One Representation to Rule Them All: Identifying Out-of-Support Examples in Few-shot Learning with Generic Representations

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

The field of few-shot learning has made remarkable strides in developing powerful models that can operate in the small data regime. Nearly all of these methods assume every unlabeled instance encountered will belong to a handful of known classes for which one has examples. This can be problematic for real-world use cases where one routinely finds ‘none-of-the-above’ examples. In this paper we describe this challenge of identifying what we term ‘out-of-support’ (OOS) examples. We describe how this problem is subtly different from out-of-distribution detection and describe a new method of identifying OOS examples within the Prototypical Networks framework using a fixed point which we call the generic representation. We show that our method outperforms other existing approaches in the literature as well as other approaches that we propose in this paper. Finally, we investigate how the use of such a generic point affects the geometry of a model’s feature space.

1 Introduction

Over the past decade, deep learning-based methods have achieved state-of-the-art performance in a range of applications including image recognition, speech recognition, and machine translation. There are many problems however, where deep learning’s utility remains limited because of its need for large amounts of labeled data. The field of few-shot learning [26] aims to develop methods for building powerful machine learning models in the limited-data regime.

The common paradigm in few-shot learning is to assume that for each unlabeled instance, one has at least one labeled example belonging to the same class. At inference time then, classification of an unlabeled example x simply involves determining which of a fixed number of known classes x is most likely to belong to. In real-world problems on the other hand, it is frequently the case that one does not have labeled examples of every possible class that has support in a data distribution. This is particularly true in science and medical applications where it is time and cost prohibitive to have a subject matter expert sift through an entire dataset and identify all classes therein. Establishing methods of detecting whether or not unlabeled input belongs to any known class is thus critical to making few-shot learning an effective tool in a broad range of applications.

We define a datapoint to be *out-of-support (OOS)* if it does not belong to a class for which we have labeled examples, but was still drawn from the same data distribution as the labeled examples we have. We call the problem of identifying such instances the *out-of-support detection problem.* As we explain in Section 2.3, OOS detection resembles, but is distinct from, out-of-distribution (OOD) detection where one attempts to identify examples which were drawn from a different data distribution entirely, (see Figure 1 for an illustration of the difference between these two types of problems). To our knowledge the OOS detection problem was first articulated in the literature only recently in [25], where two algorithms were proposed within the metric-based few-shot setting.

In this paper we describe a new approach to OOS detection which we call *Generic Representation Out-Of-Support (GROOS) Detection*. The name is inspired by the concept of generic points in algebraic geometry, which are points for which all generic properties of a geometric object are true [6]. Our method uses a so-called *generic representation* to represent the data distribution as a whole but no individual class in particular. Like the methods proposed in [25], our method can be adapted to work with a range of metric-based few-shot models. For simplicity, in this paper we focus on a Prototypical Networks [21] setting where the generic representation is simply a point in feature space. To predict whether an unlabeled instance q is OOS or not, one compares the distances from the encoding of q to each class representation and the generic representation. If the image of q is sufficiently close to the generic representation and sufficiently far from all class representations, it is predicted to be OOS. We state a pair of inequalities (1) relating the distances between query points, class prototypes, and the generic representation which need to be satisfied in order for GROOS detection to be able to correctly predict when q is OOS and also correctly predict the class of q when q is in-support. We analyze how these constraints effect the geometry of a model's feature space, characterizing its structure through three Propositions (Propositions 4.1, 4.2, and 4.3). We also show that for GROOS to be successful, additional 'second-order' relationships between prototypes and the generic representation need to hold.

We benchmark GROOS detection against two recently proposed methods - LCBO and MinDist [25] - as well as an additional method - Background OOS detection - which we describe in this paper. We find that GROOS detection not only on average outperforms previous benchmarks (Section 4.1), but an adapted version of GROOS called *Centered GROOS* tends to outperform other OOS detection methods in settings that require significant model generalization (Section 4.2). Despite the strong relative performance of Centered GROOS detection in this latter setting, it is clear that the community still has a considerable amount of work to do before few-shot models can satisfactorily detect OOS examples when evaluated on datasets significantly different from those that they were trained on.

In summary, our contributions in this paper include:

- We introduce the GROOS detection method, which is designed to solve the out-of-support detection problem in few-shot learning using a generic representation.
- We benchmark GROOS detection against existing metric-based methods in the literature and an additional OOS detection method, Background OOS Detection, which we describe in this paper. We show that GROOS out-performs these approaches both in a traditional few-shot train-evaluation setting, and in a more challenging setting where models are trained on ImageNet and then evaluated on a diverse range of datasets.
- We state two inequalities relating class prototypes, the generic representation, and encoded query points, which must be satisfied in order for both OOS detection and standard in-support classification to be effective. Motivated by these inequalities we prove three propositions which relate feature space geometries that arise from the standard Prototypical Networks problem and the feature space geometries that arise from GROOS.

2 Background and related work

2.1 Few-shot learning and Prototypical Networks

There are a range of approaches that have been used to address the challenges of few-shot learning. Fine-tuning methods [1, 2] use transfer learning followed by fine-tuning to train models with limited data. Data augmentation methods [5] leverage augmentation and generative approaches to produce additional training data. Gradient-based meta-learning [4, 19] is a class of methods that use sophisticated optimization techniques to learn strong initial weights which can be adapted to a new task with a small number of gradient steps. The algorithm we propose in this paper is related to a fourth class of algorithms called metric-based models. In these models an encoder function is trained to embed data into a space where a distance metric (either hard-coded or learned) captures some task-appropriate notion of similarity. Well-known examples of metric-based few-shot models include Prototypical Networks [21], Matching Networks [23], and Relational Network [22].

An *episode* is the basic unit of few-shot inference and training. It consists of a set S of labelled examples known as the *support set* and an unlabeled set Q known as the *query set*. Within an episode, a model uses elements in S to predict labels for elements in Q. We assume that elements of S belong to classes $C^{in} = \{1, \ldots, k\}$. For convenience, we decompose S into a disjoint union: $S = \bigcup_{c \in C^{in}} S_c$, where S_c contains only those elements of S with label c. We will assume that the size $n = |S_c|$ is constant for all $c \in C^{in}$. The integer n is known as the *shots* of the episode, while the integer k is known as the *ways*. In this paper we will assume that Q has been drawn from a distribution p, and each set S_c has been drawn from the conditional distribution $p(y = c)$.

