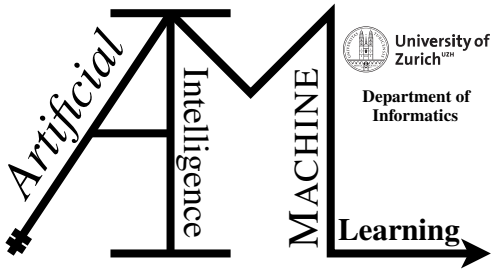
Open-Set Recognition: Modular Framework Expansion and Benchmarking

Master Project

Yuchen Qin & Qichen Zhao
23-756-695 & 23-754-401

Submitted on
October 30, 2025

Thesis Supervisor
Prof. Dr. Manuel Günther
Laurin van den Bergh



Declaration of Independence for Written Work

I hereby declare that I have **composed** this work independently and without the use of any aids other than those declared (including generative AI such as ChatGPT) – the use of generative AI to **improve** my composed work was permitted by the thesis supervisor. I am aware that I take full responsibility for the scientific character of the submitted text myself, even if AI aids were used. All passages taken verbatim or in sense from published or unpublished writings are identified as such. The work has not yet been submitted in the same or similar form or in excerpts as part of another examination.

Zürich, 24.11.2025

Place, Date

Yuchen Qin Q. Zhao

Yuchen Qin & Qichen Zhao

Master Project

Author: Yuchen Qin & Qichen Zhao, yuchen.qin@uzh.ch, qichen.zhao@uzh.ch

Project period: 13.12.2024 - 12.11.2025

Artificial Intelligence and Machine Learning Group
Department of Informatics, University of Zurich

Acknowledgements

We would like to extend our deepest gratitude to Prof. Dr. Günther, whose insightful guidance and unwavering support have been invaluable throughout the duration of this project. Prof. Guenther's expertise in artificial intelligence and machine learning not only shaped the direction of our research, but also inspired us to pursue a deeper understanding of open-set recognition. His constructive feedback, critical thinking, and encouragement motivated us to overcome challenges and strive for scientific rigor in every aspect of our work.

We are also profoundly grateful to PhD assistant Laurin, who provided us with continuous technical assistance and practical advice. Laurin's willingness to share his knowledge, answer our questions, and help us troubleshoot both conceptual and coding issues made a significant difference in the progress of our project. His patience and attention to details helped us refine our ideas, improve our implementation, and ensure the reliability of our experimental results. We would have struggled much more if it weren't for his patience. Huge thanks towards him.

Beyond direct academic support, both Prof. Günther and Laurin fostered an environment of curiosity and collaboration, which greatly enriched our learning experience. Their encouragement to think independently and creatively allowed us to develop new perspectives and approaches, and their dedication to teaching and mentoring has left a lasting impact on our personal and professional growth.

This Master Project would not have been possible without the generous support, guidance, and inspiration from all those mentioned above. We are truly grateful.

Abstract

Open-set recognition (OSR) addresses the challenge of classifying samples when some classes have not been observed during training, a scenario commonly encountered in real-world machine learning applications. This project extends the OpenOSR codebase to provide additional modular functions. Major contributions include the implementation of new dataset protocols, such as Cifar Plus U and flexible class splitting for CIFAR, MNIST, SVHN, and Tiny ImageNet; implementation of various postprocessors and losses; the integration of advanced evaluation metrics and plotting tools, including ROC curve. Extensive experiments demonstrate the correctness and flexibility of the modules, and the enhanced documentation. This work provides a robust foundation for future research and development in open-set recognition.

Contents

Chapter 1

Introduction

The ability to accurately classify samples into known categories while reliably detecting and rejecting novel, unseen classes is a fundamental requirement for deploying machine learning models in open-world scenarios. Traditional closed-set classifiers, which assume all possible classes are known at training time, often perform poorly when encountering out-of-distribution or unknown data, leading to overconfident and potentially dangerous misclassifications.

Open-set recognition (OSR) has emerged as a vital research area to address this limitation. OSR methods aim to distinguish between known and unknown classes, providing mechanisms for sample rejection and uncertainty estimation. However, the development and benchmarking of OSR algorithms have been hindered by the lack of standardized, extensible, and well-documented software frameworks, especially for large-scale and realistic datasets.

This project is dedicated to extending the **OpenOSR** package—a modular, open-source codebase for OSR research. The main objectives are as follows:

1. **Dataset Expansion:** Implementing flexible and reproducible protocols for splitting standard datasets (such as CIFAR, MNIST, SVHN, and Tiny ImageNet) into known, unknown, and negative classes, thus enabling fair and systematic comparison of OSR methods. Additionally, incorporating CIFAR+U as an independent dataset allows for further evaluation and benchmarking of OSR methods. The `splitdataset` framework is extendable to more implementations in the future.

2. **Postprocessor Framework:** Developing a flexible and reproducible framework for implementing and evaluating postprocessors (such as a base class for OOD posrprocessors, including energy-based postprocessor and GradNorm), enabling systematic benchmarking and fair comparison of various postprocessing methods for out-of-distribution (OOD) detection and open-set recognition.

3. **Energy-Based Loss Framework:** Establishing a flexible and reproducible protocol for integrating energy-based loss functions into model training pipelines.

4. **Evaluation and Visualization Tools:** Integrating advanced metrics and plotting utilities, such as ROC curve generation, to facilitate comprehensive performance analysis and result presentation.

5. **Documentation:** Updating and reorganizing code documentation using Sphinx, adding tutorials.

Through these contributions, the project aims to provide a robust, user-friendly, and extensible platform for open-set recognition research, lowering the barrier for newcomers and enabling rapid experimentation and benchmarking of novel algorithms.

DECLARATION: Generative AI tools such as ChatGPT are used in this report for grammar and formating.

Chapter 2

Literature Review

The challenge of open-set recognition (OSR) has attracted significant attention in recent years, as the limitations of traditional closed-set classification become increasingly apparent in real-world applications. This chapter reviews the key developments in OSR research, including foundational concepts, representative algorithms, evaluation protocols, and software tools.

2.1 Foundations of Open-Set Recognition

The term "open-set recognition" was formalized by Scheirer et al. (2013), who argued that real-world recognition systems must be able to handle the presence of unknown classes not seen during training. In contrast to closed-set models, which assume all test samples belong to known categories, OSR systems must provide mechanisms for rejecting or flagging out-of-distribution (OOD) inputs. This paradigm shift has led to a surge of research on both theoretical frameworks and practical algorithms for OSR.

2.2 Classical and Modern OSR Algorithms

Early OSR approaches focused on adapting classical machine learning models to support rejection. Scheirer et al. (2013) formalized open set recognition via open space risk and introduced compact abating probability (CAP) models, along with the Weibull-calibrated SVM (W-SVM). Building on these ideas for deep networks, Bendale and Boult (2016) proposed the OpenMax layer, which statistically models activation vectors to reallocate probability mass to an unknown class and improves open-set detection.

With the advent of deep learning, numerous techniques have been proposed to enhance open-set recognition (OSR) performance. Among these, confidence-based methods play a significant role by utilizing measures such as softmax probabilities, temperature scaling, or entropy to estimate uncertainty, as demonstrated by Hendrycks and Gimpel (Hendrycks and Gimpel, 2017). Generative models, such as generative adversarial networks (GANs) (Neal et al., 2018) and variational autoencoders (VAEs) (Wang et al., 2023), have also been leveraged to model the distribution of known classes and detect anomalies effectively. Another prominent direction involves metric learning and representation learning, where distance-based criteria or feature embedding regularization are employed to better separate known and unknown samples in the latent space (Perera and Patel, 2019). Furthermore, hybrid and ensemble approaches have gained attention for their robustness in open-set detection. For instance, CROSR (Yoshihashi et al., 2019) combines multiple cues by jointly learning classification and reconstruction, fusing the prediction vector y with a

reconstruction-based latent representation z to improve unknown detection. These diverse methodologies collectively push the boundaries of OSR performance and provide a wide array of tools for tackling the challenges of open-set recognition.

2.3 Benchmark Datasets and Evaluation Protocols

Neal et al. (2018) and Dhamija et al. (2018) highlighted the importance of reproducible and transparent protocols, proposing systematic ways to split datasets. However, a challenge in OSR research is the lack of standardized protocols. Early works often defined their own splits of datasets such as MNIST, CIFAR-10/100 and ImageNet into known, unknown, and (optionally) negative classes, making direct comparison under different data protocols difficult.

Also, to ensure the effectiveness of evaluation protocols, it is crucial to define appropriate metrics for assessing the performance of OSR models. Commonly used metrics include the Receiver Operating Characteristic (ROC) curve and the Area Under the ROC Curve (AUROC), which are used to evaluate the trade-off between true positive and false positive rates. Another important metric is the Open Set Classification Rate (OSCR) (Dhamija et al., 2018), which is an adaption of Detection and Identification Rate (DIR) curve from Phillips et al. (2005), it measures the correct classification rate as a function of the false positive rate for unknown samples. However, Wang et al. (2022) pointed out that AUOSCR is difficult to optimize due to its non-differentiable components, such as ranking and counting operations. Additionally, AUOSCR is limited in practical applications, especially regarding the definition and modeling of the unknown class space. To address these issues, Wang et al. (2022) proposed the OpenAUC metric, which couples correct classification of known samples with the rejection of unknown samples in a single probabilistic metric. Additionally, Open Set Accuracy (OSA) is often employed to assess model accuracy in open-set scenarios with varying decision thresholds. Despite the availability of these metrics, the lack of unified tools for implementing evaluation protocols and metrics may hinder the progress and reproducibility in the field.

2.4 Open-Source Frameworks and Tooling

Open-source frameworks play a critical role in advancing research in open-set recognition (OSR). Several toolkits, such as `OpenMax`, `ODIN`, and `OpenOOD`, provide implementations of specific algorithms and evaluation pipelines. However, each framework comes with its strengths and limitations, which are outlined below.

2.4.1 Overview of Existing Frameworks

`OpenOOD`

`OpenOOD` (Yang et al., 2022) is a comprehensive framework designed for out-of-distribution (OOD) detection, supporting a wide range of methods such as energy-based models, temperature scaling, and feature-based approaches. Its unified interface simplifies the reproduction of existing methods and provides support for multiple datasets, making it an invaluable resource for benchmarking OOD algorithms.

However, for open-set recognition (OSR), OpenOOD's default workflow is still OOD-centric. Although some versions and community extensions include OSR-related components, the end-to-end plug-and-play experience is less mature than that for OOD. In particular, template support for class-based known/unknown/negative splits and OSR-specific metrics, such as the Open

Set Classification Rate (OSCR) metrics, is not uniformly integrated across datasets. As a result, researchers often need to manually modify the codebase to adapt it for OSR tasks, which can be time-consuming and error-prone.

OpenMax

`OpenMax` (Bendale and Boult, 2016) recalibrates softmax probabilities to detect unknown classes. Its approach assumes that examples with equally likely logits are more likely to belong to the unseen or rejection class. However, this assumption has limitations. The assumption does not always hold true for examples that are difficult to classify. `OpenMax` requires validation examples from the unseen or rejection class to tune its hyperparameters. This dependency reduces its practicality and limits its extensibility to new datasets or scenarios (Shu et al., 2017).

ODIN

`ODIN` (Liang et al., 2018) relies heavily on the properties of the softmax distribution, using techniques like temperature scaling and input perturbation. While effective for OOD detection, Kim et al. (2024) points out its limitations include not leveraging deeper model features or weight information, focusing solely on the softmax layer. This dependence on softmax restricts its applicability in more complex OSR scenarios.

2.4.2 The `OpenOSR` Framework

The `OpenOSR` framework is designed to address the limitations of existing tools by providing a modular, extensible, and well-documented codebase specifically tailored for open-set recognition (OSR) research. Its key features include flexible dataset splitting, enabling seamless separation of known and unknown classes for OSR-specific evaluation, and the implementation of advanced metrics such as the Open Set Classification Rate (OSCR) and Area Under the ROC Curve (AUROC). Additionally, `OpenOSR` incorporates automated documentation using Sphinx, ensuring comprehensive and user-friendly guidance for users.

