OODEEL: A Holistic Library for Unified Post-Hoc OOD Detection Research And Application

Paul Novello* **Yannick Prudent*** **Corentin Friedrich** **Joseba Dalmau**

IRT Saint Exupéry (DEEL Team), Artificial and Natural Intelligence Toulouse Institute (ANITI)
Toulouse, France. * **Equal contributors**

Abstract

We present OODEEL, an open-source Post-hoc Out-of-Distribution (OOD) detection library. OODEEL is designed as a highly customizable tool that supports a wide range of OOD detectors and can be applied to any model classifier architectures from both PyTorch and TensorFlow. It implements unified abstractions so that every building block, such as activation shaping and layer-wise aggregation, can be used seamlessly by any detector. It also provides a user-friendly API that allows for easy integration of new OOD detectors, which can then benefit from all these building blocks, and native compatibility with most TensorFlow and PyTorch models. OODEEL seamlessly handles standard OOD evaluation settings for benchmarking, including multiple ID/OOD datasets (both near- and far-OOD). Hence, we leverage its holistic implementation to address several critical aspects of OOD evaluation that are often overlooked in current benchmarks: robustness to model variability, effect of aggregation of layer-wise scores, effect of activation shaping, and link between in-distribution accuracy and OOD detection performances.

1 Introduction

Out-of-distribution (OOD) detection has become increasingly crucial for deploying robust machine learning systems in real-world applications. Neural network models, although effective, often exhibit overconfidence when encountering inputs outside their training distribution, which can lead to significant safety and reliability issues. To mitigate this, post-hoc OOD detection methods have emerged as attractive solutions, as they require no retraining and can readily be applied to existing pretrained models. These approaches typically operate by assigning OOD scores to inputs based on either model logits or internal layer feature values.

Some libraries, such as OpenOOD [21] and PyTorch OOD [6], have been developed to foster research, benchmarking, and adoption of OOD detection. Despite their popularity, these libraries have notable limitations. Firstly, they implement OOD detectors separately, often ignoring the potential benefits of combining components such as activation shaping [15, 19, 2] and layer-wise score aggregation [5, 1, 3] across different OOD methods. Secondly, these libraries tend to restrict their usability to predefined models, neglecting the broader applicability of post-hoc methods to arbitrary user-provided classifiers. Lastly, they are generally restricted to a single deep learning framework, typically PyTorch, excluding TensorFlow practitioners from easy access to state-of-the-art OOD detection tools.

In this paper, we introduce OODEEL, an open-source, highly flexible, and holistic library that addresses these limitations. By supporting seamless integration with arbitrary pretrained models from both PyTorch and TensorFlow, it provides a unified implementation of post-hoc OOD detectors and components such as activation shaping and layer-wise aggregation. Moreover, its intuitive and unified API enables effortless application to many OOD detection test cases. OODEEL also includes

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

a benchmarking functionality that allows for easy testing of implemented OOD detectors on the OpenOOD benchmark.

Leveraging the versatility of OODEEL, we conduct extensive experiments to rigorously assess several underexplored aspects of OOD detection, including the sensitivity of methods to model variability, the impact of aggregating scores across multiple layers, and the influence of activation shaping strategies. We also provide additional insights from our experiments about the link between ID accuracy and AUROC, disparate performances on small-scale and large-scale datasets, and correlation between near and far OOD. Our findings not only reveal critical insights into the performance and robustness of current OOD methods but also highlight the practical advantages of a unified approach in developing and evaluating future OOD detection techniques. OODEEL[1] is available on GitHub, and we release an accompanying platform[2] for a comprehensive visualization of experiment results.

2 Background

Out-of-distribution detection can be interpreted as a binary classification problem. The most common way of constructing an OOD detector is by using an OOD scoring function $s : \mathcal{X} \to \mathbb{R}$ and a threshold $\tau \in \mathbb{R}$:

$$\begin{cases} x \text{ predicted ID} & \text{if } s(x) \leq \tau, \\ x \text{ predicted OOD} & \text{if } s(x) > \tau. \end{cases}$$

Throughout the paper we adopt the convention that OOD scoring functions are designed so that they give higher scores to OOD data and lower scores to ID data. Given a pair of datasets $\mathcal{D}^-$ for ID data and $\mathcal{D}^+$ for OOD data, and an OOD detector (s, τ), we can define the following metrics:

- *False Positive Rate (FPR).* The fraction of ID data that is incorrectly classified as being OOD, i.e. $FPR(\tau) = |\{x \in \mathcal{D}^- : s(x) > \tau\}|/|\mathcal{D}^-|$.
- *True Positive Rate (TPR).* The fraction of OOD data that is correctly classified as being OOD, i.e. $TPR(\tau) = |\{x \in \mathcal{D}^+ : s(x) > \tau\}|/|\mathcal{D}^+|$.

