Few-shot Classification with Hypersphere Modeling of Prototypes

Ning Ding[1,2*], **Yulin Chen**[2*], **Ganqu Cui**[1], **Xiaobin Wang**[3]
Hai-Tao Zheng[2,4†], **Zhiyuan Liu**[1,5†], **Pengjun Xie**[3]
[1]Department of Computer Science and Technology, Tsinghua University
[2]Shenzhen International Graduate School, Tsinghua University, [3]Alibaba Group,
[4]Pengcheng Laboratory, Shenzhen, [5]BNRIST, IAI, Tsinghua University
{dingn18, yl-chen21, cgq22}@mails.tsinghua.edu.cn
{xuanjie.wxb, chengchen.xpj}@alibaba-inc.com
{zheng.haitao}@sz.tsinghua.edu.cn, {liuzy}@tsinghua.edu.cn

Abstract

In few-shot classification, key points to make the learning phase effective are to construct expressive class-level representations and design appropriate metrics. However, previous studies often struggle to reconcile the expressivity of representations and the conciseness of metrics. When modeling class-level information, vanilla embeddings can make classification difficult due to the lack of capacity, whereas complex statistical modeling hinders metric interpretation. To address the issues simultaneously, this paper presents a simple and effective approach from the geometrical perspective, dubbed as hypersphere prototypes. Specifically, our method represents class information as hyperspheres, which are characterized by two sets of learnable parameters: a center and a radius. Our method enjoys the following advantages. (1) With the learnable parameters, unique class representations can be easily constructed and learned without additional restrictions. (2) Using "areas" instead of "points" as class representation, the expressive capability will be greatly enhanced, increasing the reliability of few-shot classification. (3) The metric design is intuitive for hypersphere representation, which is the distance from a data point to the surface of the hypersphere. As a fundamental method of few-shot classification, our method demonstrates remarkable effectiveness, generality, and compatibility with other technologies in experiments.

1 Introduction

Constituting cognition of novel concepts with a few examples is crucial for machines to emulate human intelligence, and with the exorbitant cost associated with annotating large amounts of data, few-shot learning has garnered considerable attention in modern deep learning (Lu et al., 2020). Despite the success under ample supervision, limited training examples remain a challenge for traditional deep neural models. Consequently, various approaches have been proposed to extend the applicability of deep neural networks to scenarios with limited data availability. One significant area of research within this domain is metric-based meta-learning (Snell et al., 2017; Ren et al., 2018; Allen et al., 2019), where models are trained to generate expressive representations and perform classification through predefined metrics.

The success of metric-based meta-learning depends on both *representation* learning and the *metrics* chosen. One straightforward approach relies on training feature representation and adopts a nearest-neighbor classifier (Vinyals et al., 2016; Yang and Katiyar, 2020; Wang et al., 2019). Other works introduce additional parameters as class representation to achieve better generalization ability. A naive way to estimate class representation is to use the mean embedding of feature representation, i.e., prototypes (Snell et al., 2017; Allen et al., 2019), while some also use second-order moments (Li et al., 2019a) or reparameterize the learning process to generate class representation in a richer semantic space (Ravichandran et al., 2019) or in the form of probability distribution (Zhang et al., 2019). Apart from traditional Euclidean and cosine distance, a variety of metric functions are also proposed (Sung et al., 2018; Zhang et al., 2020a; Xie et al., 2022). Most existing works learn class representation from the statistical perspective, making designing and implementing the metrics more difficult. For example, the proposed covariance metric in CovaMNet (Li et al., 2019a) theoretically requires a non-singular covariance matrix, which is awkward for neural-based feature extraction methods.

This paper revisits metric-based meta-learning and demonstrates that geometrical modeling can simultaneously **enhance the expressivity of representations** and **reduce the difficulty of metric calculation**, meanwhile yielding surprising per-

[*] Equal contribution
[†] Corresponding authors

Findings of the Association for Computational Linguistics: ACL 2023, pages 895–917
July 9-14, 2023

formance in few-shot classification. Specifically, we propose HyperProto , a simple and effective approach to model class representation with hyperspheres. It is equipped with three advantages: (1) Characterizing geometrical "area" as manifolds with complex boundaries can often be difficult in deep learning. Instead, we only use two sets of learnable parameters: the center and the radius, to represent a hypersphere, which is straightforward and easy to learn. (2) A hypersphere is much more expressive than a single point in the representation space. The introduction of a learnable radius parameter greatly expands the representative power. (3) Besides, hyperspheres are suitable for constructing measurements in Euclidean space. We can calculate the Euclidean distance from one feature point to the surface of the hypersphere in order to perform metric-based classification, which is difficult for other manifolds.

Along with the simplicity in metric design and the enhanced expressive power is the easiness in optimization and learning. With the metrics designed as distance to the hypersphere surface, both the radius and the center of the hypersphere will appear in the loss function and participate in the backward propagation during optimization. Intuitively, for the classes with sparse feature distributions, the corresponding radii of their prototypes are large, and the radii are small otherwise. Beyond the Euclidean space, we also develop two variants based on the general idea – cone-like prototypes with cosine similarities and Gaussian prototypes from the probability perspective (in Appendix A).

We conduct extensive experiments to evaluate the effectiveness of HyperProto on two classical information extraction tasks, few-shot named entity recognition (NER) (Ding et al., 2021c) and relation extraction (RE) (Han et al., 2018; Gao et al., 2019b). Despite the simplicity, we find that our approach is exceedingly effective, which outperforms the vanilla prototypes by 8.33 % absolute in average F1 on FEW-NERD (INTRA), 6.55% absolute in average F1 on FEW-NERD (INTER), 4.77% absolute in average accuracy on FewRel, respectively. The generality of our approach allows it to be easily integrated with other techniques. We combine our method with prompt-learning and task-specific pre-training to obtain high-quality representations, substantially outperforming many competitive baselines. We believe our approach could serve as a strong baseline for few-shot learning and inspire new ideas from the research community for representation learning.

2 Problem Setup

We consider the episodic N-way K-shot few-shot classification paradigm[1]. Given a large-scale annotated training set $\mathcal{D}_{\text{train}}$, our goal is to learn a model that can make accurate predictions for a set of new classes $\mathcal{D}_{\text{test}}$, containing only a few labeled examples for training. The model will be trained on episodes constructed using $\mathcal{D}_{\text{train}}$ and tested on episodes based on $\mathcal{D}_{\text{test}}$. Each episode contains a *support* set $\mathcal{S} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N\times K}$ for learning, with N classes and K examples for each class, and a *query* set for inference $\mathcal{Q} = \{\boldsymbol{x}_j^*, y_j^*\}_{j=1}^{N\times K'}$ of examples in the same N classes. Each input data is a vector $\boldsymbol{x}_i \in \mathbb{R}^L$ with the dimension of L and y_i is an index of the class label. For each input $\boldsymbol{x}_i$, let $f_\phi(\boldsymbol{x}_i) \in \mathbb{R}^D$ denote the D-dimensional output embedding of a neural network $f_\phi : \mathbb{R}^L \to \mathbb{R}^D$ parameterized by ϕ.

3 Methodology

This section describes the mechanisms of hypersphere modeling of prototypes. One hypersphere prototype is represented by two parameters: the center and the radius, which are first initialized via estimation and then optimized by gradient descent in conjunction with the encoder parameters.

3.1 Overview

We now introduce HyperProto , which are a set of hyperspheres in the embedding space D to abstractly represent the intrinsic features of classes. Formally, one prototype is defined by

$$\mathcal{B}^d(f_\phi, \boldsymbol{z}, \epsilon) := \{f_\phi(\boldsymbol{x}) \in \mathbb{R}^D : d(f_\phi(\boldsymbol{x}), \boldsymbol{z}) \leq \epsilon\}, \tag{1}$$

where $d : \mathbb{R}^D \times \mathbb{R}^D \to [0, +\infty)$ is the distance function in the metric space. f_ϕ is a neural encoder parameterized by ϕ, while z and ϵ denote the center and the radius of the hypersphere. We use $\mathcal{M}(\cdot)$ to denote the measurement between a data point and a hypersphere prototype based on $d(\cdot)$.

The central idea is to learn a hypersphere prototype for each class with limited episodic supervision, and each example in the query set $(\boldsymbol{x}^*, y^*)$ is predicted by the measurement to the hypersphere prototypes $\mathcal{M}(\boldsymbol{x}_j^*, \mathcal{B}^d)$, which is the

[1]For the few-shot named entity recognition task, the sampling strategy is slightly different (details in Appendix E).

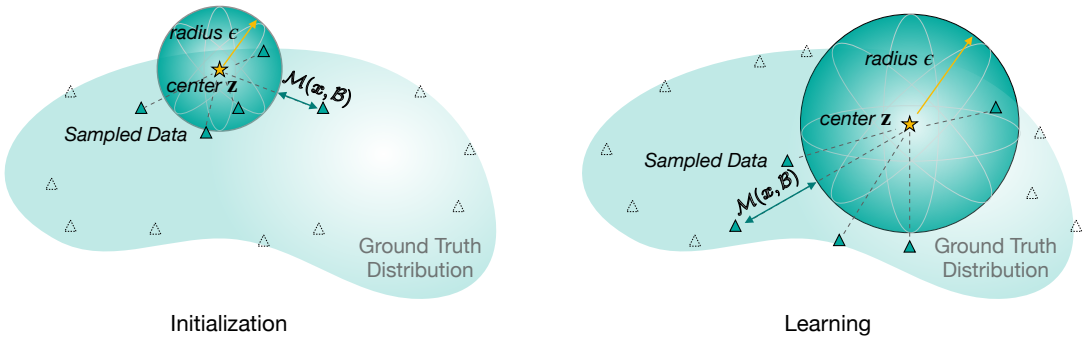

Figure 1: The illustration of our proposed *HyperProto*, where the data is sampled in 5-shot. The star symbol denotes the center of the hypersphere, the solid triangle denotes the sampled examples, and the dotted triangle denotes other examples in the whole dataset. The solid green line denotes the distance from a data embedding to the hypersphere's surface. The left part illustrates the initialization stage, where the sampled data estimate the center and radius, and the right part illustrates the learning stage, where the center and radius are simultaneously optimized.

