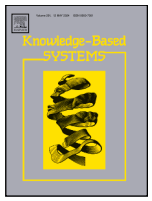
Knowledge-Based Systems 327 (2025) 114108

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



Improving open set recognition with dissimilarity-based metric learning

Lucas O. Teixeira [a,*], Diego Bertolini [b], Luiz S. Oliveira [c], George D.C. Cavalcanti [d], Yandre M.G. Costa [a]

[a] Universidade Estadual de Maringá (UEM), Maringá, PR, Brazil
[b] Universidade Tecnológica Federal do Paraná (UTFPR), Campo Mourão, PR, Brazil
[c] Universidade Federal do Paraná (UFPR), Curitiba, PR, Brazil
[d] Universidade Federal de Pernambuco (UFPE), Recife, PE, Brazil

ARTICLE INFO

Keywords:
Open set recognition
Dissimilarity
Metric learning
Representation learning

ABSTRACT

Open set recognition addresses the problem of classifying instances where the model must not only recognize and classify examples from known classes, but also handle unknown classes not present in the training set. Unlike traditional classifiers, which assume only samples from known classes appear during testing, OSR must detect and manage instances beyond the scope of the training classes. In this paper, we propose a novel approach that combines dissimilarity-based representation with task-specific metric learning in an end-to-end framework. Dissimilarity representation is an alternative to the traditional feature space representation that represents samples based on their differences. By adaptively learning a dissimilarity function specific to the task, our method improves the ability to distinguish between known and unknown classes. We evaluate the proposed method using two popular representation learning techniques, triplet loss and contrastive loss, across multiple experiments: standard OSR benchmarks (CIFAR-10 and SVHN), class-scaling scenarios (DTD and FMD), and the Semantic Shift Benchmark; our proposal consistently outperforms baseline models in both closed-set accuracy and open-set detection.

1. Introduction

Open-set Recognition (OSR) addresses a significant challenge in machine learning: the ability to correctly classify instances from known categories while also identifying instances from unknown classes that were not present during training [1]. Conventional closed-set models assume the label set is complete, which is unrealistic in dynamic, real-world environments. OSR introduces the concept of open-space risk, which is the risk of assigning a confident but incorrect known label to a sample that lies far from every known class. The objective is twofold: maximize accuracy on the known classes while minimizing false positives in this unbounded open region.

Early OSR work attacked the problem from several angles: OpenMax replaced the softmax layer with extreme-value-theory calibration [2]; class-conditional generative models and adversarial generation sought to cover the complement region explicitly [3,4]; flow-based hybrids tried to pair likelihoods with distance cues [5]. Despite this diversity, careful ablations have shown that a vanilla convolutional network, trained with strong augmentation, a cosine learning-rate schedule, and scored by simple maximum logits (MLS), can match or surpass many of these sophisticated proposals [6,7]. The fact that such a tuned baseline closes most of the gap underscores that OSR remains unresolved and motivates the development of fresh strategies that improve rejection accuracy.

In this paper, we propose a novel approach for OSR based on dissimilarity [8,9]. The core idea is straightforward: a dissimilarity score measures the degree of difference between two samples, and for each test image, we first check whether it resembles any of the stored images. If all scores are high, the image differs from every known class, and we mark it as unknown. If at least one score is low, the image is close to a known sample, and we assign the matching class. Our key innovation, and the point that distinguishes this work from earlier metric-learning systems, is that the dissimilarity function itself is learned end-to-end with explicit open-set constraints. Standard deep metric learning optimizes an embedding, then relies on a post-hoc threshold or softmax head; we instead keep the raw dissimilarities as the test-time statistic and shape it during training so that a single, tunable threshold separates known and unknown regions.

Dissimilarity can be used in two primary ways for classification [10]: dissimilarity space represents each sample by its differences from a set of

* Corresponding author.
E-mail addresses: pg54804@uem.br (L.O. Teixeira), diegobertolini@utfpr.edu.br (D. Bertolini), luiz.oliveira@ufpr.br (L.S. Oliveira), gdcc@cin.ufpe.br (G.D.C. Cavalcanti), yandre@din.uem.br (Y.M.G. Costa).

https://doi.org/10.1016/j.knosys.2025.114108
Received 6 February 2025; Received in revised form 30 June 2025; Accepted 13 July 2025
Available online 17 July 2025

prototypes, and the dissimilarity vector (dichotomization) creates positive and negative feature vectors, depending on whether they are from the same class or not, by directly computing pairwise differences between samples and prototypes. In this way, the problem is transformed into a binary classification task, which can be a good strategy for problems with a large number of classes.

Dissimilarity-based models offer several properties that map naturally onto the open-set problem. Because every test sample is represented by its distances to a fixed set of prototypes, a single threshold on the nearest-prototype score yields an immediate decision, giving control over open-space risk. The same machinery is data-type agnostic; as long as a meaningful dissimilarity can be computed, it can accommodate graphs, strings, or multimodal descriptors without relying on a feature extractor. In addition, adding a new class is trivial; one merely appends a few of its samples to the prototype bank and, if desired, trains a new detector, making it well-suited to open-world scenarios where the label set evolves [1,11].

These advantages come with corresponding challenges. The quality of the prototype set is decisive; if prototypes do not adequately cover the class manifold, an unknown sample might, by chance, fall closer to one of the sparsely placed prototypes than an actual in-class point does, resulting in a false acceptance of the unknown sample as a known class. Moreover, the dimensionality equals the number of prototypes, an overly large bank inflates estimation variance and enlarges the open space.

Traditionally, dissimilarity-based classifiers have relied on fixed, hand-chosen distance metrics, such as Euclidean distance or cosine similarity, to measure the distance between two samples. While simple to implement, these static measures cannot always capture complex, nonlinear relationships in real data, which can limit their discriminatory power. To overcome this specific shortcoming, we propose a task-specific metric learning strategy that adaptively learns a dissimilarity function tailored to the task at hand. We integrate representation learning into an end-to-end joint training framework, combining key components to enhance feature representation. This approach aims to create a feature space where dissimilarities between samples accurately reflect their class relationships.

Building upon our previous work, where we explored triplet and contrastive dissimilarity in a multiclass scenario [12,13], we now extend these techniques to open-set recognition. Triplet learning [14] involves minimizing the distance between an anchor and a positive sample while maximizing the distance to a negative sample. Contrastive learning [15,16] focuses on pulling similar pairs together and pushing dissimilar pairs apart in the embedding space.

In our experiments, we conducted a comprehensive evaluation of our dissimilarity-based approach across three distinct scenarios: i) standard OSR benchmark datasets, ii) class scaling, where the number of unseen classes varies, and iii) the Semantic Shift Benchmark (SSB) [6]. First, using the standard benchmark datasets, CIFAR-10 and SVHN, we achieved superior performance in both closed-set accuracy and open-set recognition compared to baseline models. Second, in the class scaling scenario with the DTD and FMD datasets, we achieved similar results, consistently outperforming the baseline across different proportions of unseen classes; despite not being standard datasets in OSR, they are general, challenging, and have been evaluated in many other papers. Finally, on the SSB, our approach outperformed the baseline on the Aircraft dataset, achieved better results in the most challenging settings of the Cars dataset, and was less effective on the CUB dataset.

The primary contributions of this paper are as follows:

1. We propose a novel dissimilarity-based representation for open-set recognition, incorporating task-specific metric learning to learn dissimilarity functions that are specifically adapted to the problem domain.
2. We develop an end-to-end training framework that integrates representation learning and metric learning, enabling the model to optimize both feature embeddings and dissimilarity values jointly for improved performance.
3. We conduct comprehensive experiments on standard OSR benchmarks, class scaling scenarios, and the Semantic Shift Benchmark to validate the effectiveness of our proposed approach.

The remainder of this paper is organized as follows. Section 2 brings the literature review. Section 3 details our proposed method, including the representation and metric learning components, dissimilarity representation, and open-set recognition strategy. Section 4 describes the experimental setup, including datasets, benchmarks, and evaluation metrics. Section 5 presents the experimental results and analysis. Finally, Section 6 concludes the paper and outlines potential directions for future research.

2. Literature review

This section presents a concise review of representation and metric learning, dissimilarity-based methods, and open set recognition, which lays the foundation for this work.

2.1. Representation and metric learning

Representation learning and metric learning are closely intertwined in modern machine learning, particularly with the advent of deep neural networks that can extract meaningful features from raw data [17]. While representation learning focuses on capturing the underlying structure of data to facilitate tasks such as classification and clustering, metric learning enhances this by constructing task-specific distance functions that better reflect the relationships within the data [18].

Representation learning and metric learning complement each other in modern deep learning. The first turns raw data into informative features; the second shapes a task-specific distance so that similar examples lie close together while dissimilar ones stay apart [17,18].

Before the deep learning era, traditional distance metrics like Euclidean or cosine similarity were commonly used; however, they may not adequately capture complex, non-linear relationships in high-dimensional spaces. Existing methods on this topic can be broadly classified into discriminative and generative models. Discriminative methods, such as triplet loss [14] and contrastive loss [15], structures the embedding space in a way that similar samples are closer together while dissimilar ones are pushed apart. The idea is to estimate features to better reflect the inherent similarities and differences among data points, which is crucial for tasks like face recognition, person re-identification, and image retrieval.

Triplet learning is a prevalent representation learning technique that assesses three instances at a time: an anchor, a positive sample (same class as the anchor), and a negative sample (from a different class than the anchor) [14]. The core idea is to reduce the distance between the feature representation of the anchor-positive pair while simultaneously increasing the distance for the anchor-negative pair by at least a margin, typically referred to as alpha. The margin acts as a penalization mechanism where the negative sample is closer to the anchor than the positive sample, enforcing a separation that ensures the negative sample is at least the margin distance further away from the anchor.

Contrastive learning is another popular technique for representation learning that has gained significant attention in recent years. Similar to triplet learning, its goal is to minimize the distance between similar pairs while maximizing the distance between dissimilar pairs. However, contrastive learning focuses specifically on pairs of positive and negative examples [15,19].

In recent years, contrastive learning has been particularly successful in self-supervised learning contexts [16], especially with the introduction of the SimCLR framework [20]. In these scenarios, there is no need to explicitly select negative pairs, as label availability is scarce or even non-existent. To address this, novel loss functions, such as the Normalized Temperature-scaled Cross Entropy Loss (NT-Xent loss) [21], have

been adapted from the original contrastive loss and have proven highly effective.

Generative methods create synthetic samples that are used to estimate the decision boundary between known and unknown classes [22]. GAN-based methods have shown promising results, however they often have complex architectures and demand high computational resources.

2.2. Dissimilarity

Dissimilarity-based representations focus on the differences between samples rather than their absolute features, offering an alternative perspective in feature representation [9]. This approach is particularly effective when structural differences are highly discriminative, as seen in tasks like texture classification and handwriting recognition.

Two main techniques are commonly used in dissimilarity representation: dissimilarity space and dissimilarity vector [10]. In the dissimilarity space method, each sample is represented by its dissimilarities to a set of prototypes, embedding the data into a new feature space where traditional classifiers can be applied. The dissimilarity vector approach constructs feature vectors by directly computing pairwise differences between features taken from samples and prototypes, transforming the problem into a binary classification task based on whether pairs belong to the same class.

Applications of dissimilarity representations span various domains, including handwritten digit recognition [8], content-based image retrieval [23], bird species identification [24], human pose estimation [25], and many more. The integration of deep learning has further advanced these methods, Nanni et al. [26] combined Siamese networks with metric learning to construct a dissimilarity space, achieving state-of-the-art results across multiple image datasets. In a subsequent study, Nanni et al. [27] explored the use of triplet loss and different techniques for generating the dissimilarity space, demonstrating improved performance over previous approaches. Building upon these advancements, our previous works investigated triplet and contrastive dissimilarity in multiclass scenarios [12,13].

2.3. Open set recognition

OSR addresses the challenge of accurately classifying samples from known classes while also identifying instances that belong to unknown classes. [1]. In many real-world applications, models are likely to encounter data that do not fit into any of the trained classes; traditional closed-set classifiers, which assume that all possible classes are known in advance, misclassify these samples into one of the known classes. Under the open-world assumption, a common categorization of classes is [28]:

1. Known known classes (KKC): classes with clearly labeled samples, available features, and, possibly, metadata.
2. Known unknown classes (KUC): samples labeled as negative, not necessarily organized into useful classes.
3. Unknown known classes (UKC): classes with no available samples for training but available metadata.
4. Unknown unknown classes (UUC): classes with no available samples and metadata during training.

Most research in OSR focuses on KKCs and UUCs, with some exceptions also incorporating KUCs to train detectors for negative samples. UKCs, however, are rarely addressed due to their limited practical occurrence and the often marginal utility of metadata distinguishing them from UUCs. Following the literature, in this paper, we also restrict our scope to classifying KKCs and rejecting UUCs.

