Prior Distribution and Model Confidence

Maksim Kazanskii
Independent Researcher
mkazanskii@gmail.com

Artem Kasianov
BIOPOLIS/CIBIO
Vairão, Portugal

Abstract

We study how the training data distribution affects confidence and performance in image classification models. We introduce Embedding Density, a model-agnostic framework that estimates prediction confidence by measuring the distance of test samples from the training distribution in embedding space, without requiring retraining. By filtering low-density (low-confidence) predictions, our method significantly improves classification accuracy. We evaluate Embedding Density across multiple architectures and compare it with state-of-the-art out-of-distribution (OOD) detection methods. The proposed approach is potentially generalizable beyond computer vision.

1 Introduction

In recent years, deep learning has driven progress in computer vision, enabling robust performance across tasks such as image classification, object detection, and semantic segmentation Krizhevsky et al. (2012); He et al. (2016); Long et al. (2015). Despite these advances, modern visual recognition systems remain highly sensitive to violations of the closed-set assumption, where test inputs deviate from the distribution of the training data due to changes in environment, acquisition conditions, or the presence of previously unseen semantic categories Scheirer et al. (2013); Bendale & Boult (2015); Yang et al. (2024b). Under such shifts of distribution, models often produce confident but incorrect predictions, raising critical concerns for the reliability of vision-based systems in real-world and safety-critical applications Amodei et al. (2016); Hendrycks & Dietterich (2019); Sayyed et al. (2025).

From a theoretical perspective, the behavior of deep visual models under distribution shift is tied to the geometry and structure of the latent representations induced by the training data. The distribution of features in this representation space defines the effective domain of the model and constrains the regions in which its predictions can be considered meaningful Bengio et al. (2013); Arora et al. (2019); Kirichenko et al. (2023); Zhou et al. (2024); Lu et al. (2025). Understanding and using this structure offers a path toward characterizing model predictability beyond empirical confidence measures.

In this work, we propose an approach for out-of-distribution (OOD) detection in computer vision that leverages the latent space geometry of training data to assess the reliability of model predictions at the instance level. Our method operates without additional supervision or retraining and is applicable across a range of visual tasks and network architectures. We evaluate the proposed framework on standard large-scale OOD benchmarks and under realistic distribution shifts, demonstrating consistent improvements over existing methods Hendrycks et al. (2022).

2 Related Work

Out-of-distribution (OOD) detection in computer vision has been studied in the context of deep neural networks for visual recognition. Early approaches rely on confidence-based criteria, such as thresholding the maximum softmax probability, and input perturbation and temperature scaling as in ODIN, which improve separability between in-distribution and OOD samples but often require dataset-specific calibration Hendrycks & Gimpel (2017); Liang et al. (2018); Bitterwolf et al. (2023); Jelenić et al. (2024). Feature-space methods model

Table 1: Models used in experiments (all trained on ImageNet-1K).

Model	Data	Arch.	Ref.	Params
ResNet-101	IN-1K	CNN	(He et al., 2016)	44.5M
ResNet-50	IN-1K	CNN	(He et al., 2016)	25.6M
ShuffleNet-V2	IN-1K	Lt. CNN	(Ma et al., 2018)	2.3M
DeiT-Tiny	IN-1K	ViT	(Touvron et al., 2021)	5.7M
DeiT-Small	IN-1K	ViT	(Touvron et al., 2021)	22.1M
DeiT-Base	IN-1K	ViT	(Touvron et al., 2021)	86.4M

the distribution of learned representations, for example through class-conditional Gaussian assumptions and Mahalanobis distance, enabling a unified treatment of OOD detection and adversarial examples while remaining sensitive to feature dimensionality and covariance estimation Lee et al. (2018a); Bitterwolf et al. (2023); Jelenić et al. (2024). Energy-based approaches reinterpret model outputs as unnormalized log-densities, providing a scoring function that generalizes across architectures and large-scale datasets Liu et al. (2020); Yang et al. (2024a).

Uncertainty estimation methods, such as deep ensembles, approximate predictive uncertainty by aggregating predictions from multiple independently trained models, yielding strong empirical performance at the cost of increased computational and memory overhead Lakshminarayanan et al. (2017); Ovadia et al. (2019); Fang et al. (2023). More recent work explores contrastive and self-supervised representation learning to induce feature spaces that improve robustness under distribution shift Tack et al. (2020); Aathreya & Canavan (2024). In contrast, our approach focuses on the intrinsic geometry of the latent representation induced by the training distribution, enabling instance-level reliability assessment without auxiliary outlier data, architectural modification, or ensemble training. Nearest-neighbor methods have previously been explored for OOD detection in feature space Sun et al. (2022). Unlike these approaches, which use distance-based scores within a specific classifier representation, our method estimates local embedding density and focuses on model-agnostic confidence filtering evaluated through the coverage–accuracy trade-off.

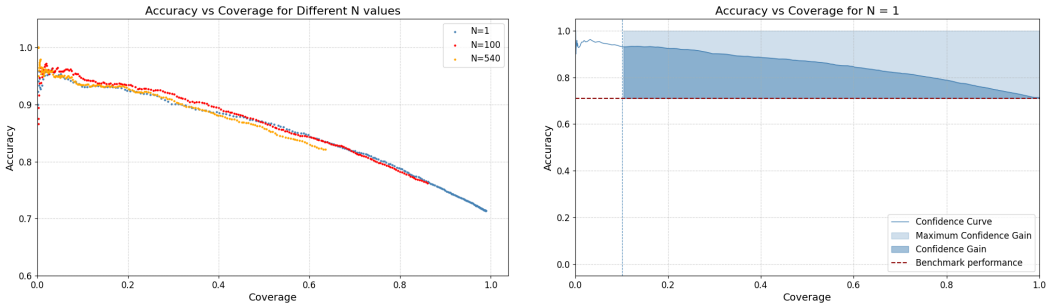
3 Methods

The proposed approach relies on a common property of representation learning: semantically similar samples form dense clusters in embedding space. Samples drawn from the training distribution therefore lie in high-density regions of the embedding manifold, while OOD samples appear in sparse regions. Counting neighbors within a fixed radius provides a simple estimate of whether a sample lies inside the training distribution. To study the effect of data distribution on model confidence, we selected several models trained on the same dataset. All models were trained on the ImageNet-1K dataset (Deng et al., 2009), ensuring a consistent training data distribution across experiments. To analyze differences in our framework's performance, we intentionally included embedding architectures from different model families with varying parameter counts. Specifically, we considered convolutional ResNet models (ResNet-50 and ResNet-101 (He et al., 2016)), Vision Transformer–based models (Touvron et al., 2021) (DeiT-T, DeiT-S, and DeiT-B), and the lightweight ShuffleNet-V2 model (Ma et al., 2018). A brief summary of the selected models is provided in Table 1. To characterize the data distribution on which the models were trained, we analyzed the structure of the ImageNet-1K training set by computing image embeddings using several pretrained models. We selected a diverse set of architectures to generate embedding representations for each image in the dataset, enabling an examination of the statistical properties of the data across different feature spaces. This embedding-based analysis provides insight into how different models encode the visual information present in the training distribution.

