Rethinking Generalizable Face Anti-spoofing via Hierarchical Prototype-guided Distribution Refinement in Hyperbolic Space

Chengyang Hu[1*], Ke-Yue Zhang[2*], Taiping Yao[2], Shouhong Ding[2†], Lizhuang Ma[1,3†]
[1]Shanghai Jiao Tong University; [2]Youtu Lab, Tencent;
[3]Shanghai Key Laboratory of Computer Software Evaluating and Testing.
{huchengyang, lzma}@sjtu.edu.cn, {zkyezhang, taipingyao, ericshding}@tencent.com

Abstract

Generalizable face anti-spoofing (FAS) approaches have drawn growing attention due to their robustness for diverse presentation attacks in unseen scenarios. Most previous methods always utilize domain generalization (DG) frameworks via directly aligning diverse source samples into a common feature space. However, these methods neglect the hierarchical relations in FAS samples which may hinder the generalization ability by direct alignment. To address these issues, we propose a novel Hierarchical Prototype-guided Distribution Refinement (HPDR) framework to learn embedding in hyperbolic space, which facilitates the hierarchical relation construction. We also collaborate with prototype learning for hierarchical distribution refinement in hyperbolic space. In detail, we propose the Hierarchical Prototype Learning to simultaneously guide domain alignment and improve the discriminative ability via constraining the multi-level relations between prototypes and instances in hyperbolic space. Moreover, we design a Prototype-oriented Classifier, which further considers relations between the sample and prototypes to improve the robustness of the final decision. Extensive experiments and visualizations demonstrate the effectiveness of our method against previous competitors.

1. Introduction

The significance of face anti-spoofing (FAS) lies in its role of protecting face recognition systems from presentation attacks, like printed photos or replayed videos. In the initial stages of FAS research, the detection of spoof patterns relies on hand-crafted features such as SIFT [42], LBP [3, 10], and HOG [30, 56]. With the development of deep learning, researchers turn to deep neural networks [1, 14, 25, 32, 41, 63, 64, 66] for FAS tasks. Though these methods have attained remarkable performance in the intra-dataset setting, the performance drops significantly when encountering unseen scenarios or attacks.

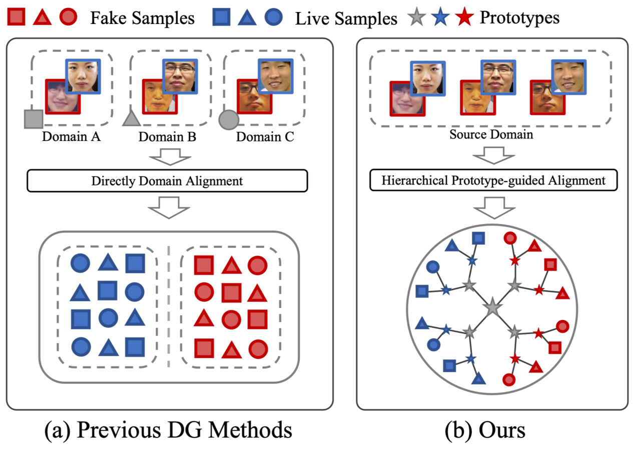

Figure 1. Previous methods map the samples into a compact feature space for generalization. While, we introduce multiple learnable prototypes in hyperbolic space to separately describe the distributions of real and spoof samples, which refines the feature alignment with hierarchical structure and promotes discrimination.

To improve the generalization, several methods introduce domain generalization techniques into FAS tasks, *e.g.,* domain adversarial learning [26, 36, 52], meta-learning [33, 47, 64], feature disentangle [35, 60] and contrastive learning [53]. It is worth noting that all of these DG-based FAS methods strive to learn a domain-invariant representation within the Euclidean space, as depicted in the Figure 1 (a). However, they often overlook the inherent hierarchical structure present in the FAS data. Such hierarchical structure of FAS data can be interpreted from different perspectives, such as the attack form and domain information. Roughly establishing a hierarchical structure for attack forms is by categorizing them into 2D and 3D attacks and both of these categories can be further divided into more fine-grained subcategories. As for domain information, the

*Equal Contribution.
†Corresponding Authors.

data might be categorized with light, background, and color, *etc*. These different inherent hierarchical structures in FAS are essential for humans to understand the data and facilitate generalization to unseen data. The absence of such hierarchical information might hinder the generalization ability of the above methods and also result in poor interpretability.

To tackle these limitations, we introduce hyperbolic space, instead of traditional Euclidean space, which takes advantage of modeling a hierarchical structure since it could objectively represent the distance relations of the tree nodes [2] and is skilled in modeling hierarchical structure without information loss. To facilitate the hierarchical structure learning in hyperbolic space, inspired by the prototype learning [19, 58], which introduces the prototypes into feature space to represent the overall feature distribution better, we carefully rethink the nature of feature alignment of hierarchical structure in the FAS task. In this work, we adopt prototype learning in hyperbolic space to model a hierarchical feature space for the generalizable FAS task. As shown in Figure 1 (b), we propose a novel Hierarchical Prototype-guided Distribution Refinement (HPDR) in hyperbolic feature space, which introduces multiple prototypes to comprehensively represent the feature distribution and further captures the hierarchical structure. Concretely, we first introduce learnable leaf prototypes in hyperbolic to bridge the relations of prototypes and instances. Also, copies of none-leaf prototypes are initialized to form the hierarchical relations among leaf prototypes in hyperbolic space. Then, to ensure the representativeness of the prototypes, we propose Hyperbolic Prototype Learning (HPL) including Prototype-based Feature Alignment and Hierarchical Structure Modeling. Prototype-based Feature Alignment focuses on obtaining effective features from samples and aligning the features via prototypes. Hierarchical Structure Modeling takes advantage of hyperbolic space to model the hierarchical structure by exploring prototype relations. HPL models the relations between the samples and prototypes in three levels including Prototype-Instance, Instance-Instance, and Prototype-Prototype levels. Finally, to fully utilize the prototype information, we design Prototype-oriented Classifier for inference.

The main contributions are summarized as follows:

- We first consider the hierarchical relations in FAS sample features and propose Hierarchical Prototype-guided Distribution Refinement (HPDR) to model to hierarchical structure in hyperbolic space without information loss.
- We propose a new perspective of DG FAS that introduces multiple prototypes in hyperbolic space to comprehensively represent the hierarchical distribution and refine the domain alignment.
- Extensive experiments and visualizations are presented to demonstrate the effectiveness of our method against state-of-the-art competitors.

2. Related Work

Face Anti-Spoofing. With the development of deep learning, first, a series of deep learning-based approaches [14, 32, 57, 61] have emerged. These methods usually extract features by stacked CNNs and make the prediction by binary classifier. Despite these methods performing well in intra-dataset scenarios, their performance degraded under unseen scenarios. To tackle these, domain-generalization based approaches [6, 51, 53, 65, 67] learn domain invariant features based on domain adversarial or meta-learning. However, most of the DG-based methods aim to learn domain-invariant representation by directly aligning diverse source domains in a unified feature space. However, considering the large domain gap, such alignment may be difficult and neglect part of discriminative information. To tackle these issues, we propose a new perspective for DG-based FAS that introduces multiple prototypes and refines the hierarchical structure in hyperbolic feature space via Hierarchical Prototype-guided Distribution Refinement (HPDR).

