Unsupervised Novelty Detection in Pretrained Representation Space with Locally Adapted Likelihood Ratio

Amirhossein Ahmadian **Yifan Ding** **Gabriel Eilertsen** **Fredrik Lindsten**
Linköping University

Abstract

Detecting novelties given unlabeled examples of normal data is a challenging task in machine learning, particularly when the novel and normal categories are semantically close. Large deep models pretrained on massive datasets can provide a rich representation space in which the simple k-nearest neighbor distance works as a novelty measure. However, as we show in this paper, the basic k-NN method might be insufficient in this context due to ignoring the 'local geometry' of the distribution over representations as well as the impact of irrelevant 'background features'. To address this, we propose a fully unsupervised novelty detection approach that integrates the flexibility of k-NN with a locally adapted scaling of dimensions based on the 'neighbors of nearest neighbor' and computing a 'likelihood ratio' in pretrained (self-supervised) representation spaces. Our experiments with image data show the advantage of this method when off-the-shelf vision transformers (e.g., pretrained by DINO) are used as the feature extractor without any fine-tuning.

1 INTRODUCTION

Detecting inputs which are dissimilar from training data is a challenging and important problem in machine learning, with various real-world applications, from boosting safety in AI systems to medical diagnosis, and discovering new phenomena/categories in science (Yang et al., 2021; Ruff et al., 2021).

In a novelty detection task, data can belong to multiple semantic classes, and the aim is to detect any novel data point at test time which comes from a class not present in the training set. This problem is particularly hard when the only training examples are unlabeled normal data, and the novel data are close to them in terms of semantic description. In the image domain, this often means that both normal and novel data are in the same coarse object category while belonging to different fine-grained classes. For instance, one might be interested in detecting novel 'species' of 'birds' given unannotated examples of several known bird species. We will refer to this problem as *unsupervised fine-grained novelty detection*, which can be also identified as a type of out-of-distribution (OOD) detection with unlabeled multi-class in-distribution data and near-OOD (Yang et al., 2021).

From a deep learning perspective, one obstacle in novelty detection is to learn a good *representation space* in absence of any supervision. Regardless of the adopted loss function (e.g., compactness, or self-supervised), training a feature extractor, or sometimes even fine-tuning it, only on unlabeled normal data can potentially result in learning features that are not informative enough for classifying normal data versus the novel ones. This is because the model is likely to become invariant to the unknown (and possibly subtle) 'directions of variations' along which novelties are observable (Reiss et al., 2021). One way to avoid such collapsed representations is to use a powerful and general-purpose deep model (so-called *foundation model*; Bommasani et al., 2021) which is *pretrained* on a broad dataset (e.g., ImageNet). The idea is that the representation space of such a feature extractor should be versatile enough to capture semantic differences between the normal and a variety of novel data.

Even with a good representation space, we still face the problem of designing a one-class classifier, or equivalently the *inference method*, that works in this space. Interestingly, it has been shown recently that once a suitable pretrained feature extractor is used, e.g., a vision transformer (ViT) (Dosovitskiy et al., 2021), fairly simple inference methods can lead to promising

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

novelty/OOD detection performance. For instance, a popular approach is to employ the *Mahalanobis distance* (Lee et al., 2018) in the representation space of deep models (Fort et al., 2021; Ren et al., 2021; Z. Zhou et al., 2021). Yet, this method assumes a Gaussian distribution for representations in each class of (labeled) normal data. When the normal data are unlabeled and perhaps coming from a multimodal distribution, the non-parametric *k-nearest neighbor* (i.e., using k-NN distance as novelty score) is one of the most feasible inference methods. Despite its simplicity, several recent studies demonstrate that k-NN leads to a competitive performance in novelty/OOD detection when it is combined with representations of deep models (Sun et al., 2022; Bergman et al., 2020; Reiss et al., 2021). Another reason for the recent interest in this classical method is the new implementations for efficient large-scale nearest neighbors search, e.g., the Faiss library (Johnson et al., 2019).

However, as we argue in this paper, a drawback of using conventional k-NN distance in unsupervised novelty detection is that the *local geometry* of normal data, or their statistical properties in each neighborhood of space, is not taken into account. Particularly, studying a type of state-of-the-art *self-supervised models*, we find that the data in their representation space seem to lie on a relatively low-dimensional manifold in each local region. The symmetric distance used in ordinary k-NN can be sufficient if normal and novel classes are far away in the representation space, but it may fail for novelties that are semantically close. Indeed, another useful source of information for detecting close novelties is the 'direction' of the difference between the input and its nearest neighbor with respect to the 'local variations' around the neighbor. Thus, we propose to enhance the k-NN by adding a *local adaptation* step, which scales the space dimensions according to local variances estimated on the neighborhood of the nearest neighbor of the input. More formally, we model the distribution of normal data through *local Gaussian* distributions on each Voronoi cell of representation space, fitted on the neighbors of each training point. Moreover, inspired by the arguments for *likelihood ratio* in some former studies (e.g., Ren et al., 2019), we conjecture that a subset of dimensions in a pretrained representation space correspond to features not semantically relevant for detecting novelties. In order to reduce the influence of such 'background' features, we propose the final score as the ratio between the local likelihood and the likelihood of a low-capacity background model (uni-modal Gaussian) that is fitted to the entire training data in representation space. More specifically, in this paper:

- We propose an unsupervised novelty detection method based on pretrained self-supervised models (particularly ViTs), and a probabilistic model fitted to the local regions of representation space. Our approach does not need any labels in either of downstream or pretraining stages, and is implementable as a lazy (training-free) algorithm.

- We evaluate the proposed method in fine-grained novelty detection on images of objects (e.g., Birds and Flowers) as well as a medical image dataset. Without any fine-tuning of the feature extractor, our method shows a considerably better performance than k-NN, and mostly outperforms other compared methods.

- We compare our method to a concurrent work by Nizan and Tal (2024) which has a similar idea of locally adapted k-NN, and show that it performs better or at least on par with their method, in addition to its different probabilistic flavor.

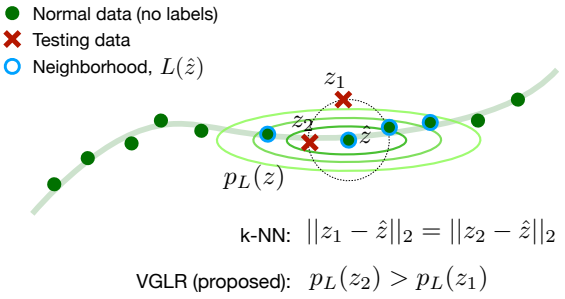

Figure 1: A 2-D toy example to show the flaw of nearest neighbor distance when normal data lie on a manifold, and the test data are close relative to distances within normal data. $\hat{z}$ is the nearest neighbor of both z_1 and z_2. Our method tackles this by fitting a local Gaussian distribution around $\hat{z}$ (section 3.2).

2 MOTIVATION: K-NN MISSES LOCAL GEOMETRY AND BACKGROUND FEATURES

Given a training set D_T with Q data points, the basic k-NN novelty/OOD detection method uses the distance of an input data point to its kth nearest neighbor in D_T as the novelty score (Sun et al., 2022). That means, the score assigned to input z is $d(z, z'_k)$ where $z'_i \in D_T$, $d(z, z'_1) \leq d(z, z'_2) \leq ... \leq d(z, z'_Q)$, and d is usually a common distance like ℓ^2. The data points can be in any representation space, and may be obtained, for instance, by encoding images using a neural network.

We first highlight a limitation of k-NN novelty detection through an example. Figure 1 illustrates a simple

toy problem in a two dimensional space where the normal data lie on a manifold (curve). Assuming $k = 1$ and using Euclidean distance here, query points z_1 and z_2 get an equal score ($||z_1 - \hat{z}||_2 = ||z_2 - \hat{z}||_2$) while z_2 obviously fits better into the local manifold structure of normal data. Thus, it is intuitive in this example to assign a higher novelty score to z_1, which is impossible for the vanilla nearest neighbor method. We should note, however, that this failure also depends on the relative distances between normal and novel points (i.e., in the same example, k-NN could work well if the normal points on the curve were much closer to each other).

Inspired by this, we form the hypothesis that the performance of k-NN novelty detection in a given representation space is particularly impaired when: 1) normal and novel data points are relatively *close* (e.g, in Euclidean distance), and 2) the normal points are *locally* structured on *low dimensional* subspaces. The first condition is likely to happen in fine-grained novelty detection. Investigating the Local Intrinsic Dimensionality (LID) (Amsaleg et al., 2018) of normal data in the representation space of our pretrained models indicates that the second condition should also hold in this problem. LID is loosely defined as the number of dimensions of underlying manifold which the neighbors of each data point are sampled from. Figure 2 (left) shows that in the case of a DINO ViT model (Caron et al., 2021), LID is varying and has an average much smaller than the dimensions of representation space.