By emphasizing reproducibility and extensibility, `OpenOSR` provides standardizing evaluation protocols and metrics and enables researchers to rapidly prototype and evaluate new OSR algorithms to overcome the challenges associated with manual codebase adaptation. This makes `OpenOSR` an essential tool for advancing OSR research and fostering reproducibility in the field.

Chapter 3

Package Contribution

This chapter details the main contributions made to the OpenOSR package during this project, including dataset expansion, postprocessing modules, loss modules, plotting utilities, and comprehensive documentation improvements.

3.1 OpenOSR Overview

The OpenOSR package is a comprehensive tool designed for open-set recognition tasks. It features a highly modular architecture, allowing users to customize each component of their model and process. The package enables users to set up their experiments using a single configuration file, simplifying workflows and improving efficiency.

The modular architecture of OpenOSR includes the following key components:

1. **Backbone:** Supports various backbone networks such as `LeNet` and `TorchModel`, providing flexibility in model selection.
2. **Dataset:** Includes support for multiple datasets such as `CIFAR+U`, `EMNIST`, and `ImageNet1K`. It also offers several dataset-splitting protocols for `CIFAR-10`, `MNIST`, `SVHN`, and `TinyImageNet`.
3. **IO (Input/Output):** Manages components through the `LoadComponent` module, supporting the loading and combination of different experimental configurations.
4. **Model:** Provides multiple open-set recognition models, including `SoftMax`, `MixUp`, `Noise`, and `Original`.
5. **Negative Samples:** Offers strategies for generating and handling negative samples, such as `MixUp` and `Noise`.
6. **Loss:** Supports various loss functions, including `CrossEntropy`, `HypersphericalSoftMax`, `EnergyLoss`, and `ObjectosphereRegularization`.
7. **PostProcessor:** Includes post-processing methods such as `OpenMax`, `ODIN`, `GradNorm`, and `EnergyScore`.
8. **Configurable Parameters:** Users can define all experimental parameters through a configuration file, including default and custom parameters.

The modular design of OpenOSR not only provides rich functionality but also ensures that users can freely combine and adjust components according to their specific needs, making it a powerful tool for research and applications in open-set recognition tasks.

Our work focuses on `CIFAR+U` and `Split Protocol` in Dataset, `Energy Score` based loss and postprocessor module, `Gradnorm`, and `ROC` implementation, shown in Figure 3.1 .

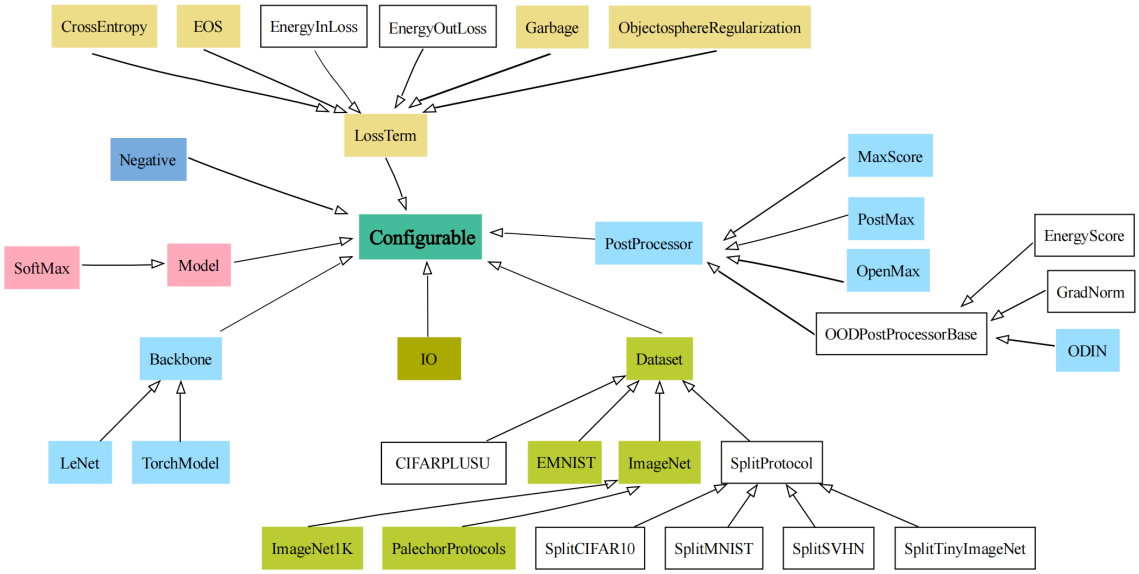


Figure 3.1: Framework of OpenOSR package. Components in transparent boxes represent our contribution.

3.2 Dataset Expansion

3.2.1 CIFAR+U

To enhance the flexibility and reproducibility of open-set recognition experiments, we introduce a new dataset class, `CIFAR+U`, which generalizes and extends the protocols described in Neal et al. (2018); Dhamija et al. (2018). This implementation allows researchers to easily construct datasets such as CIFAR-10, CIFAR-50, and their variants by incorporating additional negative classes from the remaining CIFAR-100 categories.

Motivation and Protocol

Classical OSR experiments on CIFAR typically split classes into "known" (used for training) and "unknown" (used for open-set evaluation). However, these settings often omit "negative" classes—additional distractor categories from the dataset—which can strengthen a classifier's ability to reject out-of-distribution samples.

To address this, `CIFAR+U` provides a configurable protocol that generalizes prior practice. Following the setup proposed in Dhamija et al. (2018), users may specify (i) the number of known classes drawn from CIFAR-10, (ii) the number of unknown classes drawn from CIFAR-100, and (iii) the number of negative classes taken from the remaining CIFAR-100 categories. The framework also supports class-selection constraints in the spirit of Neal et al. (2018), e.g., sampling known classes exclusively from non-animal categories while sampling unknown classes from animal

categories. Finally, reproducibility is ensured by exposing random seeds to generate consistent dataset splits across runs.

This flexible design makes `CIFAR+U` a robust and versatile tool for OSR under controlled and repeatable conditions.

Implementation Details

The `CIFARPLUSU` class is designed to integrate seamlessly with PyTorch training pipelines by inheriting from both the generic `Dataset` base and `torch.utils.data.Dataset`. It provides a flexible configuration system where all parameters can be specified through the configuration file of the original framework. Key implementation features include:

- **Flexible Configuration**:
  - All parameters (e.g., number of known, unknown, and negative classes, random seed, class selection rules) can be specified through the configuration file of the original framework.
- **Dataset Filtering and Relabeling**:
  - The dataset is filtered to include only the selected known, unknown, and negative classes.
  - Known class labels are remapped to a contiguous range (e.g. 0 to n-1).
  - Unknown and negative samples are assigned special labels -2 and -1, respectively), following the conventions of prior work.
- **Reproducibility**:
  - All random splits are determined by a user-defined seed, enabling users to reliably reproduce experiments multiple times under identical splits of known, unknown, and negative classes, ensuring consistency and facilitating comparative analysis.

This comprehensive design ensures that CIFARPLUSU is both versatile and user-friendly, enabling researchers to conduct OSR experiments with ease and reproducibility.

3.2.2 SplitProtocol-based Datasets

To facilitate systematic evaluation of OSR algorithms across diverse image classification benchmarks, a general-purpose dataset abstraction, `SplitProtocol`, and its concrete implementations for several popular datasets were introduced.

Motivation

Datasets is still to be split into known classes, unknown classes and negative classes (optional, used as additional out-of-distribution samples).

However, these splits are rarely standardized in code, making reproducibility and protocol comparison difficult. The `SplitProtocol` abstraction addresses this by providing a configurable, reproducible, and extensible class-splitting mechanism.

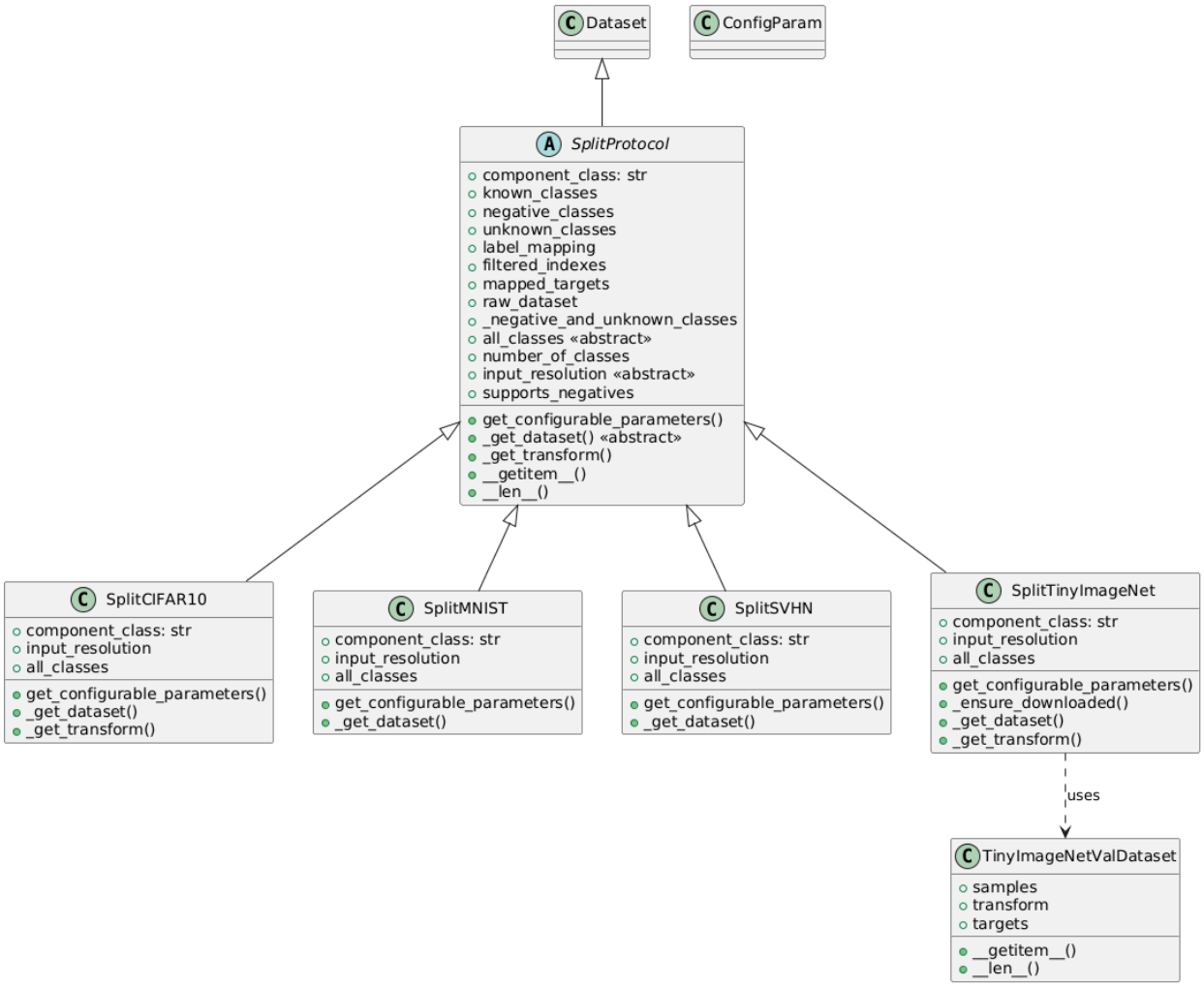


Figure 3.2: UML diagram of the Split protocol.

Implementation Details

`SplitProtocol` is an abstract base class that manages the random selection and mapping of known, unknown, and negative classes for any dataset. Its main features include allowing users to choose desirable number of each classes and normalization statistics for characterized datasets.

The core logic for class selection and mapping is handled in the base class, while each subclass implements dataset-specific details such as loading and input resolution.

Supported Datasets

- **SplitCIFAR10:** Applies the SplitProtocol logic to CIFAR-10, supporting flexible splits (e.g., 6 known, 4 unknown, 0 negative by default).
- **SplitMNIST:** Supports open-set splits for the MNIST digit dataset, with a similar interface.
- **SplitSVHN:** Provides open-set splits for the SVHN (Street View House Numbers) dataset.
- **SplitTinyImageNet:** Extends the protocol to Tiny ImageNet. Handles both training and validation splits, and parses the WordNet class mapping.