Here, current methods are can also be divided into discriminative and generative models. Discriminative methods adapt existing classification approaches to account for unknown classes. Bendale and Boult [2] introduced OpenMax, which extends the SoftMax layer to estimate the likelihood of a sample belonging to an unknown class by analyzing the activation patterns of known classes. Dhamija et al. [29] developed two loss functions to improve handling of unknown classes: the Entropic Open-Set loss, which reduces overconfidence by encouraging uniform probability distributions for unknown samples, and the Objectosphere loss, which restructures the feature space to push known class features away from the origin while pulling unknown class features toward it.

Generative approaches estimate the decision boundary between known and unknown samples, Neal et al. [3] used generative adversarial network (GAN) to create counterfactual images representing unknown classes, named Counterfactual Image Generation (OSRCI). Chen et al. [4] expanded the previous approach by using Adversarial Reciprocal Points Learning, which improves open-set detection by learning reciprocal points that capture the data distribution more effectively. Zhang et al. [5] combined flow-based models with discriminative classifiers to detect deviations from known data patterns.

However, these methods often involve complex architectures and substantial computational resources, which can limit their practicality in certain applications. Recently, the need for such complex methods has been questioned: Vaze et al. [6] conducted a critical evaluation of existing OSR techniques and found that a well-trained baseline model, with appropriate data augmentation, learning rate schedules, and the use of maximum logit scores, can achieve performance comparable to or even surpassing that of more complex methods. This finding emphasizes the importance of strong baselines and suggests that the OSR challenge remains open, highlighting the need for further research to develop methods that consistently outperform these baselines across diverse settings.

Evaluating OSR methods requires metrics that capture both the ability to classify known classes accurately and the detection of unknown classes. The Area Under the Receiver Operating Characteristic curve (AUROC) is commonly used to measure the capacity to distinguish between known and unknown classes without relying on a specific decision threshold [1]; however, it does not account for closed-set classification performance. To fill this gap, Dhamija et al. [29] introduced the Open Set Classification Rate (OSCR) by combining the true positive rate for known classes with the false positive rate for unknown classes across varying thresholds. Recently, Wang et al. [30] proposed OpenAUC, a metric that integrates both closed-set accuracy and open-set detection into a single measure, providing a more comprehensive evaluation of OSR methods.

In response to deployment constraints, several studies have proposed solutions to reduce computational footprint. He et al. [31] present Decoupled OSOD (DOSOD), a YOLO-based detector that replaces heavyweight cross-modal attention with a single MLP adaptor, enabling real-time open-set object detection on embedded computing boards. Feng et al. [32] demonstrated that even classical Random Forests can be enhanced for OSR by coupling learned distance metrics with extreme-value modeling, providing an interpretable and lightweight baseline. Bahavan et al. [33] propose SphOR, which models the feature space as a mixture of von Mises-Fisher distributions; this spherical representation is both memory-efficient and achieves competitive state-of-the-art performance.

3. Proposed method

We propose a novel approach for open-set recognition based on dissimilarity and metric learning. Dissimilarity is defined as the differences between samples and can be very useful as an alternative to the regular feature space, especially when the structural differences are highly discriminative [34]. The concept can be applied at multiple levels; for instance, one could calculate the raw difference between the pixels of two images, use a texture descriptor (like Local Binary Pattern, LBP) to extract features, and compute the difference between the two feature vectors and use it to train a machine-learning model [24,35].

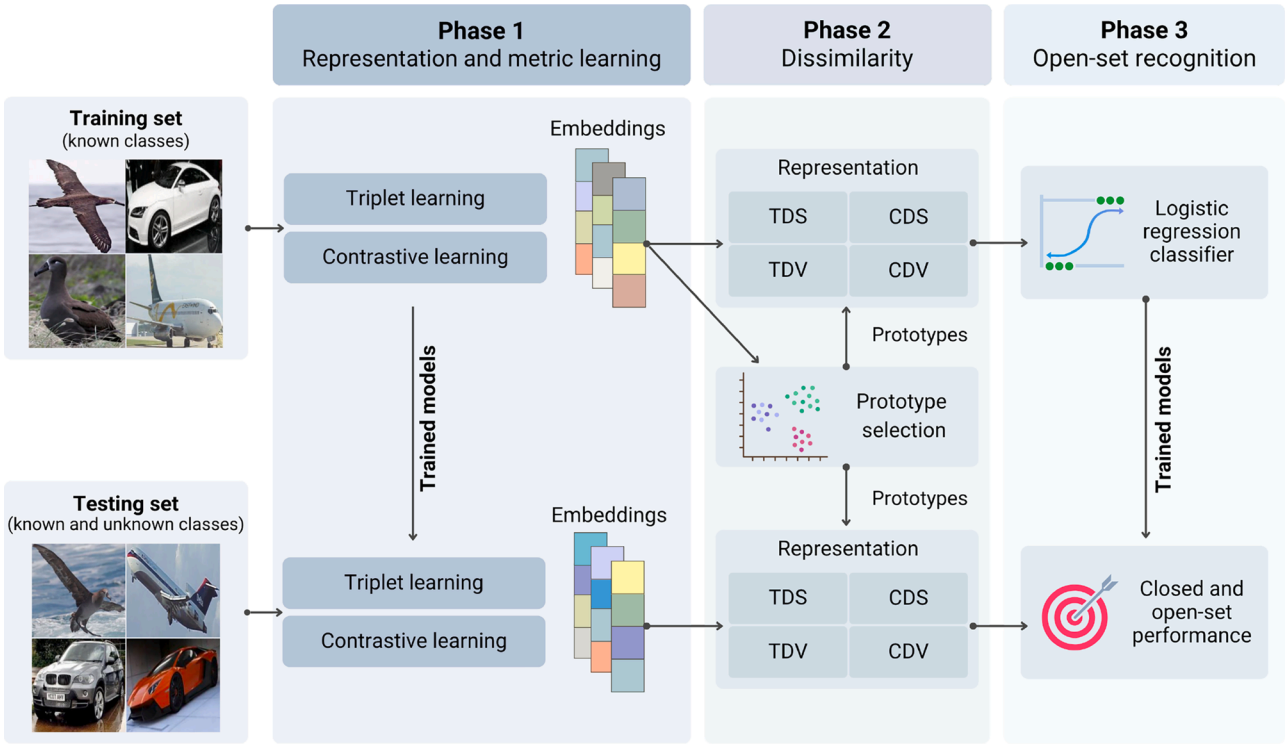


Fig. 1. Proposed method overview.

While that approach may suffice for simpler problems, tackling more complex imaging challenges may demand a more advanced solution. Deep learning offers such a solution by facilitating the extraction and learning of more complex features directly from raw data, enabling the mapping of raw input images to a feature space where semantically similar instances are clustered together while distinct instances are separated, named representation learning [17].

We can further extend this approach by combining it with a metric learning strategy that uses the derived representations to estimate a task-specific dissimilarity function in an end-to-end manner, training all components jointly. We have previously explored and evaluated this idea in our earlier works in a multiclass scenario [12,13]; now, we have expanded this approach into open-set recognition.

The rationale for our proposal is based on the fact that the dissimilarity approach classifies samples by focusing on their differences, and we hypothesize that this characteristic makes it particularly well-suited for open set recognition, improving the distinction between known and unknown classes. Further, by adopting representation and metric learning to improve the representation and dissimilarity estimation, we expect a much better model that can generalize better, even when facing the unknown.

The proposed method comprises three main phases: i) representation and metric learning, ii) dissimilarity representation, and iii) open set recognition. Fig. 1 provides a visual overview of our proposal. Phase 1 trains two separate deep learning models using triplet and contrastive learning to extract useful feature representations (embeddings) while simultaneously estimating a dissimilarity function. Phase 2 maps the embeddings into a dissimilarity space or vector representation based on a selected set of prototypes, forming multiple intermediate representations that can be used for classification. Each deep model and dissimilarity mapping combination forms a representation, resulting in four variants: Triplet Dissimilarity Space (TDS), Triplet Dissimilarity Vector (TDV), Contrastive Dissimilarity Space (CDS), and Contrastive Dissimilarity Vector (CDV). Phase 3 trains a standard classifier using the dissimilarity representation and evaluates both closed- and open-set performance.

3.1. Representation and metric learning (phase 1)

In this work, we employ two well-known metric learning methods, namely triplet and contrastive learning, in separate experiments. The model features a CNN backbone, followed by a projection head comprising fully connected layers. The projection head takes the element-wise absolute difference between two embeddings and outputs a learned dissimilarity score for the pair. The backbone and projection head are trained jointly, so the embeddings and the dissimilarity function are optimized together. This differs from classic Siamese or proxy-based metric learning, which optimizes an embedding but relies on a fixed norm (e.g., Euclidean or cosine) at test time; here, the distance itself is parameterized and becomes the decision used for open-set rejection.

Fig. 2 shows the training workflow for triplet learning, where dissimilarity scores are computed for anchor-positive and anchor-negative pairs, and Fig. 3 shows the workflow for contrastive learning, where all pairs in the batch are compared to learn positive-negative dissimilarities.

3.2. Dissimilarity representation (phase 2)

Dissimilarity representation offers an alternative way to representing sample features in a machine learning problem by focusing on the differences between samples. There are two main approaches for doing so: dissimilarity space and vector [10]. In the dissimilarity space, each sample is represented by its dissimilarities to a predefined set of prototypes, resulting in a matrix where each dimension corresponds to the dissimilarity to a specific prototype. In the dissimilarity vector, each input is represented by its differences from a set of prototypes; every combination of an input and a prototype generates a new sample, which can be labeled as either positive or negative depending on their respective classes.

The dissimilarity space is the natural choice for closed-set, multiclass problems in which the label set will not grow: by encoding each object as the vector of its distances to a fixed prototype set, it preserves a favorable parameter-to-observation ratio and allowing any off-

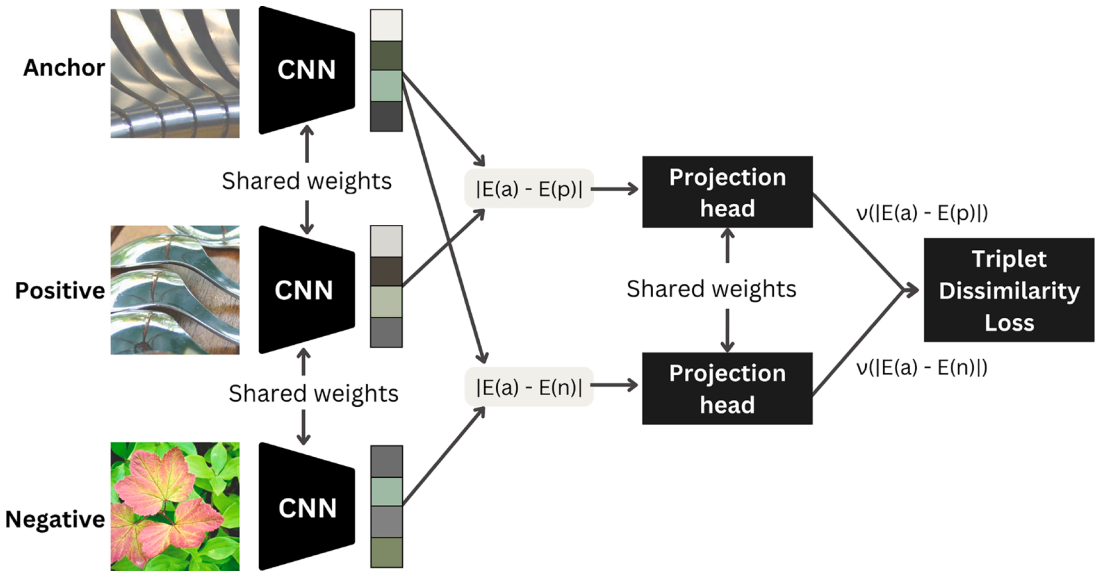


Fig. 2. Triplet dissimilarity training schema.