Another aspect that is overlooked in k-NN novelty detection is the problem of *background features*. This problem has been observed with other methods as well, such as the Mahalanobis distance and generative models (Ren et al., 2021; Schirrmeister et al., 2020). In this context, background features refer to the dimensions of representation space that are not relevant for distinguishing novel data from normal ones. In practice, a background feature in images can correspond to anything apart from the main object of interest such as the background scene, or low-level features (e.g., brightness and smoothness).

To see whether such irrelevant features can be a concern in our novelty detection tasks, we examine the contributions of different dimensions of representation space in classifying novel vs. normal data while pretending that the task is supervised. Figure 2 (right) shows the histogram of coefficients obtained by training a linear classifier on the labeled PCAM data using representations of a pretrained ViT. One can easily verify that the distribution of coefficients magnitude is far from uniform here, which suggests that different dimensions can vary largely in terms of their discriminative value for classification. This example demonstrates that the assumption of equally weighted dimensions that is implicit in k-NN novelty score can be too unrealistic in practice.

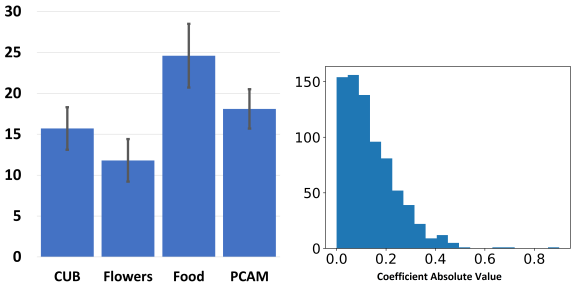

Figure 2: (*Left*) Mean and SD of Local Intrinsic Dimensionality estimated by the method of Amsaleg et al. (2018) (using 50 neighbors) on four datasets in the representation space of a DINO ViT-B/16, pretrained on ImageNet, with 768 dimensions. (*Right*) Histogram of the absolute value of coefficients of a logistic regression classifier (with test accuracy 84.7%) trained on the (standardized) representations of PCAM data with binary labels, using the same ViT model.

3 NOVELTY DETECTION USING PRETRAINED MODELS

3.1 Problem Statement

We define unsupervised novelty detection as a type of binary classification task with super-classes *normal* (in-distribution) D_I and *novel* (OOD) D_O while only samples from D_I are observed in training data. The sets D_I and D_O are disjoint, where each can be further partitioned into one or more classes as $D_I = C_1^I \cup ... \cup C_q^I$, and $D_O = C_1^O \cup ... \cup C_p^O$. However, the class labels are not available in the training data. We are specifically interested in problems where all the classes in $D_I \cup D_O$ are semantically close (e.g., each C_i^I or C_i^O is a bird species). The training set $D_T \subset D_I$ is assumed to include examples from every C_i^I class. It is noted that we allow the normal data to be multi-class, which is more challenging than the more common 'one vs. rest' (unimodal) novelty detection setup (Ruff et al., 2018; Bergman et al., 2020; Gong et al., 2019).

Moreover, we assume access to a pretrained deep *feature extractor* (encoder) model. This model returns a representation vector $z = f(x)$ given an image x, where $z \in \mathbb{R}^m$. More specifically, off-the-shelf foundation models like ViTs are employed as feature extractor in this work, which are pretrained using self-supervised methods on a large unlabeled dataset (ImageNet). We choose two recent and promising self-

distillation based methods, namely DINO (Caron et al., 2021) and iBOT (J. Zhou et al., 2021), for pretraining in practice. One particular reason for this choice is the advantage of such methods (when used with ViTs) in terms of downstream k-NN classification performance, which makes them a good candidate for an unsupervised nearest neighbor based setup as well.

Following the conventions in novelty/OOD detection, we pose the detection method at high level as a *score function* $s(x)$, and express the decision rule as: x is OOD if $s(x) > \tau$, where τ is a user specified threshold. Novelty detection methods are always applied in a given representation space in this study, and thus the score can be also written as $s(z) : \mathbb{R}^m \to \mathbb{R}$.

3.2 Voronoi-Gaussian Model

Motivated by the discussion in section 2, we propose a probabilistic model for novelty detection that combines the nearest neighbors search of k-NN with a locally adapted distance to incorporate the geometry of data around the nearest neighbor. The key idea is to partition the representation space into nearest neighbor regions, that is, *Voronoi cells* corresponding to the normal training data points, and assign a multivariate Gaussian probability density function to each region. The parameters of the Gaussian distribution of a cell are calculated based on the cell's generator data point in the training set as well as the training points which are in the local neighborhood of generator.

Let $\hat{z}$ be the *nearest neighbor* of input z among the training data in the representation space:

$$\hat{z} = \underset{\tilde{z} \in D_T}{\arg\min} ||z - \tilde{z}||_2 \tag{1}$$

This implies that z is inside the Voronoi cell generated by $\hat{z}$, which we write as $z \in \mathrm{V}(\hat{z})$. Then, we define a *local density* function as the following, which the novelty score of input is derived from:

$$p_L(z) = \mathcal{N}(z; \hat{z}, \hat{\Sigma}), \tag{2}$$

where the *local covariance* matrix $\hat{\Sigma} = \Sigma_{L(\hat{z})}$ is estimated on the set of data points that lie in the local neighborhood of $\hat{z}$, which is $L(\hat{z}) \subset D_T$. The point $\hat{z}$, the generator of the Voronoi cell, is designated as the center/mean of the Gaussian distribution. Concretely, we specify the local neighborhood of $\hat{z}$ to be the set of its k' nearest data representations in the training set, $L(\hat{z}) = \mathrm{NN}_{k'}(\hat{z})$, where k' is a hyperparameter (thus, the method looks at the 'nearest neighbors of nearest neighbor', similarly to the concurrent work by Nizan and Tal, 2024). The role of $\hat{z}$ can be interpreted as a selected prototype, to which the similarity of z is measured using eq. (2). In local learning view (Bodesheim et al., 2015), obtaining $\hat{z}$ and its neighbors is like a 'coarse recognition' stage where the ordinary ℓ^2 distance is used to consider the overall similarities of images. This is followed by another stage in which the more precise local model p_L, fitted only to a small relevant neighborhood, decides the score.

The matrix $\hat{\Sigma}$ is obtained by maximum likelihood estimation of covariance on the data representations in $L(\hat{z})$ assuming that the mean is equal to $\hat{z}$. This local parameter aims at capturing the variations of data in the neighborhood $L(\hat{z})$ with respect to $\hat{z}$, which is in turn used to evaluate the typicality of the relation between $\hat{z}$ and the test input z. From a geometrical perspective, this can be understood as using a Gaussian distribution to approximately model the manifold of data representations around the nearest neighbor of the input, and predict whether the input is likely to be in the vicinity of the manifold (figure 1). With manifold theory assumptions (Pidhorskyi et al., 2018), the normal data in $L(\hat{z})$ can exhibit two types of variations: along the directions tangent to the manifold at $\hat{z}$ as well as the ones orthogonal to it (due to noise). A locally fitted Gaussian distribution should be enough in principle to learn the direction and scale of both types of variations.

In practice, the representation space is high dimensional (e.g., $m = 768$ in our experiments) and the data for local estimation is sparse (e.g., $k' = 10$). We therefore always assume $\hat{\Sigma}$ to be diagonal [1] .

Given Q training data points, we have the same number of Voronoi cells in the representation space. This also means that Q local (unnormalized) distributions can be defined using eq. (2), where the support of each distribution is $\mathrm{V}(z_i)$ for $z_i \in D_T$. We can derive a probability density function with a support on the entire $\mathbb{R}^m$ by summing up the local densities, and dividing by a constant to normalize the distribution. Therefore, the **Voronoi-Gaussian model** of normal data is:

$$p(z) = \frac{1}{\alpha} \sum_{z_i \in D_T} \mathbf{1}\{z \in V(z_i)\} \mathcal{N}(z; z_i, \Sigma_{L(z_i)}) = \frac{1}{\alpha} p_L(z) \tag{3}$$

where $\mathbf{1}$ is the indicator function, α is a normalization constant, and p_L was defined in eq. (2).

3.3 Likelihood Ratio Score

One general approach to handle background features in novelty/OOD detection is to pose the score function as the ratio between the likelihood of a model

[1] We did not observe better results by using a full matrix with regularization (e.g., covariance shrinkage) in practice, perhaps because the problem is extremely high dimensional relative to the number of local data points.

representing normal data and an auxiliary *background model* (Ren et al., 2019; Ren et al., 2021; Serrà et al., 2020; Schirrmeister et al., 2020; Gangal et al., 2020). Following Ren et al. (2019), we assume that the data distribution (for both normal and novel classes) is *factorized* into independent background and semantic components, albeit in the representation space. That is, $z = \{z_B, z_S\}$ and $p(z) = p(z_B)p(z_S)$, where z_B corresponds to nuisance background dimensions, and z_S denotes the more interesting semantic (discriminative) ones. Moreover, the existence of a background model $p'(z)$ is assumed which aims to learn the background component but is not concerned (ideally) with the semantic part of data. More formally, the distribution over background component is shared between the data and background models, that is, $p'(z) = p(z_B)p'(z_S)$.