Hyperbolic Feature Embedding. Hyperbolic feature embedding has been successfully deployed in natural language processing fields [21, 38, 39] due to the advantages in modeling the hierarchical structure of natural language. In many computer vision tasks, image embeddings also contain hierarchical relations, previous works start to transform the essential components of deep neural networks in hyperbolic space to bridge the Euclidean space and hyperbolic space [16, 48]. Instance-to-gyroplane learning methods [15, 20, 27] try to find a hyperplane in hyperbolic space, *i.e.* gyroplane via optimizing the logistic score of samples and hyperplane. Instance-to-prototype learning methods [17, 27, 49] set multiple prototypes in hyperbolic space and optimize the distance between the instances to prototypes. Instance-to-instance learning methods [8, 13, 28] optimize the relations between samples via metric learning or contrastive learning. In this work, we propose a new Hierarchical Prototype-guided Distribution Refinement (HPDR) to take advantage of hyperbolic space in hierarchical structure models for generalization.

Prototype Learning. Prototype learning is a classical method which introduces prototypes into feature space to represent the general feature distribution. Early research mainly use machine learning to generate prototypes from feature space directly, such as K-nearest-neighbor (KNN) and learning vector quantization (LVQ) [29]. Later on, with the development of the deep-learning, some approaches [44, 45, 55] optimize the prototypes via loss function to constrain the feature distribution. Some methods [55, 58] mine representative prototypes by contrastive learning. Inspired by these methods, we introduce prototype learning into domain generation FAS task and specially design the Hierarchical Prototype-guided Distribution Refinement (HPDR) to further promote generalizable feature learning.

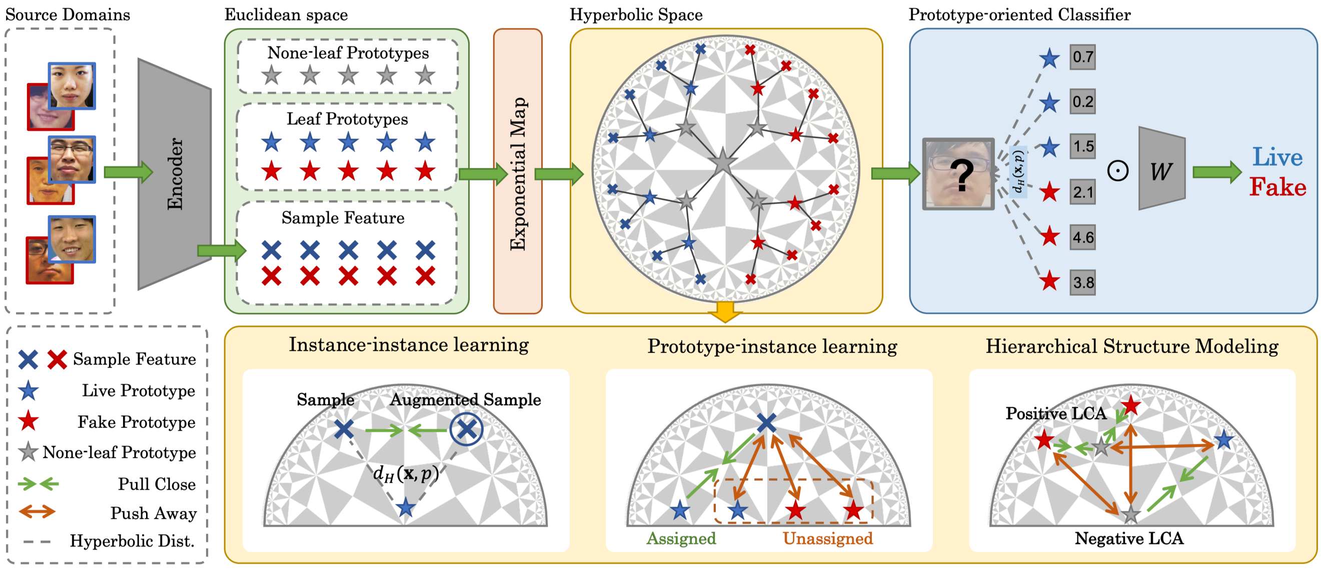

Figure 2. The overall structure of our Hierarchical Prototype-guided Distribution Refinement (HPDR) framework. After we initialize the leaf and none-leaf prototypes. Then Hyperbolic Prototype Learning (HPL) is proposed to simultaneously guide the domain alignment and improve the discriminative ability via multi-level supervision. Moreover, to utilize the relations between the samples and prototypes, we devise Prototype-oriented Classifier for robust decision.

3. Method

In this Section, we introduce Hierarchical Prototype-guided Distribution Refinement (HPDR) as shown in Figure 2. First, we will briefly introduce Hyperbolic learning and Poincaré embedding. Then, we give the definition of prototypes in hyperbolic feature space. We then propose Hyperbolic Prototype Learning (HPL) to learn a representative distribution in hierarchical space with optimization in three levels. Also, we propose a Prototype-oriented Classifier to fully utilize the advantage of prototypes.

3.1. Preliminary: Hyperbolic Learning

Hyperbolic space is defined as the Riemannian manifold with a negative curvature. In practice, the Poincaré ball model [43] is well-suited for gradient-based optimization and is a well-studied space in computer vision tasks.

Poincaré ball model $(\mathbb{D}^n, g^{\mathbb{D}})$ is defined by the manifold $\mathbb{D}^n = \{\mathbf{x} \in \mathbb{R}^n : ||\mathbf{x}|| < 1\}$ with the Riemannian metric $g^{\mathbb{D}} = \lambda_c^2 g^E$, where $\lambda_c = \frac{2}{1-c||\mathbf{x}||^2}$ is the conformal factor, c is the hyperparameter to control the curvature and radius of the ball. Due to the differential of properties in hyperbolic space and Euclidean space, vectors should be calculated by introducing gyrovector spaces [50]. For instance, the addition operation, named Möbius addition, for vectors $\mathbf{u}, \mathbf{v} \in \mathbb{D}^n$ is:

$$\mathbf{u} \oplus_c \mathbf{v} := \frac{\left(1 + 2c\langle \mathbf{u}, \mathbf{v}\rangle + c\|\mathbf{v}\|^2\right)\mathbf{u} + \left(1 - c\|\mathbf{u}\|^2\right)\mathbf{v}}{1 + 2c\langle \mathbf{u}, \mathbf{v}\rangle + c^2\|\mathbf{u}\|^2\|\mathbf{v}\|^2}. \quad (1)$$

The distance of the sample $\mathbf{u}, \mathbf{v}$ can be defined as:

$$d_H(\mathbf{u}, \mathbf{v}) = \frac{2}{\sqrt{c}}\text{arctanh}(\sqrt{c}|| - \mathbf{u} \oplus_c \mathbf{v}||). \quad (2)$$

when $c = 0$, $d_H(\mathbf{u}, \mathbf{v}) = 2||x - y||$ which is the addition operation in Euclidean space.