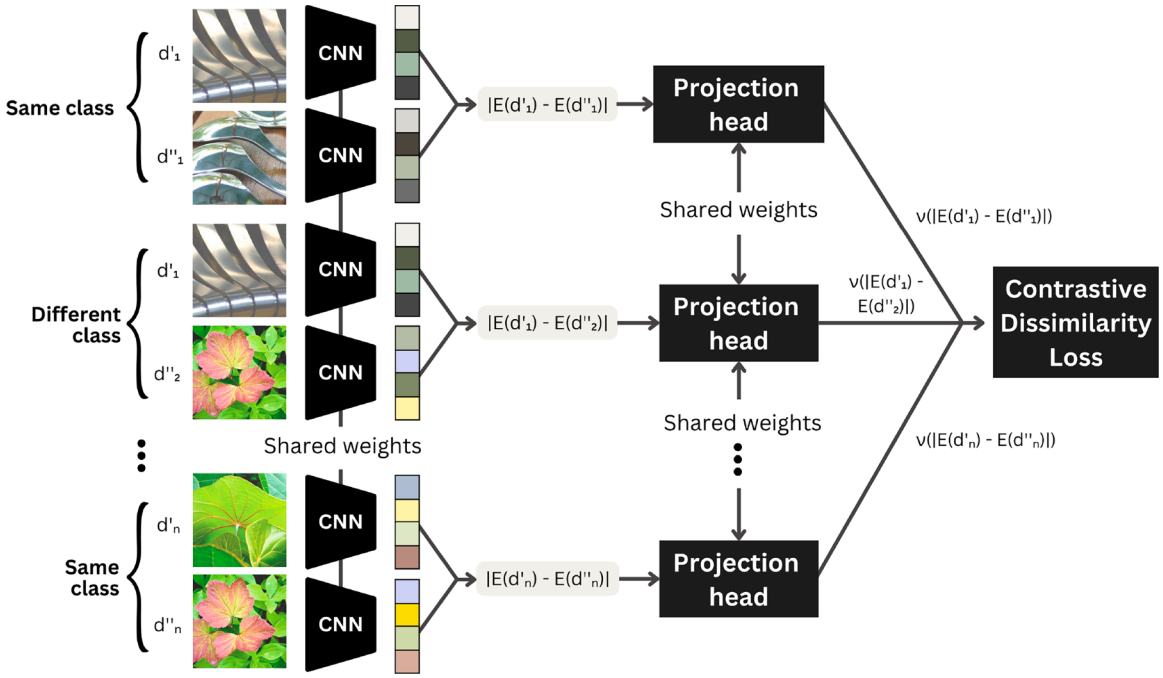


Fig. 3. Contrastive dissimilarity training schema.

the-shelf classifier, provided the prototypes adequately cover the class manifold [34]. The dissimilarity vector instead converts every sample-prototype pair into a same vs. different example, inflating the data set and thus compensating for class scarcity; more importantly, produces a binary model that generalizes to identities unseen during training, a property widely exploited in writer-independent, face- and speaker-verification, and person-re-identification systems [10,36,37]. Conceptually, the dissimilarity space captures global geometry, each coordinate represents how far apart the sample is from a given prototype, whereas the dissimilarity vector learns local margins around each prototype, making it more tolerant of overlapping classes but potentially sensitive to an excess of negative pairs [38]. Our experiments, therefore, concentrate on the dissimilarity vector formulation for its open-set flexibility while also reporting the dissimilarity space results for comparison and completeness.

3.2.1. *Prototype selection*

Prototypes are a compact representation of the training data, aiming to reduce dimensionality and decrease memory and computation requirements. They can either be selected directly from the training set or generated as artificial samples that capture key patterns in the training data.

Consider T as the training set and R as the prototype set, there are two main strategies to populate R: selecting actual training samples or generating artificial samples. The first chooses actual samples from the training set, making the prototypes a subset of the training set ($R \subseteq T$), ensuring that the prototypes are inherently consistent with the training data. The second strategy generates artificial yet representative samples, potentially making the prototype set disjoint from the training set ($R \cap T = \emptyset$), allowing for the creation of more diverse prototypes and is particularly advantageous in unbalanced scenarios, as it enables the

generation of a balanced prototype set, which would otherwise be constrained by the number of samples in the less frequent class.

In this work, we generated the prototypes by applying an unsupervised clustering algorithm to group similar samples into k clusters for each class, deriving k centroids to serve as prototypes. The number of prototypes is the number of clusters per class times the number of classes in the dataset.

3.2.2. Dissimilarity space

Assuming the training set T with n samples and the prototype set R with m samples. The dissimilarity matrix $D(T, R)$ can be defined as:

$$D(T,R) = \begin{bmatrix} v(x_1,p_1) & v(x_1,p_2) & \dots & v(x_1,p_m) \\ v(x_2,p_1) & v(x_2,p_2) & \dots & v(x_2,p_m) \\ \vdots & \vdots & \ddots & \vdots \\ v(x_n,p_1) & v(x_n,p_2) & \dots & v(x_n,p_m) \end{bmatrix}$$

x_i represents the i-th training instance, p_j the j-th prototype, and v the estimated dissimilarity function. The resulting matrix $D(T, R)$ can then be used to train a standard classification model.

The testing sample can be formulated as follows:

$$D'(t_k,R) = \begin{bmatrix} v(t_k,p_1) & v(t_k,p_2) & \dots & v(t_k,p_m) \end{bmatrix}$$

t_k and p_j represent the k-th test instance and the j-th prototype, respectively. The resulting $D'(t_k, R)$ vector has the same number of columns as $D(T, R)$, allowing the use of the previously trained classification model.

3.2.3. Dissimilarity vector

The dissimilarity vector representation generates samples by taking the differences between input features and prototypes, treating each input-prototype pair as an independent example. Rather than first extracting embeddings and then applying a fixed norm, we propose to preserve the end-to-end metric-learning pipeline. The main challenge is that our proposed dissimilarity function is designed to produce a single scalar per input-prototype pair. To improve robustness, we propose generating w small variations of each input, where w denotes the number of stochastic augmentations drawn per sample, allowing the computation of multiple dissimilarity values between the input and a given prototype.

Classical augmentation techniques, such as random crops, flips, color jitter, and others, supply the required variants. This multi-view augmentation scheme not only preserves our end-to-end learning, but also improves robustness by capturing local variation around each point, summarizing both geometric and augmentation-induced variability.

Given x_{ij} representing the i-th class and j-th training sample, p_{ik} the i-th class and k-th prototype, and v the dissimilarity function. Instead of producing a single scalar, we draw w stochastic variants of x_{ij} using random augmentations, then for each, pair with the corresponding prototype and compute their dissimilarity, forming a vector of w dissimilarity values that captures how consistently the sample diverges from prototype p_j under plausible perturbations. The resulting concatenated vector $v'(x_{ij}, p_{ik})$ can be expressed as

$$v'(x_{ij},p_{ik}) = \left[v(x_{ij}^1,p_{ik}) \quad v(x_{ij}^2,p_{ik}) \quad \dots \quad v(x_{ij}^w,p_{ik}) \right]$$

Each concatenated vector $v'(x_{ij}, p_{ik})$ is then assigned a binary label: $\oplus$ if the image and prototype belong to the same class and $\ominus$ if they come from different classes. Considering a task with n classes, m images per class, and m' prototypes per class, the positive set is defined as

$$T_\oplus = v'(x_{ij},p_{ik}) \text{ where } i = 1 \text{ to } n,\ j = 1 \text{ to } m,\ k = 1 \text{ to } m'$$

and the negative set as

$$T_\ominus = v'(x_{ij},p_{kl}) \text{ where } i,k = 1 \text{ to } n,\ i \neq k,\ j = 1 \text{ to } m,\ l = 1 \text{ to } m'$$

A conventional classifier can then be trained on $T = T_\oplus \cup T_\ominus$.

During testing, each input is again augmented w times and paired with every prototype, generating $i\,k$ vectors (where i denotes the number of classes and k the number of prototypes per class). The classifier outputs one probability per vector, representing the likelihood that the test image belongs to the prototype's class. These prototype-level probabilities are consolidated into a single score per class through an aggregation step. Two straightforward options are (i) computing the mean of the probabilities for all prototypes in the class or (ii) selecting the maximum probability among the prototypes of the class. Other pooling rules, such as the geometric mean, the median, or a weighted sum, can also be applied. After aggregation, the sample is assigned to the class with the highest pooled score.

3.3. Open set recognition (phase 3)

Closed-set recognition is a scenario where a machine learning model is trained and tested on a fixed set of classes, with every test sample belonging to one of these known classes. Let $\mathcal{X}$ denote the input space, and $\mathcal{C} = \{y_1, y_2, \dots, y_C\}$ represent the set of C known classes. We can formally define the training set as $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^N \subseteq \mathcal{X} \times \mathcal{C}$ and the test set as $\mathcal{D}_{\text{test-closed}} = \{(x_i, y_i)\}_{i=1}^M \subseteq \mathcal{X} \times \mathcal{C}$. For each test sample $t \in \mathcal{D}_{\text{test-closed}}$, the model outputs a probability distribution $p(y|t)$ over the known classes.

The open-set recognition problem is very similar, but it introduces the possibility of encountering samples during testing that do not belong to any of the known classes. The test set can be defined as $\mathcal{D}_{\text{test-open}} = \{(x_i, y_i)\}_{i=1}^M \subseteq \mathcal{X} \times (\mathcal{C} \cup \mathcal{U})$. In this setting, for each test sample $t \in \mathcal{D}_{\text{test-open}}$, the model returns a probability distribution over the known classes $p(y|t, y \in C)$ and a score $S(y \in C|t)$ that indicates whether the sample belongs to one of the known class, usually referred as open-set score.

As previously mentioned, many complex methods have been developed to tackle open-set recognition and estimate the open-set score, and it has been shown that a well-trained closed-set classifier can perform on par with many of the recent proposals [6]. Thus, we adopted a closed-set classifier as our baseline using a traditional training strategy using cross-entropy loss between a one-hot target vector and the softmax output $p(y|t)$. In this way, the probability distribution over the known classes is straightforward, and the open-set score can be calculated using two approaches: i) maximum softmax probability (MSP): $S(y \in C|t) = max_{y \in C}\ p(y|t)$; ii) maximum logit score (MLS): $S(y \in C|t) = max_{y \in C}\ z(y|t)$, where $z(y|t)$ represents the logit output for a known class given sample t [6]. The logits are the values output by the final layer of a deep classifier before softmax, and as such, they are capable of retaining more feature magnitude information.

In this work, we also propose the use of dissimilarity values as the open-set score. The idea is that the dissimilarity represents the difference between samples and serves as a reliable indicator of whether a sample belongs to a known class. Specifically, we propose the minimum dissimilarity score (MDS): $S(y \in C|t) = \min_{z \in \{1,\dots,m\}} v(t, p_z)$ where $v(t, p_z)$ represents the dissimilarity between the sample t and prototype p_z, and m is the total number of prototypes. This score reflects the smallest dissimilarity value among all prototypes, indicating the closest match, and serves as an excellent proxy for the likelihood of t belonging to the known class C.

Given that we are proposing a metric learning approach to estimate a task-specific dissimilarity function, the effectiveness is highly dependent on the loss function used. The triplet dissimilarity loss functions as expected, generating smaller values for similar samples while never producing negative values. In contrast, our proposed contrastive loss skips the cosine similarity, which makes it generate higher values for similar samples while also allowing negative values, to ensure proper dissimilarity values for the minimum dissimilarity score, we invert the contrastive loss.

4. Experimental setup

In our experiments, following prior research [12,13], we employ a standard deep model as a baseline and compare it with our proposed methods: triplet and contrastive dissimilarity. The model is trained on

a subset of classes, with the remaining classes reserved for evaluation as unseen classes. We evaluate performance under three distinct scenarios: i) standard OSR benchmark datasets, ii) class scaling, where the number of unseen classes varies, and iii) Semantic Shift Benchmark (SSB).

CIFAR-10 and SVHN serve as the standard object- and digit-recognition benchmarks. They test whether the model can draw a clear boundary between known and unknown classes in small natural images. DTD and FMD address texture and material recognition under class scaling. By withholding varying fractions of classes at evaluation, they show how performance changes as the unseen set grows. CUB, Aircraft, and Cars form the Semantic-Shift Benchmark (SSB). These fine-grained datasets contain many classes that differ only in subtle visual details, challenging the model to reject unseen classes whose appearance is close to that of known classes. The SSB further divides the unseen classes into easy and hard groups, providing a controlled range of semantic distance. Together, the three scenarios cover the key challenges in open-set recognition.

All experiments intentionally fix the representational capacity of the backbone to isolate algorithmic effects. We train a VGG-32 from scratch for the standard benchmarks (CIFAR-10, SVHN) and fine-tune ImageNet-1K pre-trained EfficientNet-V2 and ResNet-50 for the class scaling and SSB (DTD, FMD, CUB, Cars, Aircraft). Embedding size, input resolution, optimizer, and training recipes are kept constant across baselines and proposed methods. Under this controlled setting, the MLS baseline computed on the same network is the most informative baseline for our context.

Additionally, we provide details on our proposed method, its components, and the training and evaluation protocols standard across all benchmarks.

4.1. Standard benchmarks

In the first set of benchmarks, we use popular datasets for OSR: CIFAR-10 [39] and SVHN [40]. CIFAR-10 contains a diverse collection of real-world images representing various animals and vehicles, with a total of 60,000 images - 50,000 for training and 10,000 for testing. SVHN contains individual digits extracted from street-view house numbers, with a total of 73,257 images for training and 26,032 images for testing. Both datasets contain ten classes, with images sized at 32×32 pixels.

Training is conducted using six classes, with the remaining four classes serving as unseen classes ($|C| = 6, |U| = 4$), following a 5-fold cross-validation evaluation varying the selection of seen and unseen classes.

4.2. Class scaling benchmark

For the class scaling benchmark, we use two popular texture datasets: Describable Textures Dataset (DTD) [41] and Flickr Material Database (FMD) [42]. The DTD dataset includes 47 classes of textures based on their perceptual properties, containing a diverse range of patterns such as striped, dotted, and chequered, with a total of 5640 images. The FMD dataset is a collection of various real-world materials, including ten classes, such as fabric, glass, metal, wood, and plastic, containing 1000 images. In both cases, the image sizes are variable, and we standardize them to 224×224 pixels prior to training.