The choice for background model has been addressed in the literature with different heuristics and assumptions, such as training a background model on perturbed data (Ren et al., 2019), or using an external generic dataset (Schirrmeister et al., 2020). We propose to tackle this in a 'bootstrapping' fashion, where the background model is a low capacity model estimated on the normal training data themselves. Concretely, we simply use a *Gaussian distribution with diagonal covariance* fitted on training data representations as $p'(z)$. In practice, this is similar to the method of Ren et al. (2021) for enhancing Mahalanobis distance. Finally, using eq. (3) for the distribution of normal data, our **proposed novelty score** is defined as the following log-likelihood ratio:

$$s(z) = -\log \frac{p(z)}{p'(z)} = -\log p_L(z_S) + \log p'(z_S) \quad (4)$$

where α in $p(z)$ is omitted since it only shifts the score by a constant. Thus, z_B is eliminated by the likelihood ratio trick (under the idealized assumption $p(z_B) = p'(z_B)$), though at the cost of biasing the score by the term $\log p'(z_S)$. However, we argue that in effect, this term does not influence our novelty detection performance considerably. To see this, let us consider $z^I \sim p^I(z)$ and $z^O \sim p^O(z)$ where p^I and p^O are the distributions of normal and novel data representation respectively. Then, we would like to have the following expectation positive:

$$\begin{aligned}\mathbb{E}[s(z^O) - s(z^I)] &= \{\mathbb{E}_{p^I}[\log p_L(z_S^I)] - \mathbb{E}_{p^O}[\log p_L(z_S^O)]\} \\ &\quad + \{\mathbb{E}_{p^O}[\log p'(z_S^O)] - \mathbb{E}_{p^I}[\log p'(z_S^I)]\} \\ &= \Delta_1 + \Delta_2\end{aligned} \quad (5)$$

Since the p_L and p' models are fitted to the normal data (and the novel data are OOD w.r.t them), we should have $\Delta_1 > 0$ and $\Delta_2 < 0$. However, defining $p'(z)$ to be a low capacity parametric model (independent Gaussian variables) compared to the locally adapted $p_L(z)$, and also assuming that the normal and novel data are not very far in their global (coarse) statistics in semantic dimensions, we expect the log-likelihood gap between in-distribution and OOD data to be larger in the case of p_L (i.e., $|\Delta_1| > |\Delta_2|$). Therefore, at least on average, $s(z)$ should be still larger on novel inputs than the normal ones [2].

As an alternative method for obtaining a background model, we also tried a multivariate Gaussian distribution estimated on the representations of the ImageNet dataset. Its results with non-fine-tuned models (reported in the appendix) are comparable overall to the method above, though this alternative method will work poorly if the distribution of normal representations is too far from ImageNet, which specifically happens after fine-tuning the feature extractor.

Algorithm 1 VGLR Novelty Detection

Input: image x, training representations D_T, constants k', τ, and feature extractor model f.
Representation: $z = f(x)$
Background model: $p_B(z) = \mathcal{N}(z; \mu_b, \Sigma_b)$ where μ_b and diagonal Σ_b are estimated by MLE on D_T
Nearest neighbor: $\hat{z} = argmin_{\tilde{z} \in D_T} ||z - \tilde{z}||_2$
Find k' nearest neighbors of $\hat{z}$ in $D_T - \{\hat{z}\}$, that is, $\mathrm{NN}_{k'}(\hat{z})$
Local model: $p_L(z) = \mathcal{N}(z; \hat{z}, \hat{\Sigma})$, where $\hat{\Sigma}$ is diagonal, and $\hat{\Sigma}_{ii} = 1/k' \sum_{\tilde{z} \in \mathrm{NN}_{k'}(\hat{z})} (\tilde{z}_i - \hat{z}_i)^2 + \epsilon$ for each dimension i (and ϵ is a small constant)
Novelty score: $s(z) = -\log p_L(z) + \log p_B(z)$
if $s(z) > \tau$ **then**
    Classify x as novel
else
    Classify x as normal
end if

4 RELATED WORK

In this section, we use the term **out-of-distribution (OOD) detection** to refer to any problem where abnormal inputs come from a data distribution with some discrepancy from the distribution of training data (Yang et al., 2021), including novelty detection. OOD detection approaches can be divided into two broad (overlapping) categories from a *transfer learning* perspective. The methods in the first category are most suitable for training machine learning models from scratch, often with an objective function and/or

[2] Of course, this does not tell us about the overlap between the distributions of normal/novel scores. However, we also empirically observe (section 5.3) that using the likelihood ratio never impairs the baseline mean AUROC.

model tailored specifically for OOD detection. Examples of such methods are Deep Support Vector Data Description (Ruff et al., 2018), memory augmented (Gong et al., 2019) and adversarial (Pidhorskyi et al., 2018) autoencoders, augmented contrastive learning (Tack et al., 2020), self-distillation with negative sampling (Rafiee et al., 2022), and detecting 'latent holes' after compactification of the latent space of generative models (Glazunov and Zarras, 2023).

The second category, which our method falls into, comprises methods that primarily aim at leveraging a pretrained deep classifier or generic feature extractor model in OOD detection (*post hoc*, or transfer learning OOD detection). Many of these methods, e.g., ODIN (Liang et al., 2018), energy based OOD (Liu et al., 2020), and Mahalanobis distance (Lee et al., 2018) are essentially derived from a (supervised) classifier's softmax scores or representations, which means they are only applicable when the normal data have labels, or alternatively have been clustered beforehand (Sehwag et al., 2021).

A trending approach in post hoc OOD detection is to use **k-Nearest Neighbor (k-NN)** search at the core of the inference method. Sun et al. (2022) show that using k-NN distance in representation space of a deep model as the OOD score is competitive with more complicated popular methods. However, before applying the unsupervised k-NN, they use labeled normal data to train their deep model. k-NN has been also employed for unsupervised OOD (anomaly) detection with deep feature extractors (ResNet) pretrained on ImageNet by Bergman et al. (2020) and Reiss et al. (2021), where in the latter, the model is fine-tuned on normal data using compactness loss. Raghuram et al. (2021) use k-NN search on representations of labeled normal data to obtain statistics for OOD detection with pretrained deep classifiers.

Our proposed local adaptation method is related to the concept of **local learning** (Bottou and Vapnik, 1992), that is, training/adapting a model on a subset of training data which are most similar to input. Leveraging this idea, Bodesheim et al. (2015) propose the Local KNFST novelty detection method, where the input is mapped to Kernel Null Foley-Sammon space (Bodesheim et al., 2013) of its k-nearest data points, and the score is calculated as a distance to class prototypes in this space. Verdier and Ferreira (2010) propose a method to detect faults (anomalies) in semiconductor manufacturing based on estimating the mean and covariance of Mahalanobis distance locally, i.e., on the nearest neighbors of input. However, they only consider low dimensional problems in a certain industrial domain.

In a concurrent work with ours, Nizan and Tal (2024) propose the *Nearest Neighbors of Neighbors* (k-NNN) method to extend k-NN to a locally adapted operator which considers the structure of data in a pretrained representation space. To this end, PCA is run on the k'-nearest neighbors of a given ith nearest neighbor of input. Then, each dimension is weighted proportionally to its dot product with the local eigenvectors and corresponding inverse eigenvalues to obtain a distance function. k-NNN, and its underlying motivation, is very close to the local adaptation part of our proposed method. Yet, unlike their formulation, we take a probabilistic perspective to the problem, where the data distribution is modeled as the sum of defined local (Voronoi-Gaussian) densities. Moreover, in the same framework, our score function is posed as the ratio between the log-likelihood of data and background distributions. Besides, unlike our diagonal local covariance, the covariance for PCA is assumed as block diagonal in k-NNN, where the block size is determined by a hyperparameter after a heuristic sorting of variables.

5 EXPERIMENTAL RESULTS

Methods. We present the empirical results of our *Voronoi-Gaussian with Likelihood Ratio* (VGLR) method (algorithm 1) on four different unsupervised fine-grained novelty detection problems in image domain [3]. The performance is compared to the baseline *k-NN* (with Euclidean distance), and four other unsupervised methods from the literature, where all methods are applied in the pretrained representation spaces. The closely related *k-NNN* (Nizan and Tal, 2024) is one of the compared methods. The *Local KNFST* (Bodesheim et al., 2015) method is also selected, as it is an application of the local learning idea in novelty detection. We include *One-Class SVM* (Schölkopf et al., 2001) and a *k-means clustering* based method as well since they are both well-known, and easy to implement without training any extra deep models. K-means is used in combination with Mahalanobis distance in the same way as proposed in the SSD method (Sehwag et al., 2021).