By *few-shot training* we mean the process of calculating the loss for an entire episode and then using that loss to update the weights of the model. *Few-shot inference* has an analogous meaning. A *few-shot split* is a partition of a dataset into train and test sets by class, so that all examples from a given class are contained in either the train or test split, but not in both.

Prototypical networks (ProtoNets) [21] uses an encoder function $f : X \to \mathbb{R}^d$ to map elements of both Q and S into metric space $\mathbb{R}^d$ (which we will always assume is equipped with the Euclidean metric). In $\mathbb{R}^d$, a centroid γ_c is formed for each set $f_\theta(S_c)$. γ_c is referred to as the *prototype* which represents class c in $\mathbb{R}^d$. The model predicts the class of an unlabelled query point q based on the solution to $\arg\min_{c \in C^{in}} ||\gamma_c - f_\theta(q)||$. Note that in the case where one needs probabilities associated with a prediction, one can apply a softmax function to the distance vector $[-d_c]_{c \in C^{in}}$ where $d_c = ||\gamma_c - f_\theta(x)||$.

2.2 The out-of-support detection problem

As mentioned in the Introduction, it is commonly assumed in the literature that all elements of Q have a label from C^{in}. It was observed in [25] that in many real-world cases, this assumption is unrealistic. In that work the authors referred to an example $q \in Q$ that does not belong to a class in C^{in} as being "out-of-episode". We feel it is more appropriate to describe such examples as being *out-of-support* (OOS), since any elements found in Q can be said to be part of the episode. Following [25] we decompose Q as $Q = Q^{in} \cup Q^{out}$ where Q^{in} are those elements that are in-support and Q^{out} are those elements that are OOS. It is also convenient to use C to denote the set of all labels on elements from $S \cup Q$, with C decomposing as the disjoint union $C = C^{in} \cup C^{out}$ where C^{out} are simply those classes for which there are unlabeled examples in Q but no labeled examples in S. Note that the user generally does not have knowledge of C^{out}.

The *out-of-support (OOS) detection problem* then involves identifying those elements of Q that do not belong to any class in C^{in}. All the methods for OOS detection described or introduced in this paper use a confidence score $\varphi : X \to \mathbb{R}$ that maps a query point $q \in Q$ to a value in $\mathbb{R}$. In general, φ also depends on the full support set S as well as the encoder f_θ, but to simplify notation we assume these dependencies are implicit.

The authors of [25] proposed two methods for OOS detection. Both are presented as an additional component that can be added to Prototypical Networks and it is in this context that we will describe and evaluate them. The first uses a function called the *Minimum Distance Confidence Score* (MinDist), $\varphi_{dist} : X \to \mathbb{R}$ which is defined as $\varphi_{dist}(q, f_\theta) = -\min_{c \in C^{in}} ||\gamma_c - f_\theta(q)||$. A query q is predicted to be OOS if $\varphi_{dist}(q) < t$ for some $t < 0$. The second method proposed in [25] is the *Learnable Class BOundary (LCBO) Network* which is a parametric class-conditional confidence score $\varphi_{LC} : X \to \mathbb{R}$. φ_{LC} uses a small, fully-connected neural network $h_{\theta'} : \mathbb{R}^d \to \mathbb{R}$ to produce scores for each prototype/query pair (q, γ_c). The confidence score φ_{LC} is defined as $\varphi_{LC}(q) = \max_{c \in C^{in}} \big(h_{\theta'}(\gamma_c, f_\theta(q))\big)$. The model predicts that q is OOS if $\varphi_{LC}(q) < t$ for some predetermined threshold t. The authors of [20] consider methods for handling few-shot classification tasks in the presence of OOS examples. While their methods are conceptually similar to ours, their work differs in that they neither consider OOS detection nor distribution shift like we do.

2.3 Out-of-distribution detection

Out-of-distribution (OOD) detection aims to develop methods that can identify whether or not a data point x was drawn from some known distribution p. Methods for doing this within the context of deep learning models include: using a model's largest softmax output value as a confidence score [9, 13] and ODIN [15] which suggests identifying OOD examples through the use of model gradients and sofmax temperature scaling. Standard benchmarks for OOD detection focus on using OOD

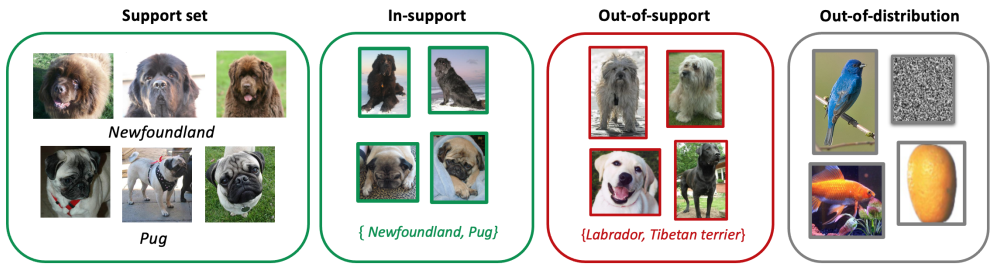

Figure 1: A diagram illustrating the difference between out-of-support detection and out-of-distribution detection for a few-shot task where one attempts to identify images of Newfoundlands and pugs from a dataset of dog images.

detection methods to identify examples drawn from very visually distinct distributions. For example, a common experiment attempts to detect Gaussian noise or MNIST [14] images injected into the CIFAR10 dataset [11].

OOS detection differs from OOD detection in that, in general, the conditional distributions corresponding to elements belonging to C^{in} and C^{out} only vary in subtle and arbitrary ways. Consider the example summarized in Figure 1 where S and Q consist of images of dogs. While it is true that the distribution of dog images belonging to classes $C^{in} = \{\text{Newfoundland, pug}\}$ is different than those belonging to classes $C^{out} = \{\text{Labrador, Tibertan terrier}\}$, these differences are slight (and focus on very specific aspects of the input) relative to differences in distribution that OOD detection methods are designed to detect. Indeed, [25] showed that a few-shot analogue of [9] applied to a ProtoNet model struggled on the OOS detection problem. Additionally, OOD detection methods generally assume that even if a model has not seen examples of OOD data, it has seen many examples of in-distribution data. This is not the case for few-shot models which only have a handful of classes that they can use to characterize "in-distribution". In fact, in the generalization-focused evaluation setting described in Section 4.2, few-shot models could be described as operating exclusively out-of-distribution in relation to their training set. Finally, while OOD examples are defined with respect to an entire dataset, OOS examples are only defined via a small support set, and this definition can vary from episode to episode. As suggested in [25], all these differences argue for identifying few-shot OOS detection as a problem which is distinct from OOD detection, requiring its own set of methods.