In this setup, we vary the proportion of classes reserved as unseen classes from 10 % ($|C| = 90\,\%, |U| = 10\,\%$) to 50 % ($|C| = 50\,\%, |U| = 50\,\%$), for each, we apply a 5-fold cross-validation evaluation varying the selection of seen and unseen classes. In the case of DTD, we gradually increase the number of unseen classes from approximately 5 ($\sim 10\,\%$) to 23 ($\sim 50\,\%$) classes; similarly, for FMD, the number of unseen classes varies from 1 to 5.

4.3. Semantic shift benchmark

The last and most crucial experiment leverages the Semantic Shift Benchmark (SSB) [6], which is designed explicitly for open-set recognition and related tasks, focusing on isolating semantic novelty from other types of distributional shifts. The SSB suite comprises three fine-grained datasets: Caltech: UCSD Birds (CUB) [43], FGVC-Aircraft [44], and Stanford Cars [45]; as well as a large-scale ImageNet benchmark. In all cases, we standardize the training and testing samples to 224×224 pixels prior to training.

The SSB suite provides specific splits for seen and unseen classes for each dataset. Unseen classes are further categorized into three levels of difficulty: easy, medium, and hard. The harder the category, the more visually and semantically similar it is to a known class. We evaluate our methods on both the easy and hard splits across all datasets, merging the medium and hard into a single group. The training/testing split used is the standard for each dataset, and training samples belonging to an unseen class are disregarded; thus, the number of images used in our experiments does not match the total number of available images per dataset.

In our experiments, we concentrated on the three fine-grained datasets: CUB, Aircraft, and Cars. The CUB dataset comprises 11,788 images spread across 200 bird species classes, of which 100 are known classes (with 2997 training images and 2884 test images), and the remaining unknown classes are divided into 32 easy classes (915 test images), 34 medium classes (1,004 test images), and 34 hard classes (991 test images). The Aircraft dataset consists of 10,200 images across 100 classes of aircraft variants, of which 50 classes are known (3,332 training images and 1668 test images), and the unknown classes are further split into 20 easy classes (667 test images), 17 medium classes (565 test images), and 13 hard classes (433 test images). The Cars dataset includes 16,185 images spanning 196 car models categorized by make, model, and year; 98 are known classes (4,020 training images and 3948 test images), with the remaining classified into 76 easy classes (3,170 test images) and 22 hard classes (923 test images).

4.4. Representation and metric learning

The representation component uses a convolutional neural network to map raw images to representative embeddings. For the standard benchmark, we trained a VGG32 network from scratch. For the class scaling and SSB benchmarks, we used EfficientNetV2 with pre-trained ImageNet-1K weights. For the SSB benchmark, we also evaluated a pre-trained ResNet50 to analyze the effect of different network architectures on performance.

Since EfficientNetV2 and ResNet50 were pre-trained on ImageNet-1K, we modified their top layers to suit our tasks. The original classification head is replaced by a three-layer fully connected network whose widths scale with the target embedding dimension e: $[4e, 2e, e]$.

To ensure consistency and facilitate comparison across methods, we standardized the embedding size to 128 across all experiments. This size matches the dimensionality of the final layer in VGG32. Although VGG32 was specifically used only in the standard benchmark, we maintained the 128-dimensional embedding size for the class scaling and SSB benchmarks as well. The reason for this choice is twofold: i) we aimed to keep as many parameters as possible consistent across methods to enable a fair comparison; ii) fine-tuning these parameters for each dataset would require substantial computational resources due to the large number of hyperparameters involved.

The projection head, which estimates the task-specific dissimilarity function, is another three-layer fully connected block sized $[e, e/2, e/4]$. The architecture is intentionally kept simple to balance computational efficiency while still providing sufficient capacity to model non-linear relationships.

4.5. Dissimilarity

The first step in the dissimilarity representation is prototype selection, where prototypes are obtained as the centroids produced by greedy K-means++ clustering [46], whose improved initialization typically yields higher quality clusters than standard K-means. For the standard and class-scaling benchmarks, the number of prototypes was set to 5, whereas for SSB, it was set to 20 due to its greater complexity. The optimal number of prototypes was determined by holding out 20 % of the training data as a validation set and selecting the number that maximized classification accuracy. In one of our previous works, we evaluated multiple clustering algorithms and found minimal differences in performance, leading us to choose K-means++ for its popularity and efficiency [13].

The dissimilarity between samples and prototypes is computed in an end-to-end manner through the projection head. For the dissimilarity space approach, as previously described, it is straightforward to create a matrix representation and apply a standard classifier. In the dissimilarity vector approach, the projection head still outputs a single score for every sample-prototype pair, but instead of stochastic augmentations, we kept the procedure deliberately simple. For each image, we systematically extracted regular patches, consisting of 25 fixed 28×28 crops on the standard benchmarks and 100 fixed 200×200 crops on the class-scaling and SSB suites, without applying any geometric or photometric transformations. To reduce local noise and improve numerical stability, we averaged the scores every five crops, reducing the resulting multi-view dimensionality to 5 for the standard benchmarks and 20 for class-scaling and SSB.

4.6. Open set recognition

As previously mentioned, following the standard method in other OSR studies, we restrict our scope to classifying known known classes (KKCs) and rejecting unknown unknown classes (UUCs).

In the baseline models, we evaluated two strategies for the open-set score: Maximum Softmax Probability (MSP) and Maximum Logit Score (MLS). In our proposed method, we introduce the Minimum Dissimilarity Score (MDS) as the primary open-set recognition strategy. Additionally, since our dissimilarity-based approaches employ a standard classifier, we can also estimate the probability of each class and apply the MSP strategy.

We report three evaluation metrics: closed-set accuracy, the threshold-free Area Under the Receiver Operating Characteristic curve (AUROC) for open-set scenarios, and OpenAUC [30]. The AUROC is standard practise in the OSR literature, however it has an important limitation: it does not take into account the closed-set performance, thus the overall performance needs to be evaluated in a separate manner.

OpenAUC is defined as the area under the Open-Set False Positive Rate (OFPR) and Conditional True Positive Rate (COTPR) curve. OFPR represents the probability that a model misclassifies an open-set sample as belonging to one of the known classes, while COTPR denotes the probability that a model correctly classifies a known-class sample given that it outputs a high confidence score for the correct class. OpenAUC thus integrates both closed-set and open-set performance by evaluating how well the model ranks open-set samples lower than close-set ones.

4.7. Training and evaluation protocol

The training protocol for the baseline models follows the protocol proposed by Vaze et al. [6] and PyTorch training primitives.[1] Training was conducted for 600 training epochs, cosine-annealed learning rate with warm restarts at epochs 200 and 400, a 20-epoch linear warmup, SGD with momentum, batch size of 128 for the standard benchmarks and 32 for the larger fine-grained sets, and five known/unknown class splits.

Key deviations from Vaze et al. [6] are as follows: (i) we omit RandAugment and rely exclusively on the Albumentations transforms listed below; (ii) following the torchvision recipe, we adopt patch training, and for the class-scaling and SSB suites initialize from ImageNet-1K weights instead of the MoCoV2-Places checkpoint; (iii) all images are resized to 224×224 (rather than 448×448) due to hardware constraints. Considering these departures, especially the lower resolution, we were unable to reproduce the exact accuracies reported in the literature and from reaching state-of-the-art performance on some datasets. Thus, our analysis focuses on the relative improvements over a consistently trained baseline, which was trained and evaluated under identical conditions as our proposal.

Table 1 presents the set of hyperparameters used in our experiments for each method. One of our primary concerns was to ensure a fair comparison between our proposed methods and the baseline; to do so, we maintained consistent hyperparameters whenever possible across different methods. For instance, the VGG32 model employs a single fully connected layer with 128 neurons; consequently, we maintained an embedding size of 128 across all benchmarks. The number of iterations was set according to the dataset size, ensuring that larger datasets received a comparable number of iterations to match the 600 epochs used for the baseline model. The hyperparameters were fine-tuned by creating a validation set from 20 % of the training data selected at random. The standard benchmark models were trained from scratch, while the models for the class scaling and SSB benchmarks were initialized with pre-trained weights from ImageNet-1K.

A brief note on optimization difficulties: while SVHN ultimately reached competitive accuracy, achieving stable convergence with the triplet dissimilarity was particularly troublesome. To mitigate this, we first trained for 50 epochs using cross-entropy loss before reverting to triplet loss, a warm-start strategy consistent with reports that triplet loss is hard to optimize from scratch [14,47,48]. By contrast, CIFAR-10 trained smoothly with triplet loss alone.

Table 1
Training and evaluation hyperparameters.

Method	Parameter	Standard	Class scaling	SSB
Common	Batch size	128	32	32
	Patch size	28×28	200×200	200×200
	Optimizer	SGD	SGD	SGD
Proposal	Embedding size	128	128	128
	Temperature τ	0.5	0.5	0.005
	Margin α	1.0	1.0	2.0
	Iterations	250,000	30,000	40,000
	Top layers warmup iters.	-	1000	1000
	Top layers warmup LR	-	0.01	0.01
	Learning rate	0.01	0.001	0.001
Baseline	Epochs	600	600	600
	Top layers warmup epochs	-	20	20
	Top layers warmup LR	-	0.1	0.01
	Learning rate	0.1	0.01	0.001

Table 2
Data augmentation parameters.

Transformation	Parameters
Flip	–
Rotate	Limit = 90
Random brightness	Limit = 20
Random contrast	Limit = 20
Gaussian blur	Limit = (3, 7)

[1] https://pytorch.org/blog/how-to-train-state-of-the-art-models-using\-torchvision-latest-primitives/

We used data augmentation techniques during training using the Albumentations library [49], which are listed in Table 2 along with their parameters. For SVHN, we turn off flips and restrict rotations to ±30° to avoid ambiguities between six and nine. Each transformation was applied with a default probability of 50 %.

To maintain fairness, we avoided using advanced data augmentation techniques, such as RandAugment [50] or ensembling methods, which could introduce additional variability and bias in performance evaluation. Instead, our approach focused on standard, well-established augmentation methods and training strategies to provide a consistent basis for comparing the proposed methods with the baseline models.

The dissimilarity-based approaches produce a feature representation that can be used to train a standard classifier. We chose logistic regression from the scikit-learn library [46] for its simplicity and effectiveness, avoiding more complex classifiers to ensure that the performance gains are attributed to our proposed method rather than the complexity of the downstream classifier.

5. Results and discussion

This section presents an extensive empirical analysis of our proposed dissimilarity-based approaches on standard OSR benchmarks, class scaling scenarios, and the Semantic Shift Benchmark (SSB). We compare our methods: Triplet Dissimilarity Space (TDS), Triplet Dissimilarity Vector (TDV), Contrastive Dissimilarity Space (CDS), and Contrastive Dissimilarity Vector (CDV); against a baseline model using Maximum Softmax Probability (MSP) and Maximum Logit Score (MLS) as open-set scoring strategies. The evaluation metrics include closed-set accuracy (Acc_k), AUROC for open-set detection, and OpenAUC, providing a holistic view of both classification performance and open-set recognition capabilities.

Table 3 summarizes the results on the CIFAR-10 and SVHN datasets. Our proposed methods consistently outperform the baseline in both closed-set accuracy and AUROC for open-set detection; thus, we decided to omit OpenAUC in this scenario due to its redundancy.

On the CIFAR-10 dataset, the contrastive dissimilarity methods (CDS and CDV) achieved the highest closed-set accuracies, with CDV reaching 91.4 %, surpassing the baseline by 2 percentage points. The AUROC values for open-set detection also improved significantly, with CDS attaining 78.3 % compared to 72.2 % for the baseline. For the SVHN dataset, CDV achieved the highest closed-set accuracy of 97.3 %, outperforming the baseline by 3.3 percentage points. The AUROC increased substantially as well, with CDS reaching 94.1 %, an improvement of 4.8 percentage points over the baseline.

Table 4 presents the results for the class scaling benchmark on the DTD and FMD datasets. Our methods consistently outperformed the baseline across different proportions of unseen classes. The contrastive dissimilarity methods (CDS and CDV) particularly demonstrated superior performance in both closed-set accuracy and AUROC.

As we increased the proportion of unseen classes in the class scaling benchmarks, our proposed methods maintained robust performance improvements over the baseline on both datasets. The CDV maintained high closed-set accuracy across the splits, starting from 75 % accuracy at 10 % unseen classes and progressively increasing to 82 % at 50 % unseen classes. In comparison, the baseline accuracy improved from 69.3 % to 80 % over the same range. The AUROC for the contrastive dissimilarity methods consistently surpassed the baseline; at 10 % unseen classes, CDS and CDV achieved an AUROC of 74.7 % and 74.9 % compared to 66.8 %, marking an 8.1 percentage points improvement. At 50 % unseen classes, both CDS and CDV attained an AUROC of approximately 80 %, 11.9 percentage points higher than the baseline.

