function to highlight the salient feature and introduce a binary-learning feature matching module to train the model smoothly. In the inference stage, both image features and binary prototypes are transformed into a discrete form, focusing on a few key features and employing a tailored similarity measure to achieve rapid classification without compromising accuracy. In addition, we propose a DRC-based class activation mapping method, called Salient-CAM, which selectively processes only salient activation maps, without relying on weight or gradient information. Experimental findings indicate that the proposed DRC performs higher inference efficiency compared to the FC classifier, along with superior classification performance in the majority of scenarios. Moreover, the Salient-CAM demonstrates superior capability in localized regions of interest, especially in natural scenes.

Overall, the main contributions of this work are listed below:

- We introduce fixed prototypes to progressively align categories with these fixed prototypes, guiding the model to train salient features.
- We design a discrete representation for features that can minimally represent salient features, significantly reduce feature dimensions, and speed up inference.
- We devise a class activation mapping method that focuses only on salient features, highlighting the most important regions of the image without needing to weight feature maps.
- We integrate the training with fixed prototypes and the discrete inference to develop the DRC, which can serve as an alternative to the FC classifier, offering comparable performance with faster inference speed.

II. Related Works

A. Fully Connected Layer (FC)

The FC layer is a core component in neural networks [19], each neuron is connected to all neurons in the previous layer. With the development of deep learning, the FC layer is applied to realize functions such as feature integration, decision making, and pattern recognition. For instance, Basha et al. [2] explore how FC layers impact the performance of CNNs [13] in image classification tasks, highlighting their role in integrating features for effective classification. Lei et al. [20] indicate that FC layers are often used at the end of the network to assimilate features for decision-making, especially in classification tasks. Ding et al. [7] point out that FC layers are fundamental in neural networks for recognizing and interpreting patterns, particularly in image recognition. However, applying FC layers inevitably increases model complexity and computational cost, while being prone to over-fitting the training data. We regard the process of learning weights inside the FC layer as a kind of prototype learning [40], and completely replace FC with the similarity calculation [5] process between fixed-prototypes and features, which greatly alleviates these problems.

B. Class Prototypes

Class prototypes [38] refer to a set of representative samples or instances used to represent different classes or concepts in data. These prototypes are typically used for several purposes, including contrastive learning, few-shot classification, category representation, and few-shot segmentation. For example,

This article has been accepted for publication in IEEE Transactions on Circuits and Systems for Video Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2024.3453052

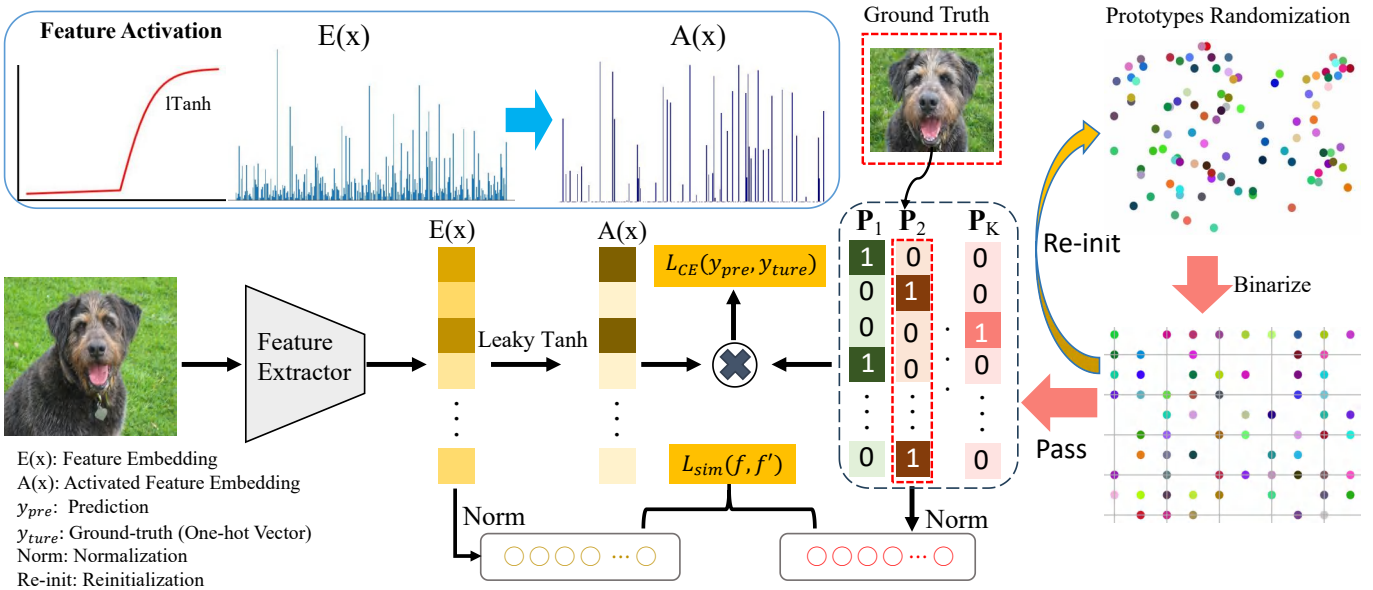


Fig. 2: The procedure of the binary-inspired learning. The qualified random prototypes are transformed into binary prototypes. Then, the obtained E(x) is transferred to A(x) by leaky Tanh. Furthermore, E(x) and A(x) are separately calculated to similarity and logit with the binary prototypes. Finally, we adopt the two loss functions, L_{sim} and L_{CE} to optimize the model.

Liu et al. [24] adopt cluster prototype as a classifier for self-supervised learning, where cluster prototypes are used to generate pseudo labels by clustering and maintaining a memory to store instance features representing the cluster centroids for contrastive learning. Snell et al. [31] obtain the prototype by calculating the mean of the feature vectors of all samples in each category, representing the center points of these categories for use in few-shot learning. Mensink et al. [27] introduce the nearest class mean approach, where each class is represented by the mean of its examples, allowing classes to have multiple prototypes. Li et al. [21] propose an adaptive prototype learning method for few-shot segmentation, using SGC and GPA to extract and allocate multiple prototypes. However, these methods typically derive prototypes by computing the centroids of sample features [40], which are time-consuming when learning distinct prototypes in standard image classification tasks. In a different manner, fixed prototypes are uniformly distributed in the feature space with appropriate spacing between them, guiding the feature gradually close to distinct fixed-prototypes.

C. Class Activation Mapping

Class activation mapping (CAM) is a visualization technique used to understand the decision process of deep convolutional [19] neural networks in image classification tasks. Zhou et al. [46] first proposed CAM, a method that locates key regions using a weighted sum of the final convolutional layer's activation maps and evaluates channel significance through weights from the next fully connected layer. Selvaraju et al. [29] present Grad-CAM, using the gradients flowing into the last convolutional layer to understand the importance of each neuron for a decision of interest. Chattopadhay et al. [4] further design Grad-CAM++, an advanced version of Grad-CAM, created to provide better visual explanations for CNN-based models. Wang et al. [34] introduce Score-CAM, a method that evaluates the importance of each activation map based on its contribution to the target class score, without using gradients. However, these methods rely on gradient information or additional weight information to weight the activation maps, which incurs additional time cost and resource computation. In contrast, Salient-CAM does not depend on any gradient or weight information and directly selects activation maps for visualization based solely on the salient features. As a result, salient-CAM not only achieves an extremely fast generation of class activation maps but also maintains an excellent ability for localizing salient regions.

III. Methodology

A. Overview

In this section, we introduce binary-inspired learning as a method to enhance the representational power of discrete features, providing a detailed explanation of the training process of DRC. Then, we illustrate the discrete inference process, discussing DRC's inference strategy, which leverages discrete representations of features, enabling a significant reduction in computation costs. Finally, we introduce Salient-CAM, a class activation technique tailored for DRC, enhancing interpretability and efficacy. It's worth noting that the 'features' mentioned are the feature vectors obtained from average pooling the final feature maps in this section. The selection of salient features is based on the intensity of each feature map, measured by its average pooling value.

B. Binary-inspired learning

We usually adopt an or more FC layer to construct a mapping for classifying, when facing a classification task.

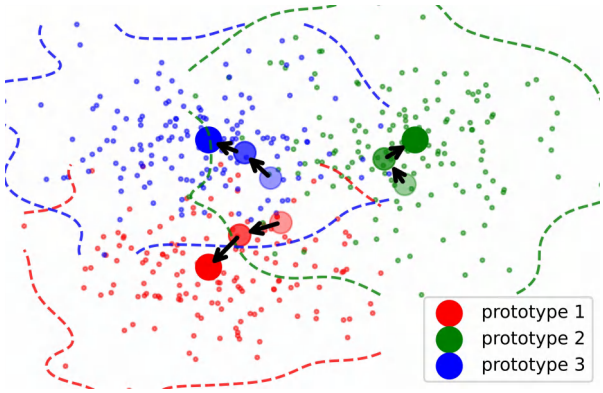
However, this simple way exists some issues. On the one hand, the FC classifier mixes all features, making it hard to trace which features contribute to decisions and posing great difficulties in understanding the classification process [22]. On the other hand, the fully connected mapping between features and classes introduces numerous parameters, which not only increases the model's deployment and inference costs but also drives the model toward over-fitting the training data [22]. According to the mechanism of the FC classifier, see Appendix A. We understand that the FC classifier's training process closely resembles learning prototypes for each category. Therefore, we may use this insight as a starting point to design a classification process that can replace the FC classifier.

To achieve this idea, we propose a binary-inspired learning process that efficiently performs classification based solely on a few distinctive features without introducing parameters. The schematic description of our proposed process is illustrated in Fig. 2. Initially, prototypes are randomly initialized and binarized (see right of the figure). If the prototypes do not meet the required standards after initialization, they will undergo reinitialization. An image, after being processed through a feature extractor, produces a feature vector E(x). This vector E(x) is then utilized to compute a similarity loss function with binary prototypes after normalization. In parallel, E(x) is also processed through a leaky tanh activation function to derive an binary-like feature vector A(x), which is then multiplied with binary prototypes to calculate the category logits for the cross-entropy loss. In the following part, we introduce three key components of our *binary-inspired learning*, including *binary prototypes*, *leaky tanh*, and *loss function*.