To utilize the operations in hyperbolic space, we need to transform the features in Euclidean space $\mathbb{R}^n$ and hyperbolic space $\mathbb{D}^n$ in a bijective way. The mapping function from $\mathbb{R}^n$ to $\mathbb{D}^n$ is called exponential map and defined as:

$$\exp_{\mathbf{x}}^c(\mathbf{v}) = \mathbf{x} \oplus_c \left(\tanh\left(\sqrt{c}\frac{\lambda_{\mathbf{x}}^c\|\mathbf{v}\|}{2}\right)\frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|}\right), \quad (3)$$

and the reverse is named as logarithmic map [27]. In practice [13, 27], we set $\mathbf{x} = \mathbf{0}$ for less cumbersome and empirically have little impact on the obtained results.

3.2. Prototype Initialization

To facilitate the hierarchical structure construction in hyperbolic space, we utilize prototype learning. To bridge the relation between samples and prototypes, we first introduce multiple prototypes in hyperbolic space $\mathbb{D}^n$ as leaf nodes in hierarchical structure respectively as:

$$\mathcal{P}_{\text{leaf}} = \{p_{ij} \mid i \in \{0, 1\}, j \in \{0, 1, \cdots, K-1\}\}, \quad (4)$$

where $p_{ij} \in \mathbb{D}^n$, i is the class index, and j is the prototype index in each class. The total number of the prototypes $|\mathcal{P}_{\text{leaf}}|$ is $2K$. For optimization difficulties when directly initializing the prototypes in hyperbolic space, we initialize all prototypes in Euclidean space and transform into hyperbolic space via Equation 3 as $p = \exp_{\mathbf{0}}^{c}(p_{\text{Euc}})$, $p_{\text{Euc}} \in \mathbb{R}^n$ is the prototype in Euclidean space. Also, to explore the hierarchical relations among the prototypes, we initialize more prototypes in hyperbolic space $\mathbb{D}^n$ as the none-leaf nodes:

$$\mathcal{P}_{\text{none-leaf}} = \{p_k \mid k \in \{0, 1, \cdots, K'\}\}. \tag{5}$$

where $|\mathcal{P}_{\text{none-leaf}}|$ is K'. Here, we assume the leaf prototypes contain the label information for they mainly reflect the distribution of the samples and the samples assigned to the same prototype should have high consistency in features. However, none-leaf prototypes are utilized to model the whole hierarchical structure without any prior knowledge (*e.g.* domain labels).

3.3. Hyperbolic Prototype Learning

To better utilize the prototypes in hyperbolic space to form a representative distribution, we propose Hyperbolic Prototype Learning (HPL) to learn the relationship between the samples and prototypes. Generally, there are two goals for HPL: 1) Aligning the features precisely via introducing prototypes; 2) Modeling the hierarchical structure of the features in hyperbolic space. To tackle these issues, we propose Prototype-instance and Instance-instance learning for prototype-based feature alignment and design Prototype-prototype learning for hierarchical structure modeling.

3.3.1 Prototype-based Feature Alignment

To align the features via prototypes in feature space, we first explore the relationship between the prototype and instance features. After initializing the prototypes into feature space, we first generate the positive pairs of the samples and prototypes by assigning features to prototypes. We assign instance features to corresponding prototypes iteratively via the rule of proximity. Concretely, given the feature z and label y, we compute the similarity between feature z to prototypes in $\mathcal{P}_{\text{leaf}}$ with the same label, and assign feature z to the closest prototype in hyperbolic distance:

$$p(z) = p_{yk} = \underset{p_{yj} \in \mathcal{P}_{\text{leaf}}}{\arg\min}\, d_H(z, p_{yj}), \tag{6}$$

where $p(z)$ is the prototype that feature z is assigned to. We expect the features to be more similar to their assigned prototype and distant to unassigned prototypes. We propose Prototype-Instance level Contrastive Learning, which pulls the features and their assigned prototype closer and pushes away features and unassigned prototypes:

$$\mathcal{L}_{PI} = -\sum_{n=1}^{N} \log \frac{e^{-d_H(z_n, p_{ck})}}{\sum_{i,j} e^{-d_H(z_n, p_{ij})}}, \tag{7}$$

where, p_{ck} is the prototype that z_n is assigned to. Moreover, to ensure the prototypes effectively represent feature distribution and avoid the trivial solution (*e.g.* unbalance assignment or assigning only one prototype), we evenly assign the samples in the mini-batch to all prototypes. With the batch size b, every prototype can be assigned by $\frac{b}{K}$ samples. Once the assigned samples to prototype p attend the $\frac{b}{K}$, $\mathcal{P}_{\text{leaf}} = \mathcal{P}_{\text{leaf}} - \{p\}$. Finally, all prototypes should be assigned samples with an even proportion.

On the other hand, although we leave some features not shared across all samples via prototypes, we aim for the left features to be task-related features. For this, we propose Instance-instance learning from the feature level and distribution level. First, we utilize inputs without and with augmentations $\mathbf{x}, \mathbf{x}_{\text{aug}}$ as pairs. To make sure the model can learn consistent task-related features, we constrain the $\mathbf{x}_{\text{aug}}$ performs in the same way as $\mathbf{x}$.

In feature level, we constrain the distance between the feature z and z_{aug} by minimizing distance:

$$\mathcal{L}_{II-feat} = d_H(\text{detach}(z), z_{\text{aug}}), \tag{8}$$

where detach$(\cdot)$ stops the gradient of variables. Also, the overall distribution of the z and z_{aug} should be the same. We restrict the distance of z_{aug} to all $\mathcal{P}_{\text{leaf}}$ should be same as z as:

$$\mathcal{L}_{II-dist} = ||d_H(\text{detach}(z), \mathcal{P}_{\text{leaf}}) - d_H(z_{\text{aug}}, \mathcal{P}_{\text{leaf}})||_2. \tag{9}$$

where $||\cdot||_2$ is the L2 distance. The total loss for Instance-instance learning is:

$$\mathcal{L}_{II} = \mathcal{L}_{II-feat} + \mathcal{L}_{II-dist}. \tag{10}$$

3.3.2 Hierarchical Structure Modeling

In this part, we model the hierarchical structure by exploring the Prototype-prototype relations. We utilize none-leaf prototypes to construct the hierarchical structure of the leaf prototypes which present sample distribution. To objectively evaluate the quality of hierarchical trees, we utilize Dasgupta cost [9] which is a widely-used metric in hierarchical clustering:

$$\mathcal{C} = \sum_{i,j} w_{ij} |\text{leaves}(p_i \vee p_j)|, \tag{11}$$

where $\mathcal{C}$ is the Dasgupta cost, w_{ij} is the ground-truth weight of the similarity between two nodes p_i and p_j, $p_i \vee p_j$ present the lowest common ancestor (LCA) and leaves$(\cdot)$ presents the number of leave node to sub-tree with the corresponding root. To build a tree with minimized Dasgupta cost, higher similarity nodes should have a smaller number of descendants of their LCA. However, it still faces two issues: 1) Dasgupta cost is unable to be optimized via gradient for its discrete formulation; 2) The ground-truth weight between nodes is unknown in this situation. Inspired by [28], we build up triplets to optimize the relations between nodes via optimizing the distance.