Similarly, on the FMD dataset, our methods maintained high closed-set accuracy across all proportions of unseen classes. The TDS method achieved accuracies ranging from 86.8 % at 10 % unseen classes to 90.1 % at 50 % unseen classes, consistently outperforming the baseline, which ranged from 71.9 % to 89.4 %. In terms of AUROC, the CDV method showed consistent improvements over the baseline. At 10 %

Table 3
Standard benchmark results.

Method		CIFAR-10		SVHN	
		Acc_k	AUROC	Acc_k	AUROC
Baseline	MSP	89.4 (2.5)	72.2 (3.6)	94.0 (0.7)	88.1 (1.6)
	MLS		71.2 (3.6)		89.3 (1.6)
TDS	MSP	89.4 (2.2)	73.8 (2.4)	96.7 (0.6)	91.7 (1.3)
	MDS		69.6 (2.6)		82.5 (7.3)
TDV	MSP	88.9 (2.5)	75.7 (3.8)	96.1 (0.9)	88.8 (2.8)
	MDS		65.1 (3.4)		80.1 (7.7)
CDS	MSP	91.3 (1.9)	**78.3** (2.4)	97.1 (0.3)	**94.1** (0.8)
	MDS		76.4 (2.6)		93.2 (0.8)
CDV	MSP	**91.4** (1.8)	77.4 (2.8)	**97.3** (0.3)	93.3 (0.9)
	MDS		76.7 (2.7)		93.0 (0.9)

Table 4
Class scaling benchmark results.

Split	Method		DTD		FMD	
			Acc_k	AUROC	Acc_k	AUROC
10 %	Baseline	MSP	69.3 (1.6)	66.8 (3.6)	71.9 (2.7)	60.2 (3.9)
		MLS		66.5 (3.8)		60.8 (3.4)
	TDS	MSP	69.6 (0.8)	64.5 (2.3)	**86.8** (0.7)	70.6 (9.9)
		MDS		73.6 (3.7)		73.1 (8.9)
	TDV	MSP	69.0 (1.4)	70.9 (5)	86.6 (1.1)	71.3 (4.9)
		MDS		73.7 (3.7)		71.9 (8.5)
	CDS	MSP	75.0 (2)	73.5 (4.2)	86.0 (1.7)	72.7 (9.6)
		MDS		74.7 (4.2)		73.8 (11)
	CDV	MSP	**75.0** (1.7)	73.7 (4.5)	86.3 (0.9)	72.3 (14.2)
		MDS		**74.9** (4)		**73.9** (11.8)
20 %	Baseline	MSP	73.6 (0.9)	67.5 (2.4)	86.0 (2.4)	73.9 (2)
		MLS		67.3 (2.4)		73.7 (2.4)
	TDS	MSP	71.8 (1.6)	66.8 (0.9)	88.6 (2.2)	79.7 (1.1)
		MDS		74.7 (3.5)		81.1 (1.3)
	TDV	MSP	71.2 (1.9)	73.6 (3.6)	**89.1** (1.9)	78.5 (2.4)
		MDS		74.8 (3.4)		**81.4** (1.7)
	CDS	MSP	76.8 (1.4)	74.9 (3.9)	88.0 (1.6)	79.3 (3)
		MDS		76.1 (4)		80.1 (2.7)
	CDV	MSP	**77.0** (1.5)	75.0 (4.5)	88.3 (2)	80.1 (2.3)
		MDS		**76.3** (4)		80.5 (2.5)
30 %	Baseline	MSP	73.8 (1.8)	68.5 (2.7)	86.9 (1.9)	75 (3.5)
		MLS		68.2 (2.8)		75.2 (3.3)
	TDS	MSP	73.0 (2.2)	68 (1.1)	89.3 (1.8)	79.3 (2.6)
		MDS		75.7 (1)		79.4 (4.5)
	TDV	MSP	72.8 (2.4)	73.3 (1.7)	**90.2** (1.2)	78.9 (2.9)
		MDS		75.7 (0.9)		**79.7** (4.7)
	CDS	MSP	77.2 (1.6)	75.3 (2.3)	89.7 (3.4)	78 (2.8)
		MDS		77.6 (2.3)		79.1 (2.9)
	CDV	MSP	**77.3** (1.7)	75.2 (2.7)	89.4 (3)	79.0 (2.2)
		MDS		**77.8** (2.4)		79.4 (3)
40 %	Baseline	MSP	75.1 (1.6)	66.9 (2.7)	75.2 (2.4)	64.4 (2)
		MLS		66.6 (2.9)		64.6 (2.4)
	TDS	MSP	75.5 (2.2)	71.2 (2)	**90.3** (1.9)	77.8 (2.3)
		MDS		76.0 (0.8)		79.0 (2.3)
	TDV	MSP	75.1 (2.4)	74 (1.2)	90.1 (2)	77.6 (2.2)
		MDS		76.1 (0.9)		**79.6** (2.7)
	CDS	MSP	78.3 (1.8)	74.4 (2)	87.6 (3.4)	76.8 (1)
		MDS		**76.3** (1.8)		77.8 (1.1)
	CDV	MSP	**79.0** (1.4)	75.0 (1.4)	87.9 (3.2)	76.9 (1)
		MDS		76.2 (2)		78.0 (1.4)
50 %	Baseline	MSP	80.0 (3.6)	68.1 (1.6)	89.4 (4.2)	76.7 (4.1)
		MLS		68.1 (1.7)		76.7 (4.1)
	TDS	MSP	80.0 (2.2)	75.3 (2.6)	90.1 (3.5)	78.9 (5.4)
		MDS		79.1 (1.9)		79.7 (5.6)
	TDV	MSP	80.5 (2.5)	76.5 (2.4)	**90.1** (3.8)	77.1 (4)
		MDS		79.2 (2)		**80.0** (5.1)
	CDS	MSP	81.8 (2.1)	78.5 (2.7)	88.8 (3)	78.6 (4)
		MDS		80.0 (2.6)		79.6 (4.3)
	CDV	MSP	**82.0** (1.9)	78.9 (3.1)	89.1 (3.4)	76.4 (7.2)
		MDS		**79.9** (2.6)		79.3 (4.5)

Table 5
Semantic shift benchmark results.

Dataset	Method		EffNetV2S			ResNet50		
			Acc_k	AUROC	OpenAUC	Acc_k	AUROC	OpenAUC
CUB	Baseline	MSP	**90.4**	88.9/82.1	83.5/78.2	86.9	86.9/79.1	79.5/73.5
		MLS		**89.6/82.2**	**84.0/78.0**		86.8/78.8	79.4/73.0
	TDS	MSP	80.8	71.3/67.1	62.8/59.4	76.6	70.9/65.9	60.0/56.2
		MDS		86.2/70.8	73.0/61.7		84.4/70.0	68.8/58.7
	TDV	MSP	81.9	86.4/71.5	73.5/62.3	78.1	85.1/71.1	69.6/59.7
		MDS		86.5/70.6	73.4/61.4		83.9/70.0	68.8/58.8
	CDS	MSP	87.7	86.5/78.9	79.8/74.0	83.3	78.2/75.5	70.2/68.4
		MDS		87.8/77.5	80.5/72.5		85.1/76.4	75.9/69.9
	CDV	MSP	88.4	88.0/78.0	81.1/73.2	85.8	85.5/77.0	77.5/71.3
		MDS		88.1/77.4	81.0/72.5		85.4/76.3	77.4/70.7
Aircraft	Baseline	MSP	85.9	86.9/76.9	79.1/71.0	82.7	81.6/74.8	72.4/67.2
		MLS		**88.0**/77.3	79.7/71.0		82.4/75.1	72.8/67.1
	TDS	MSP	77.3	55.5/54.9	49.1/48.4	75.2	53.7/53.9	46.2/46.2
		MDS		84.9/72.3	69.5/59.8		80.5/73.4	64.6/59.1
	TDV	MSP	75.8	84.5/71.8	67.5/57.9	73.3	79.9/73.6	62.1/57.4
		MDS		84.4/71.6	67.3/57.5		80.5/73.8	62.3/57.4
	CDS	MSP	86.3	84.7/75.0	77.6/69.6	82.0	70.6/69.7	62.9/62.2
		MDS		86.2/76.3	78.9/70.5		80.8/74.5	71.9/67.3
	CDV	MSP	**86.7**	86.9/76.9	79.9/71.4	84.2	80.6/74.4	72.8/68.1
		MDS		87.4/**77.4**	**80.4/71.8**		80.5/74.6	72.8/68.1
Cars	Baseline	MSP	93.6	90.7/81.8	87.1/78.9	90.4	89.1/81.1	83.5/76.7
		MLS		**90.8**/81.6	**87.1**/78.6		89.3/81.1	83.4/76.5
	TDS	MSP	82.6	68.7/67.2	61.5/60.1	78.2	64.5/63.0	55.8/54.6
		MDS		84.0/78.9	73.1/69.2		79.2/74.5	66.4/62.8
	TDV	MSP	82.8	84.9/80.0	73.5/69.8	78.6	80.2/75.4	66.6/63.0
		MDS		83.6/78.8	72.4/68.7		79.3/75.2	65.9/62.8
	CDS	MSP	92.5	87.1/82.2	83.3/79.0	91.6	88.8/81.2	84.5/77.9
		MDS		89.8/83.7	86.0/80.4		88.6/81.5	84.2/78.1
	CDV	MSP	**93.7**	90.3/**84.4**	87.0/**81.6**	92.2	88.9/82.1	84.7/78.8
		MDS		90.1/84.2	86.8/81.4		88.5/81.8	84.4/78.6

unseen classes, it achieved an AUROC of 73.9 %, compared to 60.8 %. This performance was sustained as the proportion of unseen classes increased, at 50 %, TDS achieved an AUROC of 79.7 %, surpassing the baseline by approximately 3 percentage points.

On the Semantic Shift Benchmark (SSB), we evaluated our methods on three fine-grained datasets: CUB, Aircraft, and Cars, using both EfficientNetV2S and ResNet50 architectures. Table 5 shows our results with closed-set accuracy (Acc_k), as well as AUROC and OpenAUC metrics for open-set detection, reported separately for the easy and hard unknown classes (reported as easy/hard).

On the CUB dataset, CDV achieved closed-set accuracies close to the baseline: 88.4 % with EfficientNetV2S and 85.8 % with ResNet50, compared to the baseline of 90.4 % and 86.9 %, respectively. In open-set recognition, CDV demonstrated competitive performance with EfficientNetV2S, CDV attained AUROC scores of 88.1 % (easy) and 78 % (hard), compared to the baseline of 89.6 % and 82.2 %. OpenAUC scores for CDV were 81.1 % (easy) and 73.2 % (hard), again lower than the baseline.

For the Aircraft dataset, CDV surpassed the baseline in both architectures; using EfficientNetV2S, CDV achieved a closed-set accuracy of 86.7 %, exceeding the baseline of 85.9 %; on ResNet50, CDV reached 84.2 % versus the baseline of 82.7 %. In open-set, CDV with EfficientNetV2S attained AUROC scores of 87.4 % (easy) and 77.4 % (hard), slightly higher than the baseline. OpenAUC scores for CDV were also improved, reaching 80.4 % (easy) and 71.8 % (hard).

In the Cars dataset, CDV achieved closed-set accuracies of 93.7 % with EfficientNetV2S and 92.2 % with ResNet50, closely surpassing the baseline of 93.6 % and 90.4 %. Notably, CDV improved AUROC and OpenAUC scores for hard unknown classes; with EfficientNetV2S, CDV attained AUROC scores of 90.3 % (easy) and 84.4 % (hard), compared to the baseline of 90.8 % and 81.8 %. OpenAUC scores for CDV were 87 % (easy) and 81.6 % (hard), exceeding the baseline in the hard set.

In all cases, similar trends to EfficientNetV2S were observed with ResNet50, indicating consistent performance across different network architectures.

5.1. Closed- and open-set correlation

We evaluated the correlation between closed-set classification accuracy and open-set performance (measured using AUROC), Fig. 4, by focusing exclusively on the best-performing open-set scoring methods: MLS and MDS. To keep the results clear, we used only the 60/40 in-class scaling benchmark and limited the scope to the EfficientNet architecture. This yielded a Pearson correlation of $r = 0.51$ ($p < 0.001$, 95 % CI [.37, 0.63]), indicating a moderate positive relationship between closed-set accuracy and open-set performance. These results suggest that, although improvements in closed-set accuracy generally correlate with better open-set performance, the strength of this relationship is moderate within the scope of our work.

5.2. Baseline versus dissimilarity

Across all experiments, the proposed methods demonstrated notable improvements over the baseline. First, the baseline method achieved a mean closed-set accuracy of 81.1 %. In comparison, TDS achieved a mean accuracy of 83 %, an absolute increase of 1.9 percentage points ($t(65) = 2.21$, $p = 0.031$); similarly, TDV attained a mean accuracy of 82.9 %, an increase of 1.8 percentage points ($t(65) = 2.07$, $p = 0.042$); CDS achieved a mean accuracy of 85 %, an absolute improvement of 3.9 percentage points over the baseline ($t(65) = 6.39$, $p = < 0.001$); CDV further improved the mean accuracy to 85.3 %, surpassing the baseline by 4.2 percentage points ($t(65) = 6.94$, $p = < 0.001$).