3 OOS detection with a generic point

In this section we describe our proposed *Generic Representation Out-Of-Support (GROOS) Detection* method. Let $f_\theta : X \to \mathbb{R}^d$ be the encoder (for example, when X is an image space, then f_θ might be a ResNet [7] with the final linear classification layer removed). Let $L : \mathbb{R}^d \to \mathbb{R}^d$ be an affine map, so that $L(x) = Wx + b$ for some matrix (weights) W and vector (bias) b. We construct a new encoder by composing $h_\theta = L \circ f_\theta : X \to \mathbb{R}^d$.

Next choose a point $\gamma_{oos} \in \mathbb{R}^d$ which will be called the *generic representation* and a threshold $0 \leq t \leq 1$. There are many potential choices for γ_{oos} but we find that the origin works well in practice. Inference with h_θ is similar to inference with the standard ProtoNets (Section 2.1). For an n-shot, k-way support set $S = \cup_{c \in C^{in}} S_c$, with support set labels $C^{in} = \{1, \ldots, k\}$, and query q, $h_\theta(S)$ and $h_\theta(q)$ are calculated and centroid prototypes γ_c are computed for each set $h_\theta(S_c)$ with $c \in C^{in}$. We compute the vector $\mathbf{d}_q := (d_1, \ldots, d_k, d_{oos})$ where $d_i := ||\gamma_i - h_\theta(q)||$. Finally, let $softmax : \mathbb{R}^{k+1} \to \mathbb{R}^{k+1}$ be the standard softmax function. Following the notation in Section 2.2 we define $\varphi_{gen} : X \to \mathbb{R}$ to be $\varphi_{gen}(q) := [softmax(-\mathbf{d}_q)]_{k+1}$ where $[softmax(-\mathbf{d}_q)]_{k+1}$ is the $(k+1)$st output coordinate corresponding to encoded query distance from γ_{oos}. If $\varphi_{gen}(q) > t$ then we predict that q is OOS. If $\varphi_{gen}(q) < t$, then we predict that q is in-support and we use the other k softmax outputs from $softmax(-\mathbf{d}_q)$ to predict its class. Informally, this process consists of comparing the distance of the encoded query point from the generic representation to its distance to other support prototypes. If the query is sufficiently closer to the generic representation than it is to other prototypes, then it is predicted as OOS. This process is summarized in Algorithm 1.

Algorithm 1 Generic Representation Out-Of-Support (GROOS) Detection

```
Input: Encoder function $h_\theta : X \to \mathbb{R}^d$, generic representation $\gamma_{oos} \in \mathbb{R}^d$, support set $S = S_1 \cup \dots \cup S_k$ with corresponding label set $C^{in} = \{1, \dots, k\}$, query $q$, threshold $0 \leq t \leq 1$.
  for $c \in C^{in}$ do
    Compute prototype centroid $\gamma_c$ from $h_\theta(S_c)$
    Compute $d_c = ||\gamma_c - h_\theta(q)||$
  end for
  Compute $d_{oos} = ||\gamma_{oss} - h_\theta(q)||$
  Set $\mathbf{d}_q = (d_1, \dots, d_k, d_{oos})$ and compute $\varphi_{gen}(q) = [Softmax(-\mathbf{d}_q)]_{k+1}$
  if $\varphi_{gen}(q) > t$ then
    $q$ is predicted as OOS
  else
    $q$ is predicted as in-support, belonging to class $c^* = \arg\min_{c \in C^{in}} d_c$.
  end if
```

One can ask what metric properties an encoded dataset $h_\theta(D)$ must have in order for GROOS detection to be effective on all possible combinations of support and query sets. For simplicity we assume that prototypes $\gamma_1, \dots, \gamma_k$ and generic representation γ_{oos} are fixed (empirically we find that prototypes are fairly stable when the number of shots is high enough so this is not an unreasonable approximation). (1) To ensure in-support examples are always predicted correctly, h_θ must map any $x \in D$ with label $c \in C$ closer to γ_c than to any other prototype or γ_{oos}. That is $||h_\theta(x) - \gamma_c|| < ||h_\theta(x) - \gamma_{c'}||$ for all $c' \in C \cup \{oos\}$ such that $c \neq c'$. (2) One the other hand, when c is not represented in the support set, then $h_\theta(x)$ must be closer to γ_{oos} than to any other class prototype which is not γ_c (which does not appear in the episode). Specifically, $||h_\theta(x) - \gamma_{oos}|| < ||h_\theta(x) - \gamma_{c'}||$ for all $c' \in C$ such that $c' \neq c$. These inequalities can be combined for the single expression

$$||h_\theta(x) - \gamma_c|| < ||h_\theta(x) - \gamma_{oos}|| < ||h_\theta(x) - \gamma_{c'}|| \quad \text{for } c' \in C, c' \neq c. \tag{1}$$

Inequality (1) suggests that if one is not able to actually train h_θ on dataset D (or a similar dataset), and hence h_θ is not able to learn how to arrange encoded data around γ_{oos}, then another sensible option is to choose γ_{oos} to be the centroid of $h_\theta(S \cup Q)$. We call this alternative version of GROOS detection *Centered GROOS Detection*. We will see that it works better than the standard version of GROOS detection when the test set differs significantly from the training set.

3.1 Background detection

We introduce a second OOS detection model to serve as an additional benchmark. We call it *Background Detection* since it was inspired by the "background class" described in [27]. Background detection consists of an encoder function $h_\theta : X \to \mathbb{R}^d$ such as a ResNet, with its final classification layer replaced with a linear layer $L : \mathbb{R}^d \to \mathbb{R}^d$ and two predetermined constants $M > 0$ and $0 \leq t \leq 1$. An episode with support set $S = \cup_{c \in C^{in}} S_k$ and query q proceeds with the usual calculation of class centroids γ_c for $c \in C^{in}$. Using constant M and distances $||\gamma_c - h_\theta(q)||$ between encoded query and prototypes the vector $\mathbf{d}_q := (d_1, \dots, d_k, M)$ is obtained. The confidence function $\varphi_{back} : X \to \mathbb{R}$ associated with this method is then: $\varphi_{back}(q) := \left[softmax(-\mathbf{d}_q)\right]_{k+1}$. Query q is predicted to be OOS if $\varphi_{back}(q) > t$.