In the context of OOD benchmarking, it is customary to measure the separation power of a scoring function s independently of the chosen threshold τ. The most popular such metric is the *AUROC* or area under the ROC curve, which is the parametric curve given by $\big\{\big(FPR(\tau), TPR(\tau)\big), \tau \in \mathbb{R}\big\}$.

Classical benchmarks in OOD research evaluate a score s by computing its AUROC with respect to a bunch of different pairs of ID/OOD datasets, where the ID/OOD pairs are usually chosen so that they have semantically non-overlapping classes.

2.1 Post-hoc OOD

Post-hoc OOD is a popular class of OOD detection methods (the library OpenOOD [21] implements 24) that apply to already trained models without requiring training or fine-tuning. Let f denote a classifier neural network whose weights have already been fixed through a training process. We assume the classifier f is obtained as a composition of different parametric functions called *layers*,

$$f(x) = h^\ell \circ h^{\ell-1} \circ \cdots \circ h^1(x).$$

Assume that the values $f(x) \in \mathbb{R}^{d_\ell}$ represent the (unnormalized) logits and let us denote by $g^l(x) = h^l \circ \cdots \circ h^1(x) \in \mathbb{R}^{d_l}$ the l-th layer features. Traditional post-hoc OOD detection methods can be split into two main families:

Logit-based. The OOD score $s : \mathcal{X} \to \mathbb{R}$ is obtained by applying a scoring function $\sigma : \mathbb{R}^{d_\ell} \to \mathbb{R}$ on the logit space: $s = \sigma \circ f$.

Feature-based. The OOD score $s : \mathcal{X} \to \mathbb{R}$ is obtained by applying a scoring function $\sigma : \mathbb{R}^{d_l} \to \mathbb{R}$ on an intermediate feature space: $s = \sigma \circ g^l$ for some $1 \leq l < \ell$. The most common choice is to take the penultimate layer features, i.e. $l = \ell - 1$. R

Moreover, in the OOD literature, some techniques have been applied to some of the previous OOD detectors to augment their performances:

[1] OODEEL: `https://github.com/deel-ai/oodeel`

[2] Visualization platform: `https://oodeel-benchmark.streamlit.app/`

Activation shaping. This methods can be seen as *add-ons* to logit-based scores: an additional (clipping, normalization,...) transformation $\tau : \mathbb{R}^{d_{\ell-1}} \to \mathbb{R}^{d_{\ell-1}}$ is applied to the penultimate layer $g^{\ell-1}$ before applying the last layer. The scoring function becomes $s = \sigma \circ h^{\ell} \circ \tau \circ g^{\ell-1}$.

Layer-wise score aggregation. This method can be seen as an *add-on* to feature-based scores. It aggregates the scores computed on different internal layers. For a given example x, a set (or a subset) of $\ell - 1$ different scores are computed from the different feature spaces $s_l(x) = \sigma_l \circ g^l(x)$, which are then aggregated together into a single score via an aggregation function $\mathcal{A} : \mathbb{R}^{\ell} \to \mathbb{R}$, i.e. the final score is $s(x) = \mathcal{A}\big(s_1(x), \ldots, s_{\ell-1}(x)\big)$.

Note that some of the OOD methods require *fitting* the functions σ, τ, or $\mathcal{A}$ on the ID training data, while others do not. We give references to such techniques in Section 3.3 for conciseness.

2.2 Motivations for (Another) Post-hoc OOD Detection Library

Post-hoc OOD is convenient because it theoretically applies to any pre-trained model and does not need any specific training procedure. Moreover, it has been shown by the OpenOOD benchmark that this class of OOD detectors is on par with other training-based methods [21]. Hence, some libraries such as OpenOOD and Pytorch OOD implement a large set of post-hoc OOD detectors. However, **(1)** these libraries implement OOD detectors separately, as in the original papers, whereas some ideas from one could be applied to others. For instance, activation shaping techniques can be applied to all logit-based methods, and layer-wise score aggregation to all feature-based methods. In OODEEL, we use abstractions that allow us to apply these components to any OOD detector, when appropriate. Moreover, **(2)** they restrict the application of these methods to predefined models, which are manually overloaded one by one to be equipped with OOD detection capabilities. We believe that the strength of post-hoc OOD detection lies in its broad applicability to any pre-trained model. Hence, in OODEEL, we developed a tool to build OOD detectors on top of any user-provided pre-trained model. Finally **(3)** these libraries are developed in PyTorch only, missing the TensorFlow community. As we shall see, OODEEL's engine is built to make OOD detectors available for both backends, without duplicating code.

3 A Library for Unified Post-hoc OOD Detection

In this section, we give an overview of OODEEL's usage and features. We first emphasize the simplicity of its API and how to use it for any model in both TensorFlow and PyTorch. Then, we describe its core abstractions and how they enable a unified and broad application of Post-hoc OOD detection. We also provide a list of implemented OOD detectors. Finally, we describe some additional quality-of-life features that are bundled in OODEEL. All the described features are explained in detail, together with API documentation and Jupyter tutorials, in OODEEL's documentation.