The overview of the protocol implementation is presented in the UML diagram 3.2.s

3.3 Postprocessor Implementation

3.3.1 OOD Postprocessor

Out-of-distribution (OOD) detection is crucial for Open-set Recognition (OSR) evaluation because a model must both classify in-distribution (ID, known) samples and reject out-of-distribution (OOD, unknown) samples. However, achieving this dual objective may encounter overconfidence in OOD: classifiers trained solely on ID data are forced to map unfamiliar inputs to known classes (classifier mismatch), and the softmax normalization can yield high confidence even for inputs dissimilar to the ID distribution (softmax artifacts) (Bendale and Boult, 2016). The system must reliably decide whether an input is ID or OOD while also providing accurate class predictions for ID inputs.

These key difficulties disturb traditional OOD detection methods (e.g., ODIN) typically output a single scalar score for OOD detection. While these scalar scores are effective for binary OOD detection, they cannot be directly integrated into the classification task. This limitation makes it challenging for a model to perform well on both OOD detection and ID classification simultaneously.

Motivation

To address this limitation, we introduce a flexible interface module, OOD Postprocessors, which enables the fusion of OOD scores with logits or probabilities along with OOD scores through customizable operations. Unlike traditional OOD postprocessing methods, this approach generates a refined logits or probability distribution, as open-set robustness cannot be reliably achieved by thresholding a single scalar; instead, adjusting the logit/probability space and explicitly allocating probability mass for unknowns is crucial for managing open space risk (Bendale and Boult, 2016). Additionally, this modular interface serves as a base class that can be easily extended by other classes, providing a standardized way to handle and process OOD scores.

Implementation Details

The `OODPostProcessorBase` class serves as a base for implementing Out-of-Distribution (OOD) post-processors in the OpenOSR framework. It enables the fusion of OOD scores with logits or probabilities for open-set recognition (OSR). OOD scores are computed as scalar values for each sample, indicating its likelihood of being OOD.

The `fusion_method` parameter is a key component of the `OODPostProcessorBase` class, determining how OOD scores (converted to ID probability) are fused with network outputs (logits or probabilities). Its main purposes include:

- **Controlling Fusion Strategies**: Provides three options: `none` (no fusion), `multiply_logits` (adjust logits), and `multiply_probabilities` (adjust probabilities).

- **Enhancing Compatibility**: the OOD scores obtained from different OOD algorithms can be unified by first converting the scores into ID probabilities. After this normalization step, the unified probabilities can then be processed using the specified fusion method.This ensures a consistent format for the variables returned, making them compatible with the requirements of other components in the OSR framework.

The overview of the `OODPostProcessorBase` class alongside the OOD post processor child classes it supports is shown in Fig 3.3.

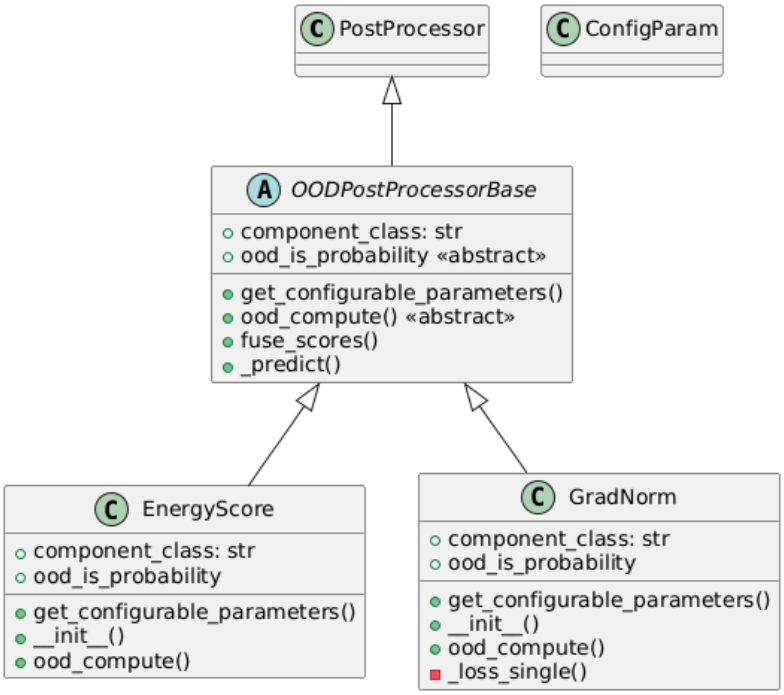


Figure 3.3: UML diagram of OODPostProcessorBase.

3.3.2 GradNorm Postprocessor

GradNorm, proposed by Huang et al. (2021), is a gradient-based post-processing method for out-of-distribution (OOD) detection. It estimates the "unknownness" of samples by measuring the magnitude of KL divergence's gradient norm (to be discussed in Implementation Details section). Typically, in-distribution (ID) samples, consistent with the training data, produce higher gradient norms due to the model's confident predictions and sensitivity near decision boundaries. On the other hand, out-of-distribution (OOD) samples, which deviate from the training distribution, lead to lower-confidence predictions and smaller gradient norms, as the model is less sensitive to these inputs.

Motivation

In many real-world scenarios, obtaining labeled out-of-distribution (OOD) samples during training is often impractical or even impossible, as OOD data is inherently unpredictable and diverse. This highlights the need for methods that can effectively detect OOD samples without relying on prior knowledge or specific training on OOD data. GradNorm addresses this challenge by leveraging the gradients of pre-trained models to estimate the "unknownness" of inputs in a label-agnostic manner. By directly utilizing the model's sensitivity to input samples, GradNorm captures the uncertainty across all categories, providing richer and more comprehensive information for OOD detection.

Beyond its practical benefits for OOD detection, GradNorm also enriches the variety of OOD scores computing available in OpenOSR, providing researchers with an additional tool to evaluate and compare different OOD detection methods. By incorporating GradNorm, OpenOSR facilitates the exploration of various OOD score combinations, helping to identify optimal strategies for diverse datasets and scenarios.

Implementation Details

- **Feature Extraction:** Extract final deep weights and logits from a best pre-trained model as the basis for subsequent calculations. The final deep features are the backbone's penultimate-layer embeddings, which are the semantically rich representation right before the classification head. The weights are chosen from the last fully connected (FC) layer, according to the original work, gradients from the last layer contain the most informative signals to effectively represent the model's behavior. The logits $f(x)$ are the raw outputs of the model before applying the softmax function.

- **KL Divergence Calculation:** The KL divergence measures how one probability distribution diverges from another. ID data is expected to have a higher KL divergence between a uniform distribution u and the model's softmax output because the predictions tend to focus on the ground-truth class, resulting in a less uniform distribution. KL Divergence is defined as:

$$D_{\mathrm{KL}}(p\|q) = \sum_i p_i \log \frac{p_i}{q_i} \tag{3.1}$$

  where:

  - p_i: Probability of class i in distribution p,
  - q_i: Probability of class i in distribution q.

For OOD detection, the softmax output is denoted as $\text{softmax}(f(x))$, the computation formula is as follows:

$$D_{\text{KL}}(u \| \text{softmax}(f(x))) = -\frac{1}{C} \sum_{c=1}^{C} \log \frac{e^{f_c(x)/T}}{\sum_{j=1}^{C} e^{f_j(x)/T}} - H(u) \tag{3.2}$$

where:

- $H(u)$: Entropy of the uniform distribution u, defined as:

$$H(u) = -\sum_{i=1}^{C} u_i \log u_i = \log C \tag{3.3}$$

- **Gradient Norm Calculation:** Huang et al. (2021) stated that through experiments, using the gradient norm of the KL divergence is more effective than directly using the KL divergence itself. Therefore, after obtaining the KL divergence, we first backpropagate it and compute its gradients with respect to weight parameters from the last linear layer (w). This gradient is given by:

$$\frac{\partial D_{\text{KL}}(u \| \text{softmax}(f(x)))}{\partial w} = \frac{1}{C} \sum_{i=1}^{C} \frac{\partial \mathcal{L}_{\text{CE}}(f(x), i)}{\partial w} \tag{3.4}$$

where:

- $\mathcal{L}_{\text{CE}}(f(x), i)$: Cross-entropy loss for class i, defined as:

$$\mathcal{L}_{\text{CE}}(f(x), i) = -\log \text{softmax}(f(x))_i \tag{3.5}$$

- w: Model parameters (e.g., weights of the last linear layer).

Then we aggregate the computed gradients using vector norms (e.g., L_1 or L_2 norm) to derive the OOD score $S(x)$ which is defined as:

$$S(x) = \left\| \frac{\partial D_{\text{KL}}\left(u \parallel \text{softmax}(f(x))\right)}{\partial w} \right\|_p \tag{3.6}$$

where:

- p: Type of norm ($p = 1$ for L_1 norm, $p = 2$ for L_2 norm),
- $\frac{\partial D_{\text{KL}}}{\partial w}$: Gradient of KL divergence with respect to model parameters w.

Challenges in Implementation

The implementation of GradNorm faced several challenges related to efficiency and flexibility.

- **Efficiency Issues:** The initial approach of sample-by-sample backpropagation was computationally inefficient, particularly for large datasets, as it required iterative gradient computations for each individual sample. This inefficiency highlighted the need for faster methods to compute gradients while maintaining accuracy. Although there are batch-based approaches designed for aggregating gradients across batches which compute gradients for the entire batch collectively, the method requires computing per-sample gradients. `vmap` allows for efficient parallelization of per-sample gradient computations while avoiding the overhead of explicit loops, making it well-suited for this task.

- **General Applicability:** While the original work of GradNorm (Huang et al., 2021) indicated that L_1-norm performs best in various norm types and provided a formula for L_1-norm-based gradient computation (as shown in Eq. 3.7), still, this formula is limited to L_1-norms and lacked general applicability for other norm types in case of users' customization. To support a wider range of scenarios, the implementation needed to accommodate arbitrary norm values. This issue was resolved by introducing dynamic norm selection in configuration and using `torch.linalg.vector_norm` to compute gradient magnitudes for arbitrary p-norms, enabling support for a wider range of norm types.

  The L_1-norm-based gradient formula:

$$S(x) = \sum_{i=1}^{m} \sum_{j=1}^{C} \left| \frac{\partial D_{\mathrm{KL}}}{\partial W_{ij}} \right| \tag{3.7}$$

  Expanding the formula:

$$S(x) = \frac{1}{CT} \sum_{i=1}^{m} |x_i| \cdot \left(\sum_{j=1}^{C} \left| 1 - C \cdot \frac{e^{f_j/T}}{\sum_{k=1}^{C} e^{f_k/T}} \right| \right) \tag{3.8}$$

  Further simplifying:

$$S(x) = \frac{1}{CT} \cdot \left(\sum_{i=1}^{m} |x_i| \right) \cdot \left(\sum_{j=1}^{C} \left| 1 - C \cdot \frac{e^{f_j/T}}{\sum_{k=1}^{C} e^{f_k/T}} \right| \right) \tag{3.9}$$

  Finally, expressing as:

$$S(x) = \frac{1}{CT} \cdot U \cdot V \tag{3.10}$$

  Where:

  - $S(x)$: The OOD score for the input sample x, which measures the magnitude of the gradient norm differences.
  - D_{KL}: KL divergence between the uniform distribution u and the model's softmax output softmax$(f(x))$.
  - W_{ij}: The weights of the last linear layer in the model.
  - x_i: The i-th feature of the input sample x.
  - m: The dimensionality of the input sample x (number of features).
  - C: The total number of classes in the classification task.
  - T: Temperature scaling factor that controls the sharpness of the softmax probabilities. We introduce a configurable parameter `temperature` that allows for customization. According to the original work, a temperature value of T=1 is found to be the optimal choice.
  - f_j: Logit (raw output) of the model for class j.
  - $e^{f_j/T}$: Exponential of the scaled logit f_j/T, used in the softmax computation.
  - $\sum_{k=1}^{C} e^{f_k/T}$: Normalization factor in the softmax computation.
  - $U = \sum_{i=1}^{m} |x_i|$: Sum of the absolute values of the input features x_i, representing the overall magnitude of the input.
  - $V = \sum_{j=1}^{C} \left| 1 - C \cdot \frac{e^{f_j/T}}{\sum_{k=1}^{C} e^{f_k/T}} \right|$: Sum of the absolute differences between 1 and the scaled softmax probabilities for each class.