First, we form the triplet relation $\{p_i, p_j, p_k\}$, where (p_i, p_j) are considered as the positive pairs and (p_i, p_k) are negative. We use K-neighbor to determine the positive as:

$$(p_i, p_j) \text{ is positive} \Leftrightarrow p_i \in N_k(p_j) \ \& \ p_j \in N_k(p_i), \quad (12)$$

where $N_k(\cdot)$ is the k nearest neighbors. We use $d_H(\cdot,\cdot)$ to measure the distance of two features. The negative pairs are features that do not match the rules of Equation 12. The probability of none-leaf prototypes $\rho \in \mathcal{P}_{\text{none-leaf}}$ is the LCA of pairs (p_i, p_j) is:

$$\pi_{ij}(\rho) = e^{-\max(d_H(p_i,\rho), d_H(p_j,\rho))}, \quad (13)$$

The final LCA is sampled from the distribution $\pi_{ij}(\rho)$ as:

$$\rho_{ij} = \arg\max_{\rho}(\pi_{ij}(\rho) + g_{ij}), \quad (14)$$

where g_{ij} is the Gumbel noise to avoid local optima. Also, we obtain ρ_{ijk} as the LCA of the ρ_{ij} and p_k, which indicates that ρ_{ijk} show have a higher hierarchy than ρ_{ij}. The loss to optimize the relationship of the triplet is:

$$\begin{aligned}\mathcal{L}_{PP-LCA} = & [d_H(p_i, \rho_{ij}) - d_H(p_i, \rho_{ijk}) + \delta] \\ & + [d_H(p_j, \rho_{ij}) - d_H(p_j, \rho_{ijk}) + \delta] \\ & + [d_H(p_k, \rho_{ijk}) - d_H(p_k, \rho_{ij}) + \delta],\end{aligned} \quad (15)$$

where δ is the margin. Also, consider the $p \in P_{\text{leaf}}$ should present high confidence in hyperbolic space and obtain high variance among other leaf prototypes. So we constrain the distances of all leaf prototypes to the origin point and the distances between leaf prototypes as far as possible,

$$\mathcal{L}_{P-Origin} = -\log \frac{1}{1 + \sum_{p_i \in \mathcal{P}_{\text{leaf}}} e^{-d_H(p_i, O)}}, \quad (16)$$

$$\mathcal{L}_{PP-leaf} = -\log \frac{1}{1 + \sum_{p_i, p_j \in \mathcal{P}_{\text{leaf}}} e^{-d_H(p_i, p_j)}}, \quad (17)$$

where O is the origin of the hyperbolic space. The total loss for Prototype-prototype learning is:

$$\mathcal{L}_{PP} = \mathcal{L}_{PP-LCA} + \mathcal{L}_{P-Origin} + \mathcal{L}_{PP-leaf}. \quad (18)$$

3.4. Prototype-oriented Classifier

To better constrain and utilize the prototype-guided distribution, we devise a Prototype-oriented Classifier. Previous vanilla binary classifier roughly predicts the label through the feature and ignores the prototype information, which may hinder the performance. Since the distance between the input feature and prototypes presents the locale of the instance feature in distribution more precisely, we predict the final task label by these similarities. We design a prediction head to adaptively aggregate similarity information and map it to the final task label. The prediction loss is formulated as:

$$\mathcal{L}_{pred} = \sum_{(x_n, y_n)} -y_n \log(\sigma(W \cdot d_H(z_n, \mathcal{P}_{\text{leaf}}))), \quad (19)$$

where $W \in \mathbb{R}^K$ is a learnable weight for feature distances to all leaf prototypes, $\sigma(\cdot)$ is the Sigmoid function.

3.5. Training Procedure

Our training procedure mainly contains three steps:
Step 1: Prototype Initialization. We initialize the $\mathcal{P}_{\text{leaf}}$ and $\mathcal{P}_{\text{none-leaf}}$ in hyperbolic space in random.
Step 2: Prototype Assignment. All features are assigned to the closest leaf prototype via the rule of proximity iteratively as Equation 6.
Step 3: Prototype Optimization. After assigning the features to corresponding prototypes, we optimize the relations between features and prototypes via previous loss functions:

$$\mathcal{L}_{all} = \mathcal{L}_{pred} + \mathcal{L}_{PI} + \mathcal{L}_{II} + \mathcal{L}_{PP} \quad (20)$$

Once the feature extractor and prototypes are updated, the features will be re-assigned to prototypes as **Step 2**.

4. Experiments

4.1. Experimental Setup

Databases. Following previous DG-based FAS works [26, 33, 34], we evaluate the proposed method and other competitors using four public FAS databases: OULU-NPU [5] (denoted by O), CASIA-FASD [62] (denoted by C), MSU-MFSD [54] (denoted by M) and Idiap Replay-Attack [7] (denoted by I). These datasets were collected under different devices, attack types, lighting conditions, backgrounds, *etc.*, leading to a significant domain shift. We conduct the experiments strictly following the same protocol of the previous DG methods [33, 34, 51, 53]. Also, to evaluate the generalizable ability in cross-attack scenarios, we conduct the evaluation on HQ-WMCA dataset [23] with eight different kinds of attacks. We conduct the experiment following the protocol as [24].
Implementation Details. We first detect facial region from the input and resize it to 256×256 with RGB channels. The batchsize of each category in each domain is 5 and the total batchsize is 30. The number of leaf prototypes K is 20, and the number of none-leaf prototypes K' are set to 256. The feature dimension d in hyperbolic space is 256. The