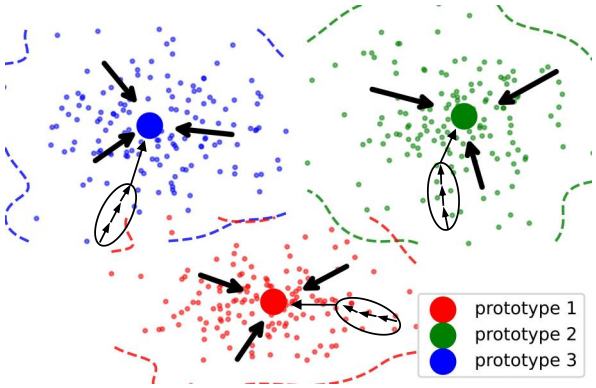
1) binary prototypes: As the common way, the class prototypes can be calculated by every category feature cluster. However, due to the larger volume of data required to be processed in standard classification tasks, it's extremely difficult to compute the centroids of each category in the feature space as exhaustively as done in few-shot learning tasks, as shown in Fig. 3(a). Besides, the presence of prototypes also occupies a non-negligible amount of space, presenting a spatial concern.

We believe that prototypes maintain appropriate distances from each other can make the class boundaries clearer. Based on this idea, we can evenly distribute the prototypes across the feature space and fix their positions, allowing the samples to gradually approach the prototypes, as shown in Fig. 3(b). Therefore, we propose a fixed prototype approach, ensure that the prototypes are broadly distributed throughout the entire feature space, with adequate distance maintained between each prototype.

In detail, we opt to generate prototypes through random initialization, requiring no computational resources or time costs. The initialized prototypes are presented in a binary form, which succinctly highlights the differences between the prototypes. Due to the natural separation distance in the random distribution of vectors in high-dimensional Euclidean space, combined with our initialization qualification check based on XOR operations, there is no need to be concerned about the coincidence of random prototypes. Furthermore, samples around the prototypes exhibit clearer boundaries after convergence, as shown in Fig. 3.



(a) Feature cluster prototype



(b) Fixed prototype

Fig. 3: The learning process for different prototypes. (a) The general feature cluster prototype, gradually finding a center of category. (b) Our proposed fixed prototype, making the feature itself gradually close to the defined prototype. The fixed prototype gives a high-dimensional spatial division, resulting in the model more easily optimizing features into the clearer category boundaries.

Overall, we denote P_k as the prototype for class *k* and *rand()* as a function that randomly generates a vector of a specified length. The P_k is defined as:

$$P_k = rand(C), s.t. \forall k \in \{1, \ldots, K\}. \tag{1}$$

where K and C indicate the number of classes and the length of the feature vector output by the backbone network, respectively. Here, we have embraced a straightforward approach aimed at preserving solely the most prominent elements within each prototype. By retaining the top-n largest elements of the prototype P_k and assigning them a value of 1, while setting the remaining elements to 0. We represent this binary vector as $\mathbf{P}_k$. The value of its *m*-th element is determined as follows:

$$\mathbf{P}_{k,m} = \begin{cases} 1, & \text{if } P_{k,m} \in top(P_k, n), \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

where top(P_k,n) represents the largest n elements of P_k.

Considering the general scenario with the number of classes equal to 1000, prototype length equal to 2048, and the number of ones equal to 50, the total number of possible

prototypes that can be generated is $\binom{2048}{50}$. The probability of generating a prototype that is exactly the same as a given prototype is $\frac{1}{\binom{2048}{50}}$, which is approximately 2.1×10^{-169}. This extremely small probability indicates that the likelihood of two prototypes being completely identical is negligible. Furthermore, we aim for initializing prototypes that have good inter-class distances, we evaluate the adequacy of prototype initialization based on the diversity of the two prototypes with the highest similarity. If the diversity does not meet the criteria, reinitialization is performed. Here is the definition of diversity for binary prototypes $\mathbf{P}$:

$$diversity(\mathbf{P}) = min\{\sum_{m=1}^{C} \mathbf{P}_{i,m} \oplus \mathbf{P}_{j,m}, \forall i, j \in \{1, \ldots, K\}\}, \quad (3)$$

where C denotes the dimension of each prototype $\mathbf{P}_k$, and i, j represent the indices of the i-th, j-th prototypes, respectively. Then, the diversity for prototypes $\mathbf{P}$ is defined using the summation after operation of XOR ($\oplus$) between the m-th elements (where m ranges from 1 to C) in each prototype.

2) leaky tanh: Directly using the feature vector E(x) to match the binary prototype forces the features to gradually binarize, limiting the expressive power of E(x) and leading to a less smooth updating process. To ensure the model's output features remain as natural as possible with minimally constrained, we believe that an adaptive transform to the similar domain for logit calculation can avoid unnatural feature representations and non-smooth learning. Based on this consideration, we propose an activation function before the similarity computation. This function is designed to amplify the influence of more critical features while reducing the impact of less significant ones, ultimately aiming to achieve a binary-like representation of the features. Additionally, the activation function needs to possess adaptive properties, as the saliency threshold for each output feature varies.

To achieve this, we adopt a modified version of the hyperbolic tangent function, as shown in top of Fig. 2. We shift the positive axis of the original Tanh function by a certain amount, equal to the activation threshold. Therefore, feature elements exceeding the threshold are activated to 1, whereas those below the threshold are suppress to 0. In the region below the activation threshold, to ensure gradient updating, we employ a proportional function. We refer to this activation function as *leaky tanh* or *lTanh*. Therefore, the calculation formula of the *leaky tanh* function is as follows:

$$lTanh(x) = \begin{cases} \alpha \cdot (x-t) \cdot [arctanh(m)/[x]_{max}), & x \leq t, \\ tanh((x-t) \cdot (arctanh(m)/[x]_{max})), & x > t, \end{cases} \quad (4)$$

where α represents the proportional coefficient we have set, *t* denotes the computed threshold, and *m* represents the desired maximum mapped value after activation. In the top of Fig.2, we present the scenario with 512 data points, and the number of activated elements is 50. The adaptive property of the function arises from the varying value of *t* based on the input feature E(x).

3) loss: We still employ the cross-entropy loss function as the primary loss for classification training. Since binary-like feature embedding A(x) = lTanh(E(x)) expresses its salient features approximately, this allows us to directly calculate the similarity between the input x and each category by performing element-wise multiplication of the A(x) and the binary prototype $\mathbf{P}_k$. The specific formula for the cross-entropy loss function is:

$$L_{CE} = \sum_{i=1}^{N} -log(\frac{exp(A(x_i) \cdot \mathbf{P}_{y_i}^T)}{\sum_{k=1}^{K} exp(A(x_i) \cdot \mathbf{P}_k^T)}, \quad (5)$$

where *N* represents the number of samples in this calculation, x_i represents the *i*-th sample, and y_i is the ground truth label for x_i, *T* is a transpose operation.

Meanwhile, in order to enhance the proximity of feature embeddings E(x) to the prototypes, we propose the incorporation of an auxiliary convergence loss function to restrict the distance between embeddings and prototypes. Initially, we compute normalized vectors for both embeddings and prototypes. We denote the normalized feature embedding as $\hat{E}$ *(x)*, and the normalized prototype as $\hat{\mathbf{P}}_k$, limited by:

$$\mid \hat{E}(x) \mid_2 = \mid \hat{\mathbf{P}}_k \mid_2 = 1. \quad (6)$$

In order to impartially quantify the degree of diversity among them, we compute the squared Euclidean distance between the embedding of the sample and the prototype while ensuring equal Euclidean norms. The specific formula is as follows:

$$L_{sim} = \frac{1}{N} \sum_{i=1}^{N} (\hat{E}(x_i) - \hat{\mathbf{P}}_j)^2, j = y(x_i), \quad (7)$$

where N denotes the batch size, x_i represents the i-th sample, and y(x_i) indicates the label of x_i. Optimizing this objective function directly facilitates the gradual convergence of samples to categories.

Overall, after defining the two loss functions, our final loss function is:

$$Loss = L_{CE} + \lambda \cdot L_{sim}, \quad (8)$$

where λ is a factor used to balance the similarity loss and the cross entropy loss.

C. Discrete inference

Maintaining identical forward propagation in both training and inference phases might not fully exploit the opportunity to optimize and simplify models for inference. Since back-propagation and gradients are not needed during inference, certain structures or features could be streamlined to make the model more lightweight [15], [35] and efficient, especially for deployment on smaller devices.