We use only the training split of ImageNet-1K, as the models listed in Table 1 were trained exclusively on this subset. For the embedding analysis, we employed the following models: DINO-v1 (Caron et al., 2021), DINO-v2 (Oquab et al., 2023) with embedding dimensions of

Table 2: Description of the models and their embedding sizes.

Model	Training Data	Emb.	Ref.
DINO-V1 ViT-S/16	ImageNet-1K (SSL)	384	(Caron et al., 2021)
DINO-V1 ViT-B/16	ImageNet-1K (SSL)	768	(Caron et al., 2021)
DINO-V1 ViT-B/8	ImageNet-1K (SSL)	768	(Caron et al., 2021)
DINO-V2 ViT-S/14	142M curated	384	(Oquab et al., 2023)
DINO-V2 ViT-B/14	142M curated	768	(Oquab et al., 2023)
MobileNet-V2	ImageNet-1K (sup.)	1000	(Sandler et al., 2018)



(a) The Confidence Curve

(b) The Confidence Curve

Figure 1: The Confidence Curves and Confidence Gain for the ResNet-101 & DINO-V2 ViT-B/14.

384, 768, and 1024, and MobileNet-V2 (Sandler et al., 2018). The selection of embedding models was guided by two main factors: model capacity and the nature of the data used for pretraining. To evaluate the sensitivity of our algorithm to embedding dimensionality, we compared performance across closely related models with different embedding sizes. Additionally, we investigated how differences in pretraining data influence the resulting embeddings. A brief summary of the embedding models is provided in Table 2. In order to investigate the effect of the prior data distribution on the prediction capabilities of the classification models, we selected two distinct datasets: ImageNet-V2 (Recht et al., 2019) and ObjectNet Barbu et al. (2019). We divided ImageNet-V2 into training and internal test sets with a split of 75% / 25% and left the ObjectNet dataset as the external test set. We used only ObjectNet classes that coincide with the classes in the original ImageNet-1K dataset (yielding approximately $10k$ images). We used the training dataset to tune two parameters: the number of neighbors N and the distance threshold L. The distance threshold L and neighbor count N were selected via validation grid search. In practice, performance is stable across a wide range of values. We divided the test sets (external and internal) into three label-disjoint subsets. Therefore, images with the same label can only be present in one subset. The goal of this division is to report variability in the metrics. Results are reported across three independent subsets to reduce bias from dataset-specific artifacts. The observed variance reflects heterogeneity between subsets rather than instability of the method.

To compute embedding similarities, we used cosine similarity (Manning et al., 2008), a commonly adopted metric for nearest-neighbor search in high-dimensional spaces. Embeddings were stored and queried using ChromaDB (Team, 2023) with a ClickHouse backend (ClickHouse, 2016), enabling efficient large-scale retrieval. All experiments were performed on a MacBook equipped with an Apple M1 processor, using either the MPS or CPU backend.

Confidence estimation was performed on both the training and test sets. On the training set, the parameters N and L were tuned to optimize performance. The optimized parameters were then applied to the test sets (internal and external), on which performance was reported. The pseudocode for the algorithm is presented in A.1.

We define the *Confidence Gain* as the area between: (i) the confidence curve (with extrapolation if needed), (ii) the vertical line coverage = 0.1, (iii) the vertical line coverage = 1, and

(iv) the horizontal line accuracy = acc_b, where acc_b denotes the baseline accuracy obtained when all data samples are used (i.e., coverage = 1). This baseline depends solely on the classifier and dataset and is independent of the embedding model. Since the confidence curve approaches acc_b as coverage approaches 1, the area can be computed directly using standard numerical integration (`Scikit-Learn` Pedregosa et al. (2011)).

We define the *Maximum Confidence Gain* as the area of the rectangle bounded by $x = 0.1$, $x = 1.0$, $y = 1.0$, and $y = acc_b$. This value represents a theoretical upper bound on the achievable *Confidence Gain*. Consequently, we introduce the *Normalized Confidence Gain* as the ratio between the *Confidence Gain* and its maximum possible value. This normalized quantity lies in the interval $[0, 1]$, where 1 indicates an ideal confidence curve and 0 indicates no gain over the baseline.

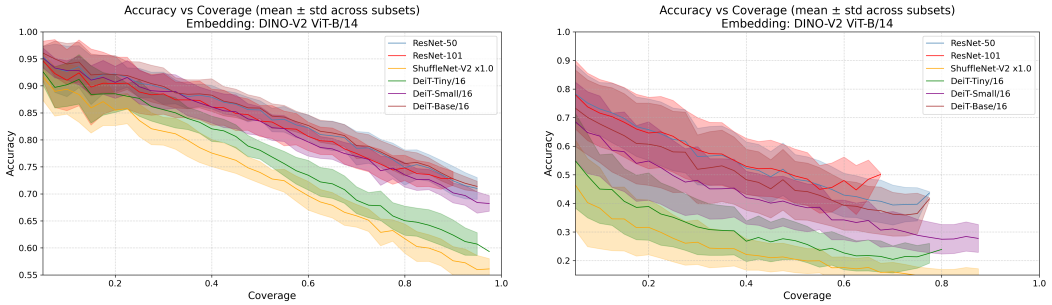
For clarity, we now detail the procedure step by step using one specific classification model as an example:

Step 1 For each image in the original training dataset ImageNet-1K (let us call this set the Base Set and the images in the set the Base Images), we calculate the embeddings for the models presented in Table 2 using cosine similarity, which is a commonly adopted metric. We store these embeddings in the Base Embedding Database. Although the Base Image Set serves as a training set for the models we investigated, we do not fine-tune their parameters. Therefore, the word 'training' could be misleading, and we used the term 'base'.

Step 2 For each image in the train set (a subset of ImageNet-V2), we compute its embedding and retrieve the $K_{\max}$ nearest neighbors from the Base Embedding Database. We refer to this collection as the train set. The value of $K_{\max}$ is chosen to be sufficiently large so that increasing it further does not change the outcome of the analysis. The term 'train' emphasizes that this dataset is used to tune the parameters N and L, which maximize the performance of the algorithm. These parameters are then kept fixed during the evaluation on the test sets. In addition, for each image in the train image set, we calculate and store the predictions of the classification model without any filtering for reference.

Step 3 For each training image, we estimate confidence using two parameters: a distance threshold L and a neighbor-count threshold N. A prediction is considered *confident* if at least N nearest neighbors lie within distance L in embedding space; otherwise, it is discarded. Accuracy is then computed over the subset of accepted predictions, and the fraction of accepted samples defines the *coverage*.