Models. We employ off-the-shelf ViTs that are pretrained using *DINO* and *iBOT* self-supervised methods on the ImageNet-1K dataset. Given an image x, the ℓ^2-normalized *CLS* token at the last layer of the ViT backbone is used as the representation vector $f(x)$. We have experiments with a DINO pretrained ResNet-50 model as well which are reported

[3] The code is available at: https://github.com/aahmadian-liu/pretrained-novelty-loclr

in the appendix, but the best performance over all methods drops considerably when ViT is replaced with ResNet. This should not be surprising as we know that ViTs generally lead to better representations in a self-distillation framework (Caron et al., 2021).

Tasks. We demonstrate novelty detection experiments using the public datasets *CUB-200-2011* (Birds) (Wah et al., 2011), *Flowers-102* (Nilsback and Zisserman, 2008), *Food-101* (Bossard et al., 2014), and *PatchCamelyon* (*PCAM*) (Veeling et al., 2018). All of the datasets consist of RGB images, resized to 224×224 pixels. PCAM is in the medical imaging domain, and is obtained from histopathologic scans of lymph node sections (Bejnordi et al., 2017). To define a novelty detection task, i.e., specifying the normal (D_I) and novel (D_O) sets, it is necessary to choose a subset of classes in each dataset as normal data. We use the following protocol to this end:

- For PCAM, class 0 is always considered *normal*, and class 1 is novel. This is a natural choice because all the samples in the latter class contain metastatic (anomalous) tissues.

- For any other dataset, *half* of the classes (rounded up) are assumed *normal*, and the remaining classes are novel. This is done by choosing the first half of the classes (in the default dataset order) as normal, in addition to randomly choosing half of the classes as normal for 4 times.

Although the datasets are labeled, the labels are *only* used to split the data into normal and novel sets. More specifically, all images in the training partition of the normal set are used for the training data (D_T). For the test data, an equal number of images is used from the test partitions of normal and novel sets. The Flowers-102 dataset has an untypical partitioning, where the 'test' part is much larger than the 'train' part. For this reason, we swap the default partitioning for this particular dataset.

5.1 Results with non-fine-tuned ViTs

Novelty detection performance on test data in terms of threshold-independent *Area under ROC curve* (AUROC) with DINO and iBOT pretrained ViTs is reported in table 1. Here, the publicly available weights of pretrained models[4] is directly used, *without* any fine-tuning. Additionally, performance on the same experiments in terms of FPR@95TPR can be found in the appendix. We choose hyperparameter values for each method such that the overall average AUROC (the mean over the four datasets) on the test data is maximized. This implies that, for a certain method and feature extractor, hyperparameter values are the *same* across all the *tasks/datasets* (more details on hyperparameters is given in the appendix). Specifically, we find that $k = 1$ is the optimal value in k-NN with both DINO and iBOT. Moreover, for k-NNN, we always assume k and s (covariance block size) equal to 1. This leads to having k' (number of neighbors of neighbor) as the only hyperparameter both in our method and k-NNN, for a more fair comparison.

The results clearly show that our method leads to improvement over k-NN, which can be as substantial as around +18% AUROC in the case of PCAM, and +11% average AUROC on Flowers. More notably, the reported performance of VGLR ranks first on all the datasets/models except for Food where it is on par with k-NNN. Interestingly, VGLR with DINO achieves an AUROC about 80% (5% higher than the second-best method) on PCAM, while this dataset is quite different from the pretraining (ImageNet) data. The average performance on Food data is the smallest among the studied datasets, meaning that it is a quite hard case for all the methods. A possible reason can be the noise (wrong labels, and clutter) in training images of Food-101 dataset, as mentioned in its documentation.

As expected, k-NNN seems to be the most competitive method to ours, and ranks second after VGLR in the majority of cases. However, we find our method to be more efficient than k-NNN in terms of number of neighbors (of nearest neighbor) even when the performance is close. Specifically, all results in table 1 are obtained with $k' = 10$ for VGLR, but either $k' = 80$ (DINO) or $k' = 40$ (iBOT) for k-NNN. Moreover, although table 1 reports the average performance over repeated normal/novel splits, we find that on almost all the splits, the best AUROC belongs to either VGLR or k-NNN (the only exception happens with iBOT-Food where k-NN ranks second once; more details given in the appendix).

5.2 Self-supervised fine-tuning of feature extractor

In the next setup, the feature extractor model is fine-tuned on the normal training data before running the methods in its representation space. The loss we use in fine-tuning is the same as the pretraining self-supervised loss. DINO models, and one task (default order of classes) from each dataset are studied under this setup. Specifically, the DINO algorithm, initialized with ImageNet weights, is run for 20 epochs on D_T images of each task (more details in the appendix). In table 2, we show how AUROC performance is changed after fine-tuning the ViT model in each task. The methods hyperparameters on each dataset are set to

[4] https://github.com/facebookresearch/dino, https://github.com/bytedance/ibot

Table 1: Unsupervised novelty detection performance using ViT-B/16 feature extractor pretrained on ImageNet by DINO and iBOT methods (without fine-tuning). The numbers are averages over 5 novel/normal splits, except for the case of PCAM.

	AUROC (%) DINO/iBOT			
Method	CUB	Flowers	Food	PCAM
VGLR (proposed)	**71.3** / **70.5**	**91.8** / **90.6**	66.4 / **65.1**	**80.2** / **78.5**
k-NN	66.0 / 65.8	80.3 / 79.1	63.3 / 63.4	62.2 / 59.9
k-NNN (k=1, s=1)	69.8 / 70.0	90.1 / 89.4	**66.5** / 64.3	74.9 / 73.2
k-means + Maha. (SSD)	63.7 / 64.4	81.6 / 80.0	59.1 / 60.2	68.8 / 66.6
OC-SVM	63.1 / 63.0	77.5 / 76.4	60.3 / 61.0	60.5 / 58.4
Local KNFST	58.8 / 59.5	65.8 / 67.0	53.8 / 52.4	74.1 / 73.1

Table 2: Change in task-specific AUROC performance (%) by fine-tuning the ViT-B/16 using DINO loss on normal data. The last column shows the change in the average over the 4 tasks.

Method	**CUB**	**Flowers**	**Food**	**PCAM**	Average
VGLR (proposed)	80.8 → **76.5**	91.5 → **92.1**	62.9 → 77.2	80.2 → 80.2	78.8 → **81.5**
k-NN	72.9 → 73.9	76.6 → 82.3	60.8 → **80.5**	62.2 → 78.5	68.1 → 78.8
k-NNN (k=1, s=1)	77 → 71.9	90.3 → 89.7	64.3 → 71.0	74.9 → 74	76.6 → 76.6
k-means + Maha. (SSD)	68.2 → 69.1	78.3 → 82.0	56.9 → 78.2	68.8 → **80.6**	68.0 → 77.4
OC-SVM	68.7 → 70.6	73 → 79.2	58.5 → 80.1	60.5 → 80.1	65.1 → 77.5
Local KNFST	58.8 → 61.7	68.3 → 71.4	55.6 → 52.1	74.1 → 78.1	64.2 → 65.8

the same values found with the non-fine-tuned model.

The numbers indicate that the performance of k-NN is consistently increased on all tasks by our self-supervised fine-tuning, though to different degrees. This is anticipated since DINO should move similar data points towards each other in representation space (into semantic clusters, ideally) (Vaze et al., 2022), which in turn means novel data points should get further from normal ones on average. More generally, fine-tuning does not impair the average performance of any method over the studied tasks. Yet, performance decrease happens in a few tasks particularly with VGLR and k-NNN (e.g., on CUB).

Although the overall gap between VGLR and k-NN looks smaller in table 2 than table 1, the proposed method still ranks first in *average* AUROC after fine-tuning. One can also see that the only case where the costly fine-tuning has led to a substantial performance gain is with Food data (around 16% improvement in best AUROC). In other cases, the top performance is either slightly increased (around 0.5% on PCAM and Flowers) or even impaired (-4.3% on CUB).

5.3 Ablation study

The proposed VGLR method was described as the combination of two distinct ideas: a locally adapted Gaussian model, and the likelihood ratio trick. Here, we investigate the role of each component by considering two ablation modes of our method:

(I) *No Local Adaptation*: This means that the background model (likelihood ratio) is used as before but no local scaling of dimensions is performed. Hence, the local covariance ($\hat{\Sigma}$ in eq. (2)) is always set to identity here.

(II) *No Likelihood Ratio*: In this case, the likelihood of the Voronoi-Gaussian model is directly used to obtain the score without dividing by $p'(z)$ (eq. (4)), that is, using $s(z) = -\log p_L(z)$.