In the inference stage, binary prototypes and image features are converted into a highly concise discrete sequences. As features are identified as salient or not, we could focus solely on the salient ones across the dimension of features. Additionally, we design a tailored similarity measure for discrete sequences to effectively compute discrete similarity. According to this idea, we design a novel discrete inference process for classification, as shown in Fig. 4. We directly extract the index positions of salient features on channel dimension, and use

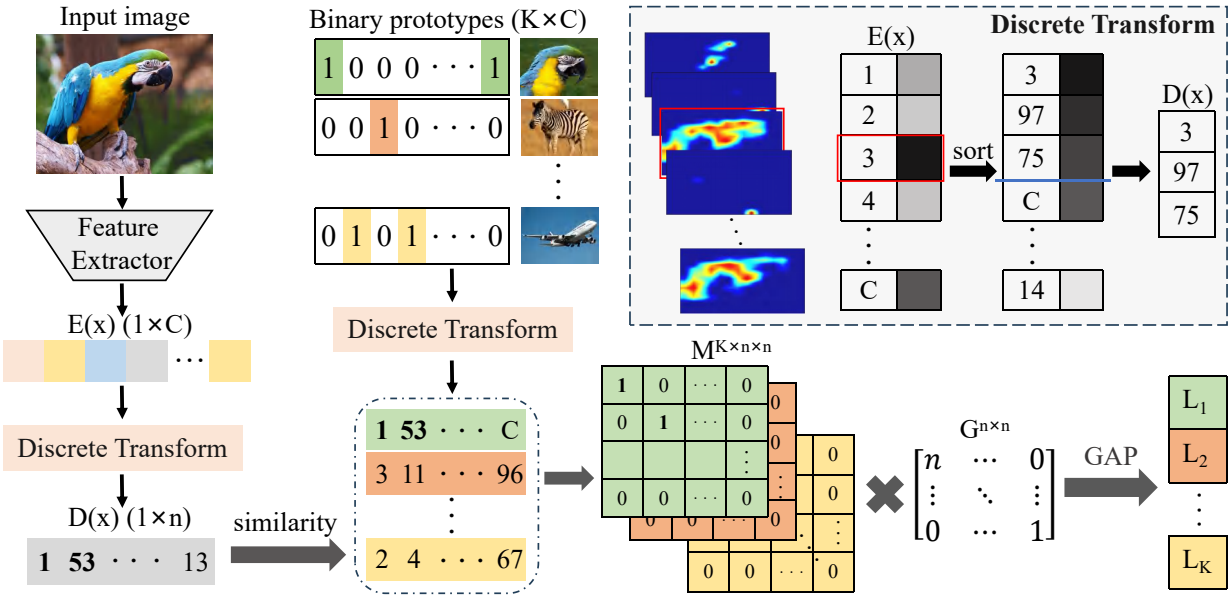


Fig. 4: The inference process after discrete transformation of features. An image undergoes feature extraction, resulting in a C-dimensional feature vector, which is then transformed into a much shorter discrete sequence. This sequence is compared with discretely transformed prototypes using discrete similarity calculations to obtain overlap matrices for each category. These matrices are then processed through weighted computation and average pooling to yield scores for each category.

Algorithm 1 Discrete Similarity Computation.

Input: GAP layer's output $X \in R^d$;
Number of salient features $n \in Z_+$;
Class binary prototypes $\boldsymbol{P} \in \{0,1\}^{K \times d}$;
Output: Logits for all categories.

Initialize: discrete similarity matrix $M \leftarrow \{0\}^{K \times n \times n}$; decreasing diagonal matrix $O \leftarrow \{0\}^{n \times n}$;
/* get prototypes' discrete representation */
$S \in Z_+^{K \times n} \leftarrow$ indices of value 1 in each prototype of $\boldsymbol{P}$;
/* Discrete similarity computation */
for all $S_i \in Z_+^n$ **in** S **do**
  Update M_i, where the rows and columns correspond to each element of S_i and X, and set the elements that are equal to 1;
end for
/* Weighted by degree of feature saliency */
for all $M_i \in \{0,1\}^{n \times n}$ **in** M **do**
  $W \leftarrow$ *matrix_multiply*(M,O);
  $L_i \leftarrow$ *sum*(W) ;
end for

the index positions as discrete representation. For example, each element of E(x)'s discrete representation is defined as:

$$D_i(x) = index\_Sort(E(x), i, n) \tag{9}$$

where D_i(x) denotes the i-th element of the discrete representation of x, and the *index_Sort(v,i,n)* is a function which first sorts the vector v in descending order, then returns the index of the i-th element after sorted. Besides, n is equal to the number of activated elements in binary-inspired learning.

At the prototype level, we can convert a binary prototype into a short discrete sequence, where each element is equally important and uniquely corresponds to a typical feature of the prototype. Similarly, the binary-like feature embedding of image can be also transformed to discrete representation. The difference is that the salient features of the image differ in degrees of significance, thus need to multiply with a weight matrix of decreasing size to simply preserve the different significance.

We designed our own algorithm to compute the similarity of two discrete sequences, as shown in Algorithm 1. It generates a set of matrices M, the elements of M can be formulated as:

$$\mathbf{M}_{i,j,k} = \begin{cases} 1, & \text{if } D_j(\mathbf{P}_i) = D_k(x), \\ 0, & \text{otherwise,} \end{cases} \tag{10}$$

where matrix M_i corresponds to the i-th class. For a matrix of certain category, each element with a value of 1 indicates the alignment between the input x and the prototype on salient features. After applying matrix multiplication with a decreasing diagonal matrix, the resulting weighted matrix for category i is obtained as:

$$\hat{\mathbf{M}}_i = \mathbf{M}_i \mathbf{G}, \tag{11}$$

where $\mathbf{G}$ represents a diagonal matrix that decreases one by one. The summation of the elements in this weighted matrix reflects the similarity between the input and the corresponding category i. We denote it as L_i, it's specific formula is:

$$L_i = \frac{1}{n^2} \sum_{j}^{n} \sum_{k}^{n} \hat{\mathbf{M}}_{i,j,k}, \tag{12}$$

Salient-CAM. Since the DRC is trained with a guiding principle that focuses exclusively on salient features, mainstream

CAM methods, which typically involve processing all features, contradict our philosophy of utilizing only important features. To address this problem, we propose a CAM visualization method specifically designed for the DRC, Salient-CAM. This method discards a large number of irrelevant feature maps, selectively processing only the important feature maps [23], without requiring additional weight or gradient information, the specific procedure is shown in Algorithm 2. Given a class of interest c, Salient-CAM $U^c_{Salient-CAM}$ can be defined as:

$$U^c_{Salient-CAM} = \sum_k \alpha^c_k A^k, \tag{13}$$

$$\alpha^c_k = \mathbf{P}_{c,k}, \tag{14}$$

where A^k denotes the k-th activation map. Since the saliency of the feature map in terms of category has already been determined by the binary prototype, we can directly obtain the binary coefficient for each feature map.

Algorithm 2 Salient CAM algorithm.

Input: Image *X*, Backbone *f(x)*, class *c*, Prototypes ***P***, overlay coefficient λ ;
Output: Salient-CAM of Image *X* corresponds to the label *c*;

Initialize: S $\leftarrow \{0\}^{W\times H}$,where W,H is the width and height of the Image X;
/* get activation of model's final layer */
M $\leftarrow$ *f(X)* ;
I $\leftarrow$ the indices of salient units in ***P*** corresponding to c;
for *k* **in** *I* **do**
  /* using bilinear interpolation for upsampling */;
  U $\leftarrow$ Upsample(M_k);
  /* each salient feature map is superimposed on S */;
  S.accumulate(*U*);
end for
/* normalizing the activation map pixels */
S $\leftarrow$ (*S*-*S*.min())* 255 / (*S*.max()-*S*.min());
/* overlay of the colored S with the original image X */
Salient-CAM $\leftarrow$ *color*(*S*)*λ + *X*(1-λ);

The computation process of Salient-CAM is remarkably straightforward and efficient due to the explicit indication of the indices of all salient feature units. By concentrating the model's expressive power on salient units, we believe that class activation maps obtained through this design of DRC can effectively highlight the prominent features of objects.

IV. Experiments

We focus on two key procedures of DRC: Binary-inspired learning and Discrete inference. For brevity, we denote them as *Binary* and *Discrete* respectively. The inference latency measurements for all models are obtained on an NVIDIA GTX1070. Our research is conducted on a Windows 10 operating system using Python 3.8, with PyCharm as the development tool. We use the PyTorch framework for deep learning. It is important to clarify that these latency results are measured from cold starts. The first inference typically represents the main bottleneck, causing significant delays in real-world applications, such as edge computing, where models are frequently initialized from scratch.

A. Dataset and setting

Dataset. We select CIFAR-100, tiny-ImageNet, and CUB-200 datasets for image classification analysis. The CIFAR-100 dataset consists of 32x32 pixel color images across 100 classes, divided into 50,000 training and 10,000 test images. The tiny-ImageNet, a larger dataset, is designed to classify images into 200 categories and includes 100K training images along with 10K test images. The CUB-200 dataset, is a benchmark dataset in the field of fine-grained visual categorization, contains images of 200 different bird species, totaling 11,788 images. To augment our data, we employ horizontal flipping and random cropping methods, resulting in output images of 224×224 pixels.

Setting in training. In our approach to training models with pre-trained weights. We conduct 120 epochs using the Adam optimizer, starting with a learning rate of 1e-4. This rate was reduced by a factor of 5 every 30 epochs, including a warm-up phase. For training from scratch on CIFAR-100, we subject all models to a 300-epoch training regimen, employing a SGD optimizer. We integrate the Nesterov [3] algorithm and set the batch size to 128. The initial learning rate was 0.1, which decreased by a factor of 10 in the 150th and 225th epochs. The weight decay and momentum settings were set to 1e-4 and 0.9, respectively.

B. Comprehensive comparison between DRC and FC

1) Effectiveness: To evaluate the transfer learning ability of the DRC, we conduct classification experiments on CIFAR-100, tiny-ImageNet, and CUB-200, using a series of representative CNN [13] models, ViT [36] models, MLP [44] model and GNN [11] model. All models are loaded with ImageNet pre-trained weights, accuracy is shown in *Top-1* and *Top-5*, and *Latency* reflects the inference latency of the entire model on 100 images. In Table I, Table II, and Table III, we present the dimensions of the input features processed by the DRC and FC, denoted as *Dim*. For the DRC, Dim represents the number of salient features. Additionally, we also show the reduction in the number of parameters in the zero-parameter DRC compared to FC, denoted as *Params*.

First, DRC performs well on a variety of neural network models with different infrastructures, confirming its universality to a variety of models. Second, DRC has a Top-1 accuracy that is competitive with FC and even exceeds FC, which proves its excellent classification capability. Because the representation power of salient features is enhanced in the Binary learning process and the design of fixed prototypes leads to more discriminative class boundaries. Third, in the Top-5 index, FC is almost always superior to DRC because the Top-5 index benefits from the broader feature information that FC utilizes, capturing a wider array of features that contribute to the classification decision. In contrast, DRC focuses primarily on a few significant features, thus naturally performing worse. Then, it is noteworthy that Discrete has the lowest inference latency on almost all models, and simply replacing the final classifier component of models can obviously accelerate the inference speed of the entire model. Last, DRC uses a much

TABLE I: The accuracy comparison on the CIFAR-100 dataset. All models are loaded with pre-trained weights. The notation Δ indicates the number of parameters reduced by DRC compared to FC. ♠CNN, ♣Transformer, ♦MLP, ★GNN.