3.1 User API

The basic workflow of using OODEEL can be broken down into three main steps:

1. **Data preparation:** Load and prepare the dataset using the `DataHandler` object. A TensorFlow `tf.data.Dataset` or a PyTorch `torch.data.Dataset` is loaded either from the pre-specified hub (`tensorflow-datasets` for TensorFlow and `torchvision` for PyTorch by default, but both can load data from HuggingFace with the argument `hub=huggingface`), or from custom input data (`tuple` or `dict` of `np.ndarray`, or existing TensorFlow or PyTorch `Dataset`). Then, `data_loader.prepare()` prepares the Dataset for scoring, and returns a prepared `tf.data.Dataset` or a `torch.data.DataLoader`.
2. **OOD Detector instantiation and scoring:** Instantiate the detector with appropriate hyperparameters depending on the chosen detector. Note that all the logit-based detectors can be used with implemented activation-shaping techniques, ReAct, ASH, and SCALE, with the arguments `use_react/ash/scale=True`. Then, fit the detector to the provided `keras` or `torch.nn` pretrained classifier. Feature-based detectors (like DKNN or Mahalanobis) leverage internal representations of neural network classifiers. For these detectors, the user has to indicate what layer(s) the detector will use to compute the score through the argument `feature_layers_id`, which is a list of names of such layers. The names can be easily found using `.summary()` functions or convenient graph visualization tools such as Netron app. When several layers are given as input, the detector aggregates the scores for each

Figure 1: User API for using Mahalanobis detector with Cifar10/SVHN as ID/OOD datasets. **(1) Data preparation:** Instantiate a `DataHandler` depending on the backend, load the ID dataset from the default hub. **(2) Detector instantiation and scoring:** Instantiate the detector, fit it to the user-provided pretrained classifier, and score the datasets. **(3) Compute OOD metrics**.

layer using an `aggregator` object. Often, these feature-based detectors need a third ID fit dataset, which can be built out of the ID training dataset. In such cases, the user needs to specify it with the argument `fit_dataset`.

3 **Computing OOD metrics:** Compute the selected OOD metrics from the scores computed in the previous step. Note that it is possible to give any metrics from `sklearn.metrics` as an argument when relevant.

The APIs are the same for TensorFlow and Pytorch, from data loading to computing metrics. All these steps are illustrated for the Mahalanobis detector in Figure 1.

3.2 Core Structure

OODEEL is built on four main core abstractions we outline in this Section. The interactions between these abstractions for a detection workflow are illustrated in Figure 2.

`DataHandler` From the user's point of view, this abstraction instantiates data from dataset hubs Torchvision, Tensorflow Datasets, or Hugging Face, or from user-provided datasets. It also prepares datasets for scoring (prepared datasets can also be used for training). It is instantiated using `load_data_handler()`, which either loads `TFDataHandler` or `TorchDataHandler` automatically depending on the backend. The two child classes share the same API, which does not depend on the backend, so all this is transparent to the user.

`FeatureExtractor` This abstraction runs completely under the hood, and extracts internal features of any pretrained models in `keras` (with `KerasFeatureExtractor` child class) or `torch.nn` (with `TorchFeatureExtractor` child class) based on layer names or indices. It is fit on a model when using `OODBaseDetector.fit()` and can be used in place of the original model. Its `.predict()` method returns the model outputs with its internal features.

`Operator` This abstraction also runs under the hood and provides a single API for tensor operations, regardless of the backend. When instanciating a detector, the backend is detected and either `TorchOperator` or `TFOperator` is loaded. All the detectors' calculations are implemented with the `Operator`'s API and hence do not need a separate implementation to work with PyTorch or Tensorflow. The spirit of this abstraction is the same as `keras.ops`[3] for Keras 3, or EagerPy [11].

`OODBaseDetector` It is the main engine of the library, and all detectors inherit from this class. It manages the use of activation-shaping and layer aggregation techniques, and loads appropriate `DataHandler`, `FeatureExtractor`, and `Operator` that will be used at runtime.

Some key practical advantages of the OODEEL implementation deserve to be highlighted.