Euclidean distance from the embedding to the *surface* of the hyperspheres,

$$\begin{aligned}\mathcal{M}(\boldsymbol{x}, \mathcal{B}) &= d(f_\phi(\boldsymbol{x}), \boldsymbol{z}) - \epsilon \\ &= \|f_\phi(\boldsymbol{x}) - \boldsymbol{z}\|_2^2 - \epsilon.\end{aligned} \tag{2}$$

Note that with such metric design, the value of $\mathcal{M}(\cdot)$ may be negative. That is, geometrically speaking, the point is contained inside the hypersphere, which does not affect the calculation of the loss function and the prediction. Generally, the idea is to use areas instead of points in the embedding space to model prototypes to enhance expressivity while preserving the convenience of Euclidean metric design. The advantages of the proposed method are two folds. First, as stated in § 1, one hypersphere prototype could be uniquely modeled by the center $\boldsymbol{z}$ and the radius ϵ, while characterizing manifolds with complex boundaries in the embedding space is intricate. Second, it is easy to optimize the parameters by conducting metric-based classification since they are naturally involved in measurement calculation.

3.2 Hypersphere Prototypes

To construct hypersphere prototypes, the first step is the initialization of the center $\boldsymbol{z}$ and the radius ϵ of the hypersphere. To start with a reasonable approximation of the data distribution, we randomly select K instances from each class for initialization. Specifically, for one class, the center of the hypersphere prototype is initialized as the mean output of the K embeddings as in Snell et al. (2017), and the radius is the mean distance from each sample to the center, as shown in Equation 3, where $\mathcal{S}_n$ is the set of sampled instances from the n-th class,

$$\mathcal{B}_n := \begin{cases} \boldsymbol{z}_n = \dfrac{1}{K} \displaystyle\sum_{\boldsymbol{x} \in \mathcal{S}_n} f_\phi(\boldsymbol{x}), \\ \epsilon_n = \dfrac{1}{K} \displaystyle\sum_{\boldsymbol{x} \in \mathcal{S}_n} d(f_\phi(\boldsymbol{x}), \boldsymbol{z}_n). \end{cases} \tag{3}$$

Once initialized, a hypersphere prototype will participate in the training process, where its center and radius are simultaneously optimized. During training, for each episode, assuming the sampled classes are $\mathcal{N} = \{n_1, n_2, ..., n_N\}$, the probability of one query point $\boldsymbol{x} \in \mathcal{Q}$ belonging to class n is calculated by softmax over the metrics to the corresponding N hypersphere prototypes.

$$p(y = n | \boldsymbol{x}; \phi) = \frac{\exp(-\mathcal{M}(\boldsymbol{x}, \mathcal{B}_n))}{\sum_{n' \in \mathcal{N}} \exp(-\mathcal{M}(\boldsymbol{x}, \mathcal{B}_{n'}))}. \tag{4}$$

And the parameters of f and hypersphere prototypes are optimized by minimizing the metric-based cross-entropy objective:

$$\mathcal{L}_{\text{cls}} = -\log p(y | \boldsymbol{x}, \phi, \boldsymbol{z}, \epsilon). \tag{5}$$

Equation 4 explains the combination of the advantages of hypersphere prototypes, where $\mathcal{M}$ is calculated by ϵ and $\boldsymbol{z}$, which will participate in the optimization. The parameters of the neural network ϕ are optimized along with the centers and radii of hypersphere prototypes through gradient descent. To sum up, in the initialization stage, the hypersphere prototypes of all classes in the training set, which are parameterized by $\boldsymbol{z}$ and ϵ, are estimated by the embeddings of randomly selected instances and *stored* for subsequent training and optimization.

Algorithm 1: Training process. f_ϕ is the feature encoder, N_{total} is the total number of classes in the training set, N is the number of classes for support and query set, K is the number of examples per class in the support set, K' is the number of examples per class in the query set, M is a hyper-parameter. RANDOMSAMPLE(S, K) denotes a set of K elements chosen uniformly at random from set S, without replacement. λ_f and λ_ϵ are separate learning rates.

Input: Training data $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_T, y_T)\}$, $y_i \in \{1, ..., N_{\text{total}}\}$. $\mathcal{D}_k$ denotes the subset of $\mathcal{D}$ containing all elements $(\boldsymbol{x}_i, y_i)$ such that $y_i = k$

Output: The updated encoder f_ϕ

// Initialization phase

for $n = 1$ *to* N_{total} **do**

$\mathcal{S}_n \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_n, K)$

$\boldsymbol{z}_n \leftarrow \frac{1}{|\mathcal{S}_n|} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{S}_n} f_\phi(\boldsymbol{x}_i)$,

$\epsilon_n \leftarrow \frac{1}{|\mathcal{S}_n|} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{S}_n} d(f_\phi(\boldsymbol{x}_i), \boldsymbol{z}_n)$,

// Learning phase

for $i = 1$ *to* M **do**

$V \leftarrow$ RANDOMSAMPLE$(\{1, ..., N_{\text{total}}\}, N)$, $\mathcal{L}_{\text{cls}} \leftarrow 0$

for n *in* $\{1, ..., N\}$ **do**

$\mathcal{Q}_n \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_n}, K')$

$\mathcal{L}_{\text{cls}} \leftarrow \mathcal{L}_{\text{cls}} + \frac{1}{NK'} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{Q}_n} [d(f_\phi(\boldsymbol{x}_i), \boldsymbol{z}_n) - \epsilon_n + \log \sum_{n'} \exp(\epsilon_{n'} - d(f_\phi(\boldsymbol{x}_i), \boldsymbol{z}_{n'}))]$

UPDATE $\boldsymbol{z}$, ϵ, f_ϕ w.r.t $\mathcal{L}_{\text{cls}}$, λ_f, λ_ϵ

In the training stage, the stored ϵ is optimized by an independent optimizer. The optimization will yield a final location and size of the hyperspheres to serve the classification performance. More importantly, the involvement of prototype centers and radii in the training process will, in turn, affect the optimization of encoder parameters, stimulating more expressive and distinguishable representations.

Algorithm 1 expresses the initialization and learning stages of hypersphere prototypes. Although the centers and radii are stored and optimized continuously in training (in contrast with vanilla prototypes where centers are re-estimated at each episode), the whole process is still episodic, as in each episode, the samples in the query set are only evaluated against the classes in that single episode instead of the global training class set.

Meanwhile, a standard episodic evaluation process is adopted to handle the unseen classes, where we estimate prototype centers and radii in closed forms. In the episodic evaluation procedure, HyperProto directly takes the mean of instance embeddings as the centers and the mean distance of each instance to the center as the radius (as in Equation 3), following previous works (Vinyals et al., 2016; Snell et al., 2017; Zhang et al., 2020a).

We also develop two variants that use "areas" to represent class-level information in few-shot classification under other measurements, details can be found in Appendix A.

4 Experiments

To evaluate the effectiveness of the proposed method, we conduct experiments on few-shot named entity recognition (NER) and few-shot relation extraction (RE) tasks, both of which are fundamental tasks of information extraction accompanied by well-established datasets. Task descriptions, datasets, and implementation details are reported in Appendix B. Apart from the experimental study in this section, we also carry out additional experiments and analyses of image classification to demonstrate the generality of our method in Appendix C.

4.1 Combination with Orthogonal Techniques

Our experiments show that by simply adding a radius parameter in the learning process, HyperProto outperforms vanilla embedding prototypes by a large margin. In addition, as a basic method of few-shot learning, HyperProto can be used successfully with other orthogonal enhancements to

Model	FEW-NERD (INTRA)				FEW-NERD (INTER)				Avg.
	5 way		10 way		5 way		10 way		
	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	
NNShot† (Yang and Katiyar, 2020)	$31.01_{1.21}$	$35.74_{2.36}$	$21.88_{0.23}$	$27.67_{1.06}$	$54.29_{0.40}$	$50.56_{3.33}$	$46.98_{1.96}$	$50.00_{0.36}$	39.77
StructShot† (Yang and Katiyar, 2020)	$35.92_{0.69}$	$38.83_{1.72}$	$25.38_{0.84}$	$26.39_{2.59}$	$57.33_{0.53}$	$57.16_{2.09}$	$49.46_{0.53}$	$49.39_{1.77}$	42.48
CONTAINER♠ (Das et al., 2021)	40.43	53.70	33.84	47.49	55.95	61.83	48.35	57.12	49.84
ESD (Wang et al., 2022)	$36.08_{1.60}$	$52.14_{1.50}$	$30.00_{0.70}$	$42.15_{2.60}$	$59.29_{1.25}$	**$69.06_{0.80}$**	$52.16_{0.79}$	$64.00_{0.43}$	50.61
Proto† (Snell et al., 2017)	$23.45_{0.92}$	$41.93_{0.55}$	$19.76_{0.59}$	$34.61_{0.59}$	$44.58_{0.26}$	$58.80_{1.42}$	$39.09_{0.87}$	$53.97_{0.38}$	39.52
HyperProto (Ours)	$32.26_{1.94}$	$50.88_{1.01}$	$24.02_{1.06}$	$42.46_{3.04}$	$52.09_{2.49}$	$65.59_{0.50}$	$44.26_{0.53}$	$60.73_{1.47}$	46.53
HyperProto +Prompt (Ours)	**$48.49_{2.75}$**	**$60.78_{1.87}$**	**$41.69_{3.45}$**	**$53.16_{3.21}$**	**$65.40_{0.08}$**	$68.34_{0.73}$	**$61.72_{0.37}$**	**$67.90_{3.90}$**	**58.44**

Table 1: Performance (F1 score) on FEW-NERD .The standard deviation is reported with 3 runs with different random seeds for each model. Results with † are reported in Ding et al. (2021c), and other baseline results are from the original papers. Results with ♠ mean that the approaches involve task-specific pre-training encoder. Best results in **bold** and the second best results are underlined.

further boost performance. We choose two techniques, prompting and task-specific pre-training to combine with our approach on NER and RE tasks, respectively. Essentially, these two methods can both be regarded as means to construct high-quality initial representations for the current task. And our approach performs metric based few-shot learning on top of the initial representations.