Method		O&M&I to C		O&C&M to I		O&C&I to M		I&C&M to O	
		HTER(%)	AUC(%)	HTER(%)	AUC(%)	HTER(%)	AUC(%)	HTER(%)	AUC(%)
Traditional Methods	MS_LBP [37]	54.28	44.98	50.30	51.64	29.76	78.50	50.29	49.31
	Binary CNN [57]	34.88	71.94	34.47	65.88	29.25	82.87	29.61	77.54
	IDA [54]	55.17	39.05	28.35	78.25	66.67	27.86	54.20	44.59
	Color Texture [4]	30.58	76.89	40.40	62.78	28.09	78.47	63.59	32.71
	LBPTOP [11]	42.60	61.05	49.45	49.54	36.90	70.80	53.15	44.09
Domain Generalization	MMD-AAE [31]	44.59	58.29	31.58	75.18	27.08	83.19	40.98	63.08
	MADDG [46]	24.50	84.51	22.19	84.99	17.69	88.06	27.98	80.02
	SSDG-M [26]	23.11	85.45	18.21	94.61	16.67	90.47	25.17	81.83
	DR-Net [52]	19.68	87.43	20.87	86.72	17.02	90.10	25.02	81.47
	RFM [47]	20.27	88.16	17.30	90.48	13.89	93.98	16.45	91.16
	D^{2}AM [6]	20.98	85.58	15.43	91.22	12.70	95.66	15.27	90.87
	ANRL [33]	17.85	89.26	16.03	91.04	10.83	96.75	15.67	91.90
	DRDG [34]	19.05	88.79	15.56	91.79	12.43	95.81	15.63	91.75
	SSAN-M [53]	16.47	90.81	14.00	**94.58**	10.42	94.76	19.51	88.17
	AMEL [64]	11.88	94.39	18.60	88.79	10.23	96.62	11.31	93.96
	EBDG [12]	18.34	90.01	18.69	92.28	9.56	**97.17**	15.66	92.02
Prototype Learning	CPL [55]	16.66	90.72	15.25	90.17	10.43	95.48	19.16	88.81
	PCL [58]	17.78	91.04	17.65	90.28	15.04	92.70	17.50	91.72
Hyperbolic Embedding	HIE [27]	18.89	91.36	23.75	72.81	17.50	93.48	18.61	88.68
	HBL [18]	13.33	93.68	20.00	80.00	17.08	92.67	15.37	91.96
Ours	**HPDR**	**11.30**	**94.42**	**11.26**	92.49	**4.58**	96.02	**9.93**	**95.26**

Table 1. Comparison with face anti-spoofing methods on four testing tasks for domain generalization.

Method	M&I to C		M&I to O	
	HTER(%)	AUC(%)	HTER(%)	AUC(%)
MS_LBP [37]	51.16	52.09	43.63	58.07
IDA [54]	45.16	58.80	54.52	42.17
CT [4]	55.17	46.89	53.31	45.16
LBPTOP [11]	45.27	54.88	47.26	50.21
MADDG [46]	41.02	64.33	39.35	65.10
SSDG-M [26]	31.89	71.29	36.01	66.88
RFM [47]	36.34	67.52	29.12	72.61
D^{2}AM [6]	32.65	72.04	27.70	75.36
ANRL [33]	31.06	72.12	30.73	74.10
DRDG [34]	31.28	71.50	33.35	69.14
SSAN-M [53]	30.00	76.20	29.44	76.62
EBDG [12]	27.97	75.84	25.94	78.28
AMEL [64]	24.52	82.12	**19.68**	87.01
CPL [55]	32.15	72.54	32.50	71.25
PCL [58]	33.58	75.41	31.05	72.03
HIE [27]	29.63	79.49	24.96	79.05
HBL [18]	25.56	80.73	25.44	80.70
Ours	**22.22**	**85.54**	21.07	**87.53**

Table 2. Comparison on limited source domains.

curvature parameter c is 0.01. We select 3 nearest neighbors in default for positive prototype pairs formulating. The margin δ in $\mathcal{L}_{PP-LCA}$ is set to 0.1. We use the same embedding part of DepthNet [36]. Also, to show the effectiveness of our methods in different backbones, we also apply ResNet-18 [22] and IADG [66] as embedding parts in Supplementary Material. Adam optimizer is applied with $\text{lr} = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$ to train the framework. We use the public Pytorch [40] framework with 24G NVIDIA 4090 GPUs on Linux to implement the proposed method. Half Total Error Rate (HTER), Area Under Curve (AUC) and Average Classification Error Rate (ACER) are used as evaluation metrics.

4.2. Comparison Results

Strictly following the previous works [24, 26, 33, 34], we evaluate the methods on Leave-One-Out (LOO) validation, Limited Source Domains, and Cross-attack Settings to indicate the generalizability to the unseen domain and attacks.

Leave-One-Out (LOO). We conduct Leave-One-Out settings of FAS task by picking up three databases as the source domains for training and leaving out the rest dataset as the target domain. As shown in Table 1, We have the following observations from the results: 1) DG-based methods have better performance than traditional methods, which illustrates that domain generalization techniques can promote generalization. 2) Compared with other DG-based methods, our method shows a promising performance. Because previous methods directly align the features which may neglect the hierarchical relations. Our method introduces prototypes into hyperbolic space to form a refined distribution. 3) Our method outperforms prototype learning methods because prototype learning focuses on relationships between the prototypes and instances. However, we are more concerned about relations and form a hierarchical view to obtain a refined distribution for generalization. 4) Our method outperforms other hyperbolic embedding methods for introducing prototypes to explore the hierarchical structure in hyperbolic space, which provides a credible feature modeling that improves generalizability.

Limited Source Domains. We evaluate our method with extremely limited two source domains. Following the previous works [33, 34], MSU and Idiap are the source domain and the remaining ones (OULU and CASIA) are used for testing. As shown in Table 2, our method obtains the best

Method	FlexibleMask	Glasses	Makeup	Mannequin	PaperMask	RigidMask	Tattoo	Replay	Mean±Std
Auxiliary(Depth) [36]	40.7	48.9	25.1	0.8	0.3	26.0	35.4	0.8	22.3±18.2
CDCN [59]	33.1	46.1	38.1	1.4	8.2	**10.0**	40.7	1.1	23.3±17.7
DCN [24]	10.4	48.6	30.7	**0.0**	**0.0**	34.3	35.4	**0.0**	19.9±18.3
Ours	**9.7**	**25.6**	**15.7**	6.8	2.1	12.7	**25.4**	19.0	**14.6**±**8.5**

Table 3. Comparison result on the cross-attack settings with ACER(%).

$\mathcal{L}_{PI}$	$\mathcal{L}_{II}$	$\mathcal{L}_{PP}$	O&M&I to C	
			HTER(%)	AUC(%)
			17.78	90.29
✓			15.56	92.07
	✓		15.64	90.76
		✓	15.74	91.25
✓	✓		14.07	92.04
✓		✓	12.78	93.47
	✓	✓	13.67	92.89
✓	✓	✓	**11.30**	**94.43**

Table 4. Abaltion study on the loss functions. The checked items (✓) indicate the corresponding loss is used for optimization.

$\lvert\mathcal{P}_{\text{leaf}}\rvert$	O&M&I to C	
	HTER(%)	AUC(%)
4	15.56	93.03
6	13.52	93.13
10	12.17	92.41
20 (Ours)	**11.30**	**94.43**
40	12.29	93.45
60	13.14	91.61

Table 5. Ablation study on the number of leaf prototypes.

performance on these more challenging cases. Compared to AEML on M&I to C task, our method gains 2.30% and 3.42% with HTER and AUC. Compared with hyperbolic embedding learning methods and prototype-based methods, our method still shows a better performance. Our model is also effective in limited domains.