[3] `https://keras.io/api/ops/`

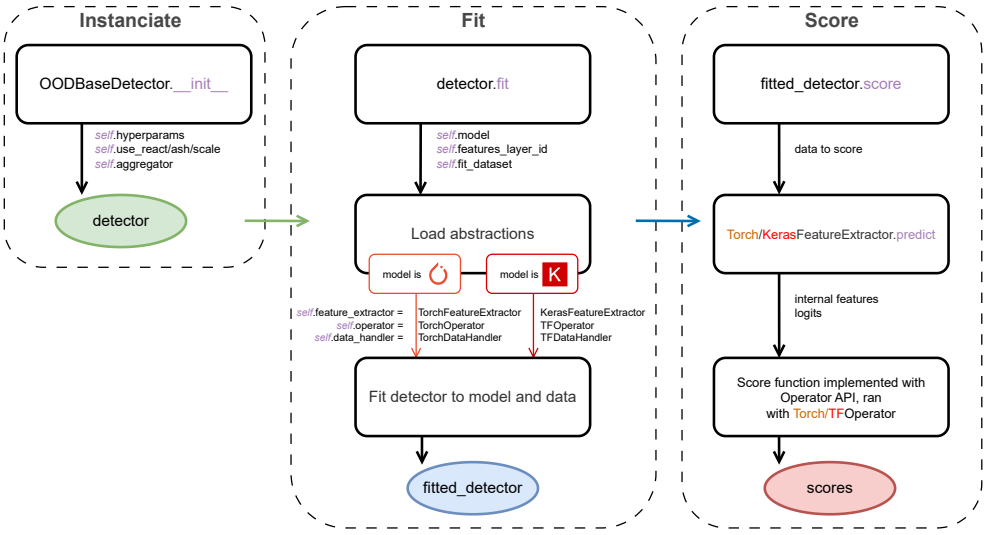

Figure 2: Details on interactions between `OODBaseDetector`, `DataHandler`, `FeatureExtractor` and `Operator` when using OODEEL. Note that for simplicity, we omit to represent `HFFeatureExtractor`

Compatibility with both PyTorch and Tensorflow: The implementation of a common API for PyTorch and TensorFlow specific classes allows us to write code using this API, without worrying about which class, and therefore which backend will be used.

Seamless data loading: Using `DataHandler`, data can be loaded from main datasets hubs: Torchvision, Tensorflow Datasets, and HuggingFace datasets, or from custom Datasets, with a backend-agnostic API.

Compatibility with any pretrained Keras and torch.nn classifier models: `FeatureExtractor` and its child classes allow the extraction of internal features from all pretrained models solely using the layer names. This is a key advantage compared to other OOD libraries that reimplement common models to make them compatible with their code.

Unified implementation: `OODBaseDetector` is implemented so that, when it is applicable, every detector's component that can be used with other detectors, even if it was not the case in these detectors' initial implementation. Hence, we can apply activation shaping techniques such as ReAct to every logit-based detector, and aggregate the scores of several layers for all feature-based detectors.

Easy implementation of new custom detectors: Using `Operator`, it takes only a few class overrides to implement a new detector that enjoys all the previous features. A tutorial in the documentation is provided to guide practitioners through such an implementation.

3.3 Implemented OOD detectors

OODEEL is not to be compared with benchmarking software like OpenOOD. It implements fewer OOD detection techniques since our focus is to unify their implementations and make them applicable to any model in both TensorFlow and PyTorch. Hence, we selected OOD detection techniques that are either the most cited or among the best of the OpenOOD benchmark. The included OOD detection techniques, sorted by category, are as follows. **Feature-based detectors**: Mahalanobis [7], DKNN [16], VIM [18], RMDS [12], SHE [20], Gram [14]. **Logit-based detectors**: MLS [17], MSP [4], Energy [9], Entropy [13], GEN [10], ODIN [8]. **Activation-shaping techniques**: ReAct [15], ASH [2], SCALE [19]. **Layer-wise aggregation techniques**: Fisher [1, 3], Normalized [14].

3.4 Additional features

OODEEL comes with additional Visualization and benchmarking features that we describe in the Appendix. We use the benchmarking feature in the following section.

4 A Unified Benchmark for Post-hoc OOD Detection

OpenOOD is the reference benchmark for OOD detection. However, the leaderboard they provide suffers from limitations. **(1)** No generic feature aggregation techniques are tested with feature-based detectors. **(2)** Activation shaping is only applied to energy, whereas it could be applied to any logit-based methods. These two limitations are design choices that are perfectly legitimate when the intention is to provide a benchmark faithful to the methods presented in the original papers. **(3)** More problematically, each experiment is run on very few different neural networks. As an example, OOD detectors are tested on Cifar10 and Cifar100 with only one model, and on Imagenet with three models, whereas, as we shall see, the OOD detector's performance rank can change a lot depending on the underlying model.