Figure 3 illustrates the performance of the two ablated methods on the previous tasks with the non-fine-tuned DINO ViT, and compares them to the original VGLR results. It is verified that generally both of the components are necessary to achieve maximum performance. Remarkably, in the case of PCAM, each of the ablated methods alone has a fairly large performance gap both with the full method and k-NN. We speculate that this is related to the type of features in tissue images that help to detect metastasis; the differences are relatively subtle, and buried in many noisy background features.

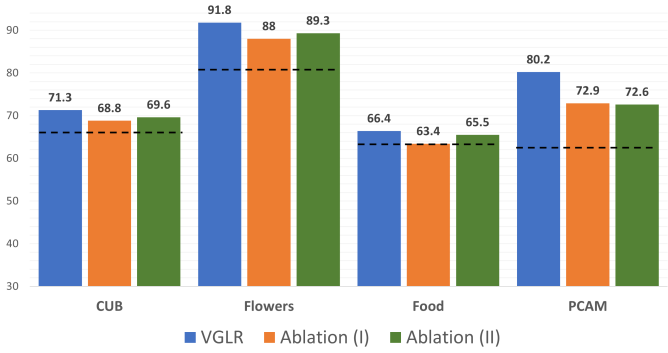

Figure 3: Average AUROC of the ablated methods I and II (i.e., removing either local adaptation or likelihood ratio) compared to the full proposed VGLR method (using DINO ViT-B/16). Dashed lines indicate the vanilla k-NN performance (as in table 1)

6 CONCLUSION

We presented a method to leverage large pretrained (self-supervised) models for novelty/OOD detection in absence of any labeled data. Our method refines the well-known k-NN score in the model representation space by fitting an on-the-fly Gaussian distribution to the neighbors of the input's nearest neighbor (to capture the local geometry), and obtaining its likelihood ratio w.r.t. a simple background Gaussian distribution. Using vision transformer models without any fine-tuning, the AUROC performance of the proposed VGLR method in fine-grained image novelty detection (with DINO/iBOT pretrained models) is remarkably better than k-NN, while k-NN is still competitive with prior methods in most cases. We showed that k-NN performance was improved by fine-tuning the ViT on normal data using DINO, but VGLR still ranked first in average over datasets. Thus, the advantage of VGLR seems most prominent with non-fine-tuned pretrained models, which should also make it a favorable method in general, considering the cost of fine-tuning foundation models.

The differences in behavior of VGLR after fine-tuning the encoder can be partly related to the collapse of representation space (i.e., a type of overfitting to in-distribution data) caused by the downstream self-supervised training. Also, it should be considered that the performance of our method can depend on how 'near' the OOD data are. As an extreme case, the in-distribution data points can be so concentrated in the representation space (w.r.t OOD points) that the vanilla nearest neighbor distance is already the optimal score. In that situation, any weighting on the space dimensions using our local/background model might lead to unintended effects. Thus, one idea for future could be to aim at a combination of the proposed VGLR and ordinary k-NN scores to enhance both near and far OOD detection.

References

Amsaleg, Laurent et al. (2018). "Extreme-value-theoretic estimation of local intrinsic dimensionality". In: *Data Mining and Knowledge Discovery* 32.6, pp. 1768–1805.

Bejnordi, Babak Ehteshami et al. (2017). "Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer". In: *Jama* 318.22, pp. 2199–2210.

Bergman, Liron, Niv Cohen, and Yedid Hoshen (2020). "Deep nearest neighbor anomaly detection". In: *arXiv preprint arXiv:2002.10445*.

Bodesheim, Paul, Alexander Freytag, Erik Rodner, and Joachim Denzler (2015). "Local novelty detection in multi-class recognition problems". In: *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE, pp. 813–820.

Bodesheim, Paul, Alexander Freytag, Erik Rodner, Michael Kemmler, and Joachim Denzler (2013). "Kernel null space methods for novelty detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3374–3381.

Bommasani, Rishi et al. (2021). "On the opportunities and risks of foundation models". In: *arXiv preprint arXiv:2108.07258*.

Bossard, Lukas, Matthieu Guillaumin, and Luc Van Gool (2014). "Food-101 – Mining Discriminative Components with Random Forests". In: *European Conference on Computer Vision*.

Bottou, Léon and Vladimir Vapnik (1992). "Local learning algorithms". In: *Neural computation* 4.6, pp. 888–900.

Caron, Mathilde et al. (2021). "Emerging properties in self-supervised vision transformers". In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660.

Dosovitskiy, Alexey et al. (2021). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*.

Fort, Stanislav, Jie Ren, and Balaji Lakshminarayanan (2021). "Exploring the limits of out-of-distribution detection". In: *Advances in Neural Information Processing Systems* 34, pp. 7068–7081.

Gangal, Varun, Abhinav Arora, Arash Einolghozati, and Sonal Gupta (2020). "Likelihood ratios and generative classifiers for unsupervised out-of-domain detection in task oriented dialog". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05, pp. 7764–7771.

Glazunov, Misha and Apostolis Zarras (2023). "Vacant holes for unsupervised detection of the outliers in compact latent representation". In: *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*. Vol. 216. Proceedings of Machine Learning Research. PMLR, pp. 701–711.

Gong, Dong et al. (2019). "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1705–1714.

Johnson, Jeff, Matthijs Douze, and Hervé Jégou (2019). "Billion-scale similarity search with GPUs". In: *IEEE Transactions on Big Data* 7.3, pp. 535–547.

Lee, Kimin, Kibok Lee, Honglak Lee, and Jinwoo Shin (2018). "A simple unified framework for detecting out-of-distribution samples and adversarial attacks". In: *Advances in neural information processing systems* 31.

Liang, Shiyu, Yixuan Li, and R Srikant (2018). "Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks". In: *International Conference on Learning Representations*.

Liu, Weitang, Xiaoyun Wang, John Owens, and Yixuan Li (2020). "Energy-based out-of-distribution detection". In: *Advances in neural information processing systems* 33, pp. 21464–21475.

Nilsback, Maria-Elena and Andrew Zisserman (2008). "Automated flower classification over a large number of classes". In: *2008 Sixth Indian conference on computer vision, graphics & image processing*. IEEE, pp. 722–729.

Nizan, Ori and Ayellet Tal (2024). "K-NNN: Nearest Neighbors of Neighbors for Anomaly Detection". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*, pp. 1005–1014.

Paszke, Adam et al. (2019). "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pidhorskyi, Stanislav, Ranya Almohsen, and Gianfranco Doretto (2018). "Generative probabilistic novelty detection with adversarial autoencoders". In: *Advances in neural information processing systems* 31.

Rafiee, Nima, Rahil Gholamipoor, Nikolas Adaloglou, Simon Jaxy, Julius Ramakers, and Markus Kollmann (2022). "Self-supervised anomaly detection by self-distillation and negative sampling". In: *International Conference on Artificial Neural Networks*. Springer, pp. 459–470.

Raghuram, Jayaram, Varun Chandrasekaran, Somesh Jha, and Suman Banerjee (2021). "A general framework for detecting anomalous inputs to dnn classifiers". In: *International Conference on Machine Learning*. PMLR, pp. 8764–8775.

Reiss, Tal, Niv Cohen, Liron Bergman, and Yedid Hoshen (2021). "Panda: Adapting pretrained features for anomaly detection and segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2806–2814.

Ren, Jie, Stanislav Fort, Jeremiah Liu, Abhijit Guha Roy, Shreyas Padhy, and Balaji Lakshminarayanan (2021). "A simple fix to mahalanobis distance for improving near-ood detection". In: *arXiv preprint arXiv:2106.09022*.

Ren, Jie et al. (2019). "Likelihood Ratios for Out-of-Distribution Detection". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc.

Ruff, Lukas et al. (2018). "Deep one-class classification". In: *International conference on machine learning*. PMLR, pp. 4393–4402.

Ruff, Lukas et al. (2021). "A unifying review of deep and shallow anomaly detection". In: *Proceedings of the IEEE* 109.5, pp. 756–795.

Schirrmeister, Robin, Yuxuan Zhou, Tonio Ball, and Dan Zhang (2020). "Understanding anomaly detection with deep invertible networks through hierarchies of distributions and features". In: *Advances in Neural Information Processing Systems* 33, pp. 21038–21049.

Schölkopf, Bernhard, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson (2001). "Estimating the support of a high-dimensional distribution". In: *Neural computation* 13.7, pp. 1443–1471.

Sehwag, Vikash, Mung Chiang, and Prateek Mittal (2021). "SSD: A Unified Framework for Self-Supervised Outlier Detection". In: *International Conference on Learning Representations*.

Serrà, Joan, David Álvarez, Vicenç Gómez, Olga Slizovskaia, José F. Núñez, and Jordi Luque (2020). "Input Complexity and Out-of-distribution Detection with Likelihood-based Generative Models". In: *International Conference on Learning Representations*.

