DEEL
DEpendable & Explainable Learning

Presentation of OODEEL

Paul Novello, DEEL, IRT Saint Exupery

Outline

1. **Introduction: Post-hoc OOD detection**
   - Model based vs Model agnostic
   - Post-hoc vs training
   - Some examples
2. Existing OOD detection libraries and positioning of OODEEL
3. OODEEL in practice

Post-hoc OOD detection

Two approaches to OOD detection:

- **Model agnostic**

  Goal: find data that does not belong to the same distribution as some input data
  Application: Anomaly detection, Outlier detection

- **Model based**

  Goal: find data that does not belong to the same distribution as some input data *that is used to train a model for some auxiliary ML task (classification, object detection)*
  Application: Robustness, selective inference, model monitoring

Post-hoc OOD detection

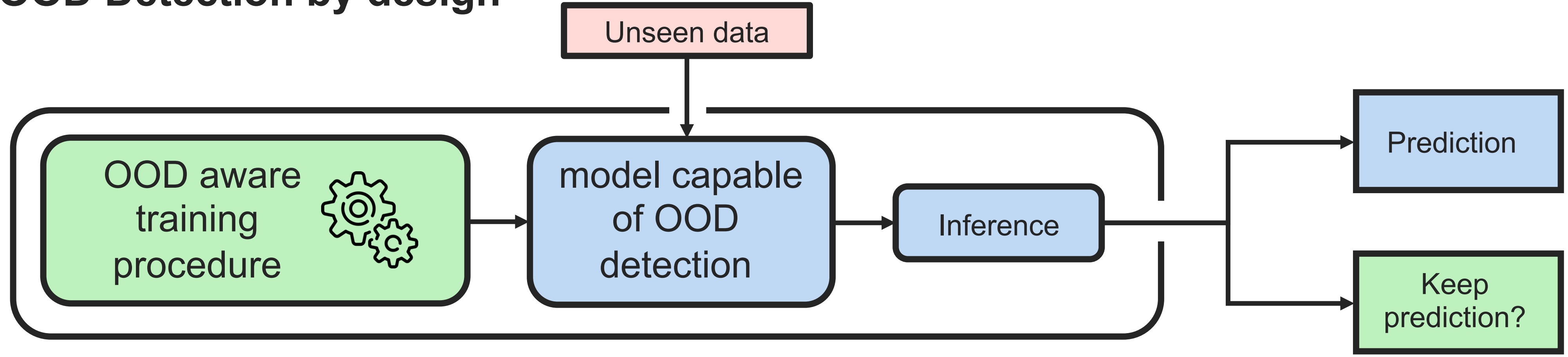
DEEL
DEpendable & Explainable Learning

Model based

Goal: find data that does not belong to the same distribution as some input data *that is used to train a model for some auxiliary ML task (classification, object detection)*

Application: Robustness, selective inference, monitoring

OOD Detection by design



Advantage:

- OOD detection by design, the model can be used as is

Drawback:

- Needs a whole new and specific training procedure

Post-hoc OOD detection

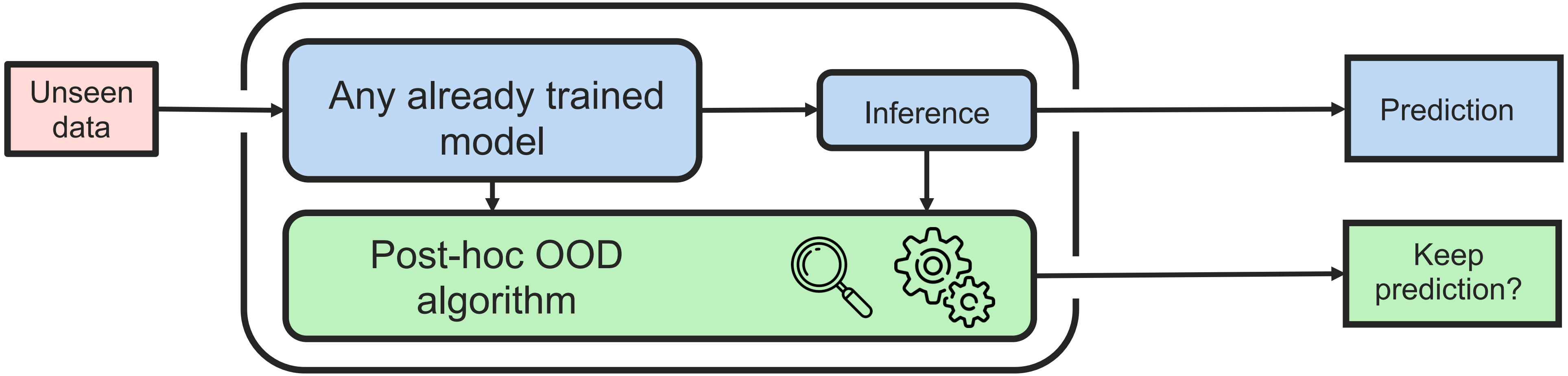
DEEL
DEpendable & Explainable Learning

Model based

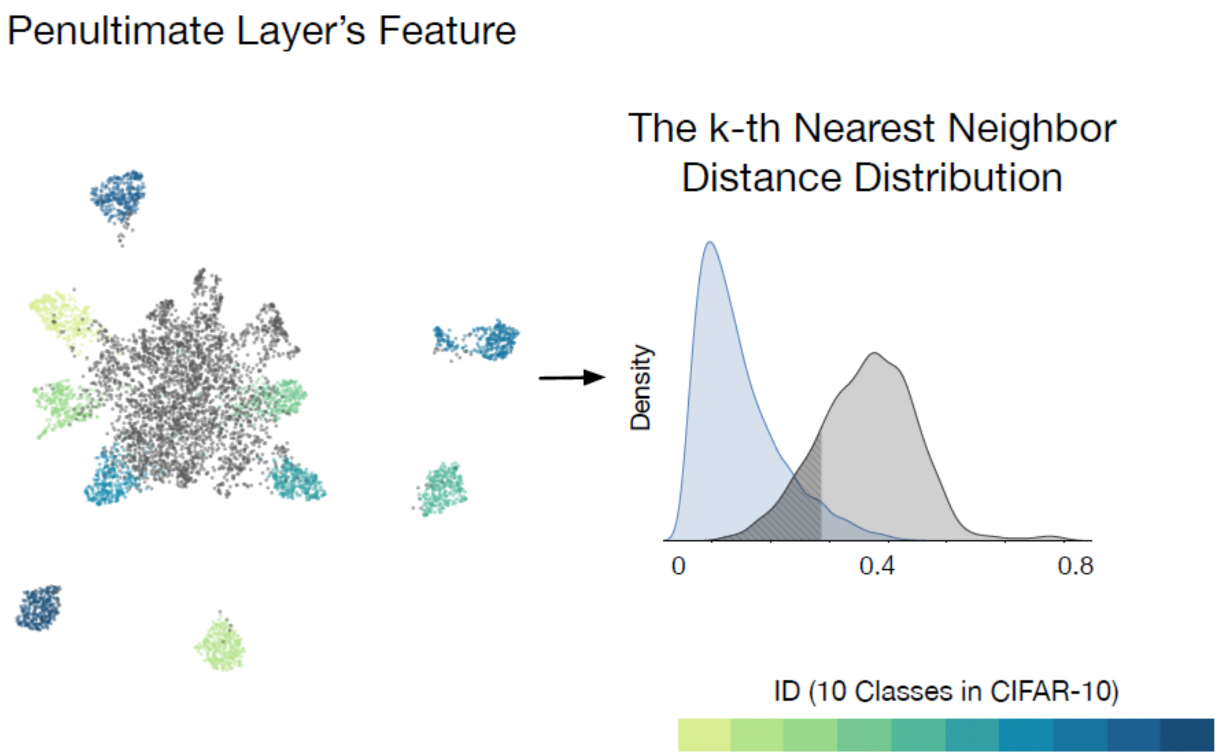
Goal: find data that does not belong to the same distribution as some input data *that is used to train a model for some auxiliary ML task (classification, object detection)*