Analyzing open-set performance, the MLS baseline achieves a mean AUROC of 72.8 %. In comparison, TDS attains a mean AUROC of 77.2 %, representing an absolute increase of 4.4 percentage points ($t(71) = 4.55$, $p < 0.001$); TDV achieves a mean AUROC of 76.7 %, an increase of 3.9 percentage points over MLS ($t(71) = 3.79$, $p < 0.001$). CDS has a mean AUROC of 79.3 %, an absolute improvement of 6.5 percentage points over MLS ($t(71) = 8.95$, $p < 0.001$); and, CDV attains the highest

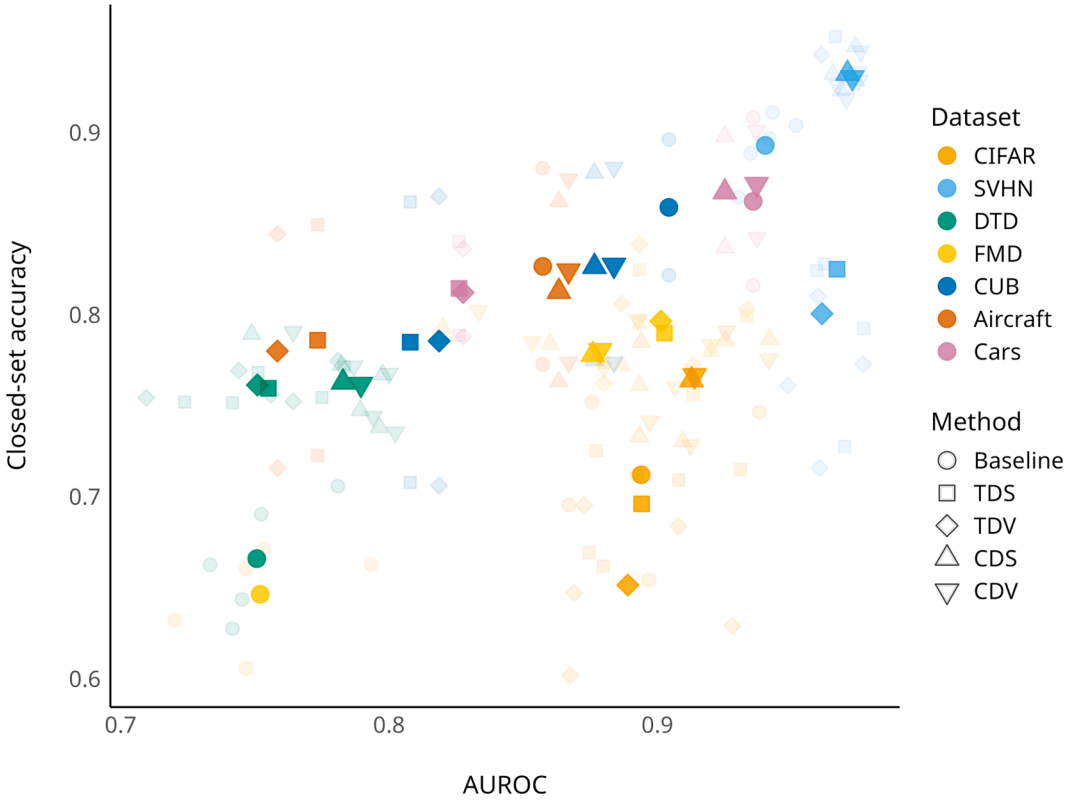


Fig. 4. Correlation between closed-set accuracy and AUROC.

mean AUROC of 79.5 %, surpassing MLS by 6.7 percentage points ($t(71) = 9$, $p < 0.001$). In both cases, the results are statistically significant, with CDS and CDV achieving the largest performance gains.

5.3. Triplet versus contrastive objectives

We compared the performance of the triplet and contrastive learning methods across all experiments. In terms of closed-set classification accuracy, triplet learning achieved a mean Acc_k of 82.9 %, while contrastive learning attained a higher mean Acc_k of 85.2 %, representing an absolute increase of 2.3 percentage points ($t(131) = 6.74$, $p < 0.001$). For open-set recognition performance, triplet learning achieved a mean AUROC of 75.9 %, and contrastive learning improved upon this with a mean AUROC of 78.9 %, an absolute increase of 3 percentage points ($t(287) = 9.09$, $p < 0.001$). In both cases, contrastive learning achieves a higher and statistically significant score.

5.4. Effect of embedding dimensionality

To isolate the influence of embedding size, we retrained models with different embedding dimensions while keeping every other hyperparameter fixed, limiting the analysis to the two best-performing contrastive variants (CDS and CDV) and evaluating them on the standard benchmarks and on SSB; for the former we used only the first cross-validation fold, and for the latter we considered only the EfficientNet-V2 backbone. Open-set performance was measured with AUROC computed from the proposed MDS score, and statistical significance was assessed with linear regression. Complete results are reported in Table 6.

For closed-set accuracy, Acc_k, a linear model with embedding dimensionality as a continuous predictor, controlling for dataset and method, found no significant effect ($b = 5.74 \times 10^{-6}$, $p = 0.162$). The changes were minor: CIFAR-10 varied by 0.2 percentage points (CDS 89.2 to 89.4 %; CDV 89.3 to 89.5 %), and SVHN by 0.1 percentage points; CUB, Aircraft and Cars by 1.1 percentage points.

Table 6
Effect of embedding dimensionality on CDS and CDV.

Dataset	Emb.	CDS		CDV	
		Acc_k	AUROC	Acc_k	AUROC
CIFAR-10	32-D	89.2	79.3	89.3	80.0
	64-D	89.2	77.6	89.5	79.2
	128-D	89.4	78.5	89.3	79.7
SVHN	32-D	97.5	94.8	97.6	94.7
	64-D	97.4	94.3	97.5	94.2
	128-D	97.4	94.7	97.6	94.5
CUB	64-D	87.9	86.6/78.0	87.9	86.4/77.4
	128-D	87.7	87.8/77.5	88.4	88.1/77.4
	256-D	88.2	87.8/79.5	88.6	87.5/79.4
	512-D	87.6	88.7/79.2	89.0	88.4/79.1
Aircraft	64-D	86.8	83.6/72.4	87.4	84.6/72.7
	128-D	86.3	86.2/76.3	86.7	87.4/77.4
	256-D	85.7	83.4/74.7	86.6	84.4/75.5
	512-D	86.3	84.1/75.7	86.6	85.1/76.7
Cars	64-D	92.4	89.4/84.1	93.3	89.5/84.6
	128-D	92.5	89.8/83.7	93.7	90.1/84.2
	256-D	93.2	89.8/83.9	93.4	89.8/84.6
	512-D	93.5	90.8/83.9	94.0	90.9/84.6

The same strategy applied to open-set AUROC with MDS scoring on the standard benchmarks (CIFAR-10 and SVHN) again showed no effect of embedding dimensionality; in fact, larger embeddings slightly reduced open-set performance ($b = -2.24 \times 10^{-5}$, $p = 0.643$). CIFAR-10 ranged from 77.6 to 79.3 % for CDS, a difference of 1.7 percentage points, and from 79.2 to 80.0 % for CDV; SVHN spanned 94.3–94.8 % (CDS) and 94.2–94.7 % (CDV), only 0.5 percentage points in both cases.

For the fine-grained SSB suite (CUB, Aircraft, Cars), the linear model likewise yielded a non-significant coefficient ($b = 2.26 \times 10^{-5}$, $p = 0.558$). In this benchmark, the improvements were larger over-

all, with the single most significant jump of 4.7 percentage points on Aircraft-Hard (CDV increased from 72.7 to 77.4 %). However, they did not translate into a consistent upward trend; some datasets peaked at 128-D, others at 512-D, and a few even dipped.

In summary, after adjustments, increasing the embedding from 32/64-D to 512-D yields, at best, marginal improvements that are rarely statistically reliable. Any AUROC gain must be weighed against the additional compute and storage, so optimal dimensionality is best selected on a case-by-case basis.

5.5. Impact of prototype count

Following the same experimental strategy as for embedding dimensionality, we assessed the impact of prototype count by building dissimilarity representations with 1, 3, 5, 10, 15, and 20 prototypes per class while keeping every other hyperparameter fixed. We again focused on the two best-performing contrastive variants (CDS and CDV) and evaluated them on the standard benchmarks and on SSB: for the standard benchmarks we considered only the first cross-validation fold, and for SSB we used the EfficientNet-V2 backbone exclusively. Open-set performance was measured with AUROC computed from the proposed MDS score, and statistical significance was assessed with linear regression. The detailed results are provided in Table 7.

For closed-set accuracy, Acc_k, we fitted a linear model with prototype count as a continuous predictor, adjusting for the dataset and method. The prototype count did not significantly predict accuracy ($b = 6.16 \times 10^{-5}$, $p = 0.259$). Across all five datasets, closed-set accuracy changed by less than one percentage point for either contrastive variant. The largest shift was observed in Cars, where CDS rose from 91.7 % using one prototype to 92.5 % using ten prototypes, a gain of 0.8 percentage points; the corresponding CDV scores moved only 0.2 percentage points (93.5 to 93.7 %). On the classic benchmarks, the changes were even smaller; for example, on CIFAR-10, CDS varied by just 0.4 percentage points (89.1 to 89.5 %), while CDV varied by only 0.2 percentage points. In short, adding prototypes produced no practically meaningful accuracy gains for either CDS or CDV.

The effect of prototype count on open-set AUROC with MDS scoring on CIFAR-10 and SVHN was likewise negligible ($b = 1.04 \times 10^{-4}$, $p = 0.545$). On CIFAR-10, CDS AUROC ranged from 77.4 % with one prototype to 79.1 % with 20 prototypes, a difference of 1.7 percentage points, whereas CDV varied by 0.6 percentage points (79.9 to 79.3 %). On SVHN, CDS fluctuated by only 0.6 percentage points (94.2 to 94.8 %), and CDV by 0.8 percentage points (93.8 to 94.6 %).

For the fine-grained SSB suite (CUB, Aircraft, Cars), we extended the model to include the difficulty subset (Easy vs. Hard) as an additional covariate. Prototype count remained non-significant ($b = 2.54 \times 10^{-4}$, $p = 0.185$). Dataset- and subset-specific AUROC ranges were all smaller than 1.3 percentage points; the largest change was observed in the Aircraft-Hard subset, which increased from 76.1 (1 prototype) to 77.4 % (20 prototypes). In many cases, the peak AUROC was achieved with just 3–5 prototypes, underscoring that adding more prototypes offers no practical benefit and can occasionally hurt performance.

Overall, adding more prototypes yields only marginal, usually non-significant, gains in both closed-set accuracy and open-set AUROC. In summary, increasing the count from 1 to 20 yields no statistically significant improvement in either metric. While a larger bank of prototypes can modestly enlarge the coverage of the open space, our results show that a small number of well-positioned prototypes per class is typically sufficient under the conditions we evaluated.

Table 7
Impact of prototype count on CDS and CDV.

Dataset	Prot. count	CDS		CDV	
		Acc_k	AUROC	Acc_k	AUROC
CIFAR-10	1	89.1	77.4	89.2	79.3
	3	89.3	78.3	89.4	79.8
	5	89.4	78.5	89.3	79.7
	10	89.5	78.9	89.3	79.9
	15	89.4	78.8	89.3	79.8
	20	89.2	79.1	89.3	79.9
SVHN	1	97.4	94.8	97.5	94.6
	3	97.4	94.5	97.6	94.3
	5	97.4	94.7	97.6	94.5
	10	97.4	94.7	97.6	94.4
	15	97.4	94.6	97.6	94.2
	20	97.4	94.2	97.5	93.8
CUB	1	87.5	87.5/77.6	88.3	87.9/77.5
	3	87.7	87.6/77.8	88.4	87.7/77.5
	5	87.7	87.7/77.8	88.4	87.9/77.6
	10	87.6	87.8/77.6	88.5	87.9/77.4
	15	87.7	87.9/77.6	88.4	87.9/77.4
	20	87.7	87.8/77.5	88.4	88.1/77.4
Aircraft	1	86.2	85.2/75.5	86.6	86.0/76.1
	3	86.4	85.4/75.2	86.5	86.8/76.7
	5	86.3	85.8/75.7	86.8	86.8/76.9
	10	86.3	85.5/75.7	86.8	87.4/77.1
	15	86.3	86.1/76.1	86.6	87.5/77.2
	20	86.3	86.2/76.3	86.7	87.4/77.4
Cars	1	91.7	89.4/83.3	93.5	89.6/83.8
	3	92.1	89.7/83.4	93.6	89.8/84.0
	5	92.2	89.7/83.9	93.6	89.9/84.2
	10	92.5	89.8/83.6	93.5	90.1/84.2
	15	92.4	89.8/83.7	93.7	90.1/84.2
	20	92.5	89.8/83.7	93.7	90.1/84.2

5.6. Clustering algorithm

To determine whether the clustering algorithm used for prototype selection matters, we evaluated four approaches: K-means, K-means++, spectral, and agglomerative clustering while leaving every other hyperparameter untouched. As this design choice is only relevant once prototypes are in place, we restricted the evaluation to the three fine-grained SSB datasets (CUB, Aircraft, Cars), used the EfficientNet-V2 backbone throughout, and reported results for the two best contrastive variants (CDS and CDV) again. Open-set performance was measured with AUROC computed from the proposed MDS score, and statistical significance was assessed with linear regression. Complete results appear in Table 8.