Sun, Yiyou, Yifei Ming, Xiaojin Zhu, and Yixuan Li (2022). "Out-of-distribution detection with deep nearest neighbors". In: *International Conference on Machine Learning*. PMLR, pp. 20827–20840.

Tack, Jihoon, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin (2020). "Csi: Novelty detection via

contrastive learning on distributionally shifted instances". In: *Advances in neural information processing systems* 33, pp. 11839–11852.

Vaze, Sagar, Kai Han, Andrea Vedaldi, and Andrew Zisserman (2022). "Generalized category discovery". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7492–7501.

Veeling, Bastiaan S, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling (June 2018). "Rotation Equivariant CNNs for Digital Pathology". In: arXiv: 1806.03962 [cs.CV].

Verdier, Ghislain and Ariane Ferreira (2010). "Adaptive Mahalanobis distance and k-nearest neighbor rule for fault detection in semiconductor manufacturing". In: *IEEE Transactions on Semiconductor Manufacturing* 24.1, pp. 59–68.

Wah, C., S. Branson, P. Welinder, P. Perona, and S. Belongie (2011). *Caltech-UCSD Birds-200-2011 Dataset*. Tech. rep. CNS-TR-2011-001. California Institute of Technology.

Yang, Jingkang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu (2021). "Generalized out-of-distribution detection: A survey". In: *arXiv preprint arXiv:2110.11334*.

Zhou, Jinghao et al. (2021). "ibot: Image bert pre-training with online tokenizer". In: *arXiv preprint arXiv:2111.07832*.

Zhou, Zhi, Lan-Zhe Guo, Zhanzhan Cheng, Yu-Feng Li, and Shiliang Pu (2021). "Step: Out-of-distribution detection in the presence of limited in-distribution labeled data". In: *Advances in Neural Information Processing Systems* 34, pp. 29168–29180.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model: **Yes**, section 3 states all the problem assumptions and formulation of the proposed method, and also more concrete details are given in section 5.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm: **Yes**, comments on complexity will be given in the appendix.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries: **Yes**, the code will be uploaded in the supplementary material.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results: **Not Applicable**, we do not make mathematical claims in this work in the form of theorems, lemmas, or similar.
   (b) Complete proofs of all theoretical results: **Not Applicable**
   (c) Clear explanations of any assumptions: **Not Applicable**

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL): **Yes**, the code will be uploaded in the supplementary material.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **Yes**, some description of the data/task settings and methods hyperparameters is given in section 5, and more details are provided in the appendix.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **Yes**, we explain how the reported means or SDs are obtained in each case (e.g., in figure 2 and table 1), and we report well-known performance measures such as AUROC as the results.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **Yes**, these details will be covered in the appendix.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. **Yes**, when using GitHub repositories and datasets, the corresponding paper is cited as instructed by their authors.
   (b) The license information of the assets, if applicable. **Yes**, the used code/model files from GitHub are open source (e.g., with Apache License). Citation to the original repository is given in our code.
   (c) New assets either in the supplemental material or as a URL, if applicable. **Yes**, the URL to our GitHub repository is provided in the paper.
   (d) Information about consent from data providers/curators. **Not Applicable**, the

used datasets are publicly available and free to use at least for non-commercial purposes.

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content: **Not Applicable**

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots: **Not Applicable**

   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable: **Not Applicable**

   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation: **Not Applicable**

A.1 Additional details on experiments and methods

In this section, we provide more details regarding the datasets, implementation, and choosing the hyperparameters of each method.

A.1.1 Datasets

Some more information about how we use the datasets in our experiments, such as number of the categories (classes), and number of images (instances) are given in table 3. The numbers are specified for the normal and novel parts of data, which are defined according to our protocol described in section 5 of the paper. Since the number of training/test images can differ among the randomized choice of normal/novel classes (the 5 splits), the rounded average is reported in the table. Moreover, figure 4 shows a few example images from each dataset.

Table 3: Number of the categories and images of each dataset used in our unsupervised novelty detection tasks. We never use novel images in the training data. The number of normal and novel images in test data are always equal.

	CUB-200	Flowers-102	Food-101	PCAM
Normal categories	100	51	51	1
Novel categories	100	51	50	1
Normal training images (average)	4417	2896	38250	131072
Normal test images (average)	1446	510	12500	16377
Novel test images (average)	1446	510	12500	16377

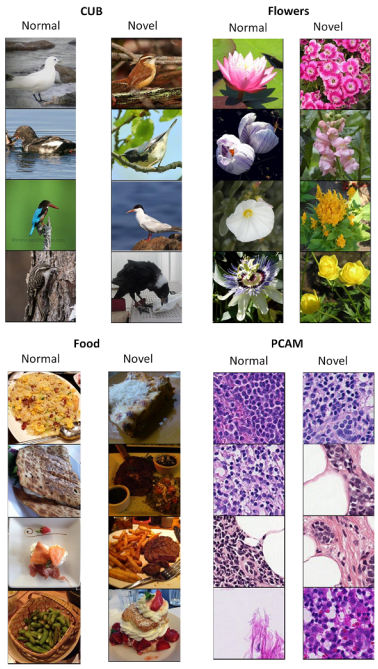

Figure 4: Example images from the normal and novel sets of the data used in our experiments. Here, the tasks considered for CUB, Flowers, and Food datasets are defined by assuming the first half of the classes as normal.

A.1.2 Software and hardware specifications

All the experiments are implemented in Python 3. The feature extraction part using pretrained models uses the PyTorch library (Paszke et al., 2019), and nearest neighbors search (in k-NN, VGLR, k-NNN, and Local

KNFST) is performed using the Faiss library (Johnson et al., 2019) . Scikit-learn library (Pedregosa et al., 2011) is used for OC-SVM and k-means.

We use a single GPU (e.g., NVIDIA RTX 2080TI) to obtain the representations vectors for one time, and store them to disk. Then, we run the novelty detection methods on CPU (e.g, Intel Xeon E5-1620 3.5GHz). Using CPU, our VGLR method takes about a few minutes (e.g., around 4 minutes) to run on the whole test data representations of PCAM dataset (the largest dataset). However, this can be speeded up by using GPU (the Faiss library has GPU support for neighbors search).

A.1.3 Methods and hyperparameters

In order to choose the value of a hyperparameter λ given a method (and feature extractor) through searching over a set of candidate values Λ, we find the $\lambda^* \in \Lambda$ that maximizes the mean AUROC over all the datasets. The mean AUROC here refers to the mean of the AUROCs of the method on the test data of each of the 4 datasets, where each AUROC value in turn corresponds to the average over 5 tasks, or a single task (in case of PCAM). When having multiple hyperparameters, the search is done over all the combinations of them. Eventually, we fix the hyperparameters in each method to the optimum values (keeping them constant across all the tasks) to obtain the final performance values reported in our results tables.

Table 4 summarizes the hyperparameter values used for each method with the pretrained DINO ViT feature extractor. All the hyperparameter values are the same in the case of iBOT ViT except k' in k-NNN, which is equal to 40 there. Here, 'standardized' refers to standardizing the representation vectors using mean and standard deviation (the vectors are always normalized in l^2 norm at first). The Euclidean (l^2) distance is used to define nearest neighbors in any method. We also provide more implementation details on each method in the following.

Table 4: Hyperparameter values of each method used in our experiments with DINO ViT-B/16 model.

Method	Hyperparameter values
VGLR	$k' = 10$, standardized
k-NN	$k = 1$
k-NNN	$k' = 80, k = 1, s = 1$
k-means	$k = 100$
OC-SVM	kernel='rbf', $\nu = 0.5, \gamma = 0.02$, standardized
Local KNFST	kernel='rbf', $\gamma = 0.02$, standardized

VGLR. The main hyperparameter of our proposed method is k', that is the number of neighbors of the nearest neighbor, used to calculate the local variances. We search over the values $k' \in \{10, 20, 40, 80\}$. As an additional hyperparameter or choice, we consider whether to standardize the representation vectors by mean and SD.

K-NN. The number of neighbors (k) is found by evaluating $k \in \{1, 2, 4, 8, 16\}$. We always observed decrease in the mean AUROC by increasing k, thus did not try larger values. Standardization of representation vectors is the other hyperparameter considered for k-NN.