	Top-1(%)			Top-5(%)			Latency(s)			Dim		Params(K)
Models	FC	Binary	Discrete	FC	Binary	Discrete	FC	Binary	Discrete	FC	DRC	Δ
♠ResNet34 [13]	84.28(±0.22)	**84.41**(±0.29)	84.37	97.03	96.17	96.16	1.368	1.367	**1.015**	512	**50**	-51
♠ResNet50 [13]	84.15(±0.14)	**84.21**(±0.13)	84.04	96.88	97	96.88	1.498	1.554	**1.144**	2048	**50**	-205
♠ResNet101 [13]	85.81(±0.27)	**85.98**(±0.33)	85.94	97.63	97.33	97.29	1.651	1.694	**1.338**	2048	**50**	-205
♠ResNeXt50(32×4d) [42]	84.3(±0.28)	**84.53**(±0.09)	84.39	96.89	96.83	96.67	1.573	1.620	**1.154**	2048	**50**	-205
♠ResNeXt101(32×8d) [42]	86.4(±0.27)	**86.71**(±0.38)	86.78	97.19	97.19	97.07	2.227	2.167	**1.820**	2048	**50**	-205
♠ResNeSt50 [45]	85.79(±0.2)	**85.82**(±0.09)	85.62	97.27	96.09	96.03	2.448	2.450	**2.030**	2048	**50**	-205
♠EfficientNet-b3 [33]	88.03(±0.21)	**88.09**(±0.22)	88.01	98.19	97.78	97.67	1.606	1.666	**1.324**	1536	**30**	-154
♠DenseNet-121 [14]	84.36(±0.25)	**84.45**(±0.17)	84.24	97.27	96.40	96.43	1.457	1.488	**1.140**	1024	**35**	-103
♣Vit-b32 [36]	**90.65**(±0.24)	90.35(±0.02)	90.3	98.96	94.89	94.92	2.851	1.428	**1.406**	768	**30**	-77
♣PVT-S [37]	86.35(±0.26)	**86.57**(±0.13)	86.53	97.62	96.65	96.29	1.648	1.638	**1.593**	512	**30**	-51
♣CVT-13 [41]	**88.31**(±0.12)	88.22(±0.2)	88.19	98.41	97.29	96.73	1.646	1.640	**1.628**	384	**20**	-39
♣Swin-T [25]	87.83((±0.19))	**88.15**(±0.28)	88.02	98.07	97.89	97.52	1.669	1.686	**1.620**	768	**25**	-77
♦Poolformer-s24 [44]	86.31((±0.23))	**86.49**((±0.34))	86.28	97.72	97.01	96.35	1.571	1.577	**1.220**	512	**20**	-51
★Pyramid-ViG-Ti [11]	**85.21**(±0.15)	85.12(±0.26)	84.41	97.48	96.77	96.55	1.860	1.855	**1.848**	1024	**30**	-103

TABLE II: The accuracy comparison on the tiny-ImageNet dataset. All models are loaded with pre-trained weights. The notation Δ indicates the number of parameters reduced by DRC compared to FC. ♠CNN, ♣Transformer, ♦MLP, ★GNN.

	Top-1(%)			Top-5(%)			Latency(s)			Dim		Params(K)
Models	FC	Binary	Discrete	FC	Binary	Discrete	FC	Binary	Discrete	FC	DRC	Δ
♠Resnet50 [13]	75.01(±0.32)	75.43(±0.36)	**75.63**	90.76	90.14	89.52	1.427	1.412	**1.089**	2048	**50**	-410
♠ResnetXt50(32×4d) [42]	**76.9**(±0.13)	76.78(±0.17)	76.61	92.84	90.34	90.22	1.507	1.469	**1.162**	2048	**50**	-410
♠ResneSt50 [45]	78.62(±0.16)	**78.83**(±0.25)	78.73	92.17	89.69	89.8	2.417	2.370	**2.016**	2048	**50**	-410
♠EfficientNet-b3 [33]	83.32(±0.08)	**83.49**(±0.22)	83.33	95.57	93.74	93.78	1.571	1.594	**1.238**	1536	**30**	-308
♠DenseNet-121 [14]	74.72(±0.15)	**74.89**(±0.13)	74.76	90.94	88.26	88.13	1.469	1.459	**1.141**	1024	**35**	-205
♣PVT-S [37]	**80.84**(±0.28)	80.75(±0.2)	80.62	94.12	90.9	90.18	1.619	1.534	**1.501**	512	**30**	-103
♣CVT-13 [41]	**85.21**(±0.19)	85.09(±0.16)	84.84	95.85	93.27	92.8	1.715	1.692	**1.598**	384	**20**	-77
♣Swin-T [25]	83.38(±0.26)	**83.69**(±0.21)	83.48	94.71	94.09	93.33	1.686	1.713	**1.568**	768	**25**	-154
♦Poolformer-s24 [44]	81.32(±0.23)	**81.47**(±0.11)	81.25	94.92	92.18	91.11	1.501	1.471	**1.139**	512	**20**	-102
★Pyramid-ViG-Ti [11]	77.77(±0.06)	**78.31**(±0.12)	78.02	92.97	90.99	90.84	3.230	1.969	**1.839**	1024	**30**	-207

TABLE III: The accuracy comparison on the CUB-200 dataset. All models are loaded with pre-trained weights. The notation Δ indicates the number of parameters reduced by DRC compared to FC. ♠CNN, ♣Transformer, ♦MLP, ★GNN.

	Top-1(%)			Top-5(%)			Latency			Dim		Params(K)
Models	FC	Binary	Discrete	FC	Binary	Discrete	FC	Binary	Discrete	FC	DRC	Δ
♠Resnet50 [13]	79.34(±0.08)	**80.63**(±0.1)	80.61	93.94	94.12	93.78	1.267	1.2669	**0.9313**	2048	**40**	-410
♠ResnetXt50 [42]	81.14(±0.15)	**81.41**(±0.23)	81.28	94.77	94.11	93.94	1.2538	1.2882	**0.9392**	2048	**40**	-410
♠ResneSt50 [45]	82.24(±0.29)	**82.41**(±0.22)	82.25	95.61	94.29	94.21	1.4999	1.5243	**1.1427**	2048	**30**	-410
♠DenseNet-121 [14]	80.49(±0.07)	**80.67**(±0.21)	80.55	95.08	93.43	93.23	1.2523	1.2497	**0.9054**	1024	**30**	-205
♣PVT-S [37]	82.98(±0.12)	83.21(±0.14)	**83.29**	95.42	94.7	94.47	1.2927	1.3009	**1.2839**	512	**25**	-103
♣Swin-T [25]	84.23(±0.06)	**85.07**(±0.14)	84.92	96.38	95.71	95.46	1.3461	1.3586	**1.295**	768	**25**	-154
♦Poolformer-s24 [44]	80.43(±0.23)	**82.29**(±0.08)	82.03	95.12	93.94	93.42	1.2755	1.3032	**0.9219**	512	**25**	-102
★Pyramid-ViG-tiny [11]	82.04(±0.13)	**82.22**(±0.32)	81.87	95.28	93.29	93.05	2.7933	1.3946	**1.3576**	1024	**20**	-207

smaller number of features than FC, and because of its zero-parameter nature it also reduces the number of parameters that cannot be ignored. Overall, the experiment results highlight the viability and potential of DRC as an alternative to traditional FC classifier.

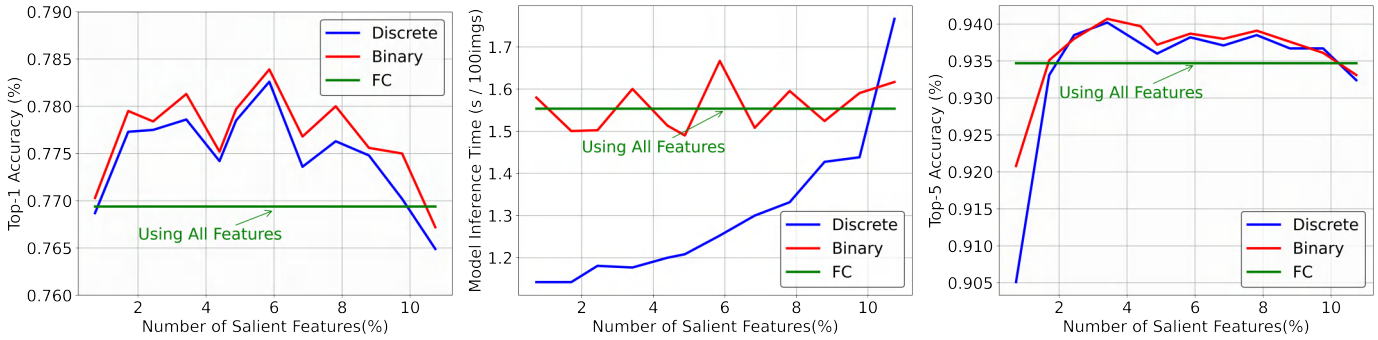
TABLE IV: The accuracy(%) of training from scratch on the CIFAR-100. All models are initialized using Kaiming Initialization [12].