As L increases, the acceptance region expands and coverage is theoretically non-decreasing. In practice, approximate nearest-neighbor search and floating-point effects can introduce small local violations. To avoid dependence on L itself, we index confidence curves by the empirically observed coverage values and map each coverage level to its corresponding effective threshold L.



(a) Confidence Curves for Internal Dataset. (b) Confidence Curves for External Dataset.

Figure 2: Accuracy vs. coverage (confidence curve) for the best embedding model (DINO-V2 ViT-B/14).

More precisely, let $d_1(x) \leq d_2(x) \leq \cdots \leq d_{K_{\max}}(x)$ denote the sorted distances to the nearest neighbors of sample x. For each value of N, we generate a coverage-ordered sequence by thresholding L at these distances. The effective threshold L inducing a desired coverage level τ is therefore defined implicitly as

$$L^*(N, \tau) = \min\{L : \text{coverage}(L; N) \geq \tau\},$$

where coverage$(L; N)$ is the fraction of samples whose N closest neighbors lie within radius L. Thus, L^* is determined automatically by sweeping through the sorted neighbor distances and selecting the smallest radius that achieves the target coverage.

The resulting *confidence curve* plots accuracy as a function of coverage (and therefore implicitly of L). Examples of confidence curves for ResNet-101 and DINO-V2 ViT-B/14 at several values of N are shown in Figure 1(a). Figure 2(b) illustrates the corresponding *maximum confidence gain* and *confidence gain* for the same embedding/model pair with $N = 1$. Each point on the curve corresponds to a specific effective value of L induced by the optimal coverage.

Step 4 We find the values of N that maximize the *NCG* (the ratio between the *Maximum Confidence Gain* and the *Confidence Gain*) for the specific Embedding Model (to be discussed further). We use the optimal values of neighbors N^* and the threshold value L^* for the test sets. For more technical details for the choice of the optimal parameters please refer to Appendix A.2

Figure 1(a) shows an example confidence curve for the ResNet-101 / DINO-V2 ViT-B/14 pair, although the same procedure applies to all model–embedding combinations. The distribution of points along the curve is uneven: at high coverage values, points become sparse, while at low coverage they are densely packed. This occurs because low coverage corresponds to highly confident predictions, which appear more frequently. To compensate for the lack of points near full coverage (i.e., near coverage = 1), we linearly extrapolate the curve using the 20 points with the highest coverage values. We also observe that for very small coverage values (below 0.1), the curve becomes unstable and noisy. For robustness, we therefore restrict the computation of the *Confidence Gain* to the interval coverage $\in [0.1, 1]$, where the accuracy varies smoothly and monotonically.

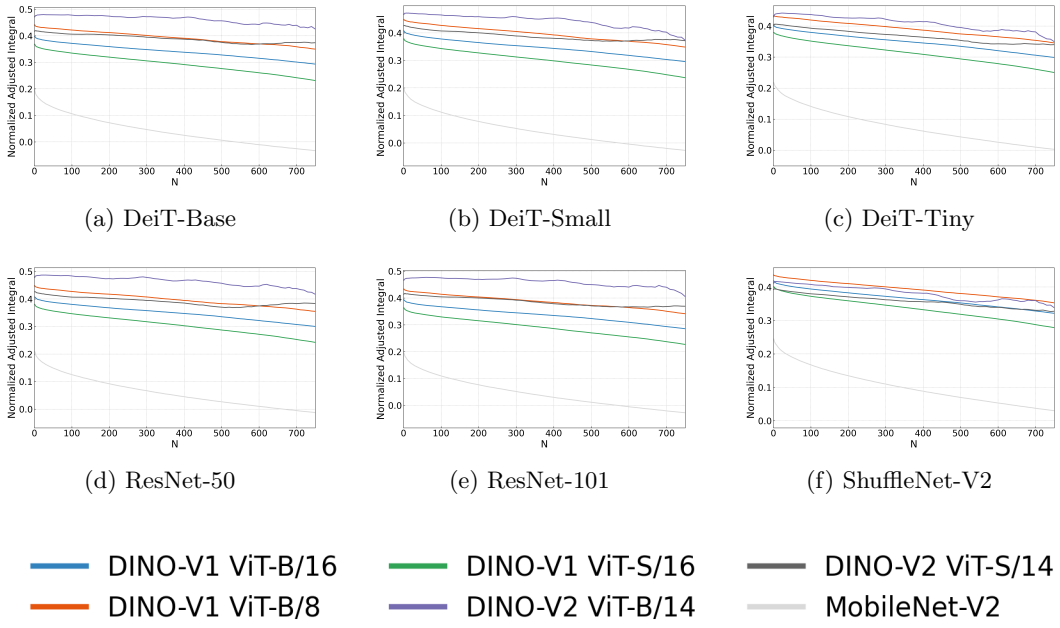
4 Results

Sensitivity of the Confidence Gain. In Figure 3, the results of the adjustment (or training) are shown. On the x-axis, the number of neighbors N is presented, and on the y-axis, the calculated *Normalized Confidence Gain* is depicted. We can observe that the highest *Normalized Confidence Gain* value is achieved for all models except the ShuffleNet-V2 by the DINO-V2 ViT-B/14 embedding model, which is so far the strongest model among embedding models in terms of the number of parameters. Interestingly, the second version of the DINO models outperforms the first version. In Table 3, the best parameter N and the corresponding *Normalized Confidence Gain* are presented.

For the evaluation pipeline, we select the number of parameters N that maximizes the *NCG* for each pair of the classification model and embedding model. The corresponding value of N is then used to compute the metrics on both the internal test and external test sets. The

Table 3: N and best *Normalized Confidence Gain* for each classification model across embedding models.

Classification Model	ViT-B/16	ViT-B/8	ViT-S/16	ViT-B/14	ViT-S/14	MobileNet-V2
deit_base_patch16_224	1/0.398	1/0.441	1/0.368	24/**0.480**	4/0.419	1/0.194
deit_small_patch16_224	1/0.408	1/0.449	1/0.379	7/**0.472**	2/0.427	1/0.199
deit_tiny_patch16_224	1/0.402	1/0.433	1/0.379	23/**0.442**	4/0.406	1/0.218
resnet101	1/0.394	1/0.434	1/0.364	60/**0.477**	4/0.417	1/0.195
resnet50	1/0.407	1/0.448	1/0.380	24/**0.487**	2/0.426	1/0.209
shufflenet_v2_x1_0	1/0.418	1/**0.436**	1/0.401	7/0.417	4/0.395	1/0.244



(a) DeiT-Base (b) DeiT-Small (c) DeiT-Tiny

(d) ResNet-50 (e) ResNet-101 (f) ShuffleNet-V2

Figure 3: Normalized Confidence Gain vs. number of neighbors N for different classification models.