3.3.3 Energy Score Postprocessor

Energy Score is a scoring mechanism to detect Out-of-Distribution (OOD) data using Energy-Based Models (EBM) proposed by Liu et al. (2020). The core idea of Energy Score is to utilize an energy function $E(x; f)$, which maps input samples to scalar values representing their energy levels. In-distribution samples typically have lower energy values, while OOD samples have higher energy values. This approach leverages the theoretical consistency between energy values and probability density based on Gibbs distribution.

Motivation

Energy Score addresses the limitations of traditional softmax-based methods in OOD detection, such as neural networks making overconfident predictions for OOD samples and softmax probabilities failing to accurately represent sample distribution characteristics. It provides a theoretically consistent measure aligned with sample probability density, avoids explicit probability density estimation (simplifying implementation compared to generative models), and effectively distinguishes between in-distribution and OOD samples using energy values.

Implementation Details

The Energy Score computation is implemented as follows:

- **Logits Extraction:** Extract logits from a pre-trained best model for the following calculations. This step ensures that the model outputs necessary information for OOD detection.
- **Energy Score Calculation:** The energy score for each input sample x is computed using the following formula:

$$E(x) = T \cdot \log \sum_{j=1}^{C} \exp\left(\frac{f_j(x)}{T}\right) \tag{3.11}$$

  where:

  - $E(x)$: The energy score for the input sample x. It is a scalar value that indicates how likely the sample is to belong to the in-distribution (ID).
  - x: The input sample, represented as a vector of features.
  - $f(x) = [f_1(x), f_2(x), \ldots, f_C(x)]$: The logits (raw outputs) of the model for the C classes. These are unnormalized scores produced by the model's final layer.
  - $f_j(x)$: The logit corresponding to class j, where $j \in \{1, 2, \ldots, C\}$.
  - C: The total number of classes in the classification task.
  - T: The temperature scaling factor. This parameter adjusts the sharpness of the energy distribution. A higher T results in smoother energy distributions, while a lower T sharpens them. This factor can be characterized by the users during configuration.
  - $\sum_{j=1}^{C} \exp\left(\frac{f_j(x)}{T}\right)$: The partition function, which sums the exponentials of the scaled logits for all C classes. This term ensures that the energy score accounts for contributions from all classes.

  Here, the original formula in the paper (Liu et al., 2020) includes a negative sign ($-T$), but the negative sign is intentionally removed in this implementation. This adjustment is made because the logic of the `OODPostProcessorBase` interface assumes that higher

energy scores represent in-distribution (ID) samples and lower energy scores represent out-of-distribution (OOD) samples. In contrast, the original energy score calculation in the paper uses a logic where lower energy scores represent ID and higher energy scores represent OOD. The Remove ensures consistency with the scoring logic of the parent OOD postprocessor.

3.4 Loss Function Extension

Apart from datasets and post-processors, we also contributed to extending loss functions in the models' training process.

3.4.1 Energy Score Loss

The Energy-score loss is from the same work of Energy-score post-processor mentioned above (Liu et al., 2020). The paper proposed a energy-scored based loss as a regularization term for CE loss (cross entropy), this term fine-tunes the neural network to explicitly create an energy gap by assigning lower energies to the in-distribution data, and higher energies to the OOD data.

Motivation

The motivation behind these loss functions is to address the limitations of standard classification losses, such as CrossEntropy, in OOD detection. While standard losses focus solely on ID samples, energy-based methods provide a principled approach to model the uncertainty of samples by leveraging logits to compute energy scores (Wu et al., 2024).

Also, although the energy score post-processor is informative for a pretrained network, just scoring method itself may not lead to optimal discrimination in energy separation between ID and OOD samples. By integrating the Energy loss with traditional losses (e.g. CE loss), it allows greater flexibility in contrastively shaping the energy surface, resulting in more distinguishable ID and OOD data.

Implementation Details

The Energy-score Loss is implemented by 2 loss terms: the `EnergyInLoss` and `EnergyOutLoss` classes. Dividing the loss into two classes leverages the modularity of the package, enabling users to customize their loss configurations when involving Energy Score loss according to their specific needs, such as configuring different weights for loss terms when constructing the entire loss function.

- **Energy Thresholds:** Two distinct energy thresholds are defined to separate in-distribution (ID) and out-of-distribution (OOD) samples: m_{in} for ID samples and m_{out} for OOD samples. For ID samples, the energy score $E(\mathbf{x})$ is encouraged to remain lower than m_{in}, while for OOD samples, $E(\mathbf{x})$ is pushed to be higher than m_{out}. These thresholds explicitly create an energy gap, enhancing the distinction between ID and OOD samples.
- **Loss Calculation:** The energy-based regularization loss is defined separately for ID and OOD samples using squared hinge losses:

$$\text{loss}_{\text{in}} = \mathbb{E}_{(\mathbf{x}_{\text{in}},y)\sim\mathcal{D}_{\text{in}}^{\text{train}}} \left(\max(0, E(\mathbf{x}_{\text{in}}) - m_{\text{in}})\right)^2, \tag{3.12}$$

$$\text{loss}_{\text{out}} = \mathbb{E}_{\mathbf{x}_{\text{out}}\sim\mathcal{D}_{\text{out}}^{\text{train}}} \left(\max(0, m_{\text{out}} - E(\mathbf{x}_{\text{out}}))\right)^2. \tag{3.13}$$

where:

- $\mathbf{x}_{\text{in}}$: Input samples from the in-distribution training dataset $\mathcal{D}_{\text{in}}^{\text{train}}$.
- $\mathbf{x}_{\text{out}}$: Input samples from the out-of-distribution training dataset $\mathcal{D}_{\text{out}}^{\text{train}}$.
- $E(\mathbf{x})$: The same equation in Eq. 3.11.
- m_{in}: Energy threshold for ID samples.
- m_{out}: Energy threshold for OOD samples.

Equation (3.12) penalizes ID samples whose energy exceeds m_{in}, while Equation (3.13) penalizes OOD samples whose energy falls below m_{out}. Together, these losses enforce a clear energy separation between ID and OOD samples.

- **Combination of Losses:** The original total loss in paper (Liu et al., 2020) combines the classification loss and the energy regularization loss. The classification loss is defined as the cross-entropy loss:

$$\mathcal{L}_{\text{classification}} = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_{\text{in}}^{\text{train}}} \left[-\log F_y(\mathbf{x})\right], \tag{3.14}$$

where:

- $F_y(\mathbf{x})$: The softmax probability of the correct class y for input $\mathbf{x}$.

The energy regularization loss L_{energy} which we focus and implement is defined as:

$$L_{\text{energy}} = \text{loss}_{\text{in}} + \text{loss}_{\text{out}}, \tag{3.15}$$

where loss_{in} and loss_{out} are computed using Equations (3.12) and (3.13).

The final objective function is expressed as:

$$\mathcal{L} = \mathcal{L}_{\text{classification}} + \lambda L_{\text{energy}}, \tag{3.16}$$

where:

- λ: A weighting factor that balances the importance of the classification loss and the energy-based regularization loss.

Challenge in Implementation

The original energy-based loss $\mathcal{L}_{\text{Energy}}$ in Liu et al. (2020) is scaled by a single hyperparameter λ, allowing adjustment of the overall regularization strength. However, when combining the energy loss with objectives beyond cross-entropy, or when assigning different weights to the ID and OOD components (i.e., $\mathcal{L}_{\text{in}}$ and $\mathcal{L}_{\text{out}}$), the original formulation is restrictive. To address this, we implement the loss as two separate terms as mentioned previously, allowing different values of λ for $\mathcal{L}_{\text{in}}$ and $\mathcal{L}_{\text{out}}$ which enables flexible integration with additional objectives and independent weighting of ID and OOD penalties.

3.5 Plot

3.5.1 ROC Curve Visualization and Evaluation

A core aspect of open-set recognition (OSR) evaluation is the ability to quantitatively and visually compare the performance of different models in distinguishing between known and unknown (or negative) samples. The Receiver Operating Characteristic (ROC) curve and its Area Under the Curve (AUC) are standard metrics for this purpose, providing an interpretable and robust summary of a model's discrimination capability Fawcett (2006).

Motivation

The ROC curve visualizes the relationship between the True Positive Rate (TPR) and the False Positive Rate (FPR) across different threshold values. In Open Set Recognition (OSR), this curve helps evaluate how effectively a model can distinguish in-distribution (known) samples from out-of-distribution (unknown or negative) samples, independent of a specific threshold. The AUC (Area Under the Curve) serves as a single scalar metric summarizing this performance.

Mathematically, for a given threshold τ, the TPR and FPR are defined as:

$$\mathrm{TPR}(\tau) = \frac{\mathrm{TP}(\tau)}{\mathrm{TP}(\tau) + \mathrm{FN}(\tau)}, \quad \mathrm{FPR}(\tau) = \frac{\mathrm{FP}(\tau)}{\mathrm{FP}(\tau) + \mathrm{TN}(\tau)} \tag{3.17}$$

where TP, FP, TN, and FN represent the numbers of true positives (correctly identified known samples), false positives (incorrectly identified unknown samples as known), true negatives (correctly identified unknown samples), and false negatives (incorrectly identified known samples as unknown), respectively, at a given threshold τ.

The ROC curve is constructed by varying τ across the range of possible detection scores, and the Area Under the Curve (AUC), specifically Area Under the ROC Curve(AUC-ROC), AUROC for short, is calculated as:

$$\mathrm{AUROC} = \int_0^1 \mathrm{TPR}(\mathrm{FPR}^{-1}(x))\, dx \tag{3.18}$$

A higher AUROC value indicates stronger overall performance in distinguishing between in-distribution and out-of-distribution samples.

Implementation Details

The ROC evaluation and visualization functionality is implemented in the consisting of two main components:

- **ROC Data Computation (`get_roc_data`):** This function processes the output scores and ground-truth labels for each model, filters the relevant known and unknown samples, and computes the FPR, TPR, and AUC using `sklearn.metrics.roc_curve` and `auc`.
- **ROC Curve Plotting (`plot_roc_curves`):** Given the computed ROC data, this function plots the ROC curves for all models on a single axis, labeling each curve with its corresponding AUC for easy comparison. The function returns a standardized `PlotInfo` object from original package framework containing plot handles, labels, and axis descriptions for further customization or integration into grid plots.

3.6 Docstring Update

Comprehensive and up-to-date documentation is essential for the usability, maintainability, and extensibility of any scientific software package, especially in collaborative and research-driven environments. During this project, some efforts were dedicated to improving the code documentation and integrating it with the Sphinx documentation framework.

3.6.1 Motivation

Initially, documentation updates were assigned as experimental work for familiarizing workflow, during which small part of the documentation was fund outdated. This allowed us to contribute

in providing clear, concise, and accessible documentation for both users and developers. These documentation updates have significantly improved our general understanding of the code in the beginning.

3.6.2 Implementation Details

- Corrected outdated instructions in `ReadMe.md` to provide accurate setup and usage guidelines.
- Integrated Sphinx for documentation, ensuring that the project documentation remains up-to-date and easily accessible online.

Chapter 4

Experiments

4.1 Analysis of Fusion Method in Postprocessors

In open-set recognition (OSR), the ability to distinguish unknown classes from known classes is crucial.Fusion methods, loss functions, and OOD post-processing techniques can each impact a model's robustness in these scenarios. In this experiment, we systematically investigate how different fusion strategies and post-processing methods affect model performance across several datasets, with particular attention to the presence of negative samples.