K-NNN. Similar to VGLR, k-NNN has a hyperparameter that determines the number of neighbors of neighbors for local adaptation, which we call k'. The same candidate values is used to tune this hyperparameter, that is, $k' \in \{10, 20, 40, 80\}$. k-NNN's authors propose two other hyperparameters as well: number of neighbors of the input (over which the score is averaged) k, and the partition size s that is used to assume uncorrelated dimensions in representation vector (bock diagonal covariance). However, we assume $k = 1$ and $s = 1$ in our implementation. This is a simplifying assumption that we justify in two ways. First, this helps to have k' as the only hyperparamter of k-NNN in common with our method, that also has a similar interpretation. This makes the comparison between the two more 'apples to apples'. Moreover, if any advantage comes from the 'block diagonal' covariance matrix or averaging over 'multiple neighbors of input', then it should be easy to add these two components into our method and rise its performance as well.

k-means + Maha. (SSD). The number of clusters (k) is the only hyperparameter that is found by considering $k \in \{1, 50, 100\}$. Clusters are initialized by 'kmeans++' method, and the algorithm is run for up to 500 iterations,

with 2 random restarts. Following the SSD method, at inference time, we use a separate full covariance matrix for each cluster to define the Mahalanobis distance. It is worth noting that with $k = 1$ this method is equivalent to One-class Mahalanobis Distance novelty detection.

OC-SVM. We use RBF kernel for OC-SVM. The ν parameter in the loss function is assumed to 0.5 (the default value in scikit-learn 1.3). The only tuned hyperparameter is γ, which is the spread of the RBF kernel, using $\gamma \in \{0.01, 0.02, 0.08\}$. The representation vectors are standardized.

Local KNFST. The number of retrieved nearest neighbors (k) is assumed as $k = 100$ in this method. The second (dummy) class used to compute the projection direction is assumed to be the origin. We try both linear and RBF kernels. When RBF kernel is used, its spread γ is set to 0.02, and the representations are standardized (following what we found the best with OC-SVM).

DINO for fine-tuning. We run the DINO self-supervised training method on a pretrained feature extractor to fine tune it. The method is run for 20 epochs starting from the DINO ImageNet weights, and the training data are the same as the ones for novelty detection methods. We use learning rate 2×10^{-4}, 3 warm up epochs, and final weight decay equal to 0.1. We keep the other hyparameters same as it is in the DINO author's code, suggested for ImageNet. The same hyperparameter values are used for all the datasets.

A.2 Performance comparison per task

As explained in section 5 of the paper, for each of the CUB-200, Flowers-102, and Food-101 datasets, we define 5 different novelty detection tasks by including different classes of the dataset in the normal and novel sets. Since only the average AUROC over the tasks was reported in the paper for each dataset, here we show the more detailed AUROC results per each individual task in figure 5, which represent the variability of performances on each dataset. We choose the three methods VGLR, k-NN, k-NNN to this end, as they appear to be the most competitive methods overall according to the average numbers.

Similar to the average results, it is verified that our proposed VGLR method outperforms k-NN in every task. The k-NNN method has a very competitive performance as well, and is only outperformed by k-NN in one case (task 5 in Food with iBOT). More specifically, VGLR ranks first in 6 out of 10 tasks on CUB, 10 out of 10 tasks on Flowers, and 6 out of 10 tasks on Food data. Despite the close performance with k-NNN, it is worth reminding that k-NNN here uses at least 4 times as many nearest neighbors (k') as our proposed method, which means possibly 4 times higher computational cost for obtaining neighbors.

A.3 Results with ResNet model

In table 5, we show the AUROC performance of methods when the feature extractor is a ResNet-50 model pretrained using DINO on ImageNet (the representation vector is the pre-logits value in the network). The protocols used in splitting the data and setting hyperparameters are the same as described before for ViT models. Most of the hyperparameter values are same as in the case of DINO ViTs (e.g. $k' = 10$ in VGLR , $k' = 80$ in k-NNN, and $k = 1$ in k-NN). One difference here is that the representation vectors are not standardized in VGLR. We found that standardization would impair the performance of k-NN considerably as well (and consequently, the nearest neighbor stage of our method) when using the ResNet model (about 6% drop in average AUROC of k-NN).

The pattern in the results is almost similar to the one with ViTs; VGLR and k-NNN performance rank as the first or second in the majority of cases. However, the best obtained mean AUROC with DINO ResNet is everywhere significantly lower than the the one with DINO ViT, as shown in the last row. Moreover, it looks that the improvement achieved by our VGLR method is generally smaller in this case, which can suggest our method is particularly suitable for 'harnessing' the power of pretrained ViTs in novelty detection.

A.4 Performance in FPR@95TPR

Tables 6 and 7 below report the performance of the experimented methods in terms of FPR@95TPR [5] before and after fine tuning the feature extractor, which means they are counterparts to the AUROC performance

[5] False Positive Rate when the score threshold (τ) is such that the True Positive Rate is 95%

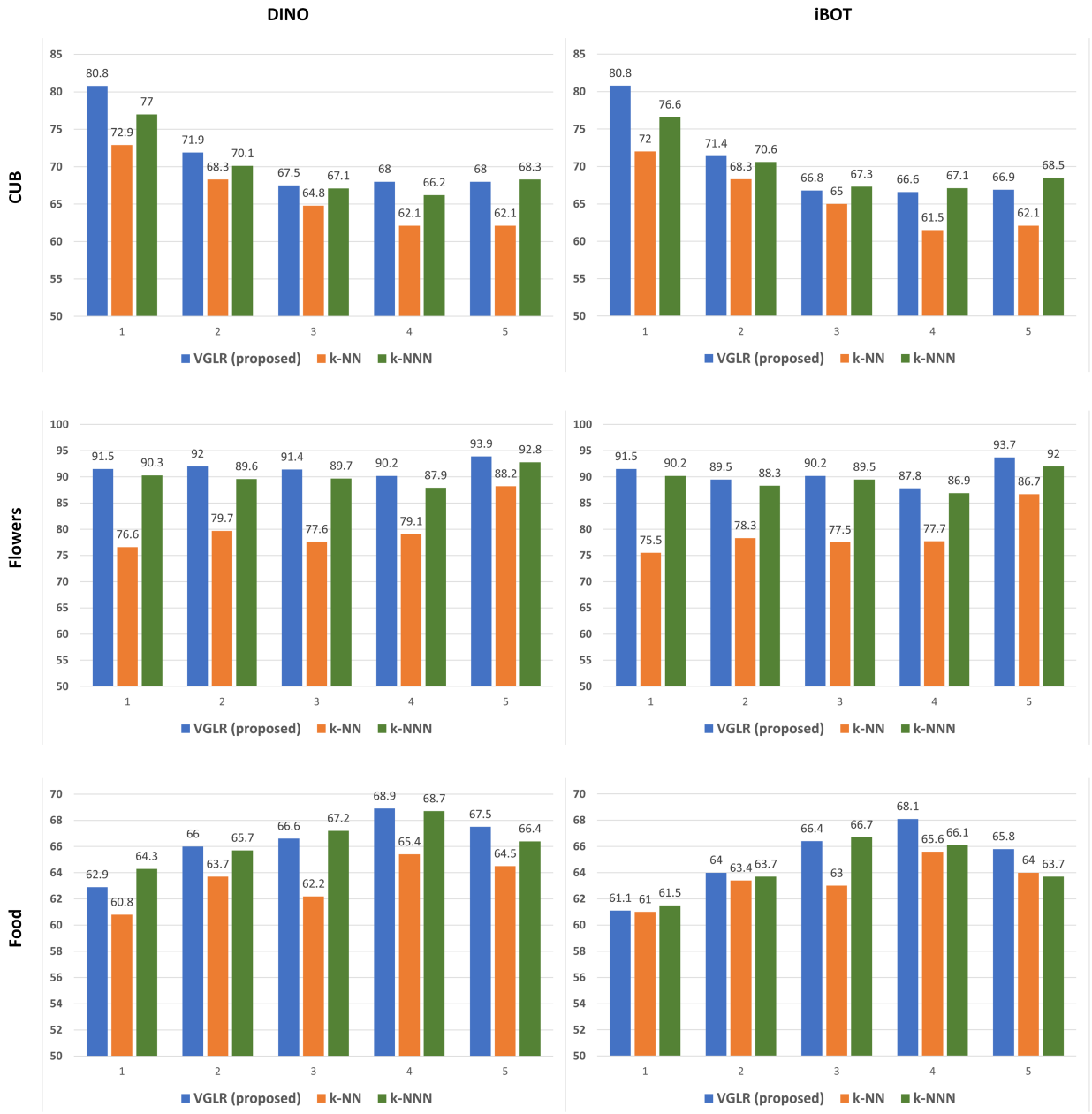

Figure 5: AUROC performance of three methods (VGLR, k-NN, and k-NNN) in 5 novelty detection tasks per each dataset (rows), using non-fine-tuned ViT-B/16 feature extractor pretrained on ImageNet with DINO (left column) and iBOT (right column) methods.

tables in the paper. Generally, VGLR ranks first in a fewer number of cases in terms of FPR@95TPR than AUROC, although its performance still ranks at least as second in quite many cases. Intuitively, the results in FPR@95TPR are not very satisfactory overall, where the best FPR (error) over all methods is always above 60%, except for the Flowers dataset. One interpretation of this observation is that there are a minority of novel images in our fine-grained (and challenging) tasks which are hard to detect for all the methods (i.e., they are only detectable at a small score threshold). Indeed, one can argue that FPR@95TPR is a more 'noisy' metric than AUROC since the former corresponds to a certain threshold, and the latter is an average (area) over different thresholds. Also, we should take into consideration that the hyperparameters of all the methods are optimized based on average AUROC in our experiments.