Motivated by these limitations, we leverage the holistic and interoperable implementation of OODEEL OOD detectors to comprehensively test the **impact of layer-wise score aggregation**, **the impact of activation shaping**, and **the effect of the model** (including on activation shaping and layer-wise aggregation). We also leverage our experiments to provide additional insights into the link between ID accuracy and AUROC, disparate performances on small-scale and large-scale datasets, and correlation between near and far OOD

4.1 Setting

We test all the implemented OOD detectors following OpenOOD benchmark settings, with Cifar10, Cifar100, and Imagenet as ID datasets and corresponding near-OOD and far-OOD sets of OOD datasets. For Cifar10, Cifar100, we test the detectors on 11 different models, and for Imagenet, on 7 different models. We select the best hyperparameters for each OOD detector using a simple grid search. Logit-based detectors are all assessed with and without activation-shaping, and feature-based detectors with three levels of score aggregations: no aggregation, partial aggregation, where we aggregate the last layers, and full aggregation, where we aggregate all the layers. We use the Fisher aggregation technique. As a result, a total of 42 different detectors are tested on 29 different models. All the following figures are extracted from our visualization platform `https://oodeel-benchmark.streamlit.app/`. The full results of our experiments can be found in the Appendix and in the platform. Below we give a summary of the main results and takeaways by displaying only a subset of the results, for the sake of readability.

4.2 Impact of the model

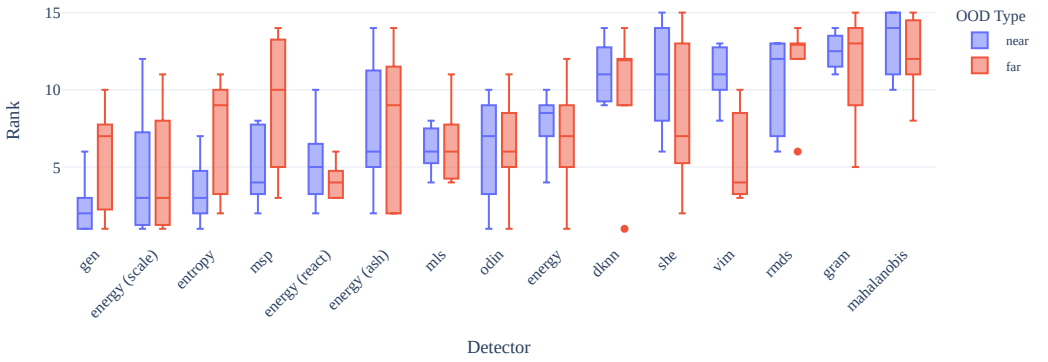

Figure 3: Box plots of the rank for each detector on Imagenet. Methods are sorted according to their mean rank on near-OOD.

For each model, we rank the different OOD detectors according to their AUROC. We then produce a box plot of these ranks with respect to the models for each OOD detector. The results of this experiment for the Imagenet benchmark and the most common OOD detectors are reported in 3. As we can see, the rank varies greatly with the model, and there is no clear winner: for the near-OOD experiments, GEN is the best detector on average, but DKNN is the best detector for one particular model while it ranks 10th on average. We also provide rank correlation heatmaps between OOD detectors for each models in Appendix to corroborate this finding.

Takeaway 1 Results on OOD detectors reported on only one or few models should be interpreted with care, and might not be representative of the real behavior on a wider range of models.

4.3 Impact of Layer-wise Score Aggregation

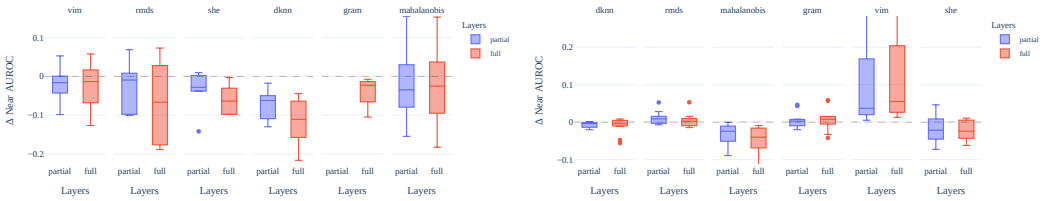

Figure 4: Box plot of the Δ in AUROC between feature-based detector applied on the penultimate layer and either a subset of the internal layers (partial) or all internal layers (full). **Left:** Imagenet, **Right:** Cifar10.

Several recent papers suggest building OOD detectors by aggregating feature-based OOD scores computed layer-wise. We illustrate the impact of layer-wise score aggregation with the following experiment. For each model and feature-based OOD detectors, we compute the gain/loss in AUROC from using layer-wise aggregation with respect to the vanilla version (i.e. computed from the penultimate layer only). For each OOD detector, we produce the box plot of these gains/losses for different levels of aggregation (partial and full) with respect to the models. The results on Imagenet and Cifar10 Near-OOD are reported in Figure 4. For Imagenet, score aggregation can improve the AUROC but hurts more often than it helps. For Cifar10 the results are more positive: it helps more often than it hurts, and greatly boosts VIM's detection capabilities.

Takeaway 2 Score aggregation can have a positive impact, but this positive impact is not systematic. For small-scale tasks, it helps most of the time, but for large-scale tasks, it tends to deteriorate the performance for most models.