4.1.1 Experimental Setup

The experiments are conducted using three datasets: CIFAR+U with negative samples, CIFAR+U without negative samples, and the MNIST dataset. For the MNIST dataset, the backbone network is LeNet, while for both CIFAR+U datasets, ResNet-18 is used as the backbone. Each model is trained for 50 epochs with a learning rate of 0.01.

For each dataset, models are trained with two types of loss functions: EOS loss and cross entropy (CE) loss. During test phase, we focus on two OOD post-processing methods: Energy Score and GradNorm—are applied, both inheriting from `OODPostProcessor`. Each post-processing method is further tested with two different fusion strategies: multiplying probabilities and multiplying logits. Since the fusion strategies fundamentally integrate the OOD score with either the logits or the softmax probabilities via multiplication, we seek to evaluate the effectiveness of these fusion approaches. To provide a fair comparison, we use the base postprocessor techniques without fusion, named MSP and Maxlogits.

Metrics and Evaluation

For open-set performance metrics, we choose AUROC and AUOSCR as metrics. AUROC, also known as AUC-ROC, representing the area under the aforementioned ROC curve, measures the model's ability to distinguish between positive and negative classes, representing the trade-off between true positive rate and false positive rate across different thresholds. AUOSCR (Area Under Open Set Classification Rate) by Dhamija et al. (2018) calculates that area under OSCR curve, indicating the model's performance in open set recognition tasks, focusing on its ability to correctly classify known samples while rejecting unknown samples across varying thresholds.

Definition of **Open Set Classification Rate (OSCR)** curve is that given a classifier $f(x)$, let $f(x)_c$ denote the confidence score for class c and let $r(x)$ represent the model's ability to reject unknowns. Typically, $f(x)_c$ is proportional to $\mathbb{P}[y = c|x]$, and $r(x)$ is proportional to $\frac{1}{\max_{c \in \mathcal{V}_k} f(x)_c}$,

Table 4.1: Open-set Recognition Performance (AUROC and AUOSCR) under Different Dataset, Fusion and Post-processing Methods

Dataset	Model		Open-set Performance	
	Postprocessor	**Loss**	**AUROC**	**AUOSCR**
MNIST				
	EnergyScore-logits	CE	0.8711	0.8447
	EnergyScore-probabilities	CE	0.8710	0.8477
	GradNorm-logits	CE	0.8711	0.8447
	GradNorm-probabilities	CE	0.8711	0.8447
	MSP	CE	0.8711	0.8447
	MaxLogits	CE	0.8328	0.8304
	EnergyScore-logits	EOS	0.9328	0.9260
	EnergyScore-probabilities	EOS	0.9321	0.9254
	GradNorm-logits	EOS	0.9330	0.9263
	GradNorm-probabilities	EOS	0.9330	0.9263
	MSP	EOS	0.9330	0.9263
	MaxLogits	EOS	0.9200	0.9128
Cifar+u with negative				
	EnergyScore-logits	CE	0.8291	0.7866
	EnergyScore-probabilities	CE	0.8319	0.7890
	GradNorm-logits	CE	0.8278	0.7857
	GradNorm-probabilities	CE	0.8278	0.7857
	MSP	CE	0.8278	0.7857
	MaxLogits	CE	0.8434	0.7952
	EnergyScore-logits	EOS	0.8815	0.8150
	EnergyScore-probabilities	EOS	0.8832	0.8162
	GradNorm-logits	EOS	0.8796	0.8139
	GradNorm-probabilities	EOS	0.8796	0.8138
	MSP	EOS	0.8796	0.8139
	MaxLogits	EOS	0.8924	0.8209
Cifar+u without negative				
	EnergyScore-logits	CE	0.8691	0.8030
	EnergyScore-probabilities	CE	0.8697	0.8020
	GradNorm-logits	CE	0.8678	0.8025
	GradNorm-probabilities	CE	0.8678	0.8025
	MSP	CE	0.8678	0.8025
	MaxLogits	CE	0.8716	0.7942

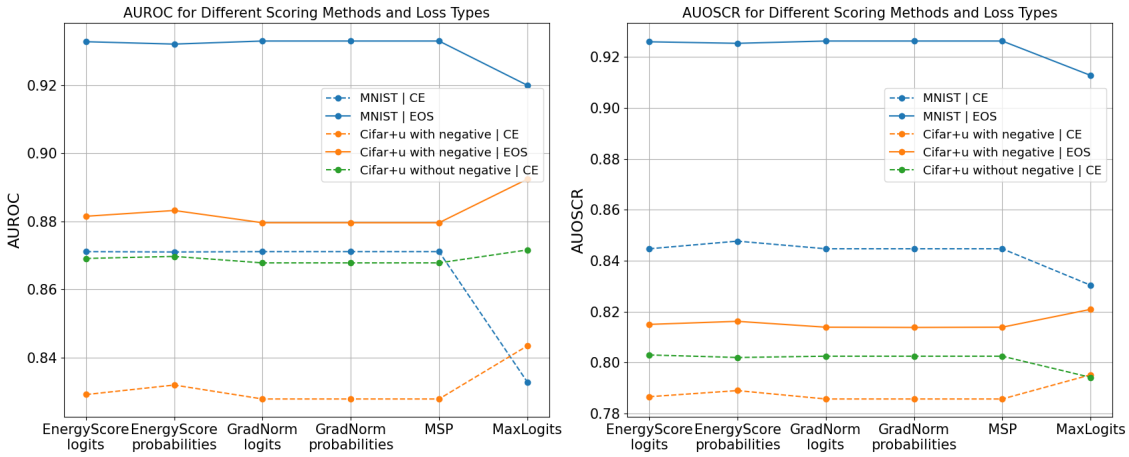


Figure 4.1: Open-set performance of post processors with different fusion methods. The left panel shows AUROC scores and the right panel shows AUOSCR scores for various OOD scoring methods across three datasets (MNIST, Cifar+u with negative, and Cifar+u without negative) and two loss functions (CE and EOS). Each line represents a combination of dataset and loss type, with colors distinguishing datasets and line styles indicating loss functions.

where $\mathcal{V}_k$ is the set of known classes. The OSCR curve plots the Correct Classification Rate (CCR) for known samples against the False Positive Rate (FPR) for unknown samples as the threshold t varies. The CCR at threshold t is defined as:

$$\mathrm{CCR}(t) = \frac{1}{N_k} \sum_{i=1}^{N_k} \mathbb{I}\left[y_i = \arg\max_{c \in \mathcal{V}_k} f(x_i)_c \right] \cdot \mathbb{I}\left[f(x_i)_{y_i} > t \right] \tag{4.1}$$

where N_k is the number of known (close-set) samples, y_i is the ground-truth label for sample x_i, and $\mathbb{I}[\cdot]$ is the indicator function.

4.1.2 Results

Table 4.1 describes the open-set performance of different fusion methods in post processors. From which, we can draw more intuitive figures, Fig. 4.1. In the figure, across all datasetsa and models, fusion-based scoring methods (EnergyScore and GradNorm with logits or probabilities) perform similarly to standard post-processors (MSP and MaxLogits), with no significant improvement observed. This suggests that in this experiments, simple fusion strategies do not substantially enhance open-set recognition performance. Additionally, We observed that EOS loss generally yields higher AUROC and AUOSCR compared to CE loss, indicating a better OSR functionality.

4.2 Analysis of the Correlation between Closed-set Accuracy and Open-set Performance

To systematically study the relationship between closed-set classification accuracy and open-set recognition performance, we design a unified experiment along three complementary axes:

1. **Cross-dataset & loss**: We evaluate multiple open-set scoring strategies and loss functions on four widely used datasets—EMNIST, CIFAR10, SVHN and ImageNet. We record the best closed-set accuracy and relate them to open-set metrics.

2. **Cross-backbone**: Aside from loss and post processors, backbone also plays a vital part in a model. In this experiment, different backbones were used to discover the classifiers' performance in close- and open-sets.

3. **Cross-training method**: To explore closed-set vs open-set performance relation under different training behaviors, we conduct experiments with varying training duration and optimizers. This experiment highlights the trade-offs between hyper-parameters and performance on both closed-set and open-set tasks.

4.2.1 Cross-dataset & loss

Experimental setup

In the cross-dataset experiment, we evaluate the aforementioned scoring methods on four datasets: Mnist, CIFAR10, SVHN, and ImageNet; alongside three loss functions: CE (cross entropy loss), EOS (entropic open-set loss) (Dhamija et al., 2018) and energy-score based loss (Liu et al., 2020) (with T $= 1$, $\lambda = 0.1$, $m_{\text{in}} = -25$, $m_{\text{out}} = -5$).

The scoring methods (post processors) are defined as follows:

- **Energy_Score**: This method uses the energy-based OOD score to distinguish samples' distribution(known, in-distribution, ID or unknown, out-of-distribution, OOD).
- **GradNorm**: Uses the GradNorm OOD score for distinguishing samples' distribution.
- **MSP**: Uses the maximum softmax probability as the score.
- **MaxLogits**: Uses the maximum logit value as the score.

For the datasets, we used the aforementioned Split Protocol to split MNIST, CIFAR-10, SVHN, and ImageNet. Specifically, we randomly chose 4 known classes, 2 negative classes, and 4 unknown classes for all datasets except ImageNet, since they all contain 10 classes. As for the ImageNet dataset, we adopted the P1 protocol from Palechor et al. (2022), in which the known and unknown classes are semantically distant and do not share too many visual features.

All models are trained using identical settings, including data splits, and hyperparameters, to ensure a fair comparison. Experiments use configuration of 50 epochs and learning rate = 0.01 with stochastic gradient descent. Experiments on MNIST use LeNet as backbone since MNIST is a small-scale dataset, the rest of the experiments use ResNet-18 as bakcbone. Evaluation is performed on both closed-set and open-set metrics, close-set performance is evaluated by model's classification accuracy of known samples on known classes; open-set performance is determined by AUROC and AUOSCR.

Results

Table 4.2: Closed-set and Open-set Performance Metrics (AUROC and AUOSCR) across Different Datasets (MNIST, CIFAR10, SVHN, and ImageNet) and Models.

Dataset	Model		Open-set Performance	
	Postprocessor	Loss	AUROC	AUOSCR
CIFAR (Closed-set Accuracy = 0.9268)				
	energy_score	CE	0.7877	0.7552
	gradnorm	CE	0.7454	0.7186
	maxscore (logits)	CE	0.7875	0.7565
	maxscore (probabilities)	CE	0.7686	0.7443
CIFAR (Closed-set Accuracy = 0.8915)				
	energy_score	Energy	0.7578	0.6924
	gradnorm	Energy	0.7627	0.7035
	maxscore (logits)	Energy	0.7599	0.6948
	maxscore (probabilities)	Energy	0.7721	0.7202
CIFAR (Closed-set Accuracy = 0.9317)				
	energy_score	EOS	0.8444	0.8035
	gradnorm	EOS	0.8311	0.7899
	maxscore (logits)	EOS	0.8469	0.8060
	maxscore (probabilities)	EOS	0.8494	0.8097
MNIST (Closed-set Accuracy = 0.9985)				
	energy_score	CE	0.9695	0.9689
	gradnorm	CE	0.9659	0.9654
	maxscore (logits)	CE	0.9695	0.9689
	maxscore (probabilities)	CE	0.9718	0.9713
MNIST (Closed-set Accuracy = 0.9951)				
	energy_score	Energy	0.9722	0.9682
	gradnorm	Energy	0.9586	0.9549
	maxscore (logits)	Energy	0.9723	0.9683
	maxscore (probabilities)	Energy	0.9501	0.9462
MNIST (Closed-set Accuracy = 0.9980)				
	energy_score	EOS	0.9959	0.9945
	gradnorm	EOS	0.9949	0.9932
	maxscore (logits)	EOS	0.9958	0.9945
	maxscore (probabilities)	EOS	0.9947	0.9936
SVHN (Closed-set Accuracy = 0.9606)				
	energy_score	CE	0.8655	0.8487
	gradnorm	CE	0.8574	0.8412
	maxscore (logits)	CE	0.8654	0.8487
	maxscore (probabilities)	CE	0.8668	0.8511
SVHN (Closed-set Accuracy = 0.9481)				
	energy_score	Energy	0.8211	0.7945
	gradnorm	Energy	0.7803	0.7572
	maxscore (logits)	Energy	0.8252	0.7988
	maxscore (probabilities)	Energy	0.8667	0.8447

Table 4.2: (continued)

Dataset	Model		Open-set Performance	
	Postprocessor	Loss	AUROC	AUOSCR
SVHN (Closed-set Accuracy = 0.9601)				
	energy_score	EOS	0.8854	0.8668
	gradnorm	EOS	0.8590	0.8405
	maxscore (logits)	EOS	0.8860	0.8675
	maxscore (probabilities)	EOS	0.8908	0.8728
ImageNet (Closed-set Accuracy = 0.8497)				
	energy_score	CE	0.6841	0.6337
	gradnorm	CE	0.6700	0.6359
	maxscore (logits)	CE	0.6713	0.6413
	maxscore (probabilities)	CE	0.6968	0.6460
ImageNet (Closed-set Accuracy = 0.8232)				
	energy_score	Energy	0.6121	0.6379
	gradnorm	Energy	0.6234	0.6138
	maxscore (logits)	Energy	0.6272	0.6288
	maxscore (probabilities)	Energy	0.6367	0.6503
ImageNet (Closed-set Accuracy = 0.8650)				
	energy_score	EOS	0.7045	0.6488
	gradnorm	EOS	0.6825	0.6647
	maxscore (logits)	EOS	0.7057	0.6652
	maxscore (probabilities)	EOS	0.7204	0.6543

Experimental results can be seen in Table 4.2. We average the open-set performance for different post processors under each dataset and loss then draw connection lines with X-axis **Close Set Accuracy** and Y-axis **Open Set Performance** to show the trend in Figure 4.2, which intuitively shows a positive association: open-set accuracy tends to increase as closed-set accuracy improves, with few outliers.