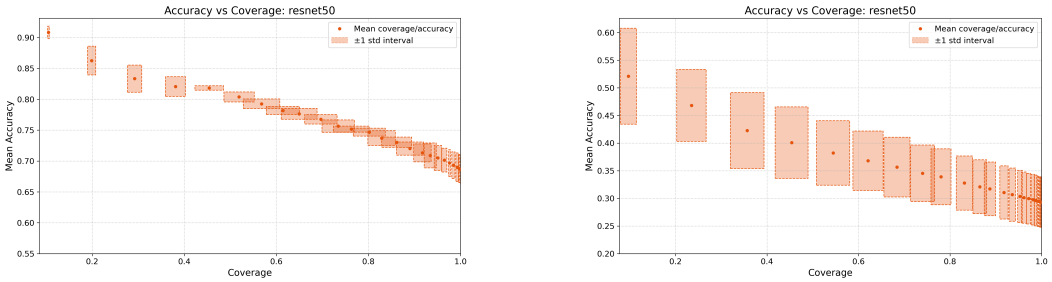
resulting scores, along with their standard deviations (computed across the three subsets of each test set), are reported in Table 4. Across nearly all model combinations, the highest value of *Normalized Confidence Gain* is obtained by DINO-v2 ViT-B/14. This behavior is expected and can be attributed to the model's higher capacity and the increased resolution of its input representations.

Ensemble confidence estimation. The next logical step would be to combine several embedding models (or use an approach similar to the ensemble of models, as in Dietterich (2000)), which has been done using greedy heuristics. We fix the same coverage for each embedding model. We take the model with the highest *Normalized Confidence Gain* and make predictions. We accept a portion of the predictions and leave the rest to other models. Note that the total coverage for all embedding models is slightly higher than the parameter *coverage* for individual models, but is not fixed. The results of the experiments (accuracy vs. coverage) are presented in Figure 4. The *Normalized Confidence Gain* for the combination of embedding models is shown in Table 4 in the column 'Combination'. The observed *Confidence*

Table 4: The Normalized Confidence Gain for different pairs of embedding/classification models and for the combination of embedding models. Benchmark accuracy is shown for reference. All values are mean ± std across three label subsets for internal and external test sets.

Set	Classification Model	Benchmark Acc.	DINO-V1 ViT-B/16	DINO-V1 ViT-B/8	DINO-V1 ViT-S/16	DINO-V2 ViT-B/14	DINO-V2 ViT-S/14	MobileNet-V2	Combination
Internal	deit_base_patch16_224	0.696 ± 0.014	0.390 ± 0.052	0.414 ± 0.048	0.357 ± 0.035	**0.449 ± 0.077**	0.404 ± 0.046	0.194 ± 0.046	0.361 ± 0.058
	deit_small_patch16_224	0.666 ± 0.019	0.376 ± 0.068	0.403 ± 0.071	0.343 ± 0.062	**0.431 ± 0.076**	0.376 ± 0.058	0.189 ± 0.058	0.320 ± 0.060
	deit_tiny_patch16_224	0.582 ± 0.037	0.376 ± 0.076	0.388 ± 0.079	0.353 ± 0.078	**0.408 ± 0.070**	0.380 ± 0.067	0.202 ± 0.069	0.366 ± 0.067
	resnet101	0.687 ± 0.025	0.345 ± 0.074	0.369 ± 0.066	0.308 ± 0.072	**0.410 ± 0.106**	0.366 ± 0.058	0.169 ± 0.065	0.316 ± 0.071
	resnet50	0.676 ± 0.014	0.377 ± 0.050	0.408 ± 0.049	0.347 ± 0.050	**0.446 ± 0.075**	0.398 ± 0.044	0.189 ± 0.068	0.344 ± 0.061
	shufflenet_v2_x1_0	0.538 ± 0.022	0.363 ± 0.056	0.374 ± 0.063	0.345 ± 0.056	**0.379 ± 0.053**	0.357 ± 0.046	0.220 ± 0.050	0.351 ± 0.046

Set	Classification Model	Benchmark Acc.	DINO-V1 ViT-B/16	DINO-V1 ViT-B/8	DINO-V1 ViT-S/16	DINO-V2 ViT-B/14	DINO-V2 ViT-S/14	MobileNet-V2	Combination
External	deit_base_patch16_224	0.269 ± 0.059	0.079 ± 0.109	0.132 ± 0.120	0.059 ± 0.107	**0.214 ± 0.153**	0.189 ± 0.149	0.044 ± 0.084	0.141 ± 0.130
	deit_small_patch16_224	0.229 ± 0.045	0.072 ± 0.092	0.120 ± 0.099	0.055 ± 0.082	**0.188 ± 0.125**	0.170 ± 0.119	0.037 ± 0.063	0.086 ± 0.087
	deit_tiny_patch16_224	0.152 ± 0.041	0.073 ± 0.077	0.102 ± 0.085	0.058 ± 0.072	0.119 ± 0.092	**0.119 ± 0.093**	0.040 ± 0.057	0.106 ± 0.090
	resnet101	0.304 ± 0.057	0.069 ± 0.112	0.131 ± 0.112	0.049 ± 0.101	**0.238 ± 0.144**	0.211 ± 0.144	0.036 ± 0.080	0.147 ± 0.123
	resnet50	0.293 ± 0.056	0.069 ± 0.102	0.127 ± 0.106	0.047 ± 0.101	**0.243 ± 0.130**	0.211 ± 0.135	0.035 ± 0.082	0.136 ± 0.114
	shufflenet_v2_x1_0	0.117 ± 0.027	0.062 ± 0.064	0.079 ± 0.065	0.047 ± 0.061	**0.096 ± 0.069**	0.090 ± 0.068	0.038 ± 0.048	0.083 ± 0.066



(a) ResNet50 — Internal

(b) ResNet50 — External

Figure 4: Confidence curves (accuracy vs. total coverage) for the ResNet50 model evaluated on the internal (**left**) and external (**right**) datasets using ensemble of embedding models.

Gain for the combination of embedding models is lower than that of the best-performing individual model. In particular, at very low coverage values for the internal dataset, the accuracy of all models converges to similar levels. This suggests that the underlying data distribution plays a more decisive role than the intrinsic performance of the models. A plausible explanation is that, within the test datasets, certain images are located in close proximity (in the embedding space) to images from the training set. Consequently, under such conditions, the specific choice of the embedding model becomes less critical. This is likely due to the fact that the images in the external dataset (ObjectNet) exhibit a greater distributional shift relative to the original training set (ImageNet-1K) than those in the internal test set.

Choice of $K_{\max}$ and its effect on the confidence rule. In our method, confidence depends only on whether at least N^* neighbors fall within distance L^*. Empirically, $N^* \leq 60$ for all model-embedding pairs, so the decision relies solely on the closest N^* neighbors. Under exact nearest-neighbor retrieval, retrieving $K_{\max}$ neighbors and truncating to N^* is equivalent to retrieving only N^*. Hence, for any $K_{\max} \geq N^*$, the confidence decision is invariant to $K_{\max}$; we set $K_{\max} = 1000$ as a conservative buffer. More generally, $K_{\max}$ can be chosen empirically by tracking Normalized Confidence Gain (NCG) versus retrieval size. NCG typically rises or plateaus before declining as additional, noisy neighbors are included; the optimal $K_{\max}$ lies near this onset. In our experiments, this stability region occurs well below 1000, further supporting $K_{\max} = 1000$ as a safe and efficient upper bound.