Cross-attack Setting. We evaluate our method on cross-attack setting with HQ-WMCA dataset [23] in Table 3 with ACER. HQ-WMCA contains eight kinds of attacks including FlexibleMask, Glasses, Makeup, Mannequin, PaperMask, RigidMask, Tattoo, and Replay. We leave one attack out in the training stage and test with the move-out attack samples to evaluate the cross-attack performance. Compared to previous methods, our method outperforms in overall performance. Our HPDR framework aligns the features in a hierarchical structure in hyperbolic space to provide more discriminating features and shows the effectiveness even tested with unseen attacks.

4.3. Ablation Study

Different Loss Functions. We explore how Hyperbolic Prototype Learning contributes to feature modeling. We conduct the experiments on O&M&I to C and remove some losses as shown in Table 4. All experiments are equipped with $\mathcal{L}_{pred}$ for classification. We have the following observations: 1) Without any loss, the result is the worst, which indicates that directly utilizing the hyperbolic space can not improve the performance. 2) With only one loss function, $\mathcal{L}_{PI}$ improves the performance most, the reason is that Prototype-instance learning bridges the relation between the prototypes and instance and causes less information loss in feature alignment. 3) With $\mathcal{L}_{PI}$ and $\mathcal{L}_{PP}$, the performance improves a lot. $\mathcal{L}_{PP}$ tasks the advantage of hyperbolic space and forms a more reasonable feature space in the hierarchical structure. 4) With all loss functions, our method gets the best performance which confirms all losses play important roles in proposed network.

Number of Leaf Prototypes. We explore the influence of leaf prototype numbers on O&M&I to C task. We apply the number of the leaf prototypes from 4 to 60. The result is shown in Table 5. We conclude that with the increase of prototypes until 20, we get the best performance. Within a certain range, more prototypes will form a better representation of the feature space. After adding more prototypes, the performance will not increase, which indicates that 20 prototypes per class are enough for formulating a representative feature space. Also, our batchsize is limited to 30 for computational resource restriction, the prototype numbers over 30 may cause unbalance assignments in every training step and lead to a trivial solution.

Curvature Parameter. We explore how the curvature parameter c influences the result on O&M&I to C with curvature parameters from 0.001 to 1 and the Euclidean space (*i.e.* $c = 0$), as shown in Table 6. We found that: 1) The result in Euclidean space degrades because Euclidean space is not suitable for hierarchical structure modeling. 2) With increase of c, the performance first increases until 0.01. A larger curvature provides flexible hyperbolic space for hierarchical structure modeling. When the curvature still increases, the performance decreases. We transform the feature to hyperbolic space with the exponential map at $\mathbf{x} = 0$ as the approximation. When the curvature increases, the approximation is not reasonable and causes the bias.

Classification Strategy. To verify the effectiveness of our Prototype-oriented Classifier, we designed the following classification strategies: 1) Classify by the average distance to live prototypes (AvgLive); 2) Classify by the maximum distance to live prototypes (MaxLive); 3) Classify by the maximum distance to attack prototypes (MaxFake); 4) Classify by the hyperbolic Multiclass Logistic Regression (Hyperbolic MLR) which is the Linear Regression in hyper-

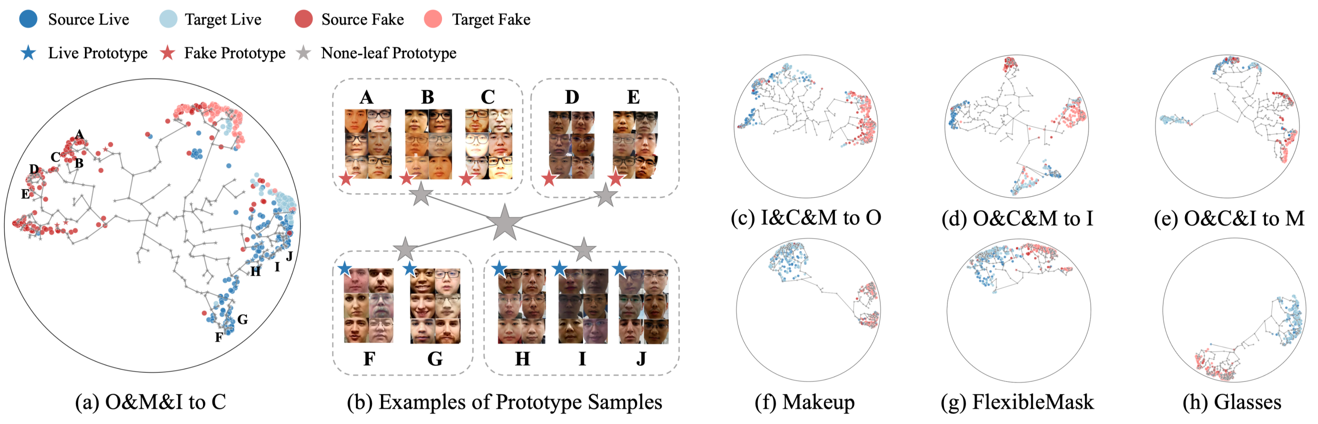

Figure 3. (a, c-h) The feature visualization on four Leave-One-Out tasks and cross-attack tasks. For every task, we select 300 samples from the source dataset and 300 samples from the target dataset. (b) Visualization of the hierarchical structure of the samples in O&M&I to C tasks. We selected 10 prototypes with 6 samples as an example.

Curvature c	**O&M&I to C**	
	HTER(%)	**AUC(%)**
0 (Euclidean)	22.22	83.74
0.001	15.56	92.19
0.01 (Ours)	**11.30**	**94.43**
0.1	14.44	92.30
1	18.15	90.83

Table 6. Ablation study on the curvature in hyperbolic space.

Classification Strategy		**O&M&I to C**	
		HTER(%)	**AUC(%)**
Hand-crafted	AvgLive	17.96	91.72
	MaxLive	14.62	91.25
	MaxFake	19.81	88.74
Deep Learning	Hyperbolic MLR	13.52	93.26
	Ours	**11.30**	**94.43**

Table 7. Ablation study on different classification strategies.

bolic space. As shown in Table 7, we have the following observations: 1) Compared to the deep learning-based methods, hand-craft methods show a limited result. The reason is that the hyperplane decided via the distance neglects the variance of different prototypes. 2) Deep learning-based methods perform better for forming a more credible hyperplane and considering the general distribution. 3) Our Prototype-oriented Classifier outperforms others in utilizing the prototype information and mitigating the influence of variance of different prototypes via learnable weight.

4.4. Visualization and Analysis

We visualize the feature distribution in Figure 3 (a, c-h) on Leave-One-Out and cross-attack tasks. Our method presents a hierarchical structure with the assistance of none-leaf prototypes. Also, our method shows an outstanding performance on cross-domain generalizability for our prototypes mitigates the information loss in feature alignment. Also, to indicate the effectiveness of the leaf prototypes, we visualize the samples assigned to different prototypes in Figure 3 (b), and will show more examples in cross-attack settings in Supplementary. All samples from the same prototypes share a high similarity. Also, for the prototypes with a lower LCA, their samples are more similar compared to the samples from a higher LCA. Our method successfully explores the hierarchical structure without prior knowledge.