To quantify the correlation behind the trend, we calculated both the Pearson correlation coefficient (r) and its associated p-value between the performances. The Pearson correlation coefficient measures the linear relationship between two variables, ranging from -1 to $+1$. A value of $+1$ indicates a perfect positive linear relationship, -1 indicates a perfect negative linear relationship, and 0 implies no linear correlation. The p-value assesses the statistical significance of the observed correlation, with a smaller p-value indicating stronger evidence that the correlation is not due to random chance.

The Pearson correlation coefficient is calculated as:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4.2}$$

where x_i and y_i are the individual data points, and $\bar{x}$ and $\bar{y}$ are the means of x and y, respectively.

The statistical significance (p-value) of the correlation is typically computed using the following t-statistic:

$$t = r\sqrt{\frac{n-2}{1-r^2}} \tag{4.3}$$

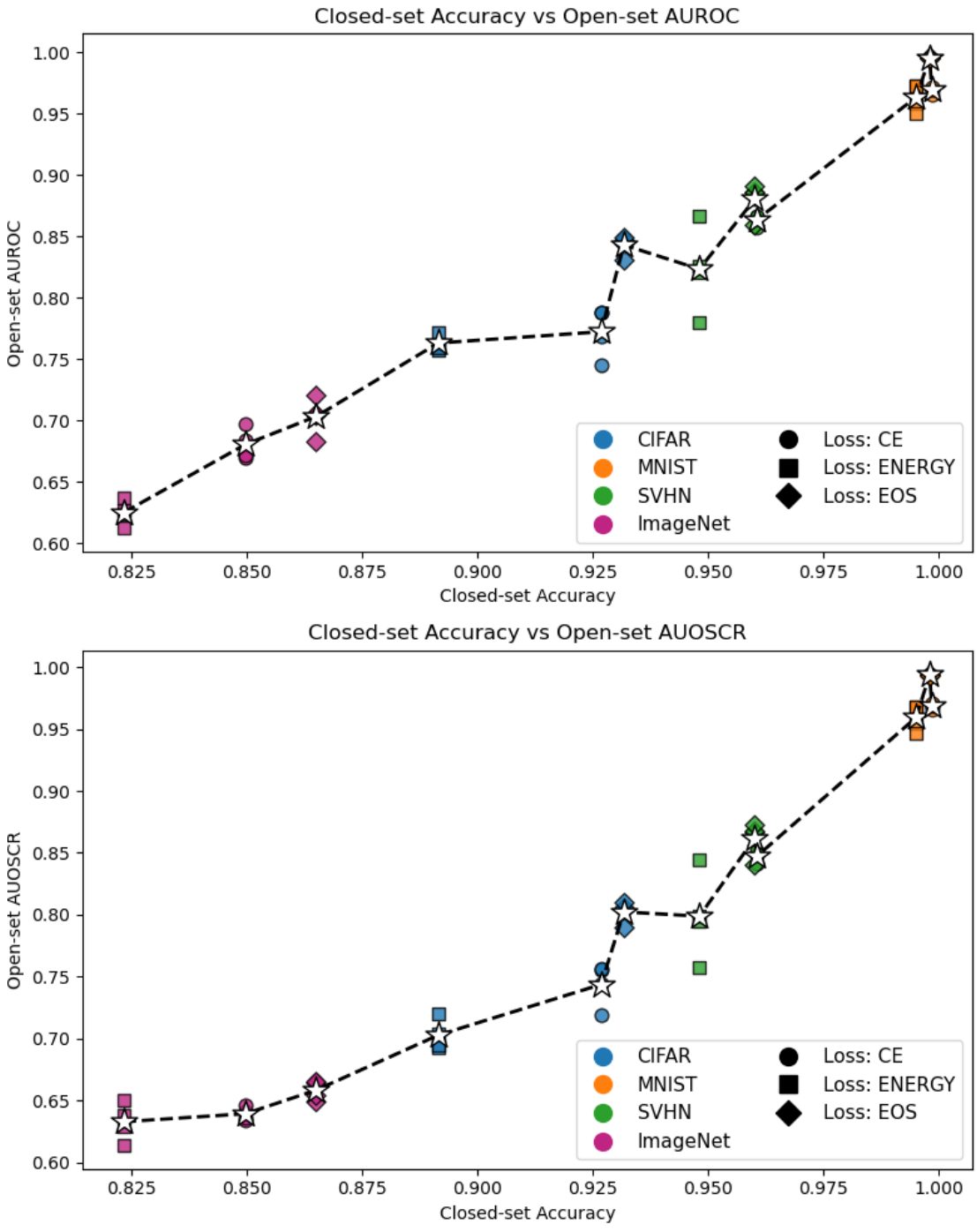


Figure 4.2: Comparison of closed-set accuracy and open-set performance metrics (AUOSCR and AUROC) across different datasets and loss with 4 different post processors. The upper plot illustrates the relationship between closed-set accuracy and open-set AUROC, while the lower plot shows the relation between closed-set accuracy and open-set AUOSCR. Each dataset is represented by colored scatter points, with the average performance marked by larger black-bordered stars. The dashed black line in both plots connects the mean points of different post processors, highlighting the general trend.

where n is the number of paired observations. The p-value is then obtained from the t-distribution with $n-2$ degrees of freedom:

$$p = 2 \cdot P\left(T \geq |t|\right) \tag{4.4}$$

where $P\left(T \geq |t|\right)$ denotes the probability that the absolute value of a t-distributed random variable with $n-2$ degrees of freedom is greater than or equal to $|t|$.

Overall, Pearson's r quantifies both the direction and strength of the linear association between the two variables, while the p-value indicates the probability that the observed correlation occurred by chance.

Table 4.3: Person r and p-value of Open-set Performance and Close-set Accuracy

Open-set Metric	Pearson r	p-value
AUROC	0.9737	3.53×10^{-31}
AUOSCR	0.9632	6.75×10^{-28}

These statistics in Table 4.3 reveal a strong positive association between closed-set accuracy and open-set performance, particularly with calculated Peaerson's r for both open-set metrics greater than 0.95 and r-values far smaller than 0.005. Indicating a strong positive correlation between performances which is highly statistically significant and unlikely to have occurred by chance.

We also observed a notable trend: both open-set and closed-set performance consistently follow the order EOS > CE > Energy loss, indicating that models trained with EOS achieve superior results compared to those using CE or Energy loss.

4.2.2 Cross-backbone

We further analyze the role of backbone architectures, LeNet, ResNet 18, 34 and 50, while comparing the close-set performance and open-set performance.

Experimental setup

Four backbones are directly configured from the original framework. The open-set experiment uses the combinations of MSP and MaxLogits as post-processors with CE and EOS as loss functions. Experimental data is obtained from CIFAR-10 dataset by using Split Protocol, setting 4 known classes, 2 negative classes and 4 unknown classes. Close-set uses the same backbones and losses. All trainings adopt 50 epochs, learning rate = 0.01 with stochastic gradient descent. Evaluation is performed on both closed-set and open-set metrics.

Results

From Table 4.4, line figure 4.3 can be easily drawn to show a clear positive correlation between close-set performance and open-set performance. In the Figure, both subplots show EOS loss outperforms CE loss in close- and open-set tasks when using the same backbone, except for ResNet50.

The Pearson correlation coefficients in Table 4.5 indicate a strong positive relationship between closed-set and open-set peformance (AUROC and AUOSCR) as Peasron r is greater than 0.85. And p-values being exponentially small confirms the statistical significance of the observed correlations. This experiment once-more suggests that close-set performance is tightly positively linked to open-set performance, backbone-wise.

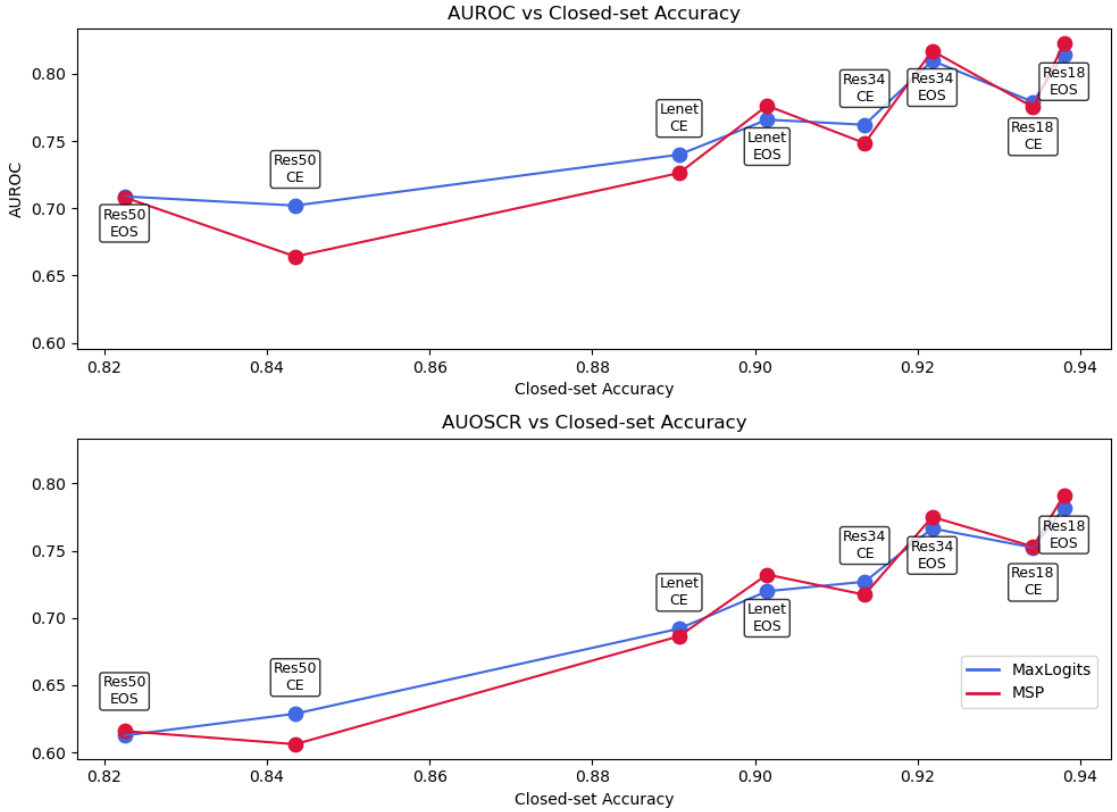


Figure 4.3: The figure shows the relationship between Closed-set Accuracy and two Open-set performance metrics: AUROC (upper) and AUOSCR (lower), for different backbones (Lenet, ResNet18, ResNet34, ResNet50) using different post processors and loss terms. The annotations above the points highlight the backbone names and loss terms.