For NER, we enhance the primitive HyperProto with prompt (Liu et al., 2023), where in the support set the label of the entity is inserted after each entity, and in the query set the label candidates are concatenated and inserted as prefixes at the beginning of the input. For RE, we apply HyperProto to a task-specific pre-trained relation encoder (Peng et al., 2020). The two experiments further show the compatibility of the proposed method and indicate its potential as a novel fundamental modeling strategy for few-shot learning.

4.2 Overall Results

Few-shot Named Entity Recognition. Table 1 shows the performance on FEW-NERD. It can be seen that HyperProto has a considerable advantage over vanilla ProtoNet, with an increase of at least 5% in f1-score across all settings. The success on both datasets demonstrates that HyperProto can learn the general distribution pattern of entities across different classes and thus can greatly improve the performance when little information is shared between the training and test set. The performances of NNShot and HyperProto are comparable when it comes to low-shot. This is probably because, in the sequence labeling task, it is more difficult to infer the class-level information from very limited tokens. In this case, the modeling ability of hypersphere prototypes degenerates towards the nearest-neighbors strategy in NNShot. As the shot number increases, the memory cost of NNShot grows quadratically and becomes unaffordable, while HyperProto keeps it in reasonable magnitude. In this sense, HyperProto is more efficient.

When HyperProto is combined with prompt, it outperforms many other strong baselines like CONTAINER (Das et al., 2021) and ESD (Wang et al., 2022), which use pre-training and additional span attention module to enhance class representation. Specifically, HyperProto is shown to be more advantageous in INTRA setting. It also fits with our intuition since less shared information between training and test set would make features learned during pre-training stage or the trained attention module less transferable. It further shows the robustness of the modeling of HyperProto . We also believe a carefully designed initialization strategy is vital for the performance of our model in low-shot settings. The impact of the number of shots is reported in Appendix C.4.

Few-shot Relation Extraction. Table 2 presents the results on two FewRel tasks. Methods that use additional data or conduct task-specific encoder pre-training are not included. HyperProto generally performs better than all baselines across all settings. In terms of backbone models, when combined with pre-trained models like BERT, hypersphere prototypes can yield a larger advantage against prototypes. It shows that the hypersphere modeling of prototypes can better approximate the real data distribution and boosts the finetuning of BERT. Meanwhile, it sheds light on the untapped ability of large pre-trained language models and stresses that a proper assumption about data distribution may help us unlock the potential. HyperProto 's outstanding performance on the Domain Adaptation task further validates the importance of a better abstraction of data in trans-

Model	FewRel 1.0				
	5 way 1 shot	**5 way 5 shot**	**10 way 1 shot**	**10 way 5 shot**	**Avg.**
Meta Net† (Munkhdalai and Yu, 2017)	64.46 ± 0.54	80.57 ± 0.48	53.96 ± 0.56	69.23 ± 0.52	67.06
SNAIL† (Mishra et al., 2017)	67.29 ± 0.26	79.40 ± 0.22	53.28 ± 0.27	68.33 ± 0.26	67.08
GNN CNN† (Satorras and Estrach, 2018)	66.23 ± 0.75	81.28 ± 0.62	46.27 ± 0.80	64.02 ± 0.77	64.45
GNN BERT (Satorras and Estrach, 2018)	75.66	89.06	70.08	76.93	77.93
Proto-HATT‡ (Gao et al., 2019a)	76.30 ± 0.06	90.12 ± 0.04	64.13 ± 0.03	83.05 ± 0.05	78.40
MLMAN (Ye and Ling, 2019)	82.98 ± 0.20	92.66 ± 0.09	73.59 ± 0.26	87.29 ± 0.15	84.13
MTB‡♠ (Soares et al., 2019)	89.80	93.59	83.37	88.64	88.85
REGRAB♣ (Qu et al., 2020)	90.30	94.25	84.09	89.93	89.64
CP♠ (Peng et al., 2020)	95.10	97.10	91.20	94.70	94.53
MIML♣ (Dong et al., 2020)	92.55 ± 0.12	96.03 ± 0.17	87.47 ± 0.21	93.22 ± 0.22	92.32
COL♠ (Ding et al., 2021b)	92.51	95.88	86.39	92.76	91.89
ProtoCNN†	69.20 ± 0.20	84.79 ± 0.16	56.44 ± 0.22	75.55 ± 0.19	71.50
HyperProto CNN (Ours)	66.05 ± 3.44	87.31 ± 0.93	56.74 ± 1.06	77.87 ± 2.60	71.99
ProtoBERT†	80.68 ± 0.28	89.60 ± 0.09	71.48 ± 0.15	82.89 ± 0.11	81.16
HyperProto BERT (Ours)	84.34 ± 1.23	93.42 ± 0.50	77.24 ± 6.05	88.71 ± 0.31	85.93
HyperProto BERT+Pretrain♠ (Ours)	**95.29 ± 0.32**	**98.15 ± 0.05**	**92.05 ± 0.13**	**96.46 ± 0.39**	**95.49**
	FewRel 2.0 Domain Adaptation				
Proto-ADV CNN† (Wang et al., 2018)	42.21 ± 0.09	58.71 ± 0.06	28.91 ± 0.10	44.35 ± 0.09	43.55
Proto-ADV BERT† (Gao et al., 2019b)	41.90 ± 0.44	54.74 ± 0.22	27.36 ± 0.50	37.40 ± 0.36	40.35
BERT-pair† (Gao et al., 2019b)	56.25 ± 0.40	67.44 ± 0.54	43.64 ± 0.46	53.17 ± 0.09	55.13
CP♠ (Peng et al., 2020)	**79.70**	<u>84.90</u>	<u>68.10</u>	<u>79.80</u>	<u>78.13</u>
HCRP♣ (Han et al., 2021a)	76.34	83.03	63.77	72.94	74.02
LPD♠ (Zhang and Lu, 2022)	77.82 ± 0.4	86.90 ±0.3	66.06 ± 0.6	78.43 ± 0.4	77.30
ProtoCNN†	35.09 ± 0.10	49.37 ± 0.10	22.98 ± 0.05	35.22 ± 0.06	35.67
HyperProto CNN (Ours)	36.41 ± 1.43	55.50 ± 1.42	22.11 ± 0.58	40.82 ± 2.50	38.71
ProtoBERT†	40.12 ± 0.19	51.50 ± 0.29	26.45 ± 0.10	36.93 ± 0.01	38.75
HyperProto BERT (Ours)	59.03 ± 3.68	74.85 ± 4.59	45.88 ± 7.43	61.61 ± 4.69	60.34
HyperProto BERT+Pretrain♠ (Ours)	<u>78.99 ± 1.26</u>	**91.65 ± 0.44**	**67.32 ± 1.90**	**84.47 ± 0.54**	**80.61**

Table 2: Accuracies on FewRel 1.0 and FewRel 2.0 under 4 different settings. The standard deviation is reported with 3 runs with different random seeds for each model. Results with † are reported in Gao et al. (2019b) and Han et al. (2018). Results with ‡ are obtained by re-running the original code. Other baseline results are from the original papers. Results with ♠ mean that the approaches involve task-specific pre-training encoder. Results with ♣ indicate that the approaches involve additional resources like knowledge graphs and relation descriptions, etc. Best results in **bold** and the second best results are <u>underlined</u>.

fer learning. Meanwhile, the large performance variation in the domain adaptation task suggests that when the domain shifts, the estimation of hypersphere prototypes becomes less stable.

To further evaluate the compatibility of our approach and other orthogonal techniques, we replace the original BERT model with the version pre-trained on relation classification task (Peng et al., 2020). It could be observed that, with this pre-trained encoder, the performance of our method boosts substantially, demonstrating the model-agnostic nature of our approach.

4.3 Experimental Analysis

Analysis of the Radius Dynamics. We demonstrate the mechanism of hypersphere prototypes by illustrating the change of radius for one specific hypersphere. In the learning phase, the radius of a hypersphere prototype changes according to the "density" of the sampled episode, which could be characterized by the mean distance of samples to the corresponding prototype center. Practically, due to randomness in sampling, the value of the mean distance may oscillate at a high frequency in this process, and the radius changes accordingly. To better visualize the changing of radius along with the mean distance at each update, for each round of training, we fix one specific class as the *anchor class* for mean distance and radius recording and apply a special sampling strategy at each episode. Specifically, we take FewRel training data and train on the 5 way 5 shot setting with CNN encoder. While training, each episode contains the *anchor class* and 4 other randomly sampled classes.

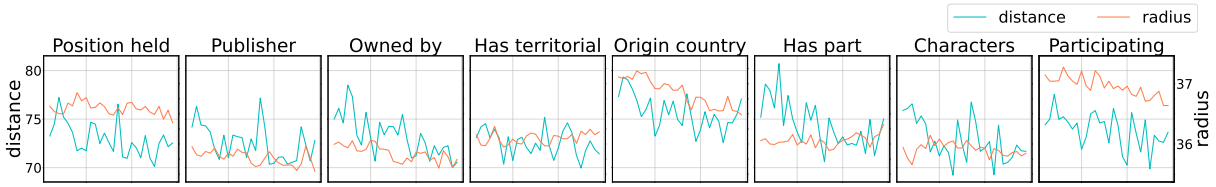

Figure 2: The illustration depicts the radius change according to the degree of sparsity of the sampled episode. Each subfigure represents a selected anchor class in FewRel. The horizontal axis represents the increase of training steps.

Training accuracy is logged every 50 steps. After a warmup training of 500 steps, we sample "good" or "bad" episodes for every 50 steps alternatively. A "good" episode has higher accuracy on the anchor class than the previously logged accuracy, while conversely, a "bad" episode has an accuracy lower than before. The mean distance to the prototype center and radius at each episode are logged every 50 steps after the warmup.