Table 5: AUROC performance using ResNet-50 feature extractor pretrained on ImageNet by DINO method (without fine-tuning). The numbers are averages over 5 novel/normal splits, except for the case of PCAM. The last row shows the best average AUROC for each dataset subtracted by the best corresponding value obtained by the DINO ViT-B/16.

Method	CUB	Flowers	Food	PCAM
VGLR (proposed)	**59.6**	**85.6**	61.8	**66.9**
k-NN	57.4	82.6	**62.5**	61.0
k-NNN (k=1, s=1)	<u>59.1</u>	80.7	<u>62.4</u>	64.8
k-means + Maha. (SSD)	55.9	83.5	60.0	63.7
OC-SVM	54.1	65.5	56.3	58.5
Local KNFST	52.7	67.0	51.9	53.5
(diff. w.r.t. ViT)	-11.7	-6.2	-3.9	-13.3

Table 6: Novelty detection performance in FPR@95TPR using ViT-B/16 feature extractor pretrained on ImageNet by DINO and iBOT methods (without fine-tuning). The numbers are averages over 5 novel/normal splits, except for the case of PCAM.

	FPR@95TPR (%) DINO/iBOT			
Method	CUB	Flowers	Food	PCAM
VGLR (proposed)	82.7 / 82.8	**31.2** / <u>37.3</u>	<u>86.9</u> / <u>88.8</u>	**64.0** / **66.0**
k-NN	80.5 / 82.3	60.3 / 61.7	88.0 / <u>88.8</u>	69.0 / 70.3
k-NNN (k=1, s=1)	**76.8** / **75.4**	35.7 / **36.3**	**83.7** / **86.5**	78.7 / 79.9
k-means + Maha. (SSD)	84.0 / 83.4	57.6 / 59.2	91.2 / 90.7	69.8 / 70.9
OC-SVM	84.0 / 84.5	65.4 / 66.8	90.1 / 90.0	70.3 / 71.7
Local KNFST	81.7 / 81.6	75.5 / 72.9	88.9 / 90.2	72.2 / 75.8

Table 7: Change in the task-specific FPR@95TPR performance (%) by fine-tuning the ViT-B/16 using DINO loss on normal data.

Method	**CUB**	**Flowers**	**Food**	**PCAM**
VGLR (proposed)	61.1 → **65.4**	32.1 → <u>33.7</u>	92.5 → 61.5	64.0 → 65.9
k-NN	67.2 → <u>69.2</u>	67.8 → 66.4	90.7 → <u>60.8</u>	69.0 → 64.5
k-NNN (k=1, s=1)	67.2 → 75.7	34.7 → **32.5**	88.9 → 75.4	78.7 → 78.8
k-means + Maha. (SSD)	76.1 → 74.4	67.8 → 71.3	92.9 → 64.7	69.8 → <u>62.1</u>
OC-SVM	72.3 → 73.2	71.9 → 75.6	91.0 → **60.5**	70.3 → **60.5**
Local KNFST	76.7 → 72.4	73.9 → 62.9	89.9 → 72.6	72.2 → 63.4

A.5 More on the ImageNet background model

The background model that we use for obtaining the likelihood ratio in our proposed method is a Gaussian distribution with diagonal covariance fitted on the normal data representations (the bootstrapped background model, as we call it). However, as pointed out in section 3 of the paper, another way that we explored in our study is to obtain the background model by fitting a Gaussian distribution on the representation vectors of ImageNet dataset (partly similar to what Schirrmeister et al. (2020) propose with generative models). The idea is that ImageNet is a broad and generic dataset that contains images of a great number of objects under a variety of conditions. Thus, this dataset can be a good source to learn the distribution of background part of representations, i.e., $p(z_B)$, which mostly corresponds to the statistics of low-level features. At the same time, due to the diversity in ImageNet, $p'(z_S)$ estimated from this dataset should tend to assign similar values to both normal and novel data.

In table 8, we show the performance of VGLR when the background model is replaced by the Gaussian distribution learned from ImageNet, and compare it to the previously reported performance with the default bootstrapped background model, using non-fine-tuned ViTs. The ImageNet background model is a multivariate Gaussian distribution with full covariance matrix. The results of the two methods are pretty close, as the table shows. However, the main flaw that we observe with the ImageNet background model is that the performance of this method drops dramatically if a feature extractor fine-tuned on normal data is used, similar to section 5.2 in the paper (for instance, with DINO, the average AUROC on CUB and Flowers will be 55.5% and 56.5% respectively after fine-tuning). This should not be surprising because the distributions of representations (particularly their background parts) are different between the non-fine-tuned and fine-tuned models while the background Gaussian distribution still comes from the non-fine-tuned model. One solution can be to estimate the mean and covariance of ImageNet representations using each fine-tuned model, though that would impose much more computational cost than the current bootstrapping method.

Table 8: The average AUROC performance (%) of our method when the alternative ImageNet estimated background model is used compared to when the current bootstrapped one (estimated on normal data) is used, with non-fine-tuned ViT-B/16 feature extractors pretrained on ImageNet.

Method	CUB	Flowers	Food	PCAM
VGLR / ImageNet Background (DINO)	72.7	91.5	67.9	79.1
VGLR / Bootstrapped Background (DINO)	71.3	91.8	66.4	80.2
VGLR / ImageNet Background (iBOT)	71.7	90.2	67.0	76.5
VGLR / Bootstrapped Background (iBOT)	70.5	90.6	65.1	78.5

A.6 Discussion and results on far-OOD tasks

So far, we have only evaluated our method in 'fine-grained novelty detection' tasks. Technically, the novel data in such tasks can be seen as an almost extreme type of 'near Out-of-Distribution' (near-OOD) inputs. Although this problem is challenging enough, one might ask if our VGLR method works reasonably also in 'far-OOD' problems where the normal and novel data are from completely different datasets.

In theory, we do not expect the method to work on far-OOD as well as the near-OOD case because of two reasons. First, as pointed out in the paper conclusion, the unmodified k-NN score should result in the optimal AUROC if the in-distribution points are sufficiently far (in Euclidean distance) from the OOD ones (consider the ideal case where distance between every pair of in-distribution points is smaller than every pair of in-distribution and OOD points). The second point is about the effect of the background model. In deriving the likelihood ratio score, we start by assuming that the representation vector can be partitioned to semantic and backgrounds dimensions in the same way for both the in-distribtuion and ODD data. But even this initial assumption can become unrealistic when these two types of data come from completely different distributions.

Table 9: AUROC performance (%) in far-OOD tasks, where one dataset is considered as novel (OOD) against another one, with non-fine-tuned ImageNet pretrained DINO ViT-B/16 feature extractor. For each multi-category dataset, we use half of the classes (in default order) as used in the normal set of previous experiments.

Normal (in-distribution)	Novel (OOD)	k-NN Performance	VGLR Performance
CUB	Flowers	99.9	99.1
Food	CUB	100	99.4
PCAM	Food	99.9	98.3
Flowers	PCAM	100	98.9
PCAM	Gaussian Noise	99.9	99.9
CUB	Gaussian Noise	100	99.9

To investigate this empirically, we run a few far-OOD detection experiments by choosing a pair of datasets as normal/novel, among the ones already used in the experiments, in addition to Gaussian noise images. Table

9 shows the AUROC performance results of our proposed method (with $k' = 10$) and its comparison to basic k-NN ($k = 1$) in this scenario. Interestingly, we see that the k-NN performance is almost perfect on these tasks, and VGLR also works very well despite of the theoretical concerns. However, VGLR is slightly outperformed by k-NN (up to 1.6%) in most of the cases.

A.7 Notes on complexity

The time/memory complexity of the method can be discussed in two parts. The first part is about finding the nearest neighbor of input and the nearest neighbors of the nearest neighbor (lines 4 and 5 in the algorithm 1). This step has a computational cost similar to finding $k' + 1$ nearest neighbors of a data point, that in practice depends on the algorithm employed for nearest neighbors search (for example, using k-d tree and sequential processing, the average time complexity will be $O((1 + k')logN)$ where N is the number of data points, but parallel computing can speed up the search largely). The second primary part of the computation is where the local variances (diagonal covariance) are obtained, i.e., line 6 of the algorithm. In case of naive sequential implementation, this should have $O(k'm)$ time and memory complexity, where m is the number of dimensions of the representation space.

An alternative way to implement the VGLR method is to pre-compute the local variances for each training data point of D_T and save them in a lookup table, similar to the method of Nizan and Tal (2024). In that case, the processing time for obtaining local variances at test time will be eliminated though at the cost of increasing the memory complexity of the whole algorithm by a factor of 2.