Table 4.4: Performance Metrics for Models with Different Backbones and Losses

Backbone	Loss	Closed-set Acc	Score Type	AUOSCR	AUROC
LENET	CE	0.8907	MaxLogits	0.6920	0.7399
			MSP	0.6864	0.7262
LENET	EOS	0.9015	MaxLogits	0.7198	0.7659
			MSP	0.7322	0.7763
RESNET18	CE	0.9342	MaxLogits	0.7523	0.7791
			MSP	0.7530	0.7752
RESNET18	EOS	0.9380	MaxLogits	0.7818	0.8142
			MSP	0.7914	0.8226
RES34	CE	0.9135	MaxLogits	0.7269	0.7621
			MSP	0.7172	0.7484
RES34	EOS	0.9218	MaxLogits	0.7665	0.8098
			MSP	0.7751	0.8171
RES50	CE	0.8435	MaxLogits	0.6287	0.7020
			MSP	0.6060	0.6641
RES50	EOS	0.8225	MaxLogits	0.6125	0.7088
			MSP	0.6158	0.7081

Table 4.5: Pearson r and p-value of Open-set Performance and Close-set Accuracy

Open-set Metric	Pearson r	p-value
AUROC	0.8693	1.23×10^{-5}
AUOSCR	0.9671	1.03×10^{-9}

4.2.3 Cross-training method

Not only different models can result in different performances during OSR tasks. Training techniques and cost can also result in such variation. We utilize this simple point and set up an experiment by varying the number of epochs and optimizers during training to further explore the close-set and open-set performance correlation.

Experimental setup

We chose MSP, MaxLogits and Energy-based OOD Score as postprocessors; CE loss and Energy-score based loss as loss terms; ResNet18 as backbone; CIFAR-10 from Split Protocol as dataset, 4 known classes, 4 unknown classes and 2 negative classes; train each method with 25 and 50 epochs; train each method using learning rate 0.01 with SGD and 0.001 withAdam optimizer. Evaluation metrics are applied on both close-set and open-set.

Results

By analyzing the data from Table 4.6, which demonstrates close- and open-set performance for different training setups, we can generate Figure 4.4. The figure demonstrates that changes in

Table 4.6: Comparison of open-set AUROC and AUOSCR for different postprocessors, losses, optimizers, and training epochs on CIFAR10.

Dataset&Train	**Postprocessor**	**Open-set Performance**	
		AUROC	**AUOSCR**
CE Loss - 25 Epochs - Adam Optimizer (Closed-set Accuracy = 0.9325)			
	Energy_Score	0.7929	0.7646
	MaxLogits	0.7903	0.7637
	MSP	0.7702	0.7481
CE Loss - 25 Epochs - SGD Optimizer (Closed-set Accuracy = 0.9030)			
	Energy_Score	0.7369	0.6942
	MaxLogits	0.7379	0.6968
	MSP	0.7237	0.6910
CE Loss - 50 Epochs - Adam Optimizer (Closed-set Accuracy = 0.9362)			
	Energy_Score	0.7971	0.7694
	MaxLogits	0.7961	0.7698
	MSP	0.7847	0.7627
CE Loss - 50 Epochs - SGD Optimizer (Closed-set Accuracy = 0.9273)			
	Energy_Score	0.8168	0.7799
	MaxLogits	0.8163	0.7803
	MSP	0.7964	0.7671
Energy Loss - 25 Epochs - Adam Optimizer (Closed-set Accuracy = 0.8643)			
	Energy_Score	0.7702	0.6857
	MaxLogits	0.7723	0.6884
	MSP	0.7605	0.6929
Energy Loss - 25 Epochs - SGD Optimizer (Closed-set Accuracy = 0.8317)			
	Energy_Score	0.7424	0.6369
	MaxLogits	0.7440	0.6391
	MSP	0.7306	0.6436
Energy Loss - 50 Epochs - Adam Optimizer (Closed-set Accuracy = 0.8878)			
	Energy_Score	0.7823	0.7095
	MaxLogits	0.7859	0.7136
	MSP	0.7857	0.7320
Energy Loss - 50 Epochs - SGD Optimizer (Closed-set Accuracy = 0.8692)			
	Energy_Score	0.7599	0.6788
	MaxLogits	0.7606	0.6801
	MSP	0.7340	0.6704

training configurations (such as training duration or optimizer) lead to variations in both close set accuracy and open-set accuracy, which, in turn, also exhibit a general positively correlated trend between these two performances.

To observe the trend in a statistical view, we calculated the Pearson r and p-values in Table. 4.7. Though AUROC shows a weaker correlation with r=0.65, intuitively caused by outlier model using CE loss, 25 epochs with SGD, the overall trend tends to be positive with statistical confidence.

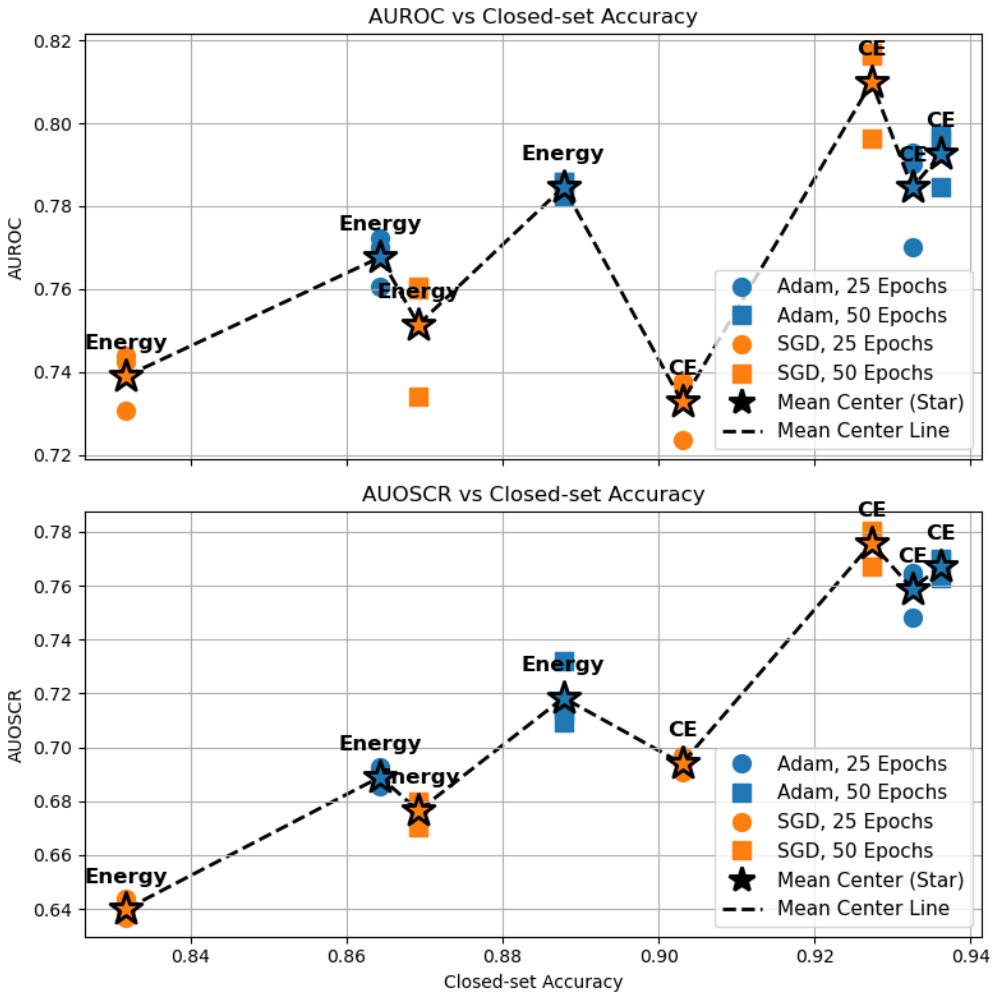


Figure 4.4: Performance Comparison of AUROC and AUOSCR vs Close Set Accuracy. The plots show the relation between close set accuracy and open-set performance metrics, AUROC (upper) and AUOSCR (lower), for different losses, optimizers (color) and training epochs (shape). The dashed line shows the mean performance across post processors.

Table 4.7: Pearson correlation between Closed-set Accuracy and Open-set metrics (AUROC, AUOSCR) on CIFAR-10 for Different Training Methods.

Metric	Pearson r	p-value
AUROC	0.6499	5.88×10^{-4}
AUOSCR	0.9396	1.02×10^{-11}

4.3 Open-set Performance for Near- and Far-OOD on CIFAR Datasets

In open-set recognition (OSR), the unknown classes can be significantly different, ranging from samples that are closely related to the known classes (Near-OOD) to samples that are entirely dissimilar (Far-OOD). A key question is whether a model can perform effectively across this diverse range of unknown classes. Specifically, we aim to investigate whether the strong performance of a model on certain unknown classes implies similarly robust performance on other unknown classes, regarding their proximity or dissimilarity to the known classes.

4.3.1 Experimental Setup

The primary objective of this experiment is to investigate how models perform in rejecting or identifying unknown classes when facing Near-OOD and Far-OOD samples. This analysis is crucial for developing robust models that can operate reliably in real world where unknown categories may vary. We utilize the CIFAR-10 and CIFAR-100 datasets for all experiments. To create meaningful open-set scenarios, we design three distinct data groups based on semantic similarity, which is done by rewriting certain parts of the package, specifying the class indexes:

Data Preparation. For each group, the training set exclusively consists of the selected known classes from CIFAR-10. For evaluation, the test set contains both ID samples and OOD samples: Near-OOD and Far-OOD images from the specified CIFAR-100 categories listed in Table 4.8. All images are preprocessed identically (e.g., normalization) to ensure consistency.

All experiments are trained for 50 epochs using a learning rate of 0.01, with SGD. Post processors include energy score, GradNorm, and MSP. Loss is CE due to the lack of negative classes and backbone is ResNet18.

Table 4.8: Class Groups for Vehicles and Animals. The table lists the known, near unknown, and far unknown classes for each group.

Group	Category	Classes
Group 1	**Known Classes (CIFAR-10)**	airplane (0), automobile (1), ship (8), truck (9)
	Near Unknown (CIFAR-100)	**vehicles 1:** bicycle (8), bus (13), motorcycle (48), pickup_truck (58), train (90) **vehicles 2:** lawn_mower (41), rocket (69), streetcar (81), tank (85), tractor (89)
	Far Unknown (CIFAR-100)	**flowers:** orchid (54), poppy (62), rose (70), sunflower (82), tulip (92) **fruit and vegetables:** apple (0), mushroom (51), orange (53), pear (57), sweet_pepper (83)
Group 2	**Known Classes (CIFAR10)**	bird (2), cat (3), deer (4), dog (5), frog (6), horse (7)
	Near Unknown (CIFAR-100)	**small mammals:** hamster (36), mouse (50), rabbit (65), shrew (74), squirrel (80) **large carnivores:** bear (3), leopard (42), lion (43), tiger (88), wolf (97)
	Far Unknown (CIFAR-100)	**household furniture:** bed (5), chair (20), couch (25), table (84), wardrobe (94) **kitchen utensils:** bottle (9), bowl (10), can (16), cup (28), plate (61)
Group 3	**Known Classes (CIFAR-10)**	bird (2), cat (3), deer (4), dog (5), frog (6), horse (7)
	Near Unknown (CIFAR-100)	**large omnivores/herbivores:** camel (15), cattle (19), chimpanzee (21), elephant (31), kangaroo (38) **mid-size mammals:** fox (34), porcupine (63), possum (64), raccoon (66), skunk (75)
	Far Unknown (CIFAR-100)	**large natural outdoor scenes:** cloud (23), forest (33), mountain (49), plain (60), sea (71) **large man-made outdoor things:** bridge (12), castle (17), house (37), road (68), skyscraper (76)

Table 4.9: Closed-set and open-set performance metrics (AUROC and AUOSCR) across different groups and post processors.