4 Experiments and analysis

4.1 Standard few-shot evaluation

Our first set of experiments look at how well OOS detection methods (MinDist, LCBO, Background Detection, GROOS, and Centered GROOS) can identify OOS examples in the setting where the base model is trained and evaluated on few-shot splits drawn from the same dataset. That is, we partition the classes of the dataset between train and test. We focus on the datasets: CIFAR100 [11] (CC-BY 4.0), CUB-200 [24] (CC0 1.0), and Omniglot [12] (MIT License).

All models were trained for four days of wall clock time on a single Tesla P100 GPU for a total of between 250,000 and 500,000 training episodes in that time. All performances stabilized around the

	CIFAR100		CUB-200		Omniglot	
	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
MinDist	88.4±0.1	88.6±0.1	89.0±0.2	89.2±0.1	99.4±0.1	**99.5±0.1**
LCBO	83.2±0.4	84.7±0.4	85.8±0.4	87.6±0.3	99.2±0.1	99.3±0.1
Background (ours)	87.2±0.2	86.9±0.2	88.8±0.6	88.0±0.7	98.9±0.1	98.9±0.1
GROOS (ours)	**90.1±0.2**	**90.2±0.3**	**90.9±0.6**	**90.6±0.7**	**99.6±0.1**	**99.5±0.1**
Centered GROOS (ours)	88.9±0.8	88.4±0.7	89.6±0.2	89.5±0.1	**99.5±0.1**	**99.5±0.1**

Table 1: The area under the ROC curve (AUROC) and area under the precision-recall curve (AUPR) for a range of few-shot OOS detection methods.

	ImageNet	CIFAR100	Omniglot	Aircraft	Textures	Fruits
MinDist	95.0±0.1	**80.1±0.5**	**85.5±0.6**	59.4±0.2	72.7±0.6	95.3±0.4
LCBO	92.6±0.1	76.3±1.2	68.5±1.4	54.7±0.4	65.3±1.3	89.0±1.9
Background (ours)	93.6±0.1	77.5±0.7	58.6±1.7	58.4±0.3	71.8±0.7	89.8±0.9
GROOS (ours)	**95.7±0.1**	74.3±1.6	74.6±1.2	53.8±0.3	75.2±0.9	91.2±0.7
Centered GROOS (ours)	95.0±0.2	**80.6±0.9**	82.1±0.4	**61.8±0.4**	**82.3±0.1**	**96.2±0.2**

Table 2: The area under the ROC curve (AUROC) for a range of few-shot OOS detection methods which were all trained on a few-shot training split of ImageNet and then evaluated on a range of datasets.

lower end of that range. All models used a ResNet18 encoder with the final linear layer removed and were initialized with the standard ImageNet (CC-BY 4.0) pre-trained weights from Torchvision [17]. We address the question of how performance differs for different sizes of encoder in Section A.1 of the Appendix. We used the Adam optimizer for training, with a learning rate of 1×10^5, a weight decay factor of 5×10^{-5}, and β values of 0.9 and 0.999. All results correspond to 5-shot, 5-way episodes, with 8 queries per support class and a total of 40 OOS images introduced per episode (that is, 50% of all images in the query were OOS). All images were resized to 224×224 before being fed through the model. To evaluate each model, we sampled 1000 episodes from the corresponding few-shot test set. To complete the evaluation, we computed the area under precision recall curve (AUPR) and area under the ROC curve (AUROC) for each model with respect to the evaluation queries and multiplied these by 100.

The result of these experiments can be found in Table 1. We bold all scores that are within 0.5 of the top model (in terms of both AUPR and AUROC), putting an $*$ on the top score for each column. As can be seen, in two of the three datasets used, GROOS outperforms other methods by at least 1.0 both in terms of AUPR and AUROC. On Omniglot, MinDist, LCBO, GROOS, and Centered GROOS all do close to perfect. We include this last experiment to demonstrate that when a sufficiently strong encoder is used for a simpler dataset, then a range of OOS detection methods can do quite well.

4.2 Generalization experiments

We also ran experiments to evaluate how adaptable MinDist, LCBO, Background Detection, GROOS Detection, and Centered GROOS Detection were when a dataset from a previously unseen distribution was introduced at inference time. We chose to train our networks on a few-shot training split of ImageNet, as ImageNet has been shown to generally produce rich and flexible feature extractors [10, 2], and then test on: the few-shot ImageNet testset, CIFAR100, Omniglot, Aircraft [16], Describable Textures [3], and Fruits 360 [18].[1] All models used the same encoder, hyperparameters, and training scheme as that described in Section 4.1.

[1]Aircraft is available exclusively for non-commercial research purposes, Describable Textures is available for research purposes, and Fruits 360 is covered by CC BY-SA 4.0

	ImageNet	CIFAR100	Omniglot	Aircraft	Textures	Fruits
MinDist	95.0±0.1	**79.4±0.5**	**86.3±0.6**	59.3±0.2	73.8±0.7	95.5±0.3
LCBO	91.8±0.1	73.3±1.2	66.4±1.6	53.8±0.4	64.5±1.6	88.6±2.3
Background (ours)	93.7±0.1	77.1±0.7	76.6±0.6	58.3±0.3	70.1±0.6	92.8±0.4
GROOS (ours)	**95.5±0.1**	72.1±2.3	75.7±1.1	54.3±0.3	71.1±0.8	92.1±0.5
Centered GROOS (ours)	94.7±0.3	**79.8±0.8**	82.1±0.6	**61.7±0.4**	**79.9±0.3**	**96.4±0.1**

Table 3: The area under the precision recall curve (AUPR) for a range of few-shot OOS detection methods which were all trained on a few-shot training split of ImageNet and then evaluated on a range of test datasets.

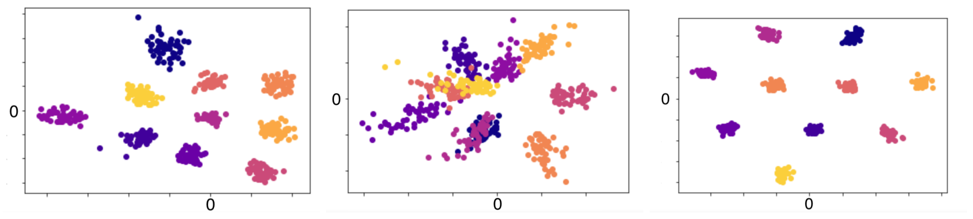

Figure 2: Visualizations of the feature space of a ResNet50 encoder (left) trained without OOS examples, (center) with OOS examples using a generic representation, (right) using a background class.