Figure 2 shows the changing of mean distance and radius for 8 classes during 600∼2000 training steps. Although the numeric values of distance and radius differ greatly and oscillate at different scales, they have similar changing patterns. Besides, it could be observed that there is often a small time lag in the change of radius, indicating that the change of radius is brought by the change in mean distance. This is in line with our expectations and perfectly demonstrates the learning mechanism of hypersphere prototypes. The experiment provides a promising idea, if we can control the sampling strategy through knowledge a priori, we may find a way to learn ideal hypersphere prototypes.

Visualization. We also use t-SNE (van der Maaten and Hinton, 2008) to visualize the embedding before and after training, by ProtoNet and HyperProto , respectively. 5 classes are sampled from the training set and test set of the Few-NERD dataset, and for each class, 500 samples are randomly chosen to be embedded by BERT trained on the 5-way-5-shot NER task. Figure 3 shows the result of embeddings in a 2-dimensional space, where different colors represent classes. Note that for the token-level NER task, the interaction between the target token and its context may result in a more mixed-up distribution compared to instance-level embedding. For both models, the representations of the same class in the training set become more compressed and easier to classify compared to their initial embeddings. While HyperProto can produce even more compact clusters. The clustering effect is also observed for novel classes. We also calculate the difference between the mean euclidean distances from each class sample to the (hypersphere) prototype of the target class and to other classes. The larger the difference, the better the samples are distinguished. For ProtoNet, the difference is 2.33 and 1.55 on the train and test set, while for HyperProto the results are 5.09 and 4.56, respectively. This can also be inferred from the t-sne result. Since samples from different classes are distributed at different densities, an extra radius parameter will help better distinguish between classes. The visualization and statistical results demonstrate the effectiveness of HyperProto in learning discriminative features, especially in learning novel class representation that considerably boosts model performance under few-shot settings.

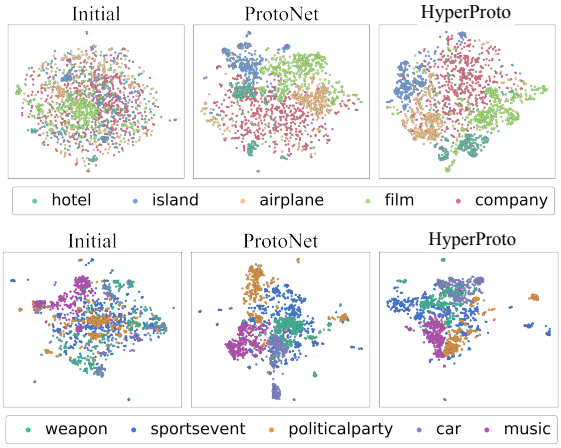

Figure 3: t-sne visualization of feature distributions. The six subfigures, from left to right, are the representations of seen data (in training set) before training, produced by ProtoNet, and produced by HyperProto ; novel data (in test set) before training, produced by ProtoNet, and produced by HyperProto . Note that even after training, the neural network has never seen the novel data and their classes.

Representation Analysis. To study if the learned representations are discriminative enough for performing few-shot classification, we illustrate the normalized distances between the learned representations and the hypersphere prototypes in Figure 4. Specifically, we randomly sample 5 classes and 25

instances (5 per class) for each dataset and produce representations for the instances and hypersphere prototypes for the classes. Then, we calculate the distance between each instance to each prototype (i.e., distance from the point to the hypersphere surface) to produce the matrix. All the values in the illustration are normalized since the absolute values may vary with the datasets. Warmer colors denote shorter distances in the illustration. The illustration shows that in all three datasets, our model could effectively learn discriminative representations and achieve stable metric-based classification.

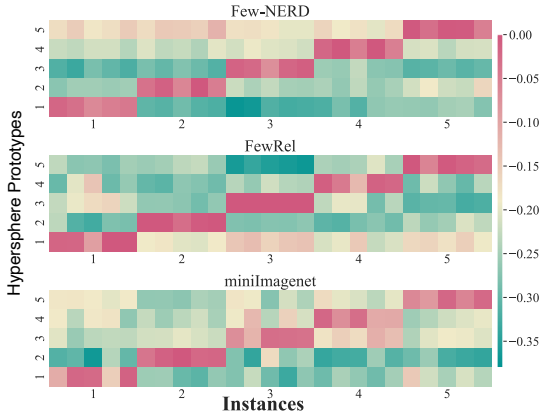

Figure 4: Normalized distances from instances to hypersphere prototypes. Horizontal axis: hypersphere prototypes of 5 classes. Vertical axis: 5 instances per class.

In order to further analyze the representations produced by HyperProto , we study the similarities of randomly sampled instance embeddings. We randomly select 4×5 classes and 5 instances per class in FEW-NERD, FewRel and *mini*ImageNet, respectively. As illustrated in Figure 5, each subfigure is a 25×25 matrix based on 5 classes. We calculate the cosine similarities of these embeddings and observe clear intra-class similarity and inter-class distinctiveness. This result confirms the robustness of our model since all the classes and instances are sampled randomly.

5 Related Work

This work is related to studies of meta-learning, whose primary goal is to quickly adapt deep neural models to new tasks with a few training examples (Hospedales et al., 2020). To this end, two branches of studies are proposed: optimization-based methods and metric-based methods. The optimization-based studies (Finn et al., 2017; Franceschi et al., 2018; Ravi and Beatson, 2018) regard few-shot learning as a bi-level optimization process, where a global optimization is conducted to learn a good initialization across various tasks, and a local optimization quickly adapts the initialization parameters to specific tasks by a few steps of gradient descent.

Compared to the mentioned studies, our work is more related to the metric-based meta-learning approaches (Vinyals et al., 2016; Snell et al., 2017; Satorras and Estrach, 2018; Sung et al., 2018), whose general idea is to learn to measure the similarity between representations and find the closest labeled example (or a derived prototype) for an unlabeled example. Typically, these methods learn a measurement function during episodic optimization. More specifically, we inherit the spirit of using prototypes to abstractly represent class-level information, which could be traced back to cognitive science (Reed, 1972; Rosch et al., 1976; Nosofsky, 1986), statistical machine learning (Graf et al., 2009) and to the Nearest Mean Classifier (Mensink et al., 2013). In the area of deep learning, Snell et al. (2017) propose the prototypical network to exploit the average of example embeddings as a prototype to perform metric-based classification in few-shot learning. In their work, prototypes are estimated by the embeddings of instances. However, it is difficult to find a satisfying location for the prototypes based on the entire dataset. Ren et al. (2018) adapt such prototype-based networks in the semi-supervised scenario where the dataset is partially annotated. Moreover, a set of prototype-based networks are proposed concentrating on the improvements of prototype estimations and application to various downstream tasks (Allen et al., 2019; Gao et al., 2019a; Li et al., 2019b; Pan et al., 2019; Seth et al., 2019; Ding et al., 2021b; Li et al., 2020c; Wertheimer and Hariharan, 2019; Xie et al., 2022; Zhang et al., 2020a). We discuss our method within the context of other prototype-enhanced methods in Section D.1. There has also been a growing body of work that considers the few-shot problem from multiple perspectives, bringing new thinking to the field (Tian et al., 2020; Yang et al., 2021; Laenen and Bertinetto, 2021; Zhang et al., 2020b; Wang et al., 2021; Das et al., 2021; Wertheimer et al., 2021; Ding et al., 2021a; Cui et al., 2022; Hu et al., 2022).There has also been a series of works that embed prototypes into a non-Euclidean output space (Mettes et al., 2019; Keller-Ressel, 2020; Atigh et al., 2021).

Figure 5: Representation similarity matrix produced by HyperProto on FEW-NERD, FewRel and *mini*ImageNet. Each row illustrates 20 classes and 100 instances in one dataset. Each subfigure contains 5 classes and 25 instances. Each unit denotes the cosine similarity of two embeddings, and each 5×5 cell indicates the comparison of two classes. The units on the diagonal represent the same instance, and the 5×5 cells on the diagonal represent the same class. Warmer color means higher similarity in this illustration.

It should be noted that these studies regard hyperspheres or other non-Euclidean manifolds as a characterization of the embedding space, while our proposed method use hyperspheres to represent prototypes and conduct metric-based classification in the Euclidean space. Therefore, the focus of our proposed HyperProto is different from the above non-Euclidean prototype-based works.

6 Conclusion

This paper proposes a novel metric-based few-shot learning method, *hypersphere prototypes*. Unlike previous metric-based methods that use dense vectors to represent the class-level semantics, we use hyperspheres to enhance the capabilities of prototypes to express the intrinsic information of the data. It is simple to model a hypersphere in the embedding space and conduct metric-based classification in few-shot learning. Our approach is easy to implement and also empirically effective, we observe significant improvements to baselines and compatibility with other techniques on downstream tasks. For potential future work, such modeling could be extended to more generalized representation learning like word embeddings.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (Grant No.62276154 and No.62236004), Research Center for Computer Network (Shenzhen) Ministry of Education, Beijing Academy of Artificial Intelligence (BAAI), the Natural Science Foundation of Guangdong Province (Grant No. 2023A1515012914), Basic Research Fund of Shenzhen City (Grant No. JCYJ20210324120012033 and JSGG20210802154402007), the Major Key Project of PCL for Experiments and Applications (PCL2021A06), Overseas Cooperation Research Fund of Tsinghua Shenzhen International Graduate

School (HW2021008), and Institute Guo Qiang at Tsinghua University.

Limitations

Compared to vanilla prototypes, the advantage of HyperProto would also rely on the additional radius parameter. Under the 1-shot setting, however, hypersphere prototypes will face challenges in estimating the radius in support sets, this is because the initial radius may be biased by the randomness of sampling. When the radius is set to exactly 0, the model will resemble a traditional prototypical network. Nevertheless, although not as large as the boost in the multi-shot setting, we find that having a consistently optimizable radius parameter at the training stage in the 1-shot scenario still delivers non-trivial results and exceeds most baselines (Table 1, Table 2, Table 3). This further points to the positive influence of the added radius parameter to learning prototype representation and hints on the possible research direction in learning a transferable radius in 1-shot scenario.