	Top-1(%)			Top-5(%)		
Models	FC	Binary	Discrete	FC	Binary	Discrete
ResNet34	77.45	**78.08**	78.02	93.52	91.54	91.6
ResNet50	76.94	**78.11**	77.93	93.47	94.1	93.92
ResNet101	78.6	**79.08**	78.88	94.08	94.73	94.62
ResNet152	79.12	**79.13**	79.04	94.14	94.46	94.4
ResNeXt50	77.09	**78.69**	78.57	93.98	94.23	94.14
ResNeXt101	79.13	**79.35**	79.34	94.68	94.56	94.55
ResNeSt50	78.13	**79.03**	78.96	94.1	94.88	94.81

2) Training from scratch: To further validate the effectiveness of our method, we compare the classification performance of DRC classifier against that of FC classifier, without loading pre-training weights. Our experimental setup encompasses a variety of ResNet architectures, including advanced ResNeXt and ResNeSt variants. For a fair comparison, we replicate the same training procedures for FC classifier. Specifically, for training DRC, we set m $\bar{3}$, $\alpha = 0.01$, $\lambda = 3$. It should be noted that, with the exception of resnet34, which has an output feature dimension of 512, the other models have an output feature dimension of 2048, and DRC only uses the most significant 60 features on all models. The results are systematically presented in Table III.

On the one hand, DRC has an obvious accuracy improvement on all models, except for ResNet152 model, revealing the superiority of DRC in the context of training from scratch. This is because the fixed prototypes used by DRC have clear and

This article has been accepted for publication in IEEE Transactions on Circuits and Systems for Video Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2024.3453052



(a) Top-1 Accuracy (b) Model Inference Latency (c) Top-5 Accuracy

Fig. 5: Comparing the performance of DRC using different quantities of salient features with FC using all features, including Top-1 accuracy, inference latency, and Top-5 accuracy.

differentiated class convergence centers at the beginning of the training of the randomly initialized model, thereby improving the classification performance. On the other hand, the DRC utilizes only a small fraction of the features processed by the FC classifier: only 3% of the features for most models, and 12% for the ResNet34 model. This indicates that DRC only receives the most important part of features that are meaningful for classification and is not affected by the interference of redundant features, thus improving the robustness of the model.

3) Number of salient features: To explore the impact of varying the number of salient features on the accuracy and inference speed of DRC, we show the change curves of Top-1 accuracy, inference latency, and Top-5 accuracy, as shown in Fig.5. The FC classifier handles all the features, serving as a baseline reference. Experimental results are obtained from a ResNet50 model trained from scratch on the CIFAR-100 dataset.

First, we plot Top-1 accuracy for cases where the number of salient features is less than 10%, as depicted in Fig.5(a). We conclude that: 1) DRC consistently maintains better Top-1 accuracy compared to FC with less than 10% salient features, underscoring DRC's ability to leverage the most influential features. 2) as more salient features are held, DRC's classification performance paradoxically declines due to a reduced average prominence of these features.

Second, we explore the trend of inference latency variations of DRC at different levels of salient features, as illustrated in Fig.5(b). We find that: 1) As the number of salient features drops below 10%, the computational efficiency of discrete inference surpasses that of FC, indicating enhanced inference speed with fewer features. 2) The inference speed of Binary learning remains comparable to that of FC and is not affected by the number of salient features, as it utilizes the entire feature vector dimension.

Last, we show the variation curve of Top-5 accuracy, as illustrated in Fig.5(c). The results show that: Both Binary learning and Discrete inference show a sharp decline in Top-5 accuracy when less than 2% of features are used, attributed to the broader range of potential labels in the Top-5 metric requiring more feature information.

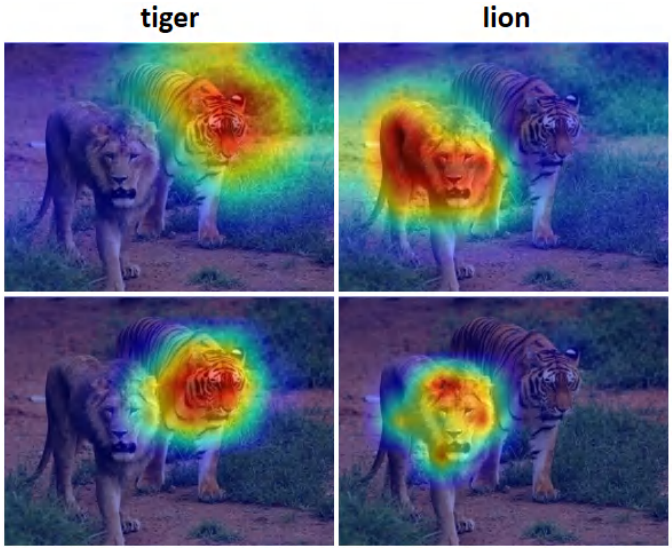


Fig. 6: The regions of interest for the two classifiers are visualized using Score-CAM. The FC classifier is shown at the top, and the DRC classifier at the bottom.

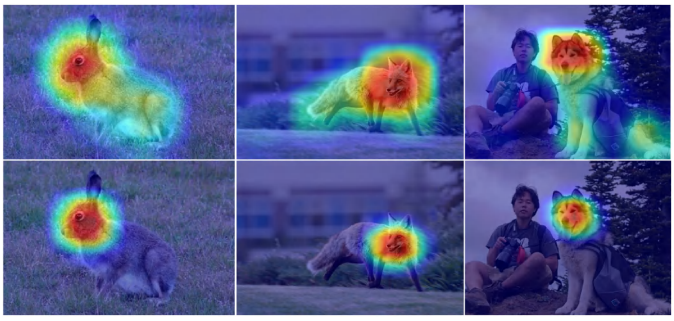


Fig. 7: The regions of interest for the two classifiers are visualized using Grad-CAM++. The FC classifier is shown at the top, and the DRC classifier at the bottom.

4) Region of interest: To test the hypothesis that the model equipped with DRC indeed focuses only on the salient features of the image during the classification decision process, we conduct experiments using two models: one with an FC classifier and one with a DRC. Both models use the ResNet50

TABLE V: The ablation study about α

	Binary		Discrete		Δ	
α	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
0.5	77.42	93.76	74.75	91.57	-2.67	-2.19
0.1	77.58	93.85	76.15	92.27	-1.43	-1.58
0.05	77.19	93.79	76.46	92.89	-0.73	-0.9
0.01	**78.36**	**94.02**	**78.24**	**94.04**	-0.12	0.02
0.005	77.8	93.93	77.72	93.78	-0.08	-0.15
0.001	77.54	93.62	77.54	93.71	0	0.09

architecture and are fine-tuned to achieve comparable levels of accuracy. We then use two advanced visualization techniques, Score-CAM in Fig.6 and Grad-CAM++ in Fig.7, to examine the model's region of interest.

In the Score-CAM visualizations (Fig. 6), the FC classifier (upper panel) concentrates on the entire contour of the target object, covering nearly half of the image area. The DRC classifier (lower panel), however, focuses primarily on the head region of the tiger or lion, disregarding the rest of the image. Furthermore, Grad-CAM++ visualizations (Fig. 7) further confirm that the different ROIs of the two classifiers stem from the essential differences between their mechanisms. These findings indicate that FC, due to its fully connected nature, focuses on all feature information, resulting in a broader region of interest. In contrast, DRC enhances salient features and reduces the impact of redundant features during training, allowing the model's region of interest to focus solely on the most important parts of objects.

C. The impact of α

We conduct experiments to investigate the impact of varying α parameters on the binary-inspired learning. The parameter α, acts as the scaling factor in the lTanh function applied to the region below the activation threshold. To observe the effect of different values of alpha on the decline in accuracy after discrete transformation, we introduce an additional metric indicator Δ, which reflects the accuracy change of the Discrete inference compared to the Binary learning. We conduct experiments on the CIFAR-100 dataset and use the ResNet50 model to train from scratch.

From Table V, it can be seen that the α value of 0.01 produces the highest Top-1 and Top-5 accuracy across all experiments. As alpha is related to the weakening strength of unimportant features, a higher value of alpha means less weakening of unimportant features, and vice versa. On the one hand, insufficiently reducing the influence of unimportant features may indirectly prevent the salient features from standing out, thereby impairing the classification mechanism of DRC. On the other hand, excessively diminishing the influence of these features can lead to a gradual slowdown or even stagnation in the gradient update, which hinders the model's optimization. In addition, as α decreases, the problem of accuracy decline after discrete transformation is gradually alleviated. A reasonable explanation is that as discrete inference relies primarily on positive activation elements, reducing the scaling factor below the activation threshold has a progressively smaller effect on discrete inference results.

TABLE VI: Different ways to generate prototypes on ImageNet-1K

generating method	Top-1(%)	Top-5(%)
50	78.93	93.11
100	79.05	93.31
150	78.88	92.37
300	78.88	93.19
500	79.03	93.25
random(ours)	78.86	93.17

D. Fixed-prototypes

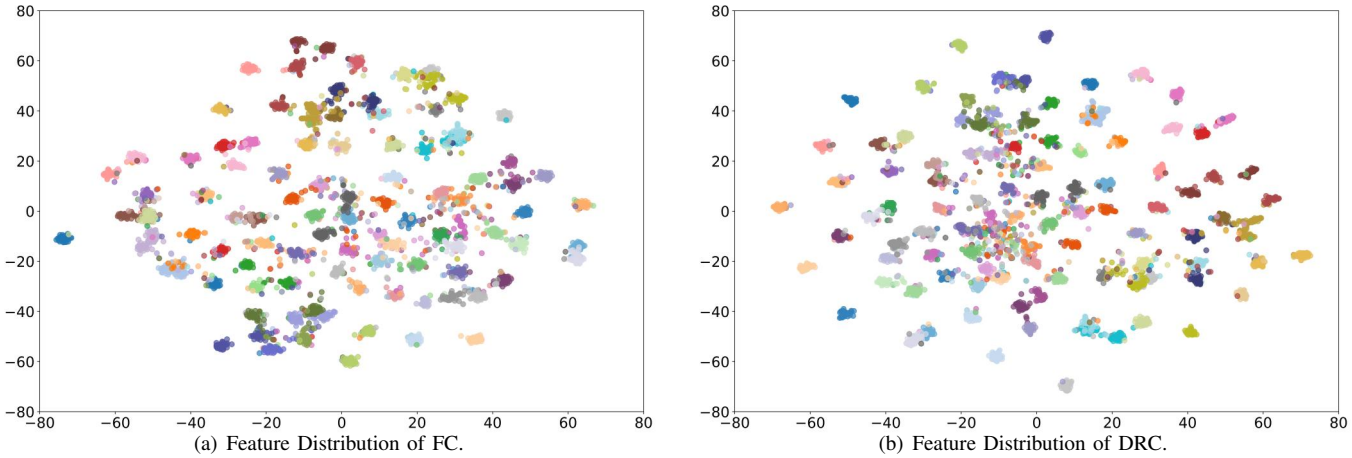
1) Generate of prototypes: To explore the effectiveness of generating prototypes in random initialization, we conduct experiments on ImageNet-1K [19], using a pre-trained ResNeSt network. Our aim is to evaluate randomization against the traditional method of using center embeddings of training examples from each category as prototypes. Recognizing the impracticality of iterating through the large number of samples in the training set, we adopt a pragmatic solution: a random selection approach to obtain a subset of samples from each class for calculating center embeddings as prototypes. We conduct experiments by randomly sourcing 50, 100, 150, 300, and 500 samples from each category to compute center embeddings as prototypes.