For the comparison with other OOD methods please refer to Appendix A.4.

5 Discussion

From the experimental results, we infer that a more sophisticated embedding model is more likely to have a better Confidence Curve (or higher *NCG*). By 'sophisticated,' we mean a larger number of dimensions, higher input image resolution, and a richer dataset on which the model was trained. In particular, the resolutions of the images and the capacity of the embedding model significantly determine the performance.

After training, most models achieved the highest *NCG* with $N = 1$, while the best-performing embedding model required larger values ($N > 1$). This suggests that more expressive embeddings benefit from aggregating information from multiple neighbors.

To determine whether the superior performance of DINO-V2 ViT-B/14 is driven by embedding dimensionality or pretraining data, we compared models with matched embedding sizes (e.g., DINO-V1 ViT-B/16 vs. DINO-V2 ViT-B/14, both 768-D). Across all classifiers and test splits, DINO-V2 consistently achieved a higher NCG margin of 0.04–0.07. In contrast, increasing dimensionality within the same model family yielded smaller improvements of 0.01–0.03. These results indicate that the primary gain arises from the broader pretraining corpus of DINO-V2 (142M images) rather than embedding dimensionality alone. This interpretation is further supported by the poor NCG performance of MobileNet-V2, which

has higher embedding dimensionality but much narrower pretraining data. Improvements on the external dataset (ObjectNet) are smaller because baseline accuracies are substantially lower than on the internal dataset, limiting the achievable relative gain under the *NCG* metric.

Combining embeddings from multiple models offers a potential way to integrate diverse representation spaces; however, in our experiments, such combinations did not outperform the best individual embedding model in terms of *NCG*. Unlike classical ensemble methods, where aggregating weaker models often yields gains, we did not observe a similar effect in this setting. This may be due to the simplicity of our greedy combination strategy, as more sophisticated approaches (e.g., per-sample model selection) could be more effective, as well as the limited diversity of pretraining data across the evaluated models, which constrains the potential for complementary information.

An important aspect of the reported OOD results is that the embedding density method employs an intentionally simple scoring function. In all experiments, the OOD score is derived from a straightforward aggregation of nearest-neighbor distances in embedding space, without any learned parameters, calibration, class conditioning, or access to classifier logits (only calibration of distance threshold L and optimal number of neighbors N). Despite this simplicity, the resulting OOD performance closely matches that of logit-based baselines in the Internal setting and remains competitive on the External dataset. This suggests that a substantial portion of the OOD signal captured by more complex methods is already present in the local geometry of the representation space. At the same time, the remaining performance gap on the External benchmark could indicate the additional information contributed by classifier-specific mechanisms, rather than a deficiency of the underlying representation geometry itself.

An additional practical advantage of the proposed embedding density score for the OOD detection is that it enables detection without performing classification at inference time. Because the score depends solely on distances in a frozen embedding space, it can be computed for arbitrary inputs even when class labels, classifier heads, or calibrated logits are unavailable. In contrast, logit-based methods such as Energy and ODIN inherently require a trained classifier and access to its output scores, which restricts their applicability to supervised classification pipelines. This distinction makes embedding-density–based OOD detection particularly suitable for representation-centric settings, such as retrieval systems, self-supervised or foundation models, and scenarios where classification is not the primary objective.

While our experiments focus on image classification, the proposed embedding density framework may extend to other modalities, particularly NLP. A central challenge in this setting is the absence of a natural discretization of text: unlike images, text forms a continuous sequence, making the choice of window size and stride nontrivial. Although multi-scale windowing could capture semantic structure at different levels, dense sliding windows quickly become computationally prohibitive at corpus scale. Large language model training datasets contain hundreds of billions of tokens. Addressing these challenges remains an important direction for future work.

6 Conclusion

We introduced a framework for estimating prediction confidence using distances in embedding space. Rather than modifying or retraining the classifier, our method identifies samples lying outside the training distribution. This leads to consistent improvements in effective accuracy while discarding a fraction of data. We further showed that a single embedding model is sufficient to filter unreliable predictions across diverse classifiers. Because confidence is tied to the geometry of the training manifold, the approach naturally reacts to distribution shift, providing an implicit mechanism for drift detection. Overall, our results demonstrate that leveraging training embeddings offers a practical and relatively lightweight alternative for confidence estimation in real-world settings where data distributions evolve over time. Code for the experiments is available at `https://github.com/maksimkazanskii/prior_hallucinations`.

References

Sriram Aathreya and Sean Canavan. Flowcon: Out-of-distribution detection using flow-based contrastive learning. *arXiv preprint arXiv:2402.XXXX*, 2024.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Sanjeev Arora, Hrushikesh Khandeparkar, Mikhail Khodak, and Orestis Plevrakis. A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning*, 2019.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Joshua Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 9448–9458, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/97af07a14cacba681feacf301271f014-Abstract.html`.

Abhijit Bendale and Terrance E. Boult. Towards open set deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

Julian Bitterwolf, Maximilian Müller, Marc Fischer, and Matthias Hein. In or out? fixing imagenet out-of-distribution detection evaluation. In *International Conference on Machine Learning*, 2023.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021. URL `https://arxiv.org/abs/2104.14294`.

Inc. ClickHouse. Clickhouse: Open-source column-oriented database management system. `https://clickhouse.com`, 2016. Accessed: 2025-07-27.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, 2009. doi: 10.1109/CVPR.2009.5206848.

Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems: First International Workshop, MCS 2000, Cagliari, Italy, June 21–23, 2000, Proceedings*, pp. 1–15. Springer, 2000. doi: 10.1007/3-540-45014-9_1.

Ke Fang, Yuxuan Chen, and Yixuan Li. Revisiting deep ensemble for out-of-distribution detection: A loss landscape perspective. *arXiv preprint arXiv:2306.XXXX*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 770–778, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

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

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Scaling out-of-distribution detection for real-world settings. In *International Conference on Machine Learning*, 2022.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

Filip Jelenić, Michael Gygli, and Dengxin Dai. Out-of-distribution detection by leveraging between-layer transformation smoothness. In *International Conference on Learning Representations*, 2024.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to distribution shift. In *Advances in Neural Information Processing Systems*, 2023.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, 2018a.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 7167–7177, 2018b.

Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. In *Advances in Neural Information Processing Systems*, 2020.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Songwei Lu et al. Out-of-distribution detection: A task-oriented survey of recent advances. *ACM Computing Surveys*, 2025.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 116–131, 2018.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. ISBN 9780521865715. URL `https://nlp.stanford.edu/IR-book/`.