4.4 Impact of Activation Shaping

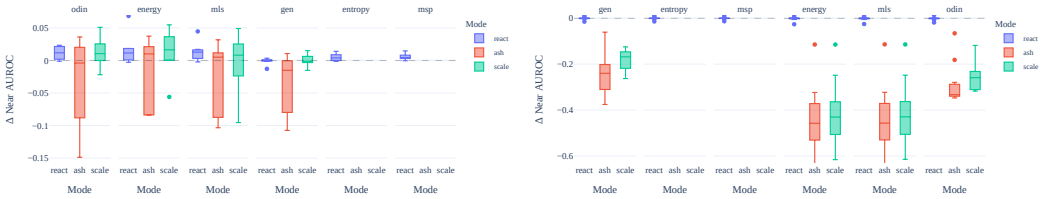

Figure 5: Box plot of the Δ in AUROC between logit-based detectors and their augmentation with activation-shaping techniques (ReAct, SCALE, and ASH). **Left:** Imagenet, **Right:** Cifar10.

Activation shaping techniques transform the penultimate layer features before applying the last linear layer and computing logit-based scores. In the original papers of the ReAct, ASH and SCALE methods, results are reported by combining these activation shaping techniques with the Energy logit-based score alone. In order to evaluate the impact of activation shaping on other logit-base scores, we perform the following experiment. For each model and feature-based OOD detectors, we compute the gain/loss in AUROC from using activation shaping with respect to the vanilla version, i.e. without activation shaping. For each OOD detector, we produce the box plot of these gains/losses for React, ASH and SCALE with respect to the models. The results on Imagenet and Cifar10 Near-OOD are reported in Figure 5. As we can see, activation shaping is mostly beneficial on Imagenet: it is systematically so for the ReAct method, while for ASH and SCALE greater care should be taken, as there are models for which they have negative effects. These results are in great contrast to those obtained for Cifar-10: ASH and SCALE have a nefarious effect on most logit-based scores and models, losing up to 0.6 in AUROC, while the effect of using ReAct is negligible.

Takeaway 3 Using Activation shaping helps on large-scale tasks (for React, consistently, and for ASH and SCALE, more randomly). For small-scale tasks, it has little effect in most cases, while SCALE and ASH have the potential to greatly damage the original detectors.

4.5 Additional Insights

4.5.1 Link Between ID Accuracy and OOD Detection Performances

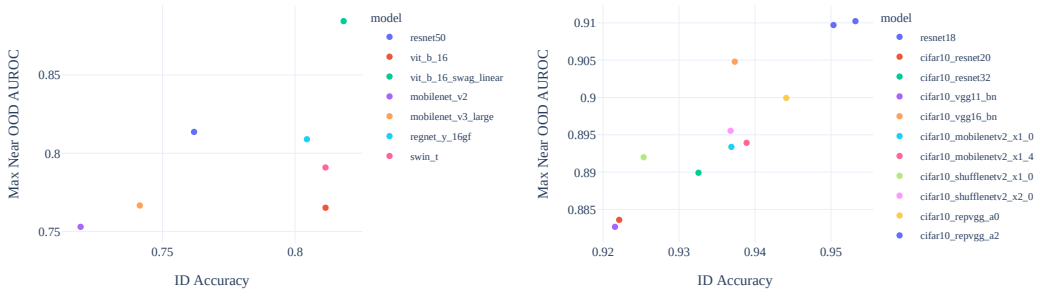

Figure 6: Scatter plot with in-distribution accuracy on x-axis and AUROC on y-axis. Each point is a model, whose assigned AUROC is the best among all OOD detectors applied to this model. **Left:** Imagenet, **Right:** Cifar10.

OODEEL allows for a thorough evaluation of ID accuracy versus OOD detection performance across different neural network architectures. In 5 we report the results of testing multiple different model architectures on the Imagenet and Cifar10 Near-OOD benchmarks. For each model architecture, the AUROC of the best OOD detector (for that model and setting) is reported. As we can see, ID accuracy and OOD detection are linearly correlated in the case of Cifar-10, while there is no clear pattern emerging from the Imagenet experiment: the four best models achieve similar ID accuracy while having very different OOD detection performances. This conclusion contrasts with [17], which observed that improving ID accuracy was an efficient way of enhancing OOD detection performances.

Takeaway 4 Correlation between ID accuracy and OOD detection seems to be near linear for small-scale tasks and absent for large-scale tasks.