We find that in this setting, performance is generally worse for all model types. This is not surprising since the models are essentially operating on out-of-distribution data at inference time. Aircraft is a particularly challenging dataset for models that have not seen the corresponding training set. A comparison of the error bars in Tables 2 and 3 on the one hand and Table 1 on the other illustrates that when operating on OOD data there is more variation between training runs. Nonetheless, Centered GROOS detection performs better than other methods on 4 out of the 5 OOD datasets, with the exception of Omniglot where MinDist does substantially better. On the in-distribution test set ImageNet test, GROOS achieves better performance than Centered GROOS, confirming our hypothesis that GROOS is better to use when inference data aligns with training data and Centered GROOS is better otherwise. Of all the datasets presented to the models in these tests, Omniglot is probably the most "unlike" ImageNet. We conjecture that for mildly OOD datasets such as CIFAR100, Aircraft, and Fruits, Centered GROOS tends to perform better, while for significantly OOD datasets such as Omniglot, the simpler MinDist model might be a better choice.

4.3 Generic points: feature space geometry and decision boundaries

In this section we analyze the geometry of the feature space induced by the use of a generic point to detect OOS examples (see Figure 2 for a low-dimensional visualization of this). Proofs for all propositions can be found in Section A.2.

Recall that an *affine hyperplane in* $\mathbb{R}^d$ is a translation of a $(d-1)$-dimensional subspace. Alternatively, a non-zero vector $v \in \mathbb{R}^d$ and constant $b \in \mathbb{R}$ define an affine hyperplane via the expression $H := \{w \in \mathbb{R}^d \mid \langle w, v\rangle = b\}$. Note that any affine hyperplane H decomposes $\mathbb{R}^d$ into two *open half-spaces*: $H^+ := \{w \in \mathbb{R}^d \mid \langle w, v\rangle > b\}$ and $H^- := \{w \in \mathbb{R}^d \mid \langle w, v\rangle < b\}$. For any two distinct points $x_1, x_2 \in \mathbb{R}^d$, one gets a hyperplane H_{x_1,x_2} defined by normal vector $x_1 - x_2$ and constant $\frac{1}{2}(||x_1||^2 - ||x_2||^2)$. In particular, when γ_1 and γ_2 are centroids for two classes, then H_{γ_1,γ_2} is the decision boundary of the associated 2-way ProtoNets model (or alternatively the Voronoi partition corresponding to two points).

Let x be a point in $\mathbb{R}^d$ and let $\gamma_1, \dots, \gamma_k, \gamma_{oos}$ be a list of prototypes and generic point. Let $\mathcal{S}_{k+1}$ be the symmetric group on (or permutations of) $k + 1$ elements. There is a trivial bijection between $\mathcal{S}_{k+1}$ and total orderings of $\gamma_1, \dots, \gamma_k, \gamma_{oos}$. In particular, for permutation $\sigma \in \mathcal{S}_{k+1}$, we associate

σ with the order $\gamma_{\sigma(1)} < \gamma_{\sigma(2)} < \cdots < \gamma_{\sigma(oos)}$ where we write $\sigma(i) = j$ to represent the value $j \in \{1, \ldots, oos\}$ that σ permutes i to (we use index oos and $k+1$ interchangeably).

Proposition 4.1. *Let $\gamma_1, \ldots, \gamma_k, \gamma_{oos} \in \mathbb{R}^d$ be a finite list of prototypes and generic point. The set of hyperplanes corresponding to each pair of $\gamma_1, \ldots, \gamma_k, \gamma_{oos}$ induce a decomposition of $\mathbb{R}^d$ into open (possibly empty) subsets* (cells) *S_σ, where $\sigma \in \mathcal{S}_{k+1}$ and*

$$S_\sigma := \{x \in \mathbb{R}^d \mid ||x - \gamma_{\sigma(1)}|| < \cdots < ||x - \gamma_{\sigma(oos)}||\},$$

as well as a measure zero, closed subset B which is the union of all H_{γ_i, γ_j} for $i, j \in \{1, \ldots, k, oos\}$.

The decomposition described in Proposition 4.1 can be used to describe those regions of $\mathbb{R}^d$ that can lead to the correct classification of an encoded point in different versions of the ProtoNet problem. As we will see, these regions differ substantially between the classic ProtoNets problem and ProtoNets with generic point. We call a point $x \in \mathbb{R}^d$, *i-viable* if encoding a class i query point q such that $h_\theta(q) = x$ results in the correct prediction that q belongs to class i, if class i is represented in the support, or that q is OOS, if class i is not represented in the support. A point is called *viable* if it is i-viable for some $i \in \{1, \ldots, k\}$. A set of points U is called *i-viable* if every point in U is i-viable and *viable* if every point in U is viable.

- *Standard ProtoNets*: For a point belonging to class i to be predicted correctly, it must lie in a cell of the form S_σ with $\sigma(1) = i$. Note that outside of measure-zero set B, every point in $\mathbb{R}^d$ is i-viable for some $i \in \{1, \ldots, k\}$ since every cell S_σ consists of points closest to some centroid (i.e. $\sigma(1) = j$ for some $j \in \{1, \ldots, k\}$) and in the setting where OOS examples do not exist, a point belonging to class i is always classified correctly if it is closer to centroid γ_i than it is to any other centroid.
- *ProtoNets with generic point*: For a point belonging to class i to be predicted correctly both when its prototype is present and also when it is not, it must satisfy inequality (1). This means that it must lie in a cell of the form S_σ with $\sigma(1) = i$ and $\sigma(2) = oos$. Note that this condition means that even outside of B, there are non-viable regions of $\mathbb{R}^d$. For example, if $\sigma(2) \neq oos$.

We illustrate these differences in Figure 3 in the Appendix.