Maxime Oquab, Thomas Darcet, Theo Moutakanni, Maciej Szafraniec, Vladislav Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Wojciech Galuba, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. URL `https://arxiv.org/abs/2304.07193`.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 5389–5400. PMLR, 2019. URL https://proceedings.mlr.press/v97/recht19a.html.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html.

A. Q. M. S. Sayyed et al. Out-of-distribution detection in computer vision: A comprehensive survey and research challenges. *arXiv preprint*, 2025.

Walter J. Scheirer, Lalit P. Jain, and Terrance E. Boult. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, 2013.

Yiyou Sun, Yixuan Li, and Hongyang Zhang. Out-of-distribution detection with deep nearest neighbors. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

Jihoon Tack, Seongho Mo, Jaehyung Jeong, and Jinwoo Shin. Csi: Novelty detection via contrastive learning on distributionally shifted instances. In *Advances in Neural Information Processing Systems*, 2020.

Chroma Team. Chroma: The ai-native open-source embedding database. https://github.com/chroma-core/chroma, 2023. Accessed: 2025-07-27.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 10347–10357. PMLR, 2021. URL https://proceedings.mlr.press/v139/touvron21a.html.

Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *International Journal of Computer Vision*, 2024a.

Jingkang Yang, Kaiyang Zhou, Yongxin Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *International Journal of Computer Vision*, 2024b.

Kaiyang Zhou, Jingkang Yang, Yu Qiao, and Tao Xiang. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

A Appendix

Use of Generative AI Tools

Generative AI tools were used only for language editing and formatting. All scientific reasoning, analysis, methodology, and conclusions are solely the responsibility of the authors.

A.1 Algorithm pseudocode

Algorithm 1 Confidence Estimation via Embedding Density Filtering (Single Embedding Model)

Require: Base set (N_{base}), training set (N_{train}), test set (N_{test}), embedding model (m), classifier (C), confidence threshold (L), neighbor count threshold (N), normalized cumulative gain (NCG), and distance function $d(\cdot, \cdot)$.

Build and store embedding index E (images from the Base set)

for all $x \in N_{\text{base}}$ **do**
  $E(x) \leftarrow m(x)$
  $E \leftarrow E \cup \{E(x)\}$
end for

Compute neighbors for tuning and tune density thresholds

for all $x \in N_{\text{train}}$ **do**
  $\hat{y} \leftarrow C(x)$
  $\mathcal{C}_x \leftarrow \{(N_i(x), L_i(x))\}_{i=1}^{K_{\max}}$
    ▷ Top-$K_{\max}$ neighbors of $m(x)$ in E
  Record $(c_x,\ \mathbf{1}[\hat{y} = y])$
end for

Find optimal parameters

$N^*, L^* \leftarrow \arg\max_{N,L} \text{NCG}(N, L)$

Evaluate on N_{test}

for all $x \in N_{\text{test}}$ **do**
  conf $\leftarrow$ **false**
  $\mathcal{N}_x \leftarrow$ top-N^* neighbors of $m(x)$
  $c \leftarrow |\{z \in \mathcal{N}_x : d(z, m(x)) \leq L^*\}|$
  if $c \geq N^*$ **then**
    conf $\leftarrow$ **true**
  end if
  $\hat{y} \leftarrow C(x)$
  Res $\leftarrow$ Res $\cup \{(x, \hat{y}, \text{conf})\}$
end for
return Res

A.2 Choosing the optimal values

Search for optimal threshold parameters. The search for the optimal pair of threshold parameters (L^*, N^*) is performed over discrete grids of candidate distance thresholds $\mathcal{L}$ and neighbor-count values $\mathcal{N}$. The distance threshold grid is defined as

$$\mathcal{L} = \{-0.20,\ -0.195,\ -0.190,\ \ldots,\ 1.000\},$$

constructed using a uniform step size of 0.005, yielding 241 distinct thresholds. This is valid for the cosine similarity metric; for other metrics the potential grid is significantly different. The lower bound of -0.20 accommodates the slight negative cosine similarities that may arise for some embedding models while avoiding the computational expense of sweeping the full $[-1, 1]$ range.

The search procedure consists of two stages. In the first stage, for each $L \in \mathcal{L}$ and for each query sample, we compute how many of its $K_{\max}$ retrieved nearest neighbors satisfy the similarity constraint (similarity $> L$). This preprocessing step has time complexity

$$O(N_{\text{train}}\, K_{\max}\, |\mathcal{L}|)\,,$$

In the second stage, the algorithm sweeps over all threshold pairs (L, N) with $L \in \mathcal{L}$ and $N \in \mathcal{N}$, evaluating coverage and accuracy using the precomputed neighbor counts. This stage has time complexity

$$O(N_{\text{train}}\, |\mathcal{L}|\, |\mathcal{N}|)\,,$$

Integral-based ranking of N. For each fixed N, the smoothed coverage–accuracy curve is denoted by $s(x)$. After extrapolating $s(x)$ to the full interval $[0.1, 1.0]$, we compute the integral score

$$I(N) = \int_{0.1}^{1.0} s(x)\, dx.$$

To account for model-dependent baseline accuracy A_{bench}, we define

$$I_{\text{adj}}(N) = I(N) - 0.9\, A_{\text{bench}},$$

$$I_{\text{norm}}(N) = \frac{I_{\text{adj}}(N)}{0.9\,(1 - A_{\text{bench}})}.$$

The optimal neighbor-count parameter is then

$$N^* = \arg\max_{N \in \mathcal{N}} I_{\text{norm}}(N).$$

The computational cost of the integral evaluation step is modest. L would be the number of coverage–accuracy points per curve. Spline fitting and numerical integration leading to a total of

$$O(|\mathcal{N}|\, |\mathcal{L}|)$$

This is negligible compared to the cost of generating the underlying coverage and accuracy values.

A.3 Computational complexity (storing and querying)

The computational cost of the proposed method has two main components: (i) storing embedding vectors for the base (training) dataset, and (ii) retrieving the nearest neighbors for each query.

Let N_{base} denote the number of stored embeddings and d the embedding dimensionality. Storing one d-dimensional vector per sample results in a memory complexity of

$$O(N_{\text{base}} d),$$

which grows linearly with dataset size. For example, DINOv2-B/14 embeddings ($d = 768$) require approximately 3 KB per image, corresponding to about 3.8 GB for the 1.28M images in ImageNet-1K.

However, the bottleneck is the computation of the embedding vector itself. For DINOv2-B, computing an embedding for a *single image* (batch size = 1) takes approximately 0.05 s, whereas storing the resulting embedding vector in the database requires only 0.005 s. These measurements were obtained without batch processing and therefore reflect the true per-image latency of the pipeline. A structured comparison of both computational and memory costs is presented in Table 5.