References

Kelsey Allen, Evan Shelhamer, Hanul Shin, and Joshua Tenenbaum. 2019. Infinite mixture prototypes for few-shot learning. In *Proceedings of ICML*, pages 232–241. PMLR.

Mina Ghadimi Atigh, Martin Keller-Ressel, and Pascal Mettes. 2021. Hyperbolic busemann learning with ideal prototypes. *arXiv preprint arXiv:2106.14472*.

Yinbo Chen, Zhuang Liu, Huijuan Xu, Trevor Darrell, and Xiaolong Wang. 2021. Meta-baseline: Exploring simple meta-learning for few-shot learning. In *Proceedings of the ICCV*, pages 9062–9071.

Ganqu Cui, Shengding Hu, Ning Ding, Longtao Huang, and Zhiyuan Liu. 2022. Prototypical verbalizer for prompt-based few-shot tuning. In *ACL*.

Debasmit Das and C. S. George Lee. 2020. A two-stage approach to few-shot learning for image recognition. *IEEE Transactions on Image Processing*, 29:3336–3350.

Sarkar Snigdha Sarathi Das, Arzoo Katiyar, Rebecca J Passonneau, and Rui Zhang. 2021. Container: Few-shot named entity recognition via contrastive learning. *arXiv preprint arXiv:2109.07589*.

Cyprien de Lichy, Hadrien Glaude, and William Campbell. 2021. Meta-learning for few-shot named entity recognition. In *Proceedings of the 1st Workshop on Meta Learning of ACL*, pages 44–58.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of CVPR*, pages 248–255.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pages 4171–4186.

Ning Ding, Yulin Chen, Xu Han, Guangwei Xu, Pengjun Xie, Hai-Tao Zheng, Zhiyuan Liu, Juanzi Li, and Hong-Gee Kim. 2021a. Prompt-learning for fine-grained entity typing. *arXiv preprint arXiv:2108.10604*.

Ning Ding, Xiaobin Wang, Yao Fu, Guangwei Xu, Rui Wang, Pengjun Xie, Ying Shen, Fei Huang, Hai-Tao Zheng, and Rui Zhang. 2021b. Prototypical representation learning for relation extraction. In *Proceedings of ICLR*.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021c. Few-NERD: A few-shot named entity recognition dataset. In *Proceedings of ACL*, page 3198–3213.

Bowen Dong, Yuan Yao, Ruobing Xie, Tianyu Gao, Xu Han, Zhiyuan Liu, Fen Lin, Leyu Lin, and Maosong Sun. 2020. Meta-information guided meta-learning for few-shot relation classification. In *Proceedings of COLING*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of ICML*, pages 1126–1135.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. 2018. Bilevel programming for hyperparameter optimization and meta-learning. In *Proceedings of ICML*, pages 1568–1577.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019a. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of AAAI*, pages 6407–6414.

Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2019b. FewRel 2.0: Towards more challenging few-shot relation classification. In *Proceedings of EMNLP*.

Arnulf BA Graf, Olivier Bousquet, Gunnar Rätsch, and Bernhard Schölkopf. 2009. Prototype classification: Insights from machine learning. *Neural computation*, pages 272–300.

Yiluan Guo and Ngai-Man Cheung. 2020. Attentive weights generation for few shot learning via information maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13499–13508.

Jiale Han, Bo Cheng, and Wei Lu. 2021a. Exploring task difficulty for few-shot relation extraction. *arXiv preprint arXiv:2109.05473*.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao Han, Minlie Huang, et al. 2021b. Pre-trained models: Past, present and future. *AI Open*.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of EMNLP*, pages 248–255.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. In *Proceedings of CVPR*, pages 770–778.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2020. Meta-learning in neural networks: A survey. *arXiv:2004.05439*.

Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Juan-Zi Li, and Maosong Sun. 2022. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. In *ACL*.

Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. 2020. Few-shot named entity recognition: A comprehensive study. *arXiv:2012.14978*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of ICML*.

Martin Keller-Ressel. 2020. A theory of hyperbolic prototype learning. *arXiv preprint arXiv:2010.07744*.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Steinar Laenen and Luca Bertinetto. 2021. On episodes, prototypical networks, and few-shot learning. *Advances in Neural Information Processing Systems*, 34.

Aoxue Li, Weiran Huang, Xu Lan, Jiashi Feng, Zhenguo Li, and Liwei Wang. 2020a. Boosting few-shot learning with adaptive margin loss. In *Proceedings of the CVPR*, pages 12576–12584.

Jing Li, Billy Chiu, Shanshan Feng, and Hao Wang. 2020b. Few-shot named entity recognition via meta-learning. *IEEE Transactions on Knowledge and Data Engineering*.

Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. 2020c. Prototypical contrastive learning of unsupervised representations. In *Proceedings of ICLR*.

Wenbin Li, Lei Wang, Jing Huo, Yinghuan Shi, Yang Gao, and Jiebo Luo. 2020d. Asymmetric distribution measure for few-shot learning. In *Proceedings of IJCAI*, pages 2957–2963.

Wenbin Li, Jinglin Xu, Jing Huo, Lei Wang, Gao Yang, and Jiebo Luo. 2019a. Distribution consistency based covariance metric networks for few-shot learning. In *AAAI*, pages 8642–8649.

Xiao Li, Min Fang, Dazheng Feng, Haikun Li, and Jinqiao Wu. 2019b. Prototype adjustment for zero shot classification. *Signal Processing: Image Communication*, pages 242–252.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Jiang Lu, Pinghua Gong, Jieping Ye, and Changshui Zhang. 2020. Learning from very few samples: A survey. *arXiv:2009.02653*.

Puneet Mangla, Nupur Kumari, Abhishek Sinha, Mayank Singh, Balaji Krishnamurthy, and Vineeth N Balasubramanian. 2020. Charting the right manifold: Manifold mixup for few-shot learning. In *Proceedings of WACV*, pages 2218–2227.

Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. 2013. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, pages 2624–2637.

Pascal Mettes, Elise van der Pol, and Cees Snoek. 2019. Hyperspherical prototype networks. *Proceedings of NeurIPS*, pages 1487–1497.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2017. A simple neural attentive meta-learner. *arXiv:1707.03141*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563. PMLR.

Robert M Nosofsky. 1986. Attention, similarity, and the identification–categorization relationship. *Journal of experimental psychology: General*.

Yingwei Pan, Ting Yao, Yehao Li, Yu Wang, Chong-Wah Ngo, and Tao Mei. 2019. Transferrable prototypical networks for unsupervised domain adaptation. In *Proceedings of CVPR*, pages 2239–2247.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of NeurIPS*, pages 8026–8037.

Hao Peng, Tianyu Gao, Xu Han, Yankai Lin, Peng Li, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2020. Learning from Context or Names? An Empirical Study on Neural Relation Extraction. In *Proceedings of EMNLP*, pages 3661–3672.

Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. 2018. Few-shot image recognition by predicting parameters from activations. In *Proceedings of the CVPR*, pages 7229–7238.

Meng Qu, Tianyu Gao, Louis-Pascal Xhonneux, and Jian Tang. 2020. Few-shot relation extraction via bayesian meta-learning on relation graphs. In *International conference on machine learning*, pages 7867–7876. PMLR.

Sachin Ravi and Alex Beatson. 2018. Amortized bayesian meta-learning. In *Proceedings of ICLR*.

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *Proceedings of ICLR*.

Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. 2019. Few-shot learning with embedded class models and shot-free meta training. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 331–339.

Stephen K Reed. 1972. Pattern recognition and categorization. *Cognitive psychology*, 3(3):382–407.

Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. 2018. Meta-learning for semi-supervised few-shot classification. In *Proceedings of ICLR*.

Eleanor Rosch, Carolyn B Mervis, Wayne D Gray, David M Johnson, and Penny Boyes-Braem. 1976. Basic objects in natural categories. *Cognitive psychology*, pages 382–439.

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. 2018. Meta-learning with latent embedding optimization. *arXiv:1807.05960*.

Victor Garcia Satorras and Joan Bruna Estrach. 2018. Few-shot learning with graph neural networks. In *Proceedings of ICLR*.

Harshita Seth, Pulkit Kumar, and Muktabh Mayank Srivastava. 2019. Prototypical metric transfer learning for continuous speech keyword spotting with limited training data. In *Proceedings of SOCO*. Springer.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of NeurIPS*.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of ACL*.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of CVPR*, pages 1199–1208.

Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B. Tenenbaum, and Phillip Isola. 2020. Rethinking few-shot image classification: A good embedding is all you need? In *Proceedings of ECCV*, pages 266–282.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Proceedings of NeurIPS*, pages 3630–3638.

Jixuan Wang, Kuan-Chieh Wang, Frank Rudzicz, and Michael Brudno. 2021. Grad2task: Improved few-shot text classification using gradients for task representation. *Advances in Neural Information Processing Systems*, 34.

Peiyi Wang, Runxin Xu, Tianyu Liu, Qingyu Zhou, Yunbo Cao, Baobao Chang, and Zhifang Sui. 2022. An enhanced span-based decomposition method for few-shot sequence labeling. In *Proceedings of NAACL*.

Xiaozhi Wang, Xu Han, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018. Adversarial multi-lingual neural relation extraction. In *Proceedings of COLING*, pages 1156–1166.

Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. 2019. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv:1911.04623*.

Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. 2010. Caltech-ucsd birds 200.

Davis Wertheimer and Bharath Hariharan. 2019. Few-shot learning with localization in realistic settings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6558–6567.

Davis Wertheimer, Luming Tang, and Bharath Hariharan. 2021. Few-shot classification with feature map reconstruction networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8012–8021.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Huggingface's transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP: System Demonstrations*, page 38–45.