Proposition 4.2. *Let $\{1, \ldots, k\}$ be a set of classes and let $\gamma_{oos} \in \mathbb{R}^d$ be a generic point.*

1. *In the standard ProtoNets problem, the set of i-viable points is always nonempty for each choice of distinct prototypes $\gamma_1, \ldots, \gamma_k \in \mathbb{R}^d$ and for all $i \in \{1, \ldots, k\}$.*
2. *In the ProtoNets with generic point problem, there are choices k and distinct $\gamma_1, \ldots, \gamma_k, \gamma_{oos} \in \mathbb{R}^d$ for which the i-viable region of $\mathbb{R}^d$ is the empty set for some $i \in \{1, \ldots, k\}$. There are also choices of distinct $\gamma_1, \ldots, \gamma_k, \gamma_{oos}$ such that there is a nonempty i-viable region for each i.*

Thus we see that the introduction of a generic points puts additional constraints on how a model can arrange prototypes in feature space, with some arrangements being not only non-optimal, but actually precluding correct predictions.

Our final proposition shows that the radial pattern shown in Figure 2 actually represents general geometric structure induced by the generic point problem. For fixed $\gamma_1, \ldots, \gamma_k, \gamma_{oos} \in \mathbb{R}^d$ we call the region of $\mathbb{R}^d$ which consists of points that are closer to γ_{oos} than to any $\gamma_1, \ldots, \gamma_k$ the *OOS-core* (note that as a corollary to Proposition 4.2.1 this always exists). We call two sets $U, V \subset \mathbb{R}^d$ *adjacent* if there is a point $p \in \mathbb{R}^d$ such that for any $\epsilon > 0$, the open ball $B_\epsilon(p)$ contains points from both U and V.

Proposition 4.3. *If $\gamma_1, \ldots, \gamma_k, \gamma_{oos} \in \mathbb{R}^d$ are a choice of distinct prototypes/generic point such that the set P_i of i-viable points is non-empty for $i \in \{1, \ldots, k\}$, then P_i is adjacent to the OOS core.*

5 Conclusion

In many situations, the ability to detect OOS examples is a necessary requirement for deployment of few-shot learning models. In this paper we showed that in the metric-based setting, GROOS and its variant Centered GROOS are two methods that begin to address this challenge. Despite the fact that our models, on average, outperformed existing approaches, we believe OOS detection is a challenge that deserves more attention within the few-shot community, since effective solutions will enable broader adoption of few-shot methods for real-world science and engineering applications.

References

[1] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv:1904.04232*, 2019.

[2] Arkabandhu Chowdhury, Mingchao Jiang, and Chris Jermaine. Few-shot image classification: Just use a library of pre-trained feature extractors and a simple classifier. *arXiv preprint arXiv:2101.00562*, 2021.

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

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[5] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017.

[6] Robin Hartshorne. *Algebraic geometry*, volume 52. Springer Science & Business Media, 2013.

[7] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[8] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. *arXiv preprint arXiv:2006.16241*, 2020.

[9] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *International Conference on Learning Representations*, 2017.

[10] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2661–2671, 2019.

[11] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[12] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[13] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6405–6416, 2017.

[14] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[15] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *International Conference on Learning Representations*, 2018.

[16] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, Johns Hopkins University, 2013.

[17] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488, 2010.

[18] Horea Mureşan and Mihai Oltean. Fruit recognition from images using deep learning. *Acta Universitatis Sapientiae, Informatica*, 10(1):26–42, 2018.

[19] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv:1803.02999*, 2018.

[20] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

[21] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[22] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[23] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3630–3638. Curran Associates, Inc., 2016.

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

[25] Kuan-Chieh Wang, Paul Vicol, Eleni Triantafillou, Chia-Cheng Liu, and Richard Zemel. Out-of-distribution detection in few-shot classification. *OpenReview.net*, 2019.

[26] Yaqing Wang, Quanming Yao, James Kwok, and Lionel M. Ni. Generalizing from a Few Examples: A Survey on Few-Shot Learning. In *Intelligent Systems Design and Applications*, pages 100–112. Springer, 2018.

[27] Xiang Zhang and Yann LeCun. Universum prescription: Regularization using unlabeled data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

	CIFAR100		CUB-200		Omniglot	
	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
MinDist	66.85	67.18	64.51	66.73	95.94	95.90
LCBO	**75.15**	**75.89**	**68.98**	**71.96**	**98.98**	**99.10**
Background (ours)	67.71	65.97	61.26	60.38	98.38	98.29
GROOS (ours)	74.15	74.95	67.27	66.55	98.81	98.77
Centered GROOS (ours)	70.36	71.41	65.34	65.38	98.65	98.62

Table 4: Results for the same set of experiments reported in Table 1 but using a 4-Conv encoder rather than a ResNet18 encoder.

A Appendix

A.1 Encoder size

Given that much of the metric-based few-shot learning literature uses small encoders, in Figure 4 we include results for the "standard" few-shot experiments using a 4-Conv encoder (as in [21, 25]) rather than the ResNet18 encoder used in Section 4.1. Interestingly, we find that with a smaller encoder the LCBO method does significantly better relative to other approaches, indicating that learning decision boundaries for OOS detection may be a more effective strategy in either a lower dimensional feature space or for less rich encoders. In future work it would be interesting to investigate whether attaching a larger MLP helps LCBO scale to larger encoders. Of course, including more fully-connected layers quickly becomes expensive which would be a potential downside of this method.

We also repeated the generalization experiments from Section 4.2 (which also used a ResNet18 encoder) with a 4-Conv encoder and a ResNet50 encoder. We summarize our results in Figures 5 and 6. The logic behind our choice to also test larger encoders in this setting stemmed from the observation that in tasks that require higher levels of generalization, large encoders can sometimes yield better results [8]. We find that larger encoders do tend to slightly improve performance in terms of AUROC and AUPR. With the exception of AUPR for the Aircraft dataset where MinDist performed slightly better than Centered GROOS when we used a ResNet50 encoder instead of a ResNet18 encoder, the top performing model on a dataset did not change based on whether one used a larger encoder. It is perhaps notable that the Aircraft dataset is also one of the few examples where model performance decreased when using a ResNet50 encoder rather than a ResNet18 encoder.

Distinct from the pattern we observed in Figure 4, in this setting using a smaller encoder did not appear to result in much better performance for LCBO. With the exception of its performance on ImageNet itself, which does not require the same level of generalization, LCBO did not out-perform other methods on any of the datasets. We suspect that this arises from the fact that learning decision boundaries is not an approach that transfers well to significantly different datasets. Similar to the results from Section 4.2 we observe that the top models in terms of generalization were Centered GROOS and MinDist suggesting that centered generic points and raw distance are better able to capture "different-ness" across datasets. We also observe that in the smaller encoder setting, MinDist is more competitive with Centered GROOS.