As detailed in TABLE VI, we find that the performance variations between the two kinds of methods are relatively marginal. On the one hand, this indicates that even models with feature extraction capabilities cannot obtain sufficiently representative prototypes in a large amount of data by aggregating feature centers, which may be due to the prevalence of noise [18] and a fraction of wrong labels in the data set [28]. On the other hand, this finding suggests that the random initialization strategy, when combined with a well-trained backbone that has already learned specific feature patterns, remains reliable. And efficiency gains from randomization do not come at the expense of distinct performance loss, making it an attractive option for scenarios that demand zero-cost of generating prototypes.

2) t-SNE visualization of feature distributions: In this section, we visualize the distribution of features learned by the DRC method in the feature space. As a comparison, we visualize the feature distribution of the FC model. We use 5000 test samples from the top 100 classes of the tiny-ImageNet dataset for this visualization, comparing the sample features output by the model guided by DRC with those of FC. Using the EfficientNet-B3 backbone network, both the DRC classifier and the FC classifier achieved an accuracy of 83%. Furthermore, we use a Siamese network with contrastive [26] loss to visualize its feature distribution, comparing it with our DRC method. Both methods use ResNet34 as the backbone and include 10 classes from the CIFAR-10 dataset.

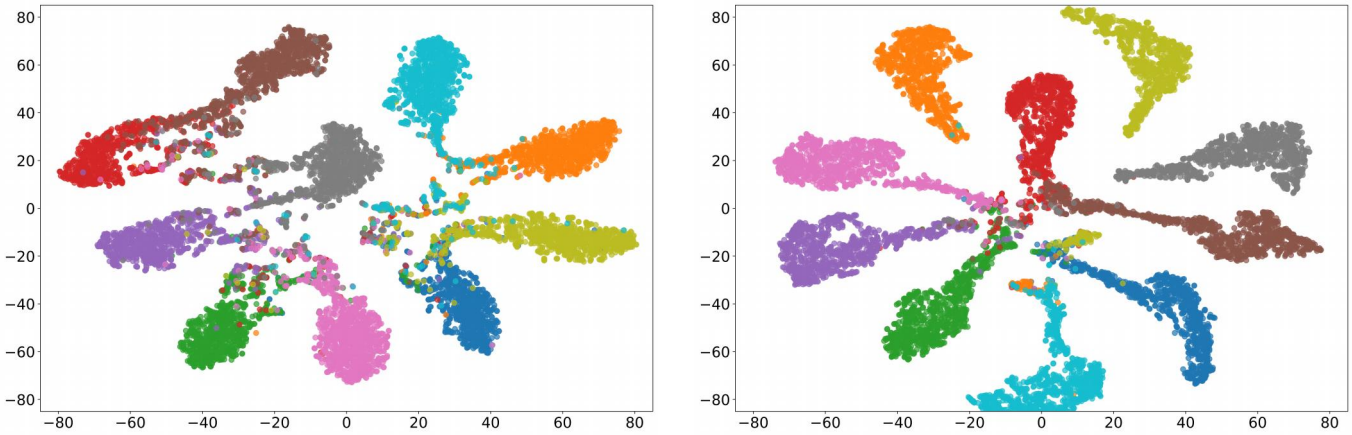
Based on the visualizations of Fig.8 and Fig.9, it can be seen that the features of the DRC occupy more space in the feature space. This aligns with our design principle: prototypes are

This article has been accepted for publication in IEEE Transactions on Circuits and Systems for Video Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2024.3453052



(a) Feature Distribution of FC. (b) Feature Distribution of DRC.

Fig. 8: Feature Space Analysis through t-SNE Visualization, including the top 100 classes from the tiny-ImageNet dataset.



(a) Siamese network utilizing ResNet34 as the backbone with contrastive loss. (b) DRC using ResNet34 as the backbone.

Fig. 9: Feature Space Analysis through t-SNE Visualization, including 10 classes from the CIFAR-10 dataset.

TABLE VII: The ablation study on L_{sim}, λ denotes the coefficient of L_{sim}.

	Binary		Discrete		Δ	
λ	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
0(baseline)	79.55	94.6	79.27	94.46	-0.28	-0.14
3	80.46	**94.81**	80.27	**94.76**	-0.19	-0.05
10	**80.63**	93.79	**80.63**	93.04	0	-0.23
20	80.55	91.44	80.51	91.22	-0.04	-0.22
40	80.49	90	80.55	89.88	0.06	-0.12

TABLE VIII: The ablation study of activation function, y=x means no activation function is used.

	Binary		Discrete	
Activation Function	Top-1(%)	Top-5(%)	Top-1(%)	Top-5(%)
y=x(baseline)	82.76	94.55	82.75	90.06
softmax	82.23	90.34	82.24	87.08
sigmoid	82.38	90.87	82.39	87.56
tanh	82.67	94.12	82.6	89.61
leaky tanh (ours)	**83.86**	**96.86**	**83.77**	**96.76**

randomly distributed in the high-dimensional space, allowing them to occupy spatial positions more evenly and thus utilize more feature space. As a result, the features learned through this method induced by fixed prototypes are more broadly distributed in the space, leading to greater separation between classes.

E. Ablation study

1) The effect of L_{sim}: In order to thoroughly evaluate the effectiveness of the proposed similarity loss function, namely L_{sim}, an extensive experimental analysis is initiated. The role of λ, as the weighing parameter that determines the equilibrium between L_{sim} and the cross-entropy loss function, L_{CE}. We conduct experiments on the CIFAR-100 dataset and use the ResNeSt50 model as our benchmark. This experiment starts with entirely novel model training, with a raw and unbiased assessment, table VII details these experimental results.

First, when λ is set to zero, resulting in the poorest Top-1 accuracy, demonstrates the pivotal role of L_{sim}, its coordination with L_{CE} can facilitate model optimization. The reason is that optimizing solely with L_{CE} may lead to getting stuck in local optima, especially when alpha is extremely low, while the role of L_{sim} can push the model to continuously move forward in a generally accurate optimization direction.

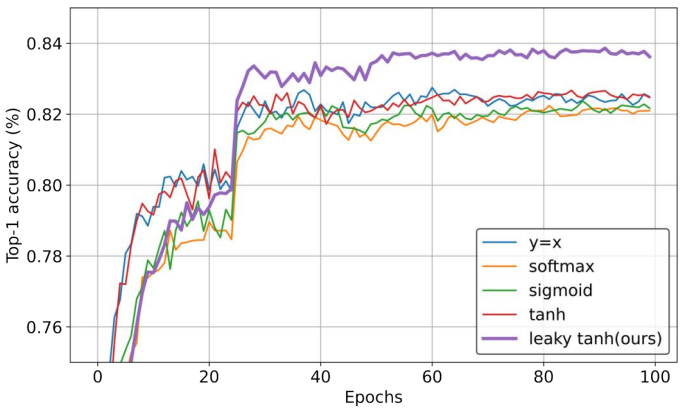


Fig. 10: The training process of activation functions.

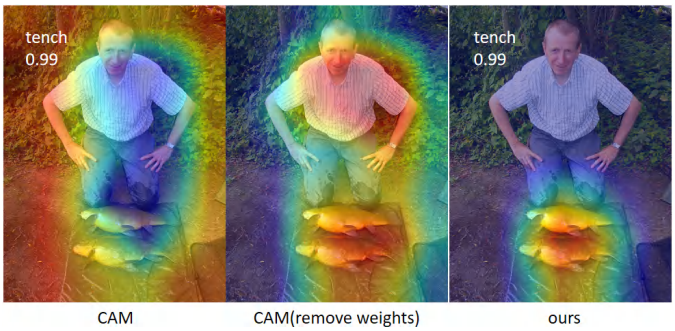


Fig. 11: Performing the CAM using the ResNet152 model(left). The overlay result of the original feature maps(center). The Salient-CAM visualization of the same level ResNet152 model(right).

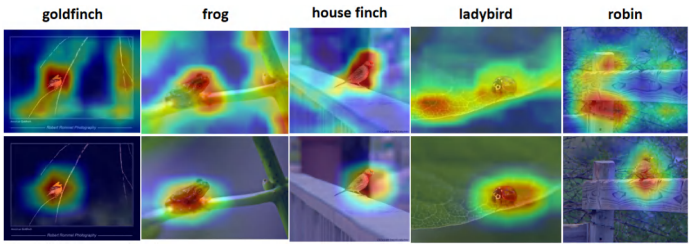


Fig. 12: Small objects scene. The top row shows the results of the Score-CAM and the bottom row shows the results of the Salient-CAM.