Let N_{base} denote the number of stored embeddings. Querying an HNSW approximate nearest-neighbor (ANN) index scales as

$$O(\log N_{\text{base}} \cdot d),$$

Table 5: Computational and memory complexity for DINOv2-B and ChromaDB on CPU (Mac M1 Pro). Values are shown per image and for the full ImageNet-1K dataset (∼1.28M images).

Category	Per image	Per dataset
Computational complexity of storing and querying		
Encoding time	0.050 s	140,800 s (39.1 h)
Recording time	0.005 s	6,400 s (1.78 h)
Total time	0.055 s	147,200 s (40.9 h)
Memory complexity		
Storage per embedding	∼3 KB	—
Total storage	—	∼3.8 GB

where d is the embedding dimension, yielding sublinear retrieval time even for databases with millions of entries.

In practice, ANN retrieval is not the dominant cost. The primary bottleneck is feature extraction: computing a DINOv2-B/14 embedding for a single image takes approximately 0.05 s, which dominates the total per-image runtime. Consequently, overall evaluation time is determined mainly by the embedding network forward pass rather than by nearest-neighbor search.

While memory usage grows linearly with the number of stored embeddings and may become challenging at very large scales, the runtime overhead of ANN retrieval remains negligible compared to embedding computation, contributing only a minor fraction of the total query time.

A.4 OOD comparison

OOD score for embedding density filtering. To relate embedding density filtering to standard OOD evaluation metrics, such as Mahalanobis distance, Energy score, and ODIN, we derive a continuous OOD score from our embedding-density-based confidence criterion. This analysis is not intended as a direct benchmark against logit-based OOD detectors, but rather as a diagnostic view of how embedding-density-based confidence behaves under standard OOD evaluation protocols.

For a query image x, we retrieve its N nearest neighbors $\mathcal{N}_x = \{z_i\}_{i=1}^{N}$ from a fixed embedding database using cosine distance. We define a continuous normalized embedding density score as

$$s(x; N) = \frac{1}{N} \sum_{i=1}^{N} \big(1 - d_i\big), \quad s(x; N) \in [0, 1],$$

where d_i denotes the cosine distance between the query embedding $m(x)$ and its i-th nearest neighbor. Higher values of $s(x; N)$ indicate dense, well-supported regions of the training distribution, while lower values correspond to sparser regions in embedding space.

Following standard OOD evaluation conventions, where larger scores indicate higher out-of-distribution likelihood, we define the OOD score as

$$\mathrm{OOD}(x; N) = 1 - s(x; N).$$

Table 6: Computational cost for DINOv2-B embeddings and ANN retrieval on CPU. Values are reported per image and for ∼1.28M images (ImageNet-1K scale).

Component	Per image	Per dataset
Embedding computation	0.050 s	∼ 140,800 s (39.1 h)
ANN retrieval (CPU)	0.003 s	∼ 3,840 s (1.07 h)

Samples that lie in low-density regions of embedding space therefore, receive high OOD scores, while samples well supported by the training data receive low OOD scores. The time complexity of the algorithm is described in Appendix A.3. Unlike most logit-based OOD metrics, the proposed approach allows the neighborhood size N and distance threshold L to be investigated cheaply as a postprocessing step. In our experiments, we evaluate this score over neighborhood sizes $N \in \{1, 5, 10, 20, 30, 50, 75, 100\}$ and distance thresholds $L \in [0.05, 0.6]$ with a step size of 0.05.

Mahalanobis distance. We employ the Mahalanobis distance method of Lee et al. (2018b). In this approach, in-distribution samples are modeled as a single multivariate Gaussian in a feature space of a pretrained model (classifier backbone). The mean vector and covariance matrix are estimated from a subset of ImageNet-1K training images, and a shrinkage term is added to the covariance matrix for numerical stability. For a query image x with feature representation $f(x)$, the OOD score is given by

$$\mathrm{OOD}_{\mathrm{Mah}}(x) = (f(x) - \mu)^\top \Sigma^{-1} (f(x) - \mu),$$

where larger values indicate a higher likelihood of being out-of-distribution. We follow the standard evaluation protocol of Lee et al. (2018b) without additional tuning.

Energy-based OOD detection. We additionally compare against the Energy-based OOD detection method proposed by Liu et al. (2020). This approach derives an OOD score directly from the classifier logits without requiring access to intermediate features or additional density estimation. Given the logits $f(x)$ of a classifier and a temperature parameter T, the Energy score is defined as

$$\mathrm{OOD}_{\mathrm{Energy}}(x) = -T \log \sum_k \exp\left(\frac{f_k(x)}{T}\right),$$

where lower energy values indicate in-distribution samples and higher values indicate out-of-distribution samples.

In our experiments, we evaluate the Energy score over a discrete set of temperature values

$$T \in \{0.5,\ 1.0,\ 5.0\},$$

and select the optimal temperature on the adjustment split based on AUROC, following the standard practice of Liu et al. (2020).

ODIN. We further compare against the ODIN method proposed by Liang et al. (2018), which improves OOD detection by combining temperature scaling of logits with a small, input-dependent perturbation. Given classifier logits $f(x)$, ODIN first applies temperature scaling with parameter T and then computes the maximum softmax probability as the confidence score.

Specifically, for a given temperature T, an input perturbation is generated by taking a single gradient step that minimizes the cross-entropy loss with respect to the predicted class. The perturbed input is given by

$$x_{\mathrm{adv}} = x - \epsilon\, \mathrm{sign}\big(\nabla_x \mathcal{L}\big(f(x)/T, \hat{y}\big)\big),$$

where $\hat{y}$ is the predicted class and ϵ controls the perturbation magnitude. The final ODIN score is defined as the maximum softmax probability computed from the perturbed input and temperature-scaled logits.

In our experiments, we evaluate ODIN over a grid of temperature and perturbation parameters,

$$T \in \{1.0,\ 10.0,\ 100.0\}, \qquad \epsilon \in \{0.0,\ 0.001,\ 0.002\},$$

and select the optimal parameter pair (T, ϵ) on the adjustment split based on AUROC, following the standard protocol of Liang et al. (2018).

Table 7: OOD detection performance on Internal and External datasets. We report AUROC (higher is better) and FPR95 (lower is better).