Jiangtao Xie, Fei Long, Jiaming Lv, Qilong Wang, and Peihua Li. 2022. Joint distribution matters: Deep brownian distance covariance for few-shot classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7972–7981.

Shuo Yang, Lu Liu, and Min Xu. 2021. Free lunch for few-shot learning: Distribution calibration. In *Proceedings of ICLR*.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *Proceedings of EMNLP*.

Zhi-Xiu Ye and Zhen-Hua Ling. 2019. Multi-level matching and aggregation network for few-shot relation classification. In *Proceedings of ACL*, pages 2872–2881.

Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. In *BMVC*, pages 87.1–87.12.

Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. 2020a. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12203–12213.

Jian Zhang, Chenglong Zhao, Bingbing Ni, Minghao Xu, and Xiaokang Yang. 2019. Variational few-shot learning. In *Proceedings of ICCV*, pages 1685–1694.

Manli Zhang, Jianhong Zhang, Zhiwu Lu, Tao Xiang, Mingyu Ding, and Songfang Huang. 2020b. Iept: Instance-level and episode-level pretext tasks for few-shot learning. In *International Conference on Learning Representations*.

Peiyuan Zhang and Wei Lu. 2022. Better few-shot relation extraction with label prompt dropout. *arXiv preprint arXiv:2210.13733*.

A Generalizations of Our Method

We have introduced the mechanisms of hypersphere prototypes in Euclidean space. In this section, we generalize this idea to construct variants with other measurements.

Cone-like HyperProto. Cosine similarity is a commonly used measurement in machine learning. Assume all the data points are distributed on a unit ball, and we use the cosine value of the intersection angle to measure the similarity of the two embeddings. While keeping the intuition of the modeling of hypersphere prototypes in mind, we introduce an additional angle parameter ϵ. We use $\theta_{\boldsymbol{a},\boldsymbol{b}}$ to denote the intersection angle of the two embeddings $\boldsymbol{a}$ and $\boldsymbol{b}$. In this way, the center point $\boldsymbol{z}$ and the angle ϵ could conjointly construct a cone-like prototype,

$$\mathcal{B}^d(\boldsymbol{z}, f_\phi, \epsilon) := \{f_\phi(x) \in \mathbb{R}^D : d(f_\phi(\boldsymbol{x}), \boldsymbol{z}) \geq \cos\epsilon\}, \tag{6}$$

where $d(f_\phi(\boldsymbol{x}), \boldsymbol{z}) = \cos(\theta_{f_\phi(\boldsymbol{x}),\boldsymbol{z}})$. The measurement $\mathcal{M}(\cdot)$ is defined as the cosine of the angle between the instance embedding and the nearest point on the border of the prototype,

$$\mathcal{M}(\boldsymbol{x}, \mathcal{B}) = \begin{cases} -\cos(\theta_{f_\phi(\boldsymbol{x}),\boldsymbol{z}} - \epsilon), \theta_{f_\phi(\boldsymbol{x}),\boldsymbol{z}} \geq |\epsilon|, \\ -1, \theta_{f_\phi(\boldsymbol{x}),\boldsymbol{z}} < |\epsilon|. \end{cases} \tag{7}$$

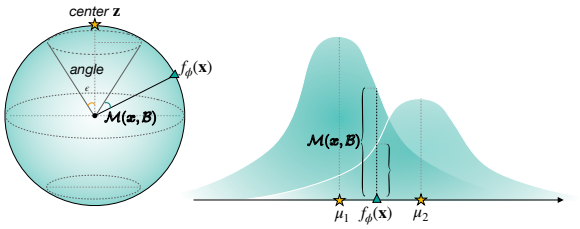

Figure 6: Two variants according to different measurements. The left is the cone-like modeling with cosine similarities, and the right is the Gaussian modeling from the probability perspective.

Similar to the vanilla hypersphere prototypes, $\boldsymbol{z}$ and ϵ need to participate in the learning process for optimization, and the angle $\theta_{\boldsymbol{x},\boldsymbol{z}}$ is computed by the inverse trigonometric function,

$$\theta_{f_\phi(\boldsymbol{x}),\boldsymbol{z}} = \arccos \frac{f_\phi(\boldsymbol{x})^T \boldsymbol{z}}{||f_\phi(\boldsymbol{x})|| \cdot ||\boldsymbol{z}||}. \tag{8}$$

The prediction for a training example is also based on the softmax over the measurements to the prototypes like Eq. 5. Note that as shown in Eq. 7, the measurement becomes -1 when a data point is "inside" the cone-like prototype. Then it is hard to make a prediction when an embedding is inside two prototypes. It thus requires that the prototypes do not intersect with each other, that is, to guarantee the angle between two center points is larger than the sum of their own parameter angles,

$$\mathcal{L}_{\text{dis}} = \frac{1}{N} \sum_{i,j} \max((|\epsilon_i| + |\epsilon_j|) - \theta_{\boldsymbol{z}_i,\boldsymbol{z}_j}, 0). \tag{9}$$

Therefore, the final loss function is $\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{dis}}$.

Gaussian HyperProto. From the probability perspective, each class can be characterized by a distribution in a multi-dimensional feature space. The measurement of a query sample to the n-th class can thus be represented by the negative log likelihood of $f_\phi(\boldsymbol{x})$ belonging to $\mathcal{B}_n$. In line with other works (Zhang et al., 2019; Li et al., 2020d), we can simply assume each class subjects to a Gaussian distribution $\mathcal{B}_n \sim \mathcal{N}(\boldsymbol{\mu}_n, \Sigma_n)$. To reduce the number of parameters and better guarantee the positive semi-definite feature, we can further restrict the covariance matrix to be a diagonal matrix such that $\Sigma_n = \sigma_n^2 I$. Then the measurement becomes

$$\begin{aligned} \mathcal{M}(\boldsymbol{x}, \mathcal{B}_n) &= -\log p(f_\phi(\boldsymbol{x}); \mathcal{B}_n) \\ &= \frac{||f_\phi(\boldsymbol{x}) - \boldsymbol{\mu}_n||_2^2}{2\sigma_n^2} + \log((2\pi)^{\frac{d}{2}} |\sigma_n|^d) \\ &= \frac{||f_\phi(\boldsymbol{x}) - \boldsymbol{\mu}_n||_2^2}{2\sigma_n^2} + d\log|\sigma_n| + \delta, \end{aligned} \tag{10}$$

where $\delta = \frac{d}{2}\log 2\pi$. The probability of target class given a query sample can be calculated by Eq. 4 in the same fashion: $p(y = n|\boldsymbol{x}) = \frac{p(f_\phi(\boldsymbol{x}); \mathcal{B}_n)}{\sum_{n'} p(f_\phi(\boldsymbol{x}); \mathcal{B}_{n'})}$. Note that the derived form of the equation is the same as directly calculating the probability of $p(y = n|\boldsymbol{x})$ under a uniform prior distribution of $p(y)$. Comparing with pure probabilistic approaches, such as variational inference that treats $\mathcal{B}$ as hidden variables and models $p(\mathcal{B}|\mathcal{S})$ and $p(\mathcal{B}|\mathcal{S}, \boldsymbol{x})$ with neural network (Zhang et al., 2019), under the framework of § 3.2, $\mathcal{B}$ is explicitly parameterized and optimized for each class during training. Moreover, comparing Eq. 10 with Eq. 2, it can be observed that when formalizing $\mathcal{B}$ as a distribution, instead of as a bias term, the original radius parameter (now the variance) functions as a scaling factor on Euclidean distance.

B Experimental Details

This section reports the experimental details of all three tasks in our evaluation. All the experiments

are conducted on NVIDIA A100 and V100 GPUs with CUDA. The main experiments are conducted on three representative tasks in NLP and CV, which are few-shot named entity recognition (NER), relation extraction (RE), and image classification. The experimental details will be presented in the following sections.

B.1 Experimental Details for Few-shot Named Entity Recognition

We assess the effectiveness of hypersphere prototypes on NLP, specifically, the first task is few-shot named entity recognition (NER) and the dataset is FEW-NERD (Ding et al., 2021c)[2]. NER aims at locating and classifying named entities (real-world objects that can be denoted with proper names) given an input sentence, which is typically regarded as a sequence labeling task. Given an input sentence ""Bill Gates is a co-founder of the American multinational technology corporation Microsoft", an named entity recognition system aims to locate the named entities (*Bill Gates*, *Microsoft*) and classify them into specific types. Conventional schema uses coarse-grained labels such that *Person* for *Bill Gates* and *Organization* for *Microsoft*. In more advanced schema like Few-NERD, models are asked to give more specific entity types, for example, *Person-Entrepreneur* for *Bill Gates* and *Organization-Company* for *Microsoft*.

Different from typical instance-level classification, few-shot NER is a sequence labeling task, where labels may share structural correlations. NER is the first step in automatic information extraction and the construction of large-scale knowledge graphs. Quickly detecting fine-grained unseen entity types is of significant importance in NLP. To capture the latent correlation, many recent efforts in this field use large pre-trained language models (Han et al., 2021b) like BERT (Devlin et al., 2019) as backbone model and have achieved remarkable performance. The original prototypical network has also been applied to this task (Li et al., 2020b; Huang et al., 2020; de Lichy et al., 2021).

Dataset. The experiment is run on FEW-NERD dataset (Ding et al., 2021c). It is a large-scale NER dataset containing over 400,000 entity mentions, across 8 coarse-grained types and 66 fine-grained types, which makes it an ideal dataset for few-shot learning. It has been shown that existing methods including prototypes are not effective enough on this dataset.

Baselines. We choose the following baselines:

- **Proto**(Snell et al., 2017) is the main baseline, which adapts the prototypical network on few-shot named entity recognition.
- **NNShot** (Yang and Katiyar, 2020) is a token-level metric-based method that is specifically designed for few-shot labeling.
- **StructShot** (Yang and Katiyar, 2020) adds a CRF layer in inference and further boosts performance of NNShot.
- **CONTaiNER** (Das et al., 2021) uses a pre-trained backbone and further finetunes on the few-shot data.
- **ESD** (Wang et al., 2022) uses attention mechanism to learn prototype representation.