Application: Robustness, selective inference, monitoring

Post-hoc OOD Detection



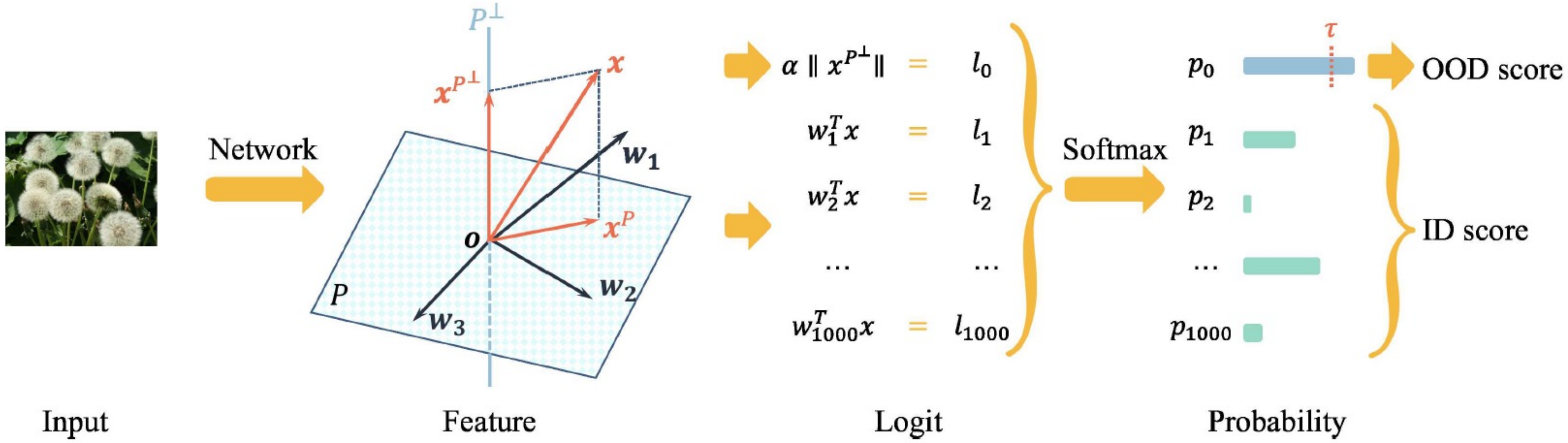
Post-hoc OOD detection: Example 1 - DKNN



Out-of-Distribution Detection with Deep Nearest Neighbors, Sun et al., ICML 2022

Post-hoc OOD detection: Example 2 - VIM

DEpendable & Explainable Learning



ViM: Out-Of-Distribution with Virtual-logit Matching, Wang et al. CVPR 2022

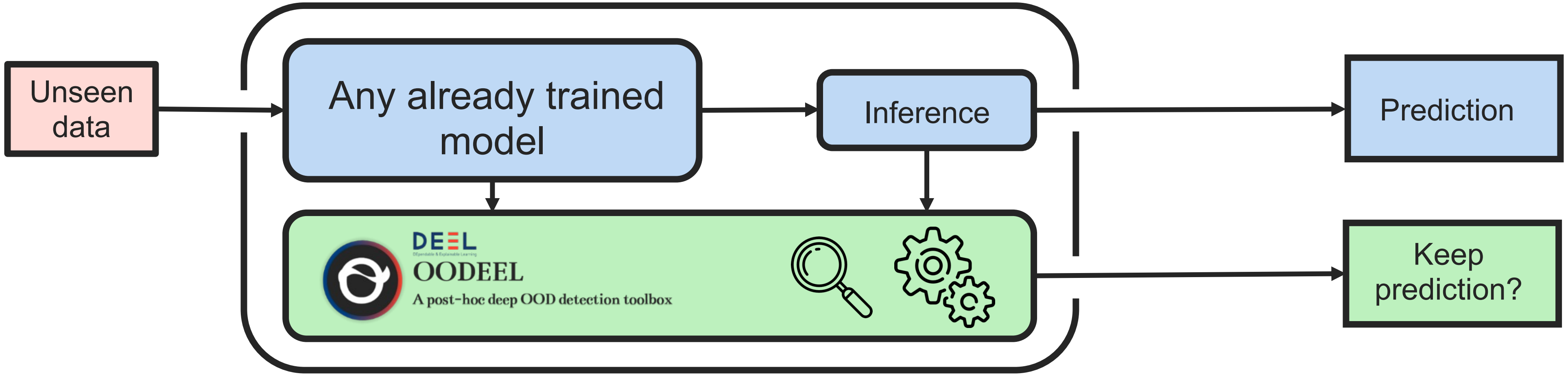
Post-hoc OOD detection

Model based

Goal: find data that does not belong to the same distribution as some input data *that is used to train a model for some auxiliary ML task (classification, object detection)*