Set	Backbone	Method	AUROC	FPR95	Best parameters
Internal	ResNet101	Energy	0.5915 ± 0.0046	0.9251 ± 0.0150	T=0.5
		Mahalanobis	0.5680 ± 0.0110	0.9432 ± 0.0073	None
		ODIN	0.6230 ± 0.0110	0.9191 ± 0.0103	T=1.0, ϵ=0.0
	DeiT-Tiny	Energy	0.5858 ± 0.0065	0.9263 ± 0.0075	T=0.5
		Mahalanobis	0.5850 ± 0.0130	0.9157 ± 0.0134	None
		ODIN	0.5878 ± 0.0053	0.9249 ± 0.0041	T=1.0, ϵ=0.0
	DINO-V2 ViT-B/14	Emb. Density (Ours)	0.5867 ± 0.0044	0.8981 ± 0.0050	N=1, L=0.05
External	ResNet101	Energy	0.9001 ± 0.0060	0.4105 ± 0.0363	T=0.5
		Mahalanobis	0.8950 ± 0.0058	0.3150 ± 0.0262	None
		ODIN	0.8891 ± 0.0069	0.5893 ± 0.0500	T=1.0, ϵ=0.0
	DeiT-Tiny	Energy	0.7979 ± 0.1368	0.6120 ± 0.2275	T=0.5
		Mahalanobis	0.7243 ± 0.0003	0.6344 ± 0.0053	None
		ODIN	0.8880 ± 0.0047	0.4577 ± 0.0261	T=10.0, ϵ=0.0
	DINO-V2 ViT-B/14	Emb. Density (Ours)	0.8512 ± 0.0005	0.4426 ± 0.0045	N=1, L=0.05

Comparison with different OOD methods. For a fair and robust evaluation, both the *Internal* and *External* datasets were partitioned into three mutually exclusive subsets based on class labels, such that each label appeared in exactly one subset. OOD detection metrics were computed independently on each subset, and the reported results correspond to the mean and standard deviation across these three splits. We report standard OOD evaluation metrics, including AUROC, and FPR95. For methods requiring hyperparameters, such as Energy-based scoring, the optimal parameters (e.g., temperature T) were selected based on validation performance and are explicitly reported in Table 7. Methods without tunable parameters are marked accordingly. For a comparison of computational efficiency and runtime, we refer the reader to Appendix A.5.

For OOD evaluation, samples from ImageNet-1K are treated as in-distribution, while samples from ObjectNet and ImageNet-V2 (validation subset) are treated as out-of-distribution. The OOD score $\mathrm{OOD}(x)$ is computed using the optimal parameters (N^*, L^*) selected on the adjustment split. Standard OOD metrics, including AUROC, and FPR95, are then computed by sweeping a threshold over $\mathrm{OOD}(x)$, following the same protocol used for the baseline OOD methods.

Table 7 reports OOD detection performance on Internal and External datasets. In the Internal setting, all methods—including Energy, Mahalanobis, ODIN, and the proposed embedding density score—exhibit near-chance AUROC (≈ 0.58–0.59) and very high FPR_{95}, indicating substantial overlap between in-distribution and internally shifted samples. In this regime, the embedding density score closely matches logit-based baselines despite relying on a deliberately simple density-based scoring function $S(x)$ defined on nearest-neighbor distances in embedding space; notably, using only the single nearest neighbor ($N = 1$) is sufficient to achieve this performance.

In contrast, performance improves markedly on the External dataset, where the distribution shift is stronger. Energy and ODIN achieve the highest AUROC (≈ 0.90), benefiting from access to classifier logits and calibration, while the proposed method attains a competitive AUROC of 0.85 under strictly weaker assumptions.

Table 8: Runtime, query-time overhead, and memory requirements for OOD methods using a ResNet101 backbone on CPU. Preprocessing time is reported per image (excluding total preprocessing time) and corresponds to offline evaluation over the full dataset (Mahalanobis: 15k images; others: ∼1.28M images).

Method	**#Samples**	**Preproc (s/img)**	**Total (h)**	**Query (s)**	**Memory**
Energy	∼1.28M	∼0.15	∼53.3	∼0	∼1 GB
Mahalanobis	15k	∼0.05	∼0.2	∼0	∼400 MB
ODIN	∼1.28M	∼0.20	∼71.4	∼0.01	∼1 GB
Ours (Emb. Density)	∼1.28M	∼0.055	∼39.1	∼0.003	∼8 GB

Overall, the embedding density score follows standard OOD behavior: it degrades when representation overlap is high and improves under larger distribution shifts, with the remaining gap highlighting the additional discriminative signal provided by classifier-specific information beyond embedding geometry alone.

Applicability of the comparison of OOD methods. OOD detection methods differ fundamentally in the information they require and the architectures on which they operate. Logit-based approaches such as Energy and ODIN rely on classifier outputs (and, in some cases, input gradients) and are therefore applicable only to supervised models with task-specific heads. In contrast, the proposed embedding density score operates solely on frozen feature embeddings and does not require classifiers, logits, or gradients. Consequently, methods are evaluated under different backbones: Energy, Mahalanobis, and ODIN use supervised classifiers, while embedding density uses a self-supervised DINO-v2 ViT backbone. Enforcing a shared representation would require auxiliary classifiers or probes, altering the assumptions of several methods and violating the embedding-only setting. Accordingly, each method is evaluated within its native regime, and the comparison **should not be interpreted as a representation-controlled benchmark**. Rather, it highlights qualitative differences between classifier-dependent and purely geometric OOD signals.

A.5 Runtime and memory comparison

Here we report the computational cost, query-time overhead, and memory requirements for OOD scoring using a ResNet101 backbone for classic OOD methods. For the embedding density we use the DINO-V2 model. Unless stated otherwise, all dataset- level measurements in Table 8 are reported for CPU execution.

For Mahalanobis, preprocessing is performed on a reduced subset of 15k in-distribution images to estimate class-conditional feature statistics, namely per-class means and a shared covariance matrix. Once computed, OOD scoring for a queried image requires only a forward pass through the backbone followed by a closed-form distance computation, resulting in negligible query-time overhead. Since only a small set of statistics is stored, both the inference-time cost and persistent memory usage are independent of the dataset size.

Energy and ODIN do not require dataset-level statistics beyond those of the trained classifier. Preprocessing corresponds to offline evaluation over the full dataset for reporting purposes, while inference consists of a forward pass through the backbone followed by lightweight score computation. ODIN additionally performs a backward pass with respect to the input to compute the perturbed score, leading to a modest increase in per-image computation. In both cases, memory usage is dominated by the backbone parameters and transient runtime buffers, with no persistent per-image storage.

The embedding density method follows the same backbone-based embedding extraction procedure during preprocessing and inference, resulting in a comparable per-image computational cost for feature extraction. At query time, an additional overhead is incurred by approximate nearest-neighbor search over the stored embedding table, followed by a simple

thresholding operation; this overhead remains small relative to the backbone forward pass. Unlike parametric methods, embedding density requires storing dataset-level representations: precomputed 768-dimensional embeddings extracted using the DINO-V2 backbone, as well as associated density statistics. In addition to storing the base embeddings, the method stores auxiliary density statistics of comparable size, resulting in a total memory footprint of 8GB, consistent with Table 8.

A practical limitation of the proposed method is the need to store training embeddings, which increases memory usage as dataset size grows. For large-scale datasets, this cost can be mitigated with approximate nearest-neighbor indexes and vector compression, such as FAISS IVF with product quantization, while maintaining high recall Jégou et al. (2011); Johnson et al. (2019).