4.5.2 Disparate Performances for detectors on Small-scale and Large-scale tasks

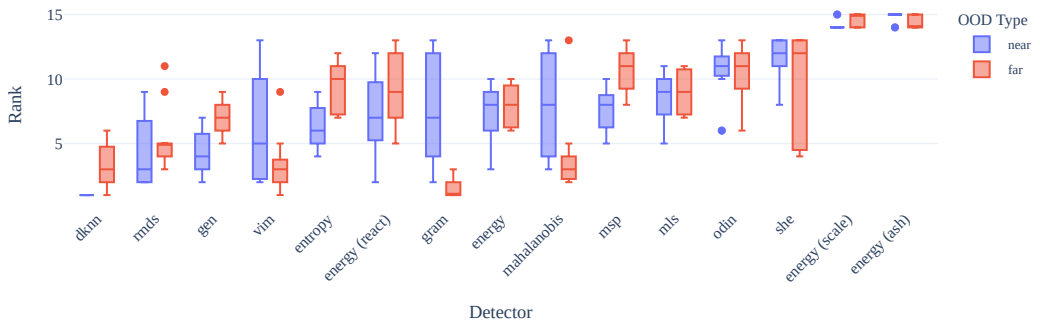

Figure 7: Box plots of the rank with standard deviation for each method (OOD detector) on Cifar10. Methods are sorted according to their mean rank on near-OOD.

By comparing the ranks of the different detectors on Imagenet as displayed in Figure 3 with those obtained by performing the equivalent experiment on Cifar10 (Figure 7), we can conclude that logit-based detectors perform better for large scale models while feature-based detectors perform

better the winners for small-scale tasks. This might explain why activation shaping dedicated to logit-based helps for Imagenet but not for Cifar10, and conversely for feature aggregation.

Takeaway 5 Logit-based detectors perform better on large-scale tasks, and feature-based detectors perform better on small-scale tasks.

4.5.3 Trade-off Between Near-OOD and Far-OOD

We can also observe from Figure 8 that for some ID datasets, there is a tradeoff between performing well on the Near-OOD datasets and performing well on the Far-OOD datasets. This is particularly clear in the case of the Cifar-100 dataset, and even if less so, it remains true for Imagenet and Cifar-10.

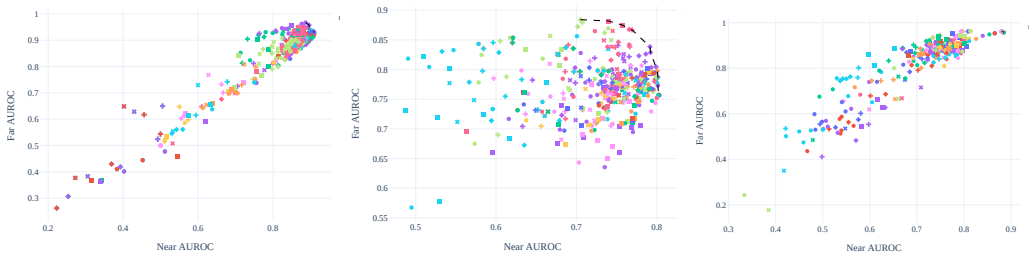

Figure 8: Scatter plot with near-OOD AUROC on x-axis and far-OOD AUROC on y-axis. EAch point is a pair model / OOD detector. **Left:** Cifar10, **Middle:** Cifar100, **Right:** Imagenet.

Takeaway 6 For some tasks, Near-OOD and Far-OOD might not be correlated.

Overall Takeaway

Perhaps the most important takeaway of this work is that in Post-Hoc OOD detection, detector performances greatly depend on the task (dataset and model). Benchmarking their performance appropriately requires testing it on at least several models, which is not classically done in post-hoc OOD detection research. But it can at most mitigate the issue. For post-hoc OOD detection on real-world applications, i.e., with user-specific datasets and models, it is crucial to be able to test detectors on the specific task at hand, hence the value of OODEEL, which can be applied to any model and datasets.

5 Conclusion and Future Works

We introduced OODEEL, a holistic and extensible library for post-hoc OOD detection, designed to work seamlessly across PyTorch and TensorFlow models. By unifying key components—such as activation shaping and layer-wise score aggregation—OODEEL provides a powerful platform for both practical application and rigorous experimentation. Our large-scale benchmarking reveals important nuances in post-hoc OOD detection performance, including the impact of model choice, the varying utility of aggregation and activation shaping techniques, the context-dependent relationship between ID accuracy and OOD detection efficacy, the disparate performances of OOD detectors on different tasks and on near-OOD and far-OOD detection. These findings emphasize the need for careful, context-aware evaluation of OOD detectors. With its open-source release and accompanying benchmarking interface, OODEEL aims to support reproducible, extensible, and insightful OOD research for the broader machine learning community.

While OODEEL offers a comprehensive and modular framework for post-hoc OOD detection, some areas of improvement remain. First, the library currently implements a subset of the most cited and best-performing OOD detectors from OpenOOD, prioritizing compatibility and unification over exhaustiveness. Consequently, newer or more experimental detectors may not be readily available. Second, although our benchmark expands beyond existing evaluations by testing across many models and configurations, it remains limited to image classification tasks. Since OODEEL's `FeatureExtractor` class can be applied to any model, this is the most direct follow-up of our work.

References

[1] E. Dadalto, F. Alberge, P. Duhamel, and P. Piantanida. Combine and Conquer: A Meta-Analysis on Data Shift and Out-of-Distribution Detection, June 2024.