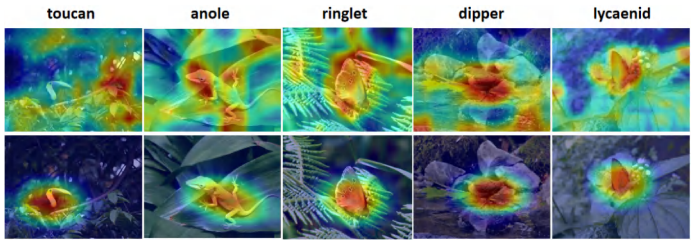


Fig. 13: Interferential background. The top row shows the results of the Score-CAM and the bottom row shows the results of the Salient-CAM.

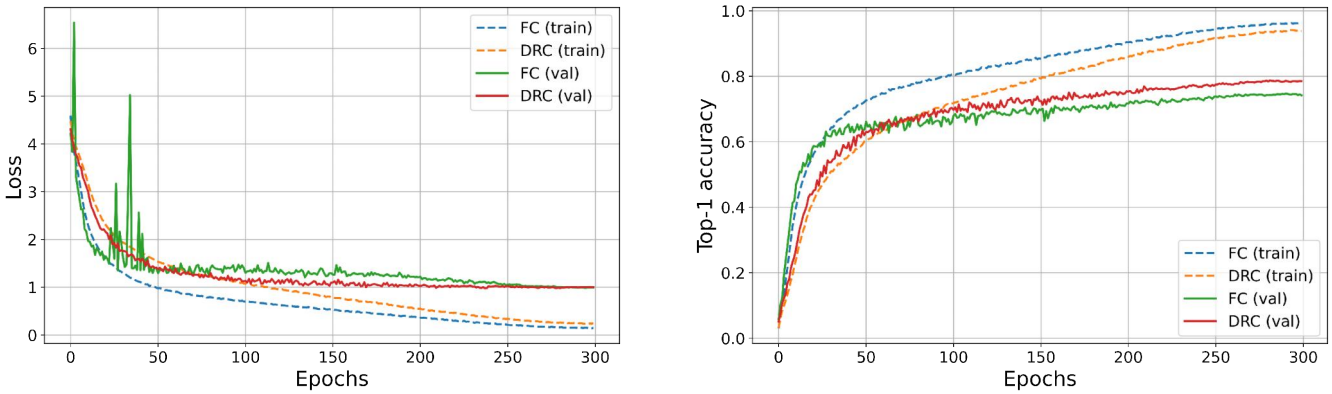
Second, as λ gradually increases, the issue of accuracy decline is gradually alleviated, and even accuracy improves when $\lambda = 40$, indicating the L_{sim} can counteract the accuracy decline caused by discrete transformations. This is due to the fact that Discrete inference essentially calculates the similarity between features and class prototypes, which is consistent with the optimization goal of L_{sim}. Then, while a higher λ level may be advantageous, excessively high λ levels may adversely affect Top-5 accuracy. This is actually caused by the difference between the optimization objective of L_{sim} and the meaning of the Top-5 metric. L_{sim} merely optimizes toward the correct unique label, while Top-5 considers more possible potential labels. Therefore, a fine calibration of L_{sim} is essential in order to optimize the model.

2) Activation function: To assess the necessity of the proposed activation function: leaky tanh, and to show the convergence of the activation function in the training process, we undertake a series of experiments involving a variety of activation functions, including sigmoid, softmax, and tanh. We also consider the scenario of not employing any activation function, represented by the identity function (y = x). We deliberately exclude the ReLU function, due to the absence of negative values in the features. We utilize a pre-trained ResNet50 model, to train on the CIFAR-100 dataset.

First, the highest accuracy achieved by different activation functions is shown in Table VIII. The leaky tanh activation function emerges as the top performer, achieving superior scores across various activation functions. This shows that leaky tanh designed by us is more suitable for the learning process of binary learning than other activation functions. It activates significant features while weakening the influence of unimportant features. Second, we give the Top-1 accuracy curve during the training of the model, as shown in Fig.10. The leaky tanh exhibits a performance deficit compared to both the identity function and standard tanh in the beginning, but as training progresses, leaky tanh dramatically outperforms other activation functions. The initial performance lag suggests an adaptation phase, due to the proportionally low slope of the Leaky Tanh function below the threshold, the convergence speed may be slowed down. Overall, this experiment not only proves that using the leaky tanh activation function is beneficial, but also proves the superiority of the leaky tanh in the Binary learning compared to other activation functions.

F. Salient-CAM

1) Comparison with CAM: Our salient CAM is as simple as the earliest CAM [46] techniques. Unlike other methods, both of them do not need to calculate the weight of the feature map. Therefore, we first conduct a comparison experiment with CAM. We visualize using the ResNet152 model equipped with an FC classifier and the ResNet152 model equipped with a DRC, respectively, both of which are at the same level of accuracy. On the one hand, as illustrated in Fig.11 (left), CAM activates almost the entire label-irrelevant background. This occurs even though the output feature map from the FC model can almost accurately locate the object, as shown in Fig. 11 (center). The underlying reason for this behavior is CAM's dependency on the sign of the weights from the FC layer,



(a) Evolution of Training and Validation Loss with Epochs

(b) Evolution of Training and Validation Top-1 Accuracy with Epochs

Fig. 14: Assessing Generalization Ability: A comparison between DRC and FC.

TABLE IX: The performance of two classifiers under data-scarce conditions, using a ResNet50 backbone.

	FC	DRC	
Dataset	Top-1(%)	Top-1(%)	Δ
CUB200 - 100%	79.47(±0.23)	80.84(±0.15)	1.37
CUB200 - 75%	76.49(±0.17)	78.19(±0.06)	1.7
CUB200 - 50%	70.87(±0.22)	72.65(±0.21)	1.78
CUB200 - 25%	56.63(±0.61)	57.53(±0.52)	0.9
CUB200 - 10%	31.06(±0.7)	30.86(±0.39)	0.2

details are described in Appendix B, where a predominance of negative weights can lead to the phenomenon of highlighting the background areas. On the other hand, as illustrated in Fig.11 (right), the Salient-CAM accurately locates the labeled object. The is because Salient-CAM does not weight and sum all feature maps, but only averages the salient feature maps.

2) Small objects scene: We compare Salient-CAM with Score-CAM using ResNet50 models on ImageNet-1K images containing small objects (Fig. 12). Score-CAM struggles to localize small objects, as it sums all feature maps, including unimportant regions. Salient-CAM, however, precisely pinpoints small objects by focusing on a subset of salient feature maps. This highlights Salient-CAM's advantage in scenarios requiring attention to small objects despite limited spatial context and background interference.

3) Interferential background: We evaluate Salient-CAM's robustness in complex natural scenes compared to Score-CAM using ResNet50 models (Fig. 13). Salient-CAM consistently localizes objects accurately, even in challenging conditions, by prioritizing salient information. This adaptability to background variability suggests its potential for improved object recognition in real-world scenarios. The study demonstrates Salient-CAM's superiority and notable performance differences, affirming its excellence as a CAM method.

G. Generalization capability of DRC

1) During the training process: To assess the generalization capability of the DRC and FC classifier, we undertake a thorough evaluation using the CIFAR-100 dataset and the ResNeSt50 model. We standardize the weight decay parameter at 1e-4 and incorporate cosine scheduler for both classifiers. A noteworthy decision in our approach is to set $\lambda = 0$ for training the DRC classifier, ensuring both FC and DRC solely use the L_{CE} loss function.

First, we analyze the development curves of loss during both training and validation processes, as depicted in Fig. 14(a). The results show that: 1) FC classifier consistently exhibits lower training loss compared to the DRC, this might be due to the stronger fitting ability of the FC classifier to the training data, possibly because of its larger parameter count. However, this lower training loss does not translate into a decrease in validation loss, indicates that the FC classifier might be over-fitting to the training data. 2) the validation loss of the FC classifier displays more pronounced fluctuations, suggesting potential issues with model generalization. This could be the FC classifier has learned specific details or noise from the training data that are not generalizable to the validation set. Second, our analysis also involves comparing the Top-1 accuracy curves for both classifiers, as illustrated in Fig. 14(b). It reveals a compelling trend: while the FC classifier achieves higher accuracy on the training set, the DRC classifier demonstrates superior performance on the validation set. To sum up, the FC classifier, due to its large parameter size and fully connected nature, exhibited excessive over-fitting to the training data, resulting in poorer generalization ability compared to DRC.

2) Under data-scarce conditions: We design and conduct additional experiments to compare the performance of DRC and FC models with varying amounts of training data. Specifically, we use the CUB-200 dataset and implement both models using the ResNet-50 architecture. Our experiments include creating subsets of the training data at different scales (100%, 75%, 50%, 25%, and 10% of the full dataset), as shown in TABLE IX. The results show that DRC outperform the traditional FC layer by 1.37, 1.7, 1.775, 0.9, and 0.2 percentage points in accuracy at the respective data scales. This indicates that as the amount of training data decreases, the advantage of DRC becomes more pronounced. However, under extremely data-scarce conditions, the performance of both DRC and FC models significantly deteriorates, and the gap between the two classifiers narrows.

TABLE X: Experimental results of multi-layer FC.

	Resnet50		ResneSt50	
Classifier	Accuracy(%)	Latency(s)	Accuracy(%)	Latency(s)
one-layer-FC	84.12	1.368	85.79	2.448
two-layer-FC	**84.33**	1.439	**86.09**	2.517
DRC	84.21	**1.015**	85.82	**2.03**

H. Multi-layer FC

Stacking multiple layers is a characteristic of FC, and typically, an FC classifier with one hidden layer, i.e., a two-layer FC, often yields better results. Therefore, we also conduct experiments comparing our method with a two-layer FC. Specifically, we use ResNet50 and ResNeSt50 models on the CIFAR-100 dataset. To ensure non-linear transformations in the two-layer FC, we add a Batch Normalization layer and a ReLU activation function between the two layers. Additionally, we set the dimension of the hidden layer to 512.

From TABLE X, it can be observed that the two-layer FC achieves the highest accuracy performance but also incurs the greatest inference latency. Additionally, the two-layer FC introduces a significantly larger number of parameters. Specifically, the two-layer FC has 1102K parameters, whereas the DRC classifier has zero parameters. Although the two-layer FC achieves slightly better classification accuracy, it incurs significant computational costs compared to the DRC classifier.