Method	**AUROC**			**AUOSCR**		
	Near	Far	Diff	Near	Far	Diff
Data Grouping 1 (Closed-set Acc = 0.9070)						
Energy	0.6601	0.8507	0.1906	0.5958	0.7657	0.1699
GradNorm	0.6340	0.8433	0.2093	0.5461	0.7184	0.1723
MaxLogits	0.6584	0.8473	0.1889	0.5952	0.7637	0.1685
MSP	0.6588	0.8158	0.1570	0.5997	0.7412	0.1415
Data Grouping 2 (Closed-set Acc = 0.8003)						
Energy	0.7521	0.7484	-0.0037	0.6421	0.6149	-0.0272
GradNorm	0.7648	0.7476	-0.0172	0.6026	0.5971	-0.0055
MaxLogits	0.7474	0.7315	-0.0159	0.6409	0.6075	-0.0334
MSP	0.7134	0.6659	-0.0475	0.6158	0.5712	-0.0446
Data Grouping 3 (Closed-set Acc = 0.7997)						
Energy	0.8790	0.8602	-0.0188	0.7300	0.7093	-0.0207
GradNorm	0.8649	0.8091	-0.0558	0.6537	0.6288	-0.0249
MaxLogits	0.8753	0.8520	-0.0233	0.7285	0.7051	-0.0234
MSP	0.8324	0.7954	-0.0370	0.6984	0.6616	-0.0368

Table 4.10: Pearson correlation coefficients (r) and p-values (p) between Near-OOD and Far-OOD performance for AUROC and AUOSCR, both overall and per data grouping.

Grouping	AUROC		AUOSCR	
	r	p	r	p
Overall	0.0633	0.8450	-0.1206	0.7090
Group 1	-0.1467	0.8533	0.8249	0.1751
Group 2	0.9687	0.0313	0.6356	0.3644
Group 3	0.8777	0.1223	0.9838	0.0162

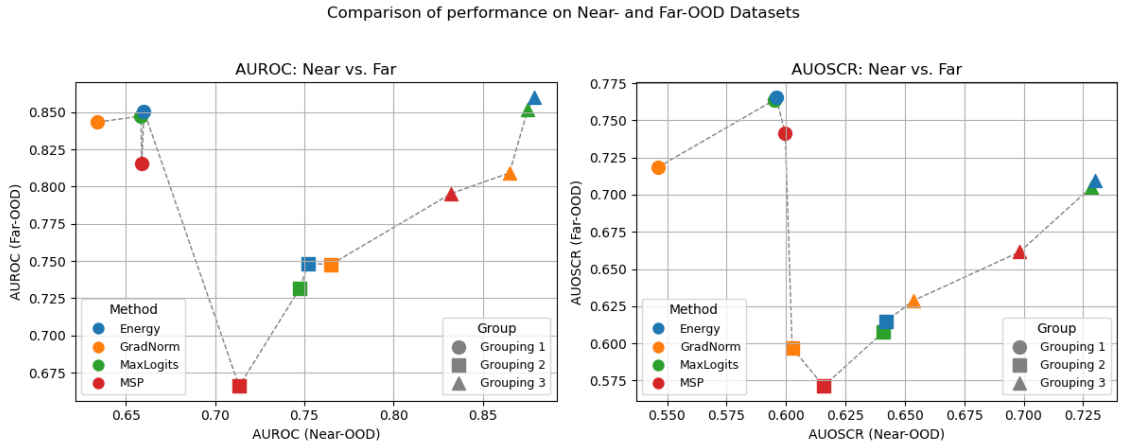


Figure 4.5: Comparison of AUROC and AUOSCR metrics for different post processors under Near-OOD and Far-OOD scenarios. The left subplot shows the AUROC performance, and the right subplot shows the AUOSCR performance. Different markers represent data groups, and different colors represent the scoring methods (post processors).

4.3.2 Results

Table 4.9 summarizes the open-set evaluation metrics for different post processors and data groups. From the table, we can conduct Figure 4.5 which visualizes the relation of open-set metrics (AUOSCR and AUROC) for all methods between Near- and Far-OOD scenarios. Each point represents a specific post processor and OOD type. In the figure, the overall trend is not clear with steep slopes. However, if we separate the data points by data groups, we may see a general upward trend between Far-OOD performance and Near-OOD performance.

Quantitative analysis was conducted using the Pearson correlation coefficient, as presented in Table 4.10. The overall correlation between Near-OOD and Far-OOD performance is not only weak but also statistically insignificant. However, a closer examination of individual groups reveals only limited positive correlation within the same data group due to certain exceptions (e.g., AUROC for Group 1 shows a negative correlation),and some correlations with large p-values (e.g., AUOSCR for Group 1 and Group 2), indicating low statistical confidence.

Overall, our experimental results provide limited support for a positive correlation between Near-OOD and Far-OOD performance within the same dataset. In other words, a model exhibiting strong Near-OOD recognition may only marginally reflect its performance in Far-OOD scenarios when the dataset remains unchanged. Furthermore, a method that performs well on one data group does not necessarily generalize to others; thus, relying solely on a single OOD scenario or metric may obscure the true strengths and weaknesses of open-set recognition approaches.

This observation appears a conflict with our findings in Section 4.2.1, where a positive correlation was identified between closed-set accuracy and open-set performance across datasets. Intuitively, robust open-set performance should imply strong closed-set accuracy, this correlation is consistent in different datasets, therefore, by extension, a good Near-OOD should imply better Far-OOD performance across data groups. However, our current experiment result does not support this hypothesis. Further studies with larger datasets and more comprehensive analyzes are needed to clarify this inconsistency

Chapter 5

Conclusion

In this report, our contributions mainly contain two aspects: code package extension and experiments.

On the code side, we enhanced the OpenOSR package by introducing new dataset protocols, postprocessors, loss functions, and evaluation tools, supplementing a modular and reproducible foundation for OSR research.For the dataset extensions, we first implemented the standardized dataset split protocol proposed in Neal et al. (2018) and also added additional configurable parameters to allow users to customize the combination of other splits. Furthermore, we also developed a general interface, `SplitProtocol`, to handle various splitting methods for different datasets. For postprocessor modules, we first implemented a general interface, `OODPostProcessorBase`, designed to fuse the single scalar OOD score obtained from different OOD methods with the logits or probabilities. This results in a logits or probability distribution enriched with OOD information, aiming to provide more informative outputs. Alongside the OOD post processor base, we implemented 2 scoring methods under its framework, Energy score and GradNorm. GradNorm GradNorm is a scoring method that utilizes the norm of gradients to distinguish samples. By leveraging the gradient information of the model with respect to input data, GradNorm effectively identifies out-of-distribution (OOD) samples. Energy score not only can be used to distinguish ID and OOD samples in post processors, but also can be utilized to form a regularization loss term - Energy-score based Loss, which we implemented it under Loss section.

On the experiments side, our extensive experiments reveal several key insights: Fusion-based scoring methods, such as EnergyScore and GradNorm fused with logits or probabilities, do not offer significant improvements over standard post-processing approaches. The results show that these fusion methods achieve performance similar to baseline approaches across all datasets and models, indicating that simple fusion strategies may not substantially enhance open-set recognition effectiveness. Through comparing closet-set performance and open-set performance across 3 dimensions: datasets, backbones and training behaviours, by varying these dimensions and retrieving results from different experimental setups, we are convinced that closed-set accuracy is a useful indicator of open-set recognition performance. A good close-set classifier usually performs well on open-set, confirming the discover from Vaze et al. (2021). Near-OOD and Far-OOD scenarios present distinct challenges, and performance in one can only partially predict performance in the other. This situation caused by the data-set change highlights the importance of evaluating and reporting results separately for different data scenarios. The analysis for Near- and Far-ood in not absolute and requires further investigation.

Future work should focus on developing customized strategies to improve OOD detection, exploring additional datasets and model architectures, and further refining evaluation protocols. Furthermore, experiments discovered inconsistencies in Near- and Far-ood scenarios, indicating the need for more experiments to get a more convincing result. And suboptimal performance using fusion methods may require further investigation in the theory level. By addressing these directions,

the research community can move towards more robust and reliable open-set recognition systems suitable for deployment in real-world environments.

Appendix A

Supplementary Note

Failure in OSR Evaluation on Split TinyImageNet

In our Open Set Recognition (OSR) evaluation on Split TinyImageNet, we observed that several common scoring methods (MSP, MaxLogits, Energy-score, GradNorm; all using ResNet18, CE loss, 50 epochs, lr = 0.01 with SGD) yielded AUROC values slightly below 0.5. This indicates that the model's ability to distinguish unknowns from knowns is close to random, or even exhibits "reverse separation". This supplementary note aims to mark the potential bug in TinyImagenet for Split Protocol for future fix.

Metric	AUROC	AUOSCR
MSP	0.4620	0.4485
MaxLogits	0.4732	0.4988
Energy-score	0.4572	0.4818
GradNorm	0.4768	0.4589

Table A.1: Open-set performance on TinyImagenet

Bibliography

Bendale, A. and Boult, T. E. (2016). Towards open set deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1563–1572.

Dhamija, A. R., Günther, M., and Boult, T. E. (2018). Reducing network agnostophobia. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9157–9168.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874.

Hendrycks, D. and Gimpel, K. (2017). A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations (ICLR)*.

Huang, R., Geng, A., and Li, Y. (2021). On the importance of gradients for detecting distributional shifts in the wild. In *Neural Information Processing Systems*.

Kim, B. C., Kim, B., and Hyun, Y. (2024). Investigation of out-of-distribution detection across various models and training methodologies. *Neural Networks*, 175:106288.

Liang, S., Li, Y., and Srikant, R. (2018). Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR (Poster)*.

Liu, W., Wang, X., Owens, J., and Finn, C. (2020). Energy-based out-of-distribution detection. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Neal, L., Olson, M., Fern, X., Wong, W.-K., and Li, F. (2018). Open set learning with counterfactual images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 613–628.

Palechor, A., Bhoumik, A., and Günther, M. (2022). Large-scale open-set classification protocols for imagenet.

Perera, P. and Patel, V. M. (2019). Learning deep features for one-class classification. *IEEE Transactions on Image Processing*, 28(11):5450–5463.

Phillips, P. J., Grother, P., and Micheals, R. (2005). *Evaluation Methods in Face Recognition*, pages 329–348. Springer New York, New York, NY.

Scheirer, W. J., Jain, L. P., and Boult, T. E. (2013). Probability models for open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2317–2324.

Shu, L., Xu, H., and Liu, B. (2017). DOC: Deep open classification of text documents. In Palmer, M., Hwa, R., and Riedel, S., editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2911–2916, Copenhagen, Denmark. Association for Computational Linguistics.

Vaze, S., Han, K., Vedaldi, A., and Zisserman, A. (2021). Open-set recognition: A good closed-set classifier is all you need. *ArXiv*, abs/2110.06207.

Wang, R., Guo, J., Zhao, R.-W., Su, L., Ye, Y., Zhang, X., Zhang, Y., and Feng, R. (2023). Class-aware variational auto-encoder for open set recognition. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*, pages 264–269.

Wang, Z., Xu, Q., Yang, Z., He, Y., and Cao, X. (2022). Openauc: Towards auc-oriented open-set recognition.

Wu, Y., Ye, X., Dai, S., Pan, D., Li, X., Zhang, W., and Chen, Y. (2024). Revisiting energy-based model for out-of-distribution detection.

Yang, J., Wang, P., Zou, D., Zhou, Z., Ding, K., Peng, W., Wang, H., Chen, G., Li, B., Sun, Y., Du, X., Zhou, K., Zhang, W., Hendrycks, D., Li, Y., and Liu, Z. (2022). Openood: benchmarking generalized out-of-distribution detection. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Yoshihashi, R., Shao, W., Kawakami, R., You, S., Iida, M., and Naemura, T. (2019). Classification-reconstruction learning for open-set recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.