A.2 Proofs from Section 4.3

Proof of Proposition 4.1. For any $x \in \mathbb{R}^d$, either (1) there are at least two γ_i, γ_j for $i, j \in \{1, \dots, k, oos\}$ such that $||x - \gamma_i|| = ||x - \gamma_j||$ or (2) for all γ_i, γ_j either $||x - \gamma_i|| > ||x - \gamma_j||$ or $||x - \gamma_i|| < ||x - \gamma_j||$. In the former case, $x \in B$ since x belongs to H_{γ_i,γ_j} as this hyperplane consists precisely of those x' such that $||x' - \gamma_i|| = ||x' - \gamma_j||$. In the latter case the set

$$D = \big\{||x - \gamma_i|| \mid i \in \{1, \dots, k, oos\}\big\}$$

consists of distinct real numbers. It is clear that these numbers can be ordered so that they are strictly increasing. Denote by σ the permutation from $\mathcal{S}_{k+1}$ such that

$$||x - \gamma_{\sigma(1)}|| < ||x - \gamma_{\sigma(2)}|| < \dots < ||x - \gamma_{\sigma(oos)}||.$$

	ImageNet	CIFAR100	Omniglot	Aircraft	Textures	Fruits
		4-Conv encoder				
MinDist	71.28	63.36	**67.45**	54.07	57.29	**95.97**
LCBO	75.51	60.36	58.68	52.49	58.16	87.35
Background (ours)	61.78	60.78	65.10	52.95	55.75	92.57
GROOS (ours)	**75.99**	59.22	61.44	53.59	59.44	90.88
Centered GROOS (ours)	61.78	**64.80**	67.03	**54.87**	**61.77**	94.80
		ResNet50 encoder				
MinDist	97.51	82.33	**84.54**	58.49	76.46	92.57
LCBO	95.66	79.54	73.93	55.72	74.48	90.72
Background (ours)	95.57	79.11	60.61	53.86	78.48	92.30
GROOS (ours)	**97.76**	79.17	78.55	53.38	80.60	94.19
Centered GROOS (ours)	96.96	**84.20**	81.36	**59.07**	**84.17**	**96.40**

Table 5: AUROC results for the same set of experiments reported on in Table 2 but using a 4-Conv encoder (top) and ResNet50 encoder (bottom) rather than a ResNet18 encoder.

	ImageNet	CIFAR100	Omniglot	Aircraft	Textures	Fruits
		4-Conv encoder				
MinDist	70.44	62.67	**69.71***	**53.84***	57.27	**95.97***
LCBO	**74.50***	58.58	57.04	52.20	57.31	87.48
Background (ours)	60.67	59.07	61.99	52.39	55.75	91.66
GROOS (ours)	**75.48**	58.62	60.55	53.10	59.12	91.66
Centered GROOS (ours)	70.44	**63.17***	64.04	**53.73**	**59.46***	94.73
		ResNet50 encoder				
MinDist	**97.63**	81.41	**85.62***	**58.72***	78.77	91.66
LCBO	95.28	77.56	71.53	55.17	73.48	91.25
Background (ours)	95.63	79.26	77.26	57.87	77.09	95.08
GROOS (ours)	**97.68***	77.16	79.53	54.63	76.87	94.51
Centered GROOS (ours)	96.75	**83.27***	81.19	**58.24**	**82.02***	**96.48***

Table 6: AUPR results for the same set of experiments reported on in Table 3 but using a 4-Conv encoder (top) and ResNet50 encoder (bottom) rather than a ResNet18 encoder.

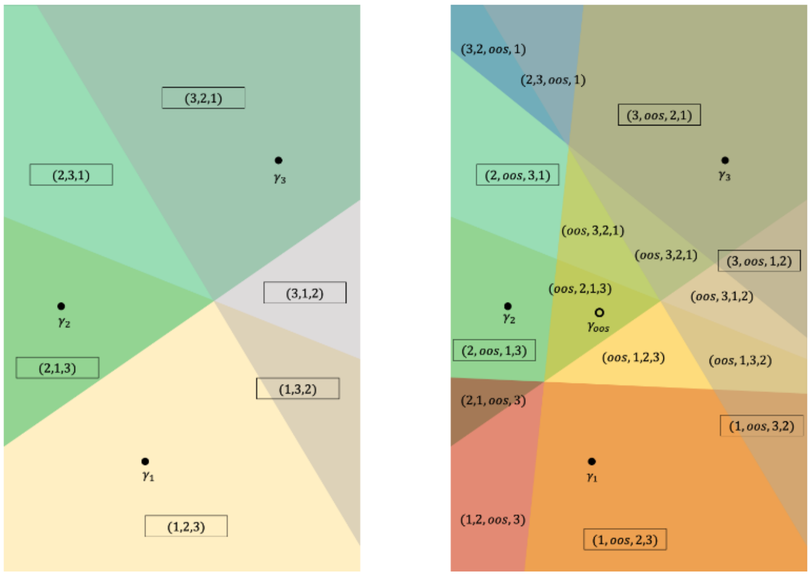

Figure 3: Low dimensional illustrations of the feature space decision boundaries of the (left) standard ProtoNet problem with three prototypes, (right) the ProtoNet problem with generic point. In each region we label the ordering the closest prototypes/generic point. We box the labels of regions which are viable.

Then $x \in S_\sigma$. This shows that the union of B and each set in $\{S_\sigma \mid \sigma \in \mathcal{S}_{k+1}\}$ is equal to $\mathbb{R}^d$. Using the distance parametrization of each S_σ based on σ, it is also clear that B is disjoint from each S_σ and that furthermore, $S_\sigma \cap S_\tau = \emptyset$ when $\sigma \neq \tau$.

The fact that each S_σ is open, and B is closed and measure zero follows from elementary topology/measure theory.

□

Proof of Proposition 4.2.

1. If $\gamma_1, \ldots, \gamma_k$ are distinct from each other, then for any $i \in \{1, \ldots, k\}$, we can choose $\epsilon > 0$ sufficiently small such that for all points $x \in B_\epsilon(\gamma_i)$ we have that $||x - \gamma_i|| < ||x - \gamma_j||$ for each $j \in \{1, \ldots, k\}$ with $j \neq i$. Observe that

$$B_\epsilon(x) \subseteq \bigcup_{\substack{\sigma \in \mathcal{S}_{k+1} \\ \sigma(1) = i}} S_\sigma,$$

that is, $B_\epsilon(x)$ belongs to the i-viable region of $\mathbb{R}^d$. Hence the i-viable region is non-empty.