V. Conclusions

We presented DRC, a discrete representation classifier with salient features via fixed-prototype. The experiments demonstrate that, despite using a very small number of features, this classifier offers competitive performance and efficiency, along with a lower risk of over-fitting. These qualities make it a compelling alternative to the fully connected layer. We have also demonstrated excellent and robust localizing capabilities with the Salient-CAM technology used with the DRC. The limitation of DRC is that it has several hyperparameters and the best parameter settings may need to be determined empirically or through multiple trials. We plan to address these limitations in future work.

References

[1] Shumin An, Qingmin Liao, Zongqing Lu, and Jing-Hao Xue. Dual correlation network for efficient video semantic segmentation. *IEEE TCSVT*, 2023.

[2] SH Shabbeer Basha, Shiv Ram Dubey, Viswanath Pulabaigari, and Snehasis Mukherjee. Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*, 378:112–119, 2020.

[3] Sébastien Bubeck. Theory of convex optimization for machine learning. *arXiv preprint arXiv:1405.4980*, 15, 2014.

[4] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *WACV*, pages 839–847. IEEE, 2018.

[5] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. *NIPS*, 32, 2019.

[6] Han Chen, Qi Wang, Kailin Xie, Liang Lei, Matthieu Gaetan Lin, Tian Lv, Yongjin Liu, and Jiebo Luo. Sd-fsod: Self-distillation paradigm via distribution calibration for few-shot object detection. *IEEE TCSVT*, 2023.

[7] Xiaohan Ding, Chunlong Xia, Xiangyu Zhang, Xiaojie Chu, Jungong Han, and Guiguang Ding. Repmlp: Re-parameterizing convolutions into fully-connected layers for image recognition. *arXiv preprint arXiv:2105.01883*, 2021.

[8] Xinyu Dong, Qi Wang, Qianding Huang, Qinglong Ge, Kejun Zhao, Xingcai Wu, Xue Wu, Liang Lei, and Gefei Hao. Pddd-pretrain: a series of commonly used pre-trained models support image-based plant disease diagnosis. *Plant Phenomics*, 5:0054, 2023.

[9] Xinyu Dong, Kejun Zhao, Qi Wang, Xingcai Wu, Yuanqin Huang, Xue Wu, Tianhan Zhang, Yawen Dong, Yangyang Gao, Panfeng Chen, et al. Plantpad: a platform for large-scale image phenomics analysis of disease in plant science. *Nucleic Acids Research*, 52(D1):D1556–D1568, 2024.

[10] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022.

[11] Kai Han, Yunhe Wang, Jianyuan Guo, Yehui Tang, and Enhua Wu. Vision gnn: An image is worth graph of nodes. *NIPS*, 35:8291–8303, 2022.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017.

[15] Qianding Huang, Xingcai Wu, Qi Wang, Xinyu Dong, Yongbin Qin, Xue Wu, Yangyang Gao, and Gefei Hao. Knowledge distillation facilitates the lightweight and efficient plant diseases detection model. *Plant phenomics*, 5:0062, 2023.

[16] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *JMLR*, 18(187):1–30, 2018.

[17] Bingliang Jiao, Lu Yang, Liying Gao, Peng Wang, Shizhou Zhang, and Yanning Zhang. Vehicle re-identification in aerial images and videos: Dataset and approach. *IEEE TCSVT*, 2023.

[18] Davood Karimi, Haoran Dou, Simon K Warfield, and Ali Gholipour. Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis. *Medical image analysis*, 65:101759, 2020.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 25, 2012.

[20] Xia Lei, Yongkai Fan, Kuan-Ching Li, Arcangelo Castiglione, and Qian Hu. High-precision linearized interpretation for fully connected neural network. *Applied Soft Computing*, 109:107572, 2021.

[21] Gen Li, Varun Jampani, Laura Sevilla-Lara, Deqing Sun, Jonghyun Kim, and Joongkyu Kim. Adaptive prototype learning and allocation for few-shot segmentation. In *CVPR*, pages 8334–8343, 2021.

[22] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[23] Yutian Lin, Xiaoyang Guo, Zheng Wang, and Bo Du. Privacy-protected person re-identification via virtual samples. *IEEE TIFS*, 2023.

[24] Tianyang Liu, Yutian Lin, and Bo Du. Unsupervised person re-identification with stochastic training strategy. *IEEE TIP*, 31:4240–4250, 2022.

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pages 10012–10022, 2021.

[26] Zhengqi Liu, Yutian Lin, Tianyang Liu, and Bo Du. Reliable cross-camera learning in random camera person re-identification. *IEEE TCSVT*, 2023.

[27] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE TPAMI*, 35(11):2624–2637, 2013.

[28] Curtis G Northcutt, Anish Athalye, and Jonas Mueller. Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749*, 2021.

[29] 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 *ICCV*, pages 618–626, 2017.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[31] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *NIPS*, 30, 2017.

[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015.

[33] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, pages 6105–6114. PMLR, 2019.

[34] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-cam: Score-weighted visual explanations for convolutional neural networks. In *CVPR workshops*, pages 24–25, 2020.

[35] Qi Wang, Hongyu Deng, Xue Wu, Zhenguo Yang, Yun Liu, Yazhou Wang, and Gefei Hao. Lcm-captioner: A lightweight text-based image captioning method with collaborative mechanism between vision and text. *Neural Networks*, 162:318–329, 2023.

[36] Qi Wang, JianJun Wang, Hongyu Deng, Xue Wu, Yazhou Wang, and Gefei Hao. Aa-trans: Core attention aggregating transformer with information entropy selector for fine-grained visual classification. *Pattern Recognition*, 140:109547, 2023.

[37] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, pages 568–578, 2021.

[38] Ye Wang, Yaxiong Wang, Guoshuai Zhao, and Xueming Qian. Improved continually evolved classifiers for few-shot class-incremental learning. *IEEE TCSVT*, 2023.

[39] Zhenyu Wang, Xuemei Xie, Qinghang Zhao, and Guangming Shi. Filter clustering for compressing cnn model with better feature diversity. *IEEE TCSVT*, 2022.

[40] Chenyue Wu and Esteban G Tabak. Prototypal analysis and prototypal regression. *arXiv preprint arXiv:1701.08916*, 2017.

[41] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *ICCV*, pages 22–31, 2021.

[42] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017.

[43] Liu Yang, Shiqiao Gu, Chenyang Shen, Xile Zhao, and Qinghua Hu. Skeleton neural networks via low-rank guided filter pruning. *IEEE TCSVT*, 2023.

[44] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *CVPR*, pages 10819–10829, 2022.

[45] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. In *CVPR*, pages 2736–2746, 2022.

[46] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, pages 2921–2929, 2016.

Qinglei Li is currently pursuing the master's degree with the College of Computer Science and Technology, Guizhou University, Guiyang, China. His research interests include artificial intelligence, computer vision, image classification, image processing, and feature representation.

Qi Wang (Member, IEEE) is currently a Special Term Professor in State Key Laboratory of Public Big Data, Guizhou University. He received Ph.D. degree in Computer Application Engineering at School of Computer Science and Technology, Guangdong University of Technology, Guangdong, China, in 2020. He also received a Ph.D. degree in Engineering Technology at Faculty of Engineering and Technology, Hasselt University, Hasselt, Belgium, in 2021. His current research interests include computer vision, intelligent agriculture.

Yongbin Qin is currently a Professor with the College of Computer Science and Technology, Guizhou University, Guiyang. He serves as the director of the State Key Laboratory of Public Big Data, Guizhou University, China. His research interests include big data processing, cloud computing, text mining, text computing and cognitive intelligence, text analysis, and legal intelligence.

Xinyu Dong is currently pursuing his master's degree at the State Key Laboratory of Public Big Data, Guizhou University, Guizhou, China. His research interests include artificial intelligence, computer vision, object detection and smart agriculture.

Xingcai Wu is a PhD student at Guizhou University. He received his M.E. degree in Computer Technology from Guangdong University of Technology, China, in 2022, and the B.E. degree in Computer Science and Technology from Dongguan University of Technology, China, in 2017. His current research interest is multimodal machine learning.

Shiming Chen (Member, IEEE) received the Ph.D. degree in the School of Electronic Information and Communications, Huazhong University of Sciences and Technology (HUST), China. His research results have been published in prestigious journals and presented at prominent conferences, such as IEEE T-EVC, T-NNLS, NeruIPS, CVPR, ICCV, AAAI, IJCAI, etc. His current research interests span computer vision and machine learning with a series of topics, such as generative modeling and learning, and zero-shot learning.

Wu Liu (Member, IEEE) received the Ph.D degrees at Multimedia Computing Group (MCG), Institute of Computing Technology (ICT), Chinese Academy of Science (CAS), China. He is now a Special Professor in School of Information Science and Technology, University of Science and Technology of China. His current research interests include human behavior analysis, vehicle search/re-identification, large multimedia model, video generation, multi-agent system, multimedia analysis and search.

Yongjin Liu (Member, IEEE) received the B.Eng. degree from Tianjin University, Tianjin, China, in 1998, and the M.Phil. and Ph.D. degrees from The Hong Kong University of Science and Technology, Hong Kong, China, in 2000 and 2004, respectively. He is currently a Professor with the BNRist, Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include computational geometry, computer vision, cognitive computation, and pattern analysis.

Jiebo Luo (Fellow, IEEE) received the B.S. and M.S. degrees in electrical engineering from the University of Science and Technology of China, Hefei, respectively, and the Ph.D. degree in electrical engineering from the University of Rochester, Rochester, NY. Currently, he is on the editorial boards of the IEEE Transactions on Multimedia, Pattern Recognition, and the Journal of Electronic Imaging, as well as a Co-Chair of the 2007 SPIE International Symposium on Visual Communication and Image Processing. His research interests include image processing, pattern recognition, computer vision, medical imaging, and multimedia communication.