Implementation Details. We run experiments under four settings on the two released benchmarks, FEW-NERD (INTRA) and FEW-NERD (INTER). Specifically, we use uncased BERT-base as the backbone encoder and 1e-4 as the encoder learning rate. As for learning rate for radius parameter, we use 20.0 for HyperProto+Prompt 10-way-5-shot INTER setting and 10.0 for other settings. AdamW is used as the BERT encoder optimizer, and Adam (Kingma and Ba, 2017) is used to optimize prototype radius. The batch size is set to 2 across all settings. All models are trained for 10000 steps and validated every 1000 steps. The results are reported on 5000 steps of the test episode. For each setting, we run the experiment with 3 different random seeds and report the average F1-score the standard error. We use PyTorch (Paszke et al., 2019) and huggingface transformers (Wolf et al., 2020) to implement the backbone encoder $\text{BERT}_{\text{base}}$.

B.2 Experimental Details for Few-shot Relation Extraction

The other common NLP task is relation extraction (RE), which aims at correctly classifying the relation between two given entities in a sentence. For example, given an input sentence with marked entities "[*Bill Gates*] is a co-founder of the American multinational technology corporation [*Microsoft*]", the relation extraction system aims to give the relationship between *Bill Gates* and *Microsoft*. This is a fundamental task in information extraction. RE is an important form of

[2]FEW-NERD is distributed under CC BY-SA 4.0 license

learning structured knowledge from unstructured text. We use FewRel (Han et al., 2018)[3] and FewRel 2.0 (Gao et al., 2019b) as the datasets.

Dataset. We adopt the FewRel dataset (Han et al., 2018; Gao et al., 2019b), a relation extraction dataset specifically designed for few-shot learning. FewRel has 100 relations with 700 labeled instances each. The sentences are extracted from Wikipedia and the relational entities are obtained from Wikidata. FewRel 1.0 (Han et al., 2018) is released as a standard few-shot learning benchmark. FewRel 2.0 (Gao et al., 2019b) adds domain adaptation task and NOTA task on top of FewRel 1.0 with the newly released test dataset on PubMed corpus.

Baselines. In addition to the main baseline, prototypical network (Snell et al., 2017), we also choose the following few-shot learning methods as the baselines in relation extraction.

- **Proto-HATT** (Gao et al., 2019a) is a neural model with hybrid prototypical attention.
- **MLMAN** (Ye and Ling, 2019) is a multi-level matching and aggregation network for few-shot relation classification. Note that Proto-HATT and MLMAN are not model-agnostic.
- **GNN** (Satorras and Estrach, 2018) is a meta-learning model with a graph neural network as the prediction head.
- **SNAIL** (Mishra et al., 2017) is a meta-learning model with attention mechanisms.
- **Meta Net** (Munkhdalai and Yu, 2017) is a classical meta-learning model with meta information.
- **Proto-ADV** (Gao et al., 2019b) is a prototype-based method enhanced by adversarial learning.
- **BERT-pair** (Gao et al., 2019b) is a strong baseline that uses BERT for few-shot relation classification.
- **MTB** (Soares et al., 2019) pre-trains on sentence pairs constructed by entity linking technique.
- **REGRAB** (Qu et al., 2020) uses external knowledge in KBs.
- **CP** (Peng et al., 2020) pre-trains a relation classification model with contrastive learning.
- **MIML** (Dong et al., 2020) uses additional semantic information of each class.
- **COL** (Ding et al., 2021b) assumes that prototypes distribute uniformly on a unit ball surface and pre-trains the prototype representation.
- **HCRP** (Han et al., 2021a) uses contrastive learning to learn better prototype representations, while focusing more on hard cases.
- **LPD** (Zhang and Lu, 2022) adopts relation description as prompt and randomly drops labels in the support set to derive better class prototype.

Implementation Details. The experiments are conducted on FewRel 1.0 and FewRel 2.0 domain adaptation tasks. For FewRel 1.0, we follow the official splits in Han et al. (2018). For FewRel2.0, we follow Gao et al. (2019b), training the model on wiki data, validating on SemEval data, and testing on the PubMed data. We use the same CNN structure and BERT as encoders. The learning rate for hypersphere prototype radius is 0.1 and 0.01 for CNN and BERT encoder, respectively. Adam (Kingma and Ba, 2017) is used as radius optimizer. We train the model for 10000 steps, validate every 1000 steps, and test for 5000 steps. The other hyperparameters are the same as in the original paper.

B.3 Experimental Details for Few-shot Image Classification

Image classification is one of the most classical tasks in few-shot learning research. Seeking a better solution for few-shot image classification is beneficial in two ways: (1) to alleviate the need for data augmentation, which is a standard technique to enrich the labeled data by performing transformations on a given image; (2) to facilitate the application where the labeled image is expensive. To demonstrate the effectiveness of HyperProto, we also conduct experiment on few-shot image classification with *mini*ImageNet (Vinyals et al., 2016) dataset. The results of the experiment are shown in C.

Dataset. *mini*ImageNet (Vinyals et al., 2016) is used as a common benchmark for few-shot learning. The dataset is extracted from the full ImageNet dataset (Deng et al., 2009), and consists of 100 randomly chosen classes, with 600 instances each. Each image is of size $3\times84\times84$. We follow the split in Ravi and Larochelle (2017) and use 64, 16, and 20 classes for training, validation, and testing.

Baselines. The baselines we choose are as follows:

- **Prototypical network** (Snell et al., 2017) uses the vanilla prototypes as representations and is our main baseline.

[3] FewRel is distributed under MIT license

- **IMP** (Allen et al., 2019) is a prototype-enhanced method that models an infinite mixture of prototypes for few-shot learning.
- **CovaMNet** (Li et al., 2019a) is a few-shot learning method that uses covariance to model the distribution information to enhance few-shot learning performance.
- **SNAIL** (Mishra et al., 2017) is an attention-based classical meta-learning method.
- **Variational FSL** (Zhang et al., 2019) is a variational Bayesian framework for few-shot learning, which contains a pre-training stage.
- **Activation to Parameter** (Qiao et al., 2018) predicts parameters from activations in few-shot learning.
- **LEO** (Rusu et al., 2018) optimizes latent embeddings for few-shot learning.
- **TRAML** (Li et al., 2020a) uses adaptive margin loss to boost few-shot learning, and Prototypes + TRAML is a strong baseline in recent years.
- **SimpleShot** (Wang et al., 2019) combines vanilla prototypes with simple feature transformation.
- **AWGIM** (Guo and Cheung, 2020) follows the idea of LEO (Rusu et al., 2018) but generates different classification representations for different queries by maximizing mutual information.

Implementation Details. The experiments are conducted on 5 way 1 shot and 5 way 5 shot settings. To ensure validity and fairness, we implement hypersphere prototypes with various backbone models including CNN, ResNet-12, and WideResNet (Zagoruyko and Komodakis, 2016) to make it comparable to all baseline results, and we also re-run some of the baselines including prototypical network (Snell et al., 2017), infinite mixture prototypes (Allen et al., 2019), and CovaMNet (Li et al., 2019a) under our settings based on their original code. Other baseline results are taken from the original paper. Each model is trained on 10,000 randomly sampled episodes for 30∼40 epochs and tested on 1000 episodes. The result is reported with 95% confidence interval. Note that both ResNet-12 and WideResNet (Zagoruyko and Komodakis, 2016) are pretrained on the training data, where the pretrained ResNet-12 is identical to Chen et al. (2021) and the pretrained WideResNet follows Mangla et al. (2020). The CNN structure is the same as Snell et al. (2017), which is composed of 4 convolutional blocks each with a 64-filter 3×3 convolution, a batch normalization layer (Ioffe and Szegedy, 2015), a ReLU nonlinearity, and a 2×2 max-pooling layer. We use SGD optimizer for the encoder and Adam (Kingma and Ba, 2017) optimizer for the prototype radius. The learning rate for the backbone model is 1e-3. The learning rate for radius is manually tuned and the reported result in Table 3 has a learning rate of 10. For cone-like and gaussian prototypes, we use 1e-1 and 1e-3. At the training stage, the prototype center is re-initialized at each episode as the mean vector of the support embeddings.

C Additional Experiments and Analysis

This section provides additional experiments and analysis. We first present results on image classification, then we compare generalized HyperProto. We also experiment on cross-dataset setting and provide analysis on impact of training data volume on model performance and instance-level representation.

C.1 Few-shot Image Classification.

Table 3 shows the result on *mini*ImageNet few-shot classification under 2 settings. HyperProto substantially outperforms the primary baselines in most settings, displaying their ability to model the class distribution of images. We observe that compared to NLP, image classification results are more stable both for vanilla prototypes and hypersphere prototypes. This observation may indicate the difference in encoding between the two technologies. Token representations in BERT are contextualized and changeable around different contexts, yet the image representation produced by deep CNNs aims to capture the global and local features thoroughly. Under the 5-way 5-shot setting, the improvements of HyperProto are significant. The effectiveness of our method is also demonstrated by the comparisons with other previous few-shot learning methods with the same backbones. In particular, HyperProto yields the best results of all the compared methods with the WideResNet (Zagoruyko and Komodakis, 2016) backbone, suggesting that the expressive capability of hypersphere prototypes can be enhanced with a more powerful encoder. Compared to the 5-shot setting, our model improves

mediocrely in the 1-shot setting of ConvNet and ResNet-12 (He et al., 2015). The phenomenon is consistent with the intuition that more examples would be more favorable to the learning of radius. We further analyze the dynamics of the radius of our method in Appendix 4.3.

C.2 Comparison of Generalized HyperProto.