5. Conclusion

In this paper, we propose a novel Hierarchical Prototype-guided Distribution Refinement (HPDR) framework, introducing the prototypes into hyperbolic feature space to obtain a refined domain distribution with a hierarchical structure for generalization. Specifically, we first introduce the leaf and none-leaf prototypes into hyperbolic features space separately for hierarchical structure construction. Then, we propose Hyperbolic Prototype Learning (HPL) to constrain the multi-level relations between features and prototypes. Moreover, we devise Prototype-oriented Classifiers to consider the relations between features and prototypes to improve the robustness of the final decision. Extensive experiments and analysis demonstrate the effectiveness of our method against state-of-the-art competitors.

Acknowledgements

This work is supported by National Natural Science Foundation of China (No. 72192821), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), Shanghai Science and Technology Commission (21511101200), YuCaiKe[2023] Project Number: 14105167-2023.

References

[1] Yousef Atoum, Yaojie Liu, Amin Jourabloo, and Xiaoming Liu. Face anti-spoofing using patch and depth-based cnns. In *IJCB*, 2017. 1

[2] Gregor Bachmann, Gary Becigneul, and Octavian Ganea. Constant Curvature Graph Convolutional Networks. In *Proceedings of the 37th International Conference on Machine Learning*, pages 486–496. PMLR. 2

[3] Zinelabidine Boulkenafet, Jukka Komulainen, and Abdenour Hadid. Face anti-spoofing based on color texture analysis. In *ICIP*, 2015. 1

[4] Zinelabidine Boulkenafet, Jukka Komulainen, Abdenour Hadid, et al. Face spoofing detection using colour texture analysis. *TIFS*, 2016. 6

[5] Zinelabinde Boulkenafet, Jukka Komulainen, Lei Li, Xiaoyi Feng, and Abdenour Hadid. Oulu-npu: A mobile face presentation attack database with real-world variations. In *FG*, 2017. 5

[6] Zhihong Chen, Taiping Yao, Kekai Sheng, Shouhong Ding, Ying Tai, Jilin Li, Feiyue Huang, and Xinyu Jin. Generalizable representation learning for mixture domain face anti-spoofing. In *AAAI*, 2021. 2, 6

[7] Ivana Chingovska, André Anjos, and Sébastien Marcel. On the effectiveness of local binary patterns in face anti-spoofing. In *BIOSIG*, 2012. 5

[8] Yawen Cui, Zitong Yu, Wei Peng, and Li Liu. Rethinking few-shot class-incremental learning with open-set hypothesis in hyperbolic geometry. *arXiv*, 2022. 2

[9] Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. In *ACM STOC*, pages 118–127, 2016. 4

[10] Tiago de Freitas Pereira, André Anjos, José Mario De Martino, and Sébastien Marcel. Lbp-top based countermeasure against face spoofing attacks. In *ACCV*, 2012. 1

[11] Tiago de Freitas Pereira, Jukka Komulainen, André Anjos, José Mario De Martino, Abdenour Hadid, Matti Pietikäinen, and Sébastien Marcel. Face liveness detection using dynamic texture. *EURASIP J IMAGE VIDE*, 2014. 6

[12] Zhekai Du, Jingjing Li, Lin Zuo, Lei Zhu, and Ke Lu. Energy-based domain generalization for face anti-spoofing. In *ACM MM*, pages 1749–1757, 2022. 6

[13] Aleksandr Ermolov, Leyla Mirvakhabova, Valentin Khrulkov, Nicu Sebe, and Ivan Oseledets. Hyperbolic vision transformers: Combining improvements in metric learning. In *CVPR*, 2022. 2, 3

[14] Litong Feng, Lai-Man Po, Yuming Li, Xuyuan Xu, Fang Yuan, Terence Chun-Ho Cheung, and Kwok-Wai Cheung. Integration of image quality and motion cues for face anti-spoofing: A neural network approach. *JVCIR*, 2016. 1, 2

[15] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *NIPS*, 2018. 2

[16] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *NIPS*, 31, 2018. 2

[17] Zhi Gao, Yuwei Wu, Yunde Jia, and Mehrtash Harandi. Curvature generation in curved spaces for few-shot learning. In *ICCV*, 2021. 2

[18] Mina Ghadimi Atigh, Martin Keller-Ressel, and Pascal Mettes. Hyperbolic busemann learning with ideal prototypes. In *NIPS*, 2021. 6

[19] Dandan Guo, Long Tian, Minghe Zhang, Mingyuan Zhou, and Hongyuan Zha. Learning prototype-oriented set representations for meta-learning. In *ICLR*. OpenReview.net, 2022. 2

[20] Yunhui Guo, Xudong Wang, Yubei Chen, and Stella X Yu. Clipped hyperbolic classifiers are super-hyperbolic classifiers. In *CVPR*, 2022. 2

[21] Shuangpeng Han, Rizhao Cai, Yawen Cui, Zitong Yu, Yongjian Hu, and Alex Kot. Hyperbolic face anti-spoofing. *arXiv preprint arXiv:2308.09107*, 2023. 2

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 6

[23] Guillaume Heusch, Anjith George, David Geissbühler, Zohreh Mostaani, and Sébastien Marcel. Deep models and shortwave infrared information to detect face presentation attacks. *IEEE TBIOM*, 2020. 5, 7

[24] Chengyang Hu, Junyi Cao, Ke-Yue Zhang, Taiping Yao, Shouhong Ding, and Lizhuang Ma. Structure destruction and content combination for generalizable anti-spoofing. *IEEE TBIOM*, 4(4):508–521, 2022. 5, 6, 7

[25] Chengyang Hu, Ke-Yue Zhang, Taiping Yao, Shice Liu, Shouhong Ding, Xin Tan, and Lizhuang Ma. Domain-hallucinated updating for multi-domain face anti-spoofing. In *AAAI*, 2024. 1

[26] Yunpei Jia, Jie Zhang, Shiguang Shan, and Xilin Chen. Single-side domain generalization for face anti-spoofing. In *CVPR*, 2020. 1, 5, 6

[27] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *CVPR*, 2020. 2, 3, 6

[28] Sungyeon Kim, Boseung Jung, and Suha Kwak. Hier: Metric learning beyond class labels via hierarchical regularization. *arXiv*, 2022. 2, 5

[29] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990. 2

[30] Jukka Komulainen, Abdenour Hadid, and Matti Pietikäinen. Context based face anti-spoofing. In *BTAS*, 2013. 1

[31] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *CVPR*, 2018. 6

[32] Lei Li, Xiaoyi Feng, Zinelabidine Boulkenafet, Zhaoqiang Xia, Mingming Li, and Abdenour Hadid. An original face anti-spoofing approach using partial convolutional neural network. In *IPTA*, 2016. 1, 2