[2] A. Djurisic, N. Bozanic, A. Ashok, and R. Liu. Extremely Simple Activation Shaping for Out-of-Distribution Detection. In *The Eleventh International Conference on Learning Representations*, Sept. 2022.

[3] M. Haroush, T. Frostig, R. Heller, and D. Soudry. A statistical framework for efficient out of distribution detection in deep neural networks, Mar. 2022.

[4] D. Hendrycks and K. Gimpel. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks, Oct. 2018.

[5] A. Himmi, G. Staerman, M. Picot, P. Colombo, and N. M. Guerreiro. Enhanced hallucination detection in neural machine translation through simple detector aggregation. *arXiv preprint arXiv:2402.13331*, 2024.

[6] K. Kirchheim, M. Filax, and F. Ortmeier. Pytorch-ood: A library for out-of-distribution detection based on pytorch. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4351–4360, June 2022.

[7] K. Lee, K. Lee, H. Lee, and J. Shin. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. *arXiv:1807.03888 [cs, stat]*, Oct. 2018.

[8] S. Liang, Y. Li, and R. Srikant. Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks, Aug. 2020.

[9] W. Liu, X. Wang, J. D. Owens, and Y. Li. Energy-based Out-of-distribution Detection, Apr. 2021.

[10] X. Liu, Y. Lochman, and C. Zach. GEN: Pushing the Limits of Softmax-Based Out-of-Distribution Detection. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23946–23955, Vancouver, BC, Canada, June 2023. IEEE.

[11] J. Rauber, M. Bethge, and W. Brendel. EagerPy: Writing code that works natively with PyTorch, TensorFlow, JAX, and NumPy. *arXiv preprint arXiv:2008.04175*, 2020.

[12] J. Ren, S. Fort, J. Liu, A. G. Roy, S. Padhy, and B. Lakshminarayanan. A Simple Fix to Mahalanobis Distance for Improving Near-OOD Detection, June 2021.

[13] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. Depristo, J. Dillon, and B. Lakshminarayanan. Likelihood Ratios for Out-of-Distribution Detection. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[14] C. S. Sastry and S. Oore. Detecting Out-of-Distribution Examples with Gram Matrices. In *Proceedings of the 37th International Conference on Machine Learning*, pages 8491–8501. PMLR, Nov. 2020.

[15] Y. Sun, C. Guo, and Y. Li. ReAct: Out-of-distribution Detection With Rectified Activations, Nov. 2021.

[16] Y. Sun, Y. Ming, X. Zhu, and Y. Li. Out-of-Distribution Detection with Deep Nearest Neighbors, Dec. 2022.

[17] S. Vaze, K. Han, A. Vedaldi, and A. Zisserman. Open-Set Recognition: A Good Closed-Set Classifier is All You Need?, Apr. 2022.

[18] H. Wang, Z. Li, L. Feng, and W. Zhang. ViM: Out-Of-Distribution with Virtual-logit Matching, Mar. 2022.

[19] K. Xu, R. Chen, G. Franchi, and A. Yao. Scaling for Training Time and Post-hoc Out-of-distribution Detection Enhancement, Sept. 2023.

[20] J. Zhang, Q. Fu, X. Chen, L. Du, Z. Li, G. Wang, X. Liu, S. Han, and D. Zhang. Out-of-Distribution Detection based on In-Distribution Data Patterns Memorization with Modern Hopfield Energy. In *The Eleventh International Conference on Learning Representations*, Sept. 2022.

[21] J. Zhang, J. Yang, P. Wang, H. Wang, Y. Lin, H. Zhang, Y. Sun, X. Du, Y. Li, Z. Liu, et al. Openood v1. 5: Enhanced benchmark for out-of-distribution detection. *arXiv preprint arXiv:2306.09301*, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main contribution is the library OODEEL. The library as well as its main features are accurately reported in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We added a few limitations in the Conclusion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA] .

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Not only does the paper include all information needed to reproduce the main experimental results. The OODEEL library as well as the repository we use for the benchmarking experiments are open-source and public.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All code is provided in the OODEEL library and the additional repository for benchmarking experimets, the necessary information to reproduce the experiments is also included.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The main details are provided in the paper in the appendix. The full details are provided in the public repository

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: There is no randomness involved in our experiments, as OOD detection is performed by computing metrics over the entire datasets in question. Thus it is standard practice not to include error bars in expereiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: **[TODO]**

   Justification: All this information is detailed in the Appendix found in the supplementary materials

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We perform no human experiments. All datasets used are well-established, public academic research datasets. The algorithms implemented are state-of-the-art published algorithms.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The paper does not introduce any new algorithms or methods, it only introduces a library where published state-of-the-art methods are implemented.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We use standard benchmarking datasets such as Imagenet or Cifar-10 which is explicitly stated and credited in the paper.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We introduce the OODEEL library which is an open-source publicly available library and a companion benchmarking repo, both of therm are documented.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.