For closed-set accuracy, Acc_k, a linear model with clustering algorithm as a categorical predictor, controlling for dataset and method, yielded no significant effect for any of the three comparisons. Across all datasets, accuracy varied by at most 0.6 percentage points; the largest change occurred on Cars, where CDS rose from 92.4 % with vanilla K-means to 93.0 % with agglomerative clustering; on CUB and Aircraft, both variants fluctuated by ≤ 0.2 percentage points.

Table 8
Comparison of clustering algorithms for prototype selection on CDS and CDV.

Dataset	Clustering	CDS		CDV	
		Acc_k	AUROC	Acc_k	AUROC
CUB	K-means	87.6	87.8/77.5	88.3	88.1/77.4
	K-means++	87.7	87.8/77.5	88.4	88.1/77.4
	Spectral	87.6	87.7/77.8	88.2	88.0/77.4
	Hierarchical	87.7	87.9/77.6	88.3	88.0/77.4
Aircraft	K-means	86.3	86.3/76.2	86.8	87.3/77.3
	K-means++	86.3	86.2/76.3	86.7	87.4/77.4
	Spectral	86.3	86.4/76.1	86.9	87.3/77.2
	Hierarchical	86.4	86.6/76.2	86.8	87.4/77.3
Cars	K-means	92.4	89.9/83.7	93.6	90.0/84.2
	K-means++	92.5	89.8/83.7	93.7	90.1/84.2
	Spectral	92.9	89.9/83.9	93.6	90.0/84.2
	Hierarchical	93.0	89.9/83.6	93.6	90.1/84.3

The same strategy applied to open-set AUROC on SSB, now including the difficulty subset as an additional covariate, again found no significant effect of clustering choice. Dataset- and subset-specific AUROC ranges were similarly small: on Aircraft-Hard, CDS spanned 75.5–76.3 %, while CDV varied by just 0.3 percentage points; on Cars-Hard, the spread was 0.7 percentage points (83.6–84.3 %) for CDS and 0.1 percentage points for CDV.

Overall, changing the clustering algorithm produced no measurable effect on either closed-set accuracy or open-set AUROC under our experimental conditions.

5.7. Multi-view augmentation

Following the same evaluation protocol used in the previous ablations, to evaluate the robustness of the trained models and the effect of multi-view dimensionality, we varied the number of aggregated views presented at inference time. Keeping the backbone, prototypes, and all other hyperparameters fixed, we tested $\{1, 5, 10, 20, 50, 100\}$ views with the CDV variant on the three SSB datasets. Detailed results are provided in Table 9.

For closed-set accuracy, a linear model with aggregated-view count as a continuous predictor ,controlling for dataset, found no significant effect ($b = 9.4 \times 10^{-6}$, $p = 0.469$). Across all datasets the maximum change was 0.6 percentage points: CUB rose from 88.3 to 88.6 %, Aircraft from 86.5 to 87.1 %, and Cars from 93.4 to 93.7 %.

The same model fitted to open-set AUROC, including subset difficulty as an additional covariate, likewise yielded a non-significant coefficient ($b = 2.2 \times 10^{-5}$, $p = 0.695$). AUROC spreads never exceeded 0.4 percentage points on any dataset-subset combination.

In summary, increasing the number of aggregated views beyond five does not improve either closed- or open-set performance under the conditions we evaluated.

5.8. Limited training data

Next, to understand how the methods behave when training data are scarce, we randomly sampled $\{5, 10, 20, 50\}\%$ of the original training images for each dataset and retrained all models from scratch. Because tiny splits leave too few examples to place many reliable prototypes, at the 5 % budget, the majority of classes contain only a single labeled image. We proportionally reduced the number of prototypes per class to $1, 2, 5, 10$, respectively, while keeping all other hyperparameters unchanged. Fig. 5 displays the closed-set accuracy and AUROC learning curves for the Hard split of the SSB suite only, whereas Table 10 reports the complete results.

Table 9
Effect of multi-view dimensionality on CDV performance.

Dataset	Multi-view	CDV	
		Acc_k	AUROC
CUB	1	88.3	87.5/77.4
	5	88.6	88.1/77.5
	10	88.5	88.1/77.4
	20	88.4	88.1/77.4
	50	88.2	88.1/77.3
	100	88.4	88.0/77.2
Aircraft	1	86.5	86.4/76.5
	5	87.0	87.1/77.3
	10	86.8	87.3/77.4
	20	86.7	87.4/77.4
	50	86.4	87.7/77.5
	100	87.1	87.9/77.6
Cars	1	93.4	90.0/84.2
	5	93.6	90.2/84.3
	10	93.5	90.0/84.4
	20	93.7	90.1/84.2
	50	93.6	90.0/84.2
	100	93.6	90.0/84.1

Table 10
Performance when training data are limited on Baseline, CDS and CDV.

Dataset	Training	Baseline		CDS		CDV	
	data	Acc_k	AUROC	Acc_k	AUROC	Acc_k	AUROC
CUB	5 %	39.0	66.5/57.6	27.9	60.3/56.5	41.9	61.0/56.2
	10 %	53.5	67.4/57.1	53.0	68.1/62.1	56.9	68.3/62.6
	20 %	71.0	75.1/60.6	70.0	75.1/68.8	70.9	75.2/69.1
	50 %	84.2	85.5/65.2	82.4	82.5/75.7	83.0	82.7/75.9
	100 %	90.4	89.6/82.2	87.7	87.8/77.5	88.4	88.1/77.4
Aircraft	5 %	27.1	58.2/58.2	11.6	55.8/58.4	26.9	53.5/57.8
	10 %	44.3	58.5/59.4	36.8	58.5/59.3	41.9	58.6/59.5
	20 %	56.1	70.3/61.6	56.2	66.0/65.7	58.6	64.6/64.9
	50 %	75.4	80.3/71.8	77.2	76.8/73.2	78.4	76.0/73.2
	100 %	85.7	88.0/77.3	86.3	86.2/76.3	86.7	87.4/77.4
Cars	5 %	20.4	53.4/56.3	4.5	54.1/54.0	23.7	54.0/54.1
	10 %	35.0	55.6/61.1	29.6	59.5/60.0	38.7	59.3/60.5
	20 %	58.7	68.3/66.8	60.5	70.3/67.5	63.5	71.1/68.0
	50 %	84.6	80.5/76.3	85.8	83.6/78.4	86.5	84.0/78.7
	100 %	93.6	90.8/81.6	82.5	89.8/83.7	93.7	90.1/84.2

For closed-set accuracy (Acc_k), performance declines steeply when only 5 % of the labels are available; CDV already surpasses the baseline on two of the three datasets: it scores 41.9 % on CUB versus 39.0 % for the baseline and 23.7 % versus 20.4 % on Cars, while trailing the baseline by just 0.2 percentage points on Aircraft (26.9 vs 27.1 %). As the budget rises to 10 %, CDV extends its lead on CUB (+3.4 percentage points) and Cars (+3.7 percentage points). At 20 %, the three methods roughly converge on CUB, but CDV pulls ahead on Aircraft (+2.5 percentage points over the baseline) and Cars (+4.8 percentage points). With 50 % of the data, CDV is now best on both Aircraft (78.4 % vs. 75.4 %) and Cars (86.5 % vs. 84.6 %) while sitting only 1.2 percentage points below the baseline on CUB. In contrast, CDS lags markedly at the smallest budgets but closes the gap once five or more prototypes per class are available, even overtaking the baseline on Cars at medium and high budgets.

On the hard subset, AUROC rises smoothly as more labels enter the training pool, but the gains materialize sooner for the dissimilarity-based methods. With only 5 % of the data, the baseline posts the highest AUROC on all three datasets by a small margin; as soon as the budget doubles to 10 %, both CDS and CDV jump ahead on CUB (+5.0 percentage points and +5.5 percentage points, respectively) and pull even with the baseline on Aircraft and Cars. At 20 %, they are clearly superior across the board, leading by up to 8.5 percentage points on CUB and by 3–4 percentage points on Aircraft and Cars. The gap widens further at 50 %, where dissimilarity boosts CUB-Hard AUROC by +10.7 percentage points (75.9 vs. 65.2) and adds roughly +1.4-2.4 percentage points on the other two datasets.

CDV, and to a slightly lesser extent CDS, surpass the baseline and hold that advantage across almost every data budget, still matching or surpassing it on two of the three datasets even under full supervision. Although the baseline maintains a slight edge in the most extreme low-data setting, a small bank of well-chosen prototypes quickly proves to be the more reliable choice for open-set detection, with CDV delivering the most consistent gains overall.

5.9. Limitations and future work

This study is confined to medium-scale image datasets and CNN backbones. Although transformer architectures and large-scale open-world benchmarks may raise the absolute performance ceiling, evaluating these settings lies outside the scope of the present work.

Furthermore, our method has several limitations: i) it depends on many hyper-parameters, including projection-head depth, temperature,

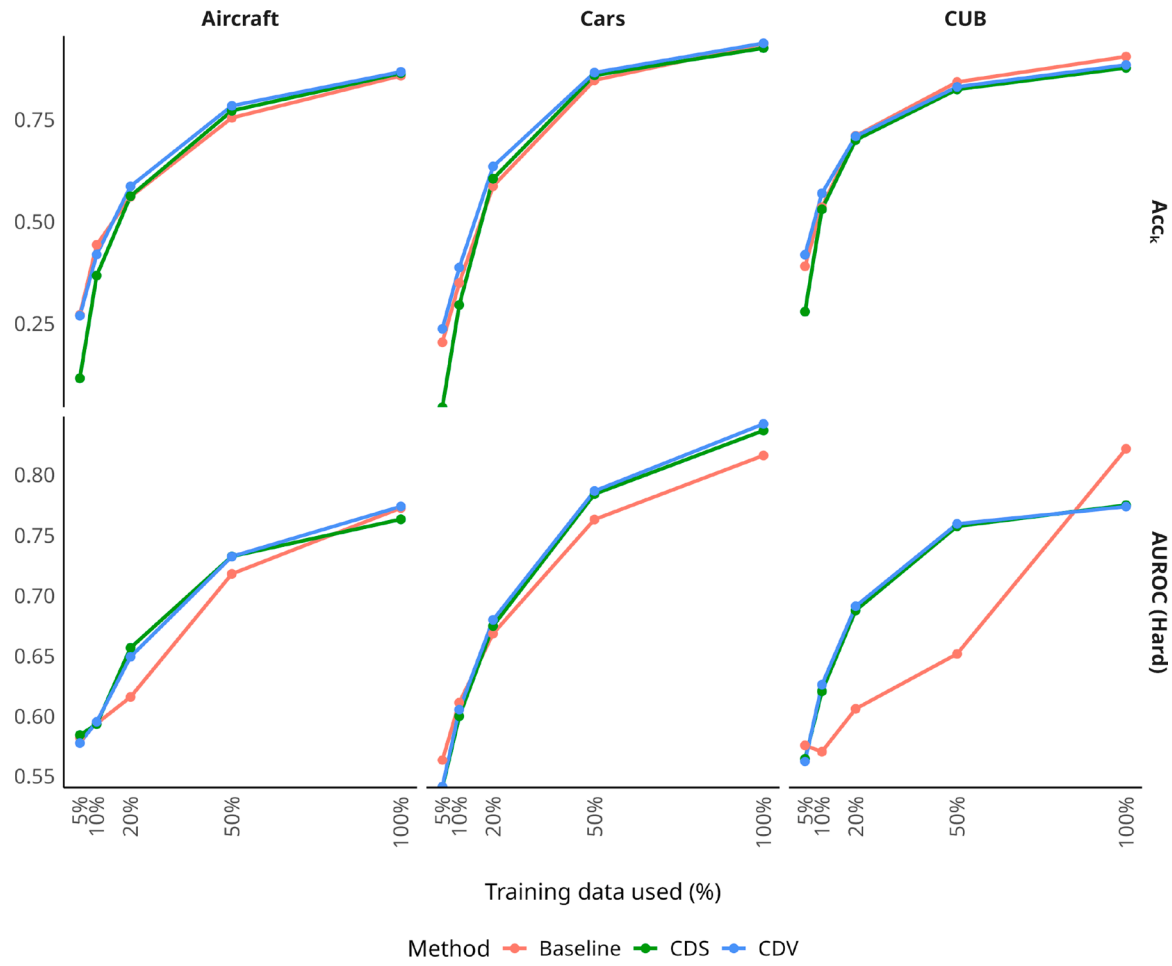


Fig. 5. Learning curves for Acc_k and AUROC vs. data budget.

learning rate, batch size, and prototype selection, that must be tuned for each task; ii) because every prediction is driven by the dissimilarity between learned embeddings, any weakness in those embeddings directly degrades performance; iii) the curse of dimensionality persists, as redundant features inflate the dissimilarity vector, add noise, and lower accuracy, though feature selection or pooling can curb this effect; and iv) the MDS is only as reliable as its prototypes, so a poor or imbalanced prototype set weakens open-set rejection, although automated pruning or periodic re-clustering could reduce that sensitivity.