To further compare the variants of our approach, we conduct experiments for cone-like and gaussian HyperProto with WideResNet-28-10 on *mini*ImageNet as well. Table 4 presents results across three measurement settings. Although the two variants do not perform better than our main method, they still considerably outperform many baselines in Table 3. While the three models' performance is close under the 1-shot setting, the Cone HyperProto model performs worse in the 5-shot setting. It could be attributed to unsatisfying radius learning. It is found that the Cone Hyperproto model is susceptible to radius learning rate and is prone to overfitting.

C.3 Cross-dataset Few-shot Learning

We also conduct experiments on the more difficult cross-dataset setting. Specifically, the model trained on *mini*Imagenet is tested on the CUB dataset (Welinder et al., 2010) under the 5-way 5-shot setting. We use ResNet-12 (RN-12) (He et al., 2015) as the backbone in our experiment. Table 5 shows the results compared with several baselines. It can be seen that HyperProto outperforms the baselines by a large margin even with less powerful encoder (RN-12), indicating the ability to learn representations that are transferrable to new domains. The results also echo the performance of HyperProto for the cross-domain relation extraction in Table 2.

C.4 Impact of Number of Shots

We conduct additional experiments on FEW-NERD (INTRA) 5-way setting with 10, 15, 20 shots. Since NNShot becomes too memory-intensive to run when shot reaches 15, we provide results on Proto and HyperProto . Figure 7 shows both models perform better when more data are available, while HyperProto performs consistently better than vanilla prototypes.

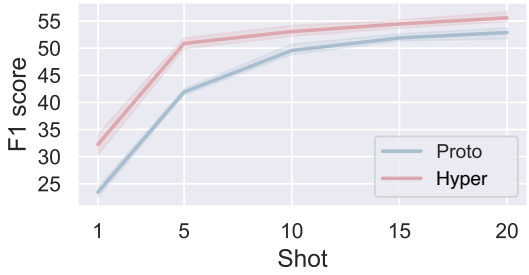

Figure 7: Impact of shot number on model performance for FEW-NERD (INTRA) 5-way setting.

D Discussion

This section discusses related prototype-based methods in detail, and the broader impact of our work.

D.1 Other Prototype-enhanced Methods

In this section, we discuss the difference between hypersphere prototypes with four prototype-enhanced methods in few-shot learning: infinite mixture prototypes (Allen et al., 2019), CovaMNet (Li et al., 2019a), Variational Few-Shot Learning (Zhang et al., 2019), and Two-Stage (Das and Lee, 2020).

Infinite mixture prototypes (Allen et al., 2019) model each class as an indefinite number of clusters and the prediction is obtained by computing and comparing the distance to the nearest cluster in each class. This method is an intermediate model between prototypes and the nearest neighbor model, whereas hypersphere prototypes alleviate the over-generalization problem of vanilla prototypes with a single additional parameter that turns a single point modeling into a hypersphere. The essential prototype-based feature of hypersphere prototypes allows faster computation and easier training.

CovaMNet (Li et al., 2019a) calculates local variance for each class based on support samples and conducts metric-based learning via covariance metric, which basically evaluates the cosine similarity between query samples and the eigenvectors of the local covariance matrix. To ensure the non-singularity of the covariance matrix, the feature of each sample is represented with a matrix, generated by a number of local descriptors, with each extracting a feature vector. Compared to hypersphere prototypes, both methods attempt to model more variance based on vanilla prototypes, while the idea of hypersphere prototypes is more straightforward and requires fewer parameters. On the other hand, the multi-channel features adopted by CovaMNet are less natural for NLP tasks.

Variational Few-Shot Learning (Zhang et al.,

Model	Backbone	*mini*ImageNet		
		5 way 1 shot	5 way 5 shot	Average
Infinite Mixture Prototypes† (Allen et al., 2019)	ConvNet	33.30 ± 0.71	65.88 ± 0.71	49.59
ProtoNet† (Snell et al., 2017)	ConvNet	46.44 ± 0.60	63.72 ± 0.55	55.08
CovaMNet† (Li et al., 2019a)	ConvNet	51.83 ± 0.64	65.65 ± 0.77	58.74
HyperProto (Ours)	ConvNet	50.21 ± 0.31	66.48 ± 0.71	58.35
SNAIL (Mishra et al., 2017)	ResNet-12	55.71 ± 0.99	68.88 ± 0.92	62.30
ProtoNet† (Snell et al., 2017)	ResNet-12	53.81 ± 0.23	75.68 ± 0.17	64.75
Variational FSL (Zhang et al., 2019)	ResNet-12	61.23 ± 0.26	77.69 ± 0.17	69.46
Prototypes + TRAML (Li et al., 2020a)	ResNet-12	60.31 ± 0.48	77.94 ± 0.57	69.13
HyperProto (Ours)	ResNet-12	59.65 ± 0.62	78.24 ± 0.47	68.95
ProtoNet† (Snell et al., 2017)	WideResNet-28-10	59.09 ± 0.64	79.09 ± 0.46	69.09
Activation to Parameter (Qiao et al., 2018)	WideResNet-28-10	59.60 ± 0.41	73.74 ± 0.19	66.67
LEO (Rusu et al., 2018)	WideResNet-28-10	61.76 ± 0.08	77.59 ± 0.12	69.68
SimpleShot (Wang et al., 2019)	WideResNet-28-10	63.50 ± 0.20	80.33 ± 0.14	71.92
AWGIM (Guo and Cheung, 2020)	WideResNet-28-10	63.12 ± 0.08	78.40 ± 0.11	70.76
HyperProto (Ours)	WideResNet-28-10	**63.78 ± 0.63**	**81.29 ± 0.46**	**72.54**

Table 3: Accuracies with 95% confidence interval on 1000 test episodes of HyperProto and baselines on *mini*ImageNet. Results with † are obtained by re-running the original code unider our experimental settings. Other baselines are reported in their original papers.

Methods	*mini*ImageNet	
	5 way 1 shot	5 way 5 shot
Cone HyperProto	62.43 ± 0.63	76.03 ± 0.50
Gaussian HyperProto	60.34 ± 0.64	80.43 ± 0.45
HyperProto	**63.78 ± 0.63**	**81.29 ± 0.46**

Table 4: Accuracies with 95% confidence interval of generalized HyperProto on *mini*ImageNet.

Methods	Backbone	5 way 5 shot
	*mini*ImageNet → CUB	
MatchingNet (Vinyals et al., 2016)	RN-12	53.07 ± 0.74
ProtoNet (Snell et al., 2017)	RN-12	62.02 ± 0.70
MAML (Finn et al., 2017)	RN-18	52.34 ± 0.72
RelationNet (Sung et al., 2018)	RN-18	57.71 ± 0.73
Baseline++ (Chen et al., 2021)	RN-18	62.04 ± 0.76
SimpleShot (Wang et al., 2019)	RN-18	65.56 ± 0.70
HyperProto (Ours)	RN-12	63.22 ± 0.77

Table 5: Results on cross-dataset classification.

2019) tackles the few-shot learning problem under a bayesian framework. In order to improve single point-based estimation, a class-specific latent variable representing the class distribution is introduced and is assumed to be Gaussian. The method parameterizes the mean and variance of the latent variable distribution with neural networks that take the feature of a single instance as input. The learning and inference processes are both conducted on the latent variable level. The method adopts variational inference and is built on modeling distribution as a latent variable, where the metric calculation highly relies on the Gaussian assumption. Hypersphere prototypes, on the other hand, model the distribution with a center vector and a radius parameter in the actual embedding space, which is more tangible and easier to calculate. It is worth noting that this work also points that a single embedding is insufficient to represent a class, and samples the prototype from a high-dimensional distribution. This is actually similar to our starting point, the difference is that our approach turns out to consider the problem from the geometric point of view based on the original embedding space, and proves that such simple geometric modeling could be very efficient in the few-shot scenarios.

Two-Stage Approach first trains feature encoder and variance estimator on training data in an episodic manner with extracted absolute and relative features. Then in the second stage, training data are split into "novel" class, and base class, novel class prototypes are learned from both sample mean and base class features. The classification is carried out with integrated prototypes. This method improves on vanilla prototypes by extracting more features and combining information from base classes, but still follows single-point-based metric learning. Our approach extends a single point to a hypersphere in the embedding space and

therefore, better captures within-class variance.

D.2 Broader Impact

Our method focuses on the method of few-shot learning, which enables machine learning systems to learn with few examples, and could be applied to many downstream applications. The technique itself does not have a direct negative impact, i.e., its impact stems primarily from the intent of the user, and there may be potential pitfalls when the method is applied to certain malicious applications.

E K∼2K Sampling for Few-NERD

In the sequence labeling task FEW-NERD, the sampling strategy is slightly different from other classification tasks. Because in the named entity recognition, each token in a sequence is asked to be labeled as if it is a part of a named entity. And the context is crucial for the classification of each entity, thus the examples are sampled at the sequence level. Under this circumstance, it is difficult to operate accurate N way N shot sampling. Ding et al. (2021c) propose a greedy algorithm to conduct N way $K \sim 2K$ shot sampling for the FEW-NERD dataset. We follow the strategy of the original paper (Ding et al., 2021c) and report it in Algorithm 2.

Algorithm 2: Greedy N-way $K\sim2K$-shot sampling algorithm for FEW-NERD

Input: Dataset $\mathcal{X}$, Label set $\mathcal{Y}$, N, K
Output: output result
$\mathcal{S} \leftarrow \varnothing$; `// Init the support set`
`// Init the count of entity types`
for $i = 1$ *to* N **do**
    Count$[i] = 0$;
repeat
    Randomly sample $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}$;
    Compute |Count| and Count$_i$ after update ;
    if $|Count| > N$ *or* $\exists Count[i] > 2K$ **then**
        Continue ;
    else
        $\mathcal{S} = \mathcal{S} \bigcup (\boldsymbol{x}, \boldsymbol{y})$;
        Update Count$_i$;
until $Count_i \geq K$ *for* $i = 1$ *to* N;

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Page 9 (Limitations)

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4

☑ B1. Did you cite the creators of artifacts you used?
Section 4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Section 4

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Appendix B

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix B

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix B

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix B

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.