[33] Shubao Liu, Ke-Yue Zhang, Taiping Yao, Mingwei Bi, Shouhong Ding, Jilin Li, Feiyue Huang, and Lizhuang Ma. Adaptive normalized representation learning for generalizable face anti-spoofing. *ACM MM*, 2021. 1, 5, 6

[34] Shubao Liu, Ke-Yue Zhang, Taiping Yao, Kekai Sheng, Shouhong Ding, Ying Tai, Jilin Li, Yuan Xie, and Lizhuang Ma. Dual reweighting domain generalization for face presentation attack detection. *IJCAI*, 2021. 5, 6

[35] Shice Liu, Shitao Lu, Hongyi Xu, Jing Yang, Shouhong Ding, and Lizhuang Ma. Feature generation and hypothesis verification for reliable face anti-spoofing. In *AAAI*, pages 1782–1791, 2022. 1

[36] Yaojie Liu*, Amin Jourabloo*, and Xiaoming Liu. Learning deep models for face anti-spoofing: Binary or auxiliary supervision. In *CVPR*, Salt Lake City, UT, 2018. 1, 6, 7

[37] Jukka Määttä, Abdenour Hadid, Matti Pietikäinen, et al. Face spoofing detection from single images using micro-texture analysis. In *IJCB*, 2011. 6

[38] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *NIPS*, 30, 2017. 2

[39] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *ICML*, pages 3779–3788. PMLR, 2018. 2

[40] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 6

[41] Keyurkumar Patel, Hu Han, and Anil K Jain. Cross-database face antispoofing with robust feature representation. In *CCBR*, 2016. 1

[42] Keyurkumar Patel, Hu Han, and Anil K Jain. Secure face unlock: Spoof detection on smartphones. *TIFS*, 2016. 1

[43] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *GraphDrawing*, pages 355–366. Springer, 2011. 3

[44] Atsushi Sato and Keiji Yamada. Generalized learning vector quantization. *NIPS*, 8, 1995. 2

[45] Atsushi Sato and Keiji Yamada. A formulation of learning vector quantization using a new misclassification measure. In *ICPR*, pages 322–325. IEEE, 1998. 2

[46] Rui Shao, Xiangyuan Lan, Jiawei Li, and Pong C Yuen. Multi-adversarial discriminative deep domain generalization for face presentation attack detection. In *CVPR*, 2019. 6

[47] Rui Shao, Xiangyuan Lan, Pong C Yuen, et al. Regularized fine-grained meta face anti-spoofing. In *AAAI*, 2020. 1, 6

[48] Ryohei Shimizu, Yusuke Mukuta, and Tatsuya Harada. Hyperbolic neural networks++. *arXiv preprint arXiv:2006.08210*, 2020. 2

[49] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017. 2

[50] Abraham Albert Ungar. A gyrovector space approach to hyperbolic geometry. *Synth. Lect. Math. Stat.*, 1(1):1–194, 2008. 3

[51] Chien-Yi Wang, Yu-Ding Lu, Shang-Ta Yang, and Shang-Hong Lai. Patchnet: A simple face anti-spoofing framework via fine-grained patch recognition. In *CVPR*, pages 20281–20290, 2022. 2, 5

[52] Guoqing Wang, Hu Han, Shiguang Shan, and Xilin Chen. Cross-domain face presentation attack detection via multi-domain disentangled representation learning. In *CVPR*, 2020. 1, 6

[53] Zhuo Wang, Zezheng Wang, Zitong Yu, Weihong Deng, Jiahong Li, Tingting Gao, and Zhongyuan Wang. Domain generalization via shuffled style assembly for face anti-spoofing. In *CVPR*, pages 4123–4133, 2022. 1, 2, 5, 6

[54] Di Wen, Hu Han, Anil K Jain, et al. Face spoof detection with image distortion analysis. *TIFS*, 2015. 5, 6

[55] Hong-Ming Yang, Xu-Yao Zhang, Fei Yin, and Cheng-Lin Liu. Robust classification with convolutional prototype learning. In *CVPR*, pages 3474–3482, 2018. 2, 6

[56] Jianwei Yang, Zhen Lei, Shengcai Liao, and Stan Z Li. Face liveness detection with component dependent descriptor. In *ICB*, 2013. 1

[57] Jianwei Yang, Zhen Lei, Stan Z Li, et al. Learn convolutional neural network for face anti-spoofing. *arXiv preprint arXiv:1408.5601*, 2014. 2, 6

[58] Xufeng Yao, Yang Bai, Xinyun Zhang, Yuechen Zhang, Qi Sun, Ran Chen, Ruiyu Li, and Bei Yu. Pcl: Proxy-based contrastive learning for domain generalization. In *CVPR*, pages 7097–7107, 2022. 2, 6

[59] Zitong Yu, Chenxu Zhao, Zezheng Wang, Yunxiao Qin, Zhuo Su, Xiaobai Li, Feng Zhou, and Guoying Zhao. Searching central difference convolutional networks for face anti-spoofing. In *CVPR*, 2020. 7

[60] Ke-Yue Zhang, Taiping Yao, Jian Zhang, Ying Tai, Shouhong Ding, Jilin Li, Feiyue Huang, Haichuan Song, and Lizhuang Ma. Face anti-spoofing via disentangled representation learning. *ECCV*, 2020. 1

[61] Ke-Yue Zhang, Taiping Yao, Jian Zhang, Shice Liu, Bangjie Yin, Shouhong Ding, and Jilin Li. Structure destruction and content combination for face anti-spoofing. In *IJCB*, 2021. 2

[62] Zhiwei Zhang, Junjie Yan, Sifei Liu, Zhen Lei, Dong Yi, and Stan Z Li. A face antispoofing database with diverse attacks. In *ICB*, 2012. 5

[63] Tianyi Zheng, Bo Li, Shuang Wu, Ben Wan, Guodong Mu, Shice Liu, Shouhong Ding, and Jia Wang. Mfae: Masked frequency autoencoders for domain generalization face anti-spoofing. *TIP*, pages 1–1, 2024. 1

[64] Qianyu Zhou, Ke-Yue Zhang, Taiping Yao, Ran Yi, Shouhong Ding, and Lizhuang Ma. Adaptive mixture of experts learning for generalizable face anti-spoofing. In *ACM MM*, pages 6009–6018, 2022. 1, 6

[65] Qianyu Zhou, Ke-Yue Zhang, Taiping Yao, Ran Yi, Kekai Sheng, Shouhong Ding, and Lizhuang Ma. Generative domain adaptation for face anti-spoofing. In *ECCV*, pages 335–356. Springer, 2022. 2

[66] Qianyu Zhou, Ke-Yue Zhang, Taiping Yao, Xuequan Lu, Ran Yi, Shouhong Ding, and Lizhuang Ma. Instance-aware domain generalization for face anti-spoofing. In *CVPR*, pages 20453–20463, 2023. 1, 6

[67] Qianyu Zhou, Ke-Yue Zhang, Taiping Yao, Xuequan Lu, Shouhong Ding, and Lizhuang Ma. Test-time domain generalization for face anti-spoofing. In *CVPR*, 2024. 2