As future work, methodologically, we see two main avenues for improvement: first, the method could leverage KUCs in two complementary ways: during training, KUC images can be used as hard negatives to increase the margin around each known class, and at inference, prototypes drawn from the same KUC pool can serve as hard negatives when computing the minimum dissimilarity score. Second, architectural enhancements could be explored: more metric learning objectives (e.g., proxy-based, angular-margin, adaptive-radius), stronger representation learners (e.g., generative models, masked-image pretraining, self-distillation), and more sophisticated prototype selection strategies beyond unsupervised clustering (e.g., diversity-aware sampling, supervised pruning, or adversarially synthesized examples) to further broaden coverage. Empirically, a comprehensive cross-domain evaluation spanning vision, text, and audio benchmarks under varying noise and imbalance levels remains an important next step for rigorously stress-testing the framework in realistic, continually evolving scenarios.

6. Conclusion

In this paper, we proposed a novel dissimilarity-based framework for open-set recognition that unifies representation and metric learning to derive a task-specific dissimilarity function; by mapping data into a space of pairwise differences, the method more effectively separates known from unknown classes, and our experiments confirm that the contrastive dissimilarity variants (CDS and CDV) consistently surpass strong baselines across diverse datasets and openness levels. Building on previous research, we also observed a clear positive correlation between closed-set accuracy and open-set performance, suggesting that advances in standard classification accuracy directly translate into improved rejection of unseen categories. Finally, we note an important open challenge: metrics optimized only on seen classes do not always extrapolate gracefully to unseen semantics, and this shortcoming grows with semantic shift; developing approaches that mitigate this limitation remains a promising direction for future research.

CRediT authorship contribution statement

Lucas O. Teixeira: Writing – original draft, Visualization, Methodology, Investigation, Formal analysis, Conceptualization; **Diego Bertolini:** Writing – review & editing, Visualization, Validation; **Luiz S. Oliveira:** Writing – review & editing, Validation, Methodology; **George D.C. Cavalcanti:** Writing – review & editing, Validation, Methodology; **Yandre M.G. Costa:** Writing – review & editing, Validation, Supervision, Methodology, Conceptualization.

Data availability

The data used is publicly available, and the code is available at https://github.com/lucasxteixeira/openset-dissimilarity.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This study was financed in part by the National Council for Scientific and Technological Development (CNPq) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Appendix A. Toy example of dissimilarity-based open-set recognition

Fig. A.6 shows a synthetic two-dimensional data set with two known Gaussian clusters, drawn as green and blue circles, and one unknown cluster, drawn as red triangles. Each column corresponds to a different number of prototypes per known class: one, two, and three, represented as black crosses.

In the first row, samples appear in their original coordinates. The color of each point is the decision of a nearest-prototype classifier without any rejection rule, so every sample receives the class of the closest prototype, even when that prototype is far away.

In the second row, every sample is embedded in a dissimilarity space whose coordinates are Euclidean distances to the prototype set. For display, these vectors are reduced to two principal components. The shading encodes the open-set decision: a point is colored green or blue if it is within a threshold of a prototype and light gray if it lies outside, in which case the model outputs it as unknown. Increasing the number of prototypes tightens the coverage of the known space and reduces the gray area to some extent.

The third row represents the dissimilarity vector. In this view, each original sample is transformed into multiple points, one for each prototype, by taking the absolute difference between their feature values. As a result, class labels are reduced to a binary tag indicating whether the sample and prototype come from the same class (orange) or different classes (purple). The open-set decision then combines the outcomes of all related pairs; however, the plot still shows a clear separation between same-class and different-class pairs.

The number and placement of prototypes strongly affect the decision boundary. Fewer prototypes create a more simplistic boundary and a larger rejected region, reducing the risk of false accepts but potentially under-fitting known classes. More prototypes provide greater coverage of the known samples but increase the probability of accepting unknown points as known, and they can introduce more noise, especially in the dissimilarity vector representation where each sample generates multiple pairwise points.

Fig. A.6. Toy example of open-set boundaries with one, two, and three prototypes per class.

References

[1] W.J. Scheirer, A. de Rezen, A. Sapkota, T.E. Boult, Toward open set recognition, IEEE Trans. Pattern Anal. Mach. Intell. 35 (2013) 1757–1772.

[2] A. Bendale, T.E. Boult, Towards open set deep networks, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1563–1572.

[3] L. Neal, M. Olson, X. Fern, W.-K. Wong, F. Li, Open set learning with counterfactual images, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018.

[4] G. Chen, P. Peng, X. Wang, Y. Tian, Adversarial reciprocal points learning for open set recognition, IEEE Trans. Pattern Anal. Mach. Intell. 44 (2022) 8065–8081.

[5] H. Zhang, A. Li, J. Guo, Y. Guo, Hybrid models for open set recognition, Springer, 2020. Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16, 102–117.

[6] S. Vaze, K. Han, A. Vedaldi, A. Zisserman, Open-set recognition: a good closed-set classifier is all you need?, in: International Conference on Learning Representations, 2022.

[7] P.F. Jaeger, C.T. Lüth, L. Klein, T.J. Bungert, A call to reflect on evaluation practices for failure Detection in Image Classification, in: The Eleventh International Conference on Learning Representations, 2023.

[8] E. Pękalska, P. Paclik, R.P.W. Duin, A generalized kernel approach to dissimilarity-based classification, J. Mach. Learn. Res. 2 (2002) 175–211.

[9] E. Pękalska, R. Duin, Dissimilarity-based classification for vectorial representations, in: 18th International Conference on Pattern Recognition (ICPR'06), IEEE, 2006.

[10] Y.M.G. Costa, D. Bertolini, A.S. Britto, G.D.C. Cavalcanti, L.E.S. Oliveira, The dissimilarity approach: a review, Artif. Intell. Rev. 53 (2019) 2783–2808.

[11] E. Pękalska, R.P. Duin, P. Paclík, Prototype selection for dissimilarity-based classifiers, Pattern Recognit. 39 (2006) 189–208. Part Special Issue: Complexity Reduction.

[12] L.O. Teixeira, D. Bertolini, L.S. Oliveira, G.D.C. Cavalcanti, Y.M.G. Costa, Contrastive dissimilarity: optimizing performance on imbalanced and limited data sets, Neural Comput. Appl. 36 (2024) 20439–20456.

[13] L.O. Teixeira, D. Bertolini, L.S. Oliveira, G.D.C. Cavalcanti, Y.M.G. Costa, Triplet dissimilarity: a texture classification approach using dissimilarity and siamese networks, Soft comput.. (2025). Accepted for publication (in press).

[14] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: a unified embedding for face recognition and clustering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[15] R. Hadsell, S. Chopra, Y. LeCun, Dimensionality reduction by learning an invariant mapping, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), volume 2, 2006, pp. 1735–1742.

[16] A. Dosovitskiy, J.T. Springenberg, M. Riedmiller, T. Brox, Discriminative unsupervised feature learning with convolutional neural networks, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, volume 27, Curran Associates, Inc., 2014.

[17] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (2013) 1798–1828.

[18] B. Kulis, et al., Metric learning: a survey, Foundations Trends Mach. Learn. 5 (2013) 287–364.

[19] A.v.d. Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding, 2018, https://arxiv.org/abs/1807.03748.

[20] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: Proceedings of the 37th International Conference on Machine Learning, JMLR.org, 2020.

[21] K. Sohn, Improved deep metric learning with multi-class n-pair loss objective, in: D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 29, Curran Associates, Inc., 2016.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Commun. ACM 63 (2020) 139–144.

[23] G.P. Nguyen, M. Worring, A.W.M. Smeulders, Similarity learning via dissimilarity space in cbir, in: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval, Association for Computing Machinery, New York, NY, USA, 2006, p. 107–116.

[24] R.H. Zottesso, Y.M. Costa, D. Bertolini, L.E. Oliveira, Bird species identification using spectrogram and dissimilarity approach, Ecol. Inform. 48 (2018) 187–197.

[25] I. Theodorakopoulos, D. Kastaniotis, G. Economou, S. Fotopoulos, Pose-based human action recognition via sparse representation in dissimilarity space, J. Vis. Commun. Image Represent. 25 (2014) 12–23.

[26] L. Nanni, G. Minchio, S. Brahnam, G. Maguolo, A. Lumini, Experiments of image classification using dissimilarity spaces built with siamese networks, Sensors 21 (2021) 1573.

[27] L. Nanni, G. Minchio, S. Brahnam, D. Sarraggiotto, A. Lumini, Closing the performance gap between siamese networks for dissimilarity image classification and convolutional neural networks, Sensors 21 (2021) 5809.

[28] C. Geng, S.-J. Huang, S. Chen, Recent advances in open set recognition: a survey, IEEE Trans. Pattern Anal. Mach. Intell. 43 (2021) 3614–3631.

[29] A.R. Dhamija, M. Günther, T.E. Boult, Reducing network agnostophobia, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2018, p. 9175–9186.

[30] Z. Wang, Q. Xu, Z. Yang, Y. He, X. Cao, Q. Huang, Openauc: towards auc-oriented open-set recognition, in: Proceedings of the 36th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2024.

[31] Y. He, H. Su, H. Yu, C. Yang, W. Sui, C. Wang, S. Liu, A light-weight framework for open-set object detection with decoupled feature alignment in joint space, 2024, https://arxiv.org/abs/2412.14680.

[32] G. Feng, D. Desai, S. Pasquali, D. Mehta, Open set recognition for random forest, in: Proceedings of the 5th ACM International Conference on AI in Finance, Association for Computing Machinery, New York, NY, USA, 2024, p. 45–53.

[33] N. Bahavan, S. Seneviratne, S. Halgamuge, Sphor: A representation learning perspective on open-set recognition for identifying unknown classes in deep learning models, 2025, https://arxiv.org/abs/2503.08049.

[34] E. Pękalska, R.P.W. Duin, The Dissimilarity Representation for Pattern Recognition: Foundations and Applications, WORLD SCIENTIFIC, 2005.

[35] D. Bertolini, L.S. Oliveira, R. Sabourin, Multi-script writer identification using dissimilarity, in: 2016 23rd International Conference on Pattern Recognition (ICPR), 2016, pp. 3025–3030.

[36] T.B. Viana, V.L. Souza, A.L. Oliveira, R.M. Cruz, R. Sabourin, A multi-task approach for contrastive learning of handwritten signature feature representations, Expert Syst. Appl. 217 (2023) 119589.

[37] V.L. Souza, A.L. Oliveira, R.M. Cruz, R. Sabourin, A white-box analysis on the writer-independent dichotomy transformation applied to offline handwritten signature verification, Expert. Syst. Appl. 154 (2020) 113397.

[38] R.P. Duin, E. Pękalska, The dissimilarity space: bridging structural and statistical pattern recognition, Pattern Recognit. Lett. 33 (2012) 826–832. Special Issue on Awards from ICPR 2010.

[39] A. Krizhevsky, Learning multiple layers of features from tiny images, 2009. Technical Rep., University of Toronto.

[40] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, et al., Reading digits in natural images with unsupervised feature learning, Granada, 2011. NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011, 4.

[41] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, A. Vedaldi, Describing textures in the wild, in: Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2014.

[42] L. Sharan, R. Rosenholtz, E.H. Adelson, Accuracy and speed of material categorization in real-world images, J. Vis. 14 (2014).

[43] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The Caltech-UCSD Birds-200-2011 Dataset, 2011, Technical Report CNS-TR-2011-001, California Institute of Technology.

[44] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, A. Vedaldi, Fine-grained visual classification of aircraft, 2013. http://arxiv.org/abs/1306.5151.

[45] J. Krause, M. Stark, J. Deng, L. Fei-Fei, 3D object representations for fine-grained categorization, in: 2013 IEEE International Conference on Computer Vision Workshops, 2013, pp. 554–561.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[47] Y. Movshovitz-Attias, A. Toshev, T.K. Leung, S. Ioffe, S. Singh, No fuss distance metric learning using proxies, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 360–368.

[48] Y. Zhai, X. Guo, Y. Lu, H. Li, In defense of the classification loss for person re-identification, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2019, pp. 1526–1535.

[49] A. Buslaev, V.I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, A.A. Kalinin, Albumentations: fast and flexible image augmentations, Information 11 (2020) 125.

[50] E.D. Cubuk, B. Zoph, J. Shlens, Q. Le, Randaugment: practical automated data augmentation with a reduced search space, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 18613–18624.