2. We give two examples, in the first there exists an element $i \in \{1, \ldots, k\}$ such that the i-viable region is empty. In the second, for each $i \in \{1, \ldots, k\}$, the i-viable region is not empty. In both cases we leave it to the reader to verify the example.

   (a) Consider the case $d = 2$, $k = 2$, $\gamma_{oos} = (1, 0), \gamma_1 = (0, 0)$, and $\gamma_2 = (-1, 0)$. It can be checked that in this case the 2-viable region consists of those points that are both to the left of the line $x = (0, 0)$ and to the right of the line $x = (\frac{1}{2}, 0)$. This set is of course empty.

   (b) Consider the case $d = 2$, $k = 4$, $\gamma_{oos} = (0, 0), \gamma_1 = (1, 0), \gamma_2 = (0, 1), \gamma_3 = (-1, 0)$, and $\gamma_4 = (0, -1)$. Elementary calculations show that the 1-viable region is nonzero and defined by the inequalities $y > -\frac{1}{2}$, $y < \frac{1}{2}$, and $x > \frac{1}{2}$. The 2, 3, and 4-viable regions can be obtained from the 1-viable region via symmetry transformations.

□

To prove Proposition 4.3, we need to establish a couple short lemmas:

Lemma A.1. *Let γ_i and γ_j be distinct prototypes. If two points x and y satisfy the inequalities*

$$||x - \gamma_i|| < ||x - \gamma_j|| \quad \textit{and} \quad ||y - \gamma_i|| < ||y - \gamma_j||,$$

then for any point z on the line segment ℓ connecting these two points,

$$||z - \gamma_i|| < ||z - \gamma_j||. \tag{2}$$

If, instead,

$$||x - \gamma_i|| = ||x - \gamma_j|| \quad \textit{and} \quad ||y - \gamma_i|| < ||y - \gamma_j||,$$

the strict inequality (2) *holds at every point on $\ell \setminus \{x\}$.*

Proof. To prove the first part of the Lemma, notice that both x and y lie on the same side of the hyperplane H_{γ_i,γ_j}. Since a hyperplane splits $\mathbb{R}^d$ into two convex half-spaces, the entire segment ℓ lies on a single side of this hyperplane and the result follows.

The last statement is true since, if the segment does not lie entirely in the plane H_{γ_i,γ_j}, it can only intersect at a single point, x (note that ℓ could also lie entirely within H_{γ_i,γ_j} but we know that the other end point of ℓ, y, is not in H_{γ_i,γ_j}). Since x is the endpoint of the segment, the rest lies in one of the open half spaces, in this case, that whose points satisfy (2). □

Lemma A.2. *Let the γ_i, γ_j be as in Lemma A.1, x a point in the i-viable region and γ_{oos} be a distinct generic representation. Let ℓ be the line segment between x and γ_{oos} and z be the point on ℓ where it intersects $H_{\gamma_{oos}\gamma_i}$. Then the line segment ℓ' from x to z is such that for any point w on this segment and for all $j \in \{1, \dots, k\}$ with $j \neq i$,*

$$||w - \gamma_i|| < ||w - \gamma_{oos}|| < ||w - \gamma_j||.$$

Similarly, if ℓ'' is the line segment from z to γ_{oos}, then all w on ℓ'' satisfy

$$||w - \gamma_{oos}|| < ||w - \gamma_j||$$

for all $j \in \{1, \dots, k\}$ (including $j = i$).

Proof. Notice that γ_{oos} and x satisfy the inequalities

$$0 = ||\gamma_{oos} - \gamma_{oos}|| < ||\gamma_{oos} - \gamma_j|| \quad \text{and} \quad ||x - \gamma_{oos}|| < ||x - \gamma_j||$$

for any $j \in \{1, \dots, k\}$ with $j \neq i$. Applying Lemma A.1, this implies that z, which lies on the line segment connecting x an γ_{oos}, satisfies

$$||z - \gamma_{oos}|| < ||z - \gamma_j||.$$

Since it lies on $H_{\gamma_{oos},\gamma_i}$ as well,

$$||z - \gamma_i|| = ||z - \gamma_{oos}|| < ||z - \gamma_j||. \tag{3}$$

But since x satisfies

$$||x - \gamma_i|| < ||x - \gamma_{oos}|| < ||x - \gamma_j||,$$

two applications of Lemma A.1 yield that for any point w on ℓ',

$$||w - \gamma_i|| < ||w - \gamma_{oos}|| < ||w - \gamma_j||.$$

This proves the first statement.

Next, returning to (3), we see that since

$$0 = ||\gamma_{oos} - \gamma_{oos}|| < ||\gamma_{oos} - \gamma_j||$$

for any $j \in \{1, \dots, k\}$ (including $i = j$), then by Lemma A.1, for all w on ℓ'' we must have that

$$||w - \gamma_{oos}|| < ||w - \gamma_j||,$$

which proves the second statement. □

Using these lemmas, we can prove Proposition 4.3:

Proof. Let x be a point in the i-viable region of $\mathbb{R}^d$ and z be the point on the segment ℓ between x and γ_{oos} that lies on the hyperplane $H_{\gamma_i,\gamma_{oos}}$. Note that ℓ must cross this hyperplane since x lies on one side of $H_{\gamma_i,\gamma_{oos}}$, being closer to γ_i than to γ_{oos}, and γ_{oos} lies on the other.

By Lemma A.2, all points w of ℓ on the same side of $H_{\gamma_i,\gamma_{oos}}$ as x satisfy

$$||w - \gamma_i|| < ||w - \gamma_{oos}|| < ||w - \gamma_j||,$$

for all $j \in \{1, \ldots, k\}$ with $j \neq i$. All such points are i-viable. All points on ℓ on the same side of $H_{\gamma_i,\gamma_{oos}}$ as γ_{oos} satisfy

$$||w - \gamma_{oos}|| < ||w - \gamma_j||$$

for all $j \in \{1, \ldots, k\}$ including $j = i$. It follows that these points are in the OOS-core. It is clear then that for any $\epsilon > 0$, the ball $B_\epsilon(z)$ contains both points from the i-viable region of $\mathbb{R}^d$ and the OOS-core. This proves the Proposition.

$\square$