Application: Robustness, selective inference, monitoring

Post-hoc OOD Detection



Advantage:

- Can be applied to any pretrained model

e.g. hugging face, keras.applications…

Drawback:

- Less accurate OOD performances ???

Post-hoc OOD detection

DEel
DEpendable & Explainable Learning

Is Post-hoc OOD Detection really less accurate ?

> “ **Post-Hoc Methods Outperform Training in General** For OOD and OSR methods without extra data, we further split them into two parts, one that needs a training process and the other does not. Surprisingly, those methods that require training do not necessarily obtain higher performance. Generally, methods that require training do not outperform inference-only methods. Nevertheless, the trained models can be generally used in a combined way with the post-hoc methods, which could potentially further increase their performance. ”

Yang, Jingkang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, Wenxuan Peng, Haoqi Wang, et al. “OpenOOD: Benchmarking Generalized Out-of-Distribution Detection,” Neurips 2022

… not so clear

1. Post-hoc methods are at least as good as training based
2. They are way more convenient because they do not require any training and can be applied to already trained networks

Table 1: **Experimental Results on the Generalized OOD Detection Benchmark.** The generalized OOD detection benchmark composes 9 benchmarks that are popular in the subfields of AD, OSR, and OOD detection. To save space, we use M-6 to denote MNIST-6/4, C-6 to CIFAR-6/4, C-50 to CIFAR-50/50, TIN-20 to TinyImageNet-20/180 benchmark. We only report the metric of AUROC.

	AD	OSR				OOD Detection (Near-OOD / Far-OOD)				Avg
	MVTec	M-6	C-6	C-50	T-20	MNIST	CIFAR-10	CIFAR-100	ImageNet	
- Anomaly Detection										
DeepSVDD (ICML'18)	90.80	55.83	48.41	46.42	52.73	54.82 / 54.97	56.37 / 58.90	53.45 / 49.10	N/A	56.53
CutPaste (CVPR'21)	91.24	46.53	**83.99**	66.36	56.17	**85.11 / 92.38**	**80.27** / 83.22	71.73 / 83.25	N/A	76.39
PatchCore (arXiv'21)	**98.01**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	-
DRÆM (ICCV'21)	97.03	**57.76**	63.49	**72.31**	**75.12**	79.32 / 99.12	77.30 / **83.33**	**72.73 / 85.39**	N/A	**78.45**
- Open Set Recognition & Out-of-Distribution Detection (w/o Extra Data)										
❄ OpenMax (CVPR'16)	N/A	2.23	14.37	24.76	34.67	6.80 / 1.20	36.60 / 43.22	25.00 / 19.43	24.63 / 13.45	20.52
❄ MSP (ICLR'17)	N/A	96.23	85.33	80.98	73.02	91.45 / 98.51	86.87 / 89.64	80.05 / 77.55	69.33 / 86.16	84.59
❄ ODIN (ICLR'18)	N/A	98.00	72.09	80.31	75.66	92.38 / 99.02	77.51 / 81.87	79.79 / 78.50	73.15 / 94.42	83.56
❄ MDS (NeurIPS'18)	N/A	89.79	42.91	55.12	57.60	**98.00** / 98.12	66.54 / 88.78	51.36 / 70.14	68.27 / 93.96	73.38
❄ Gram (ICML'20)	N/A	82.27	61.03	57.46	63.66	73.90 / **99.75**	58.57 / 67.51	55.35 / 72.70	48+hours	-
❄ EBO (NeurIPS'20)	N/A	**98.07**	84.86	82.68	75.61	90.77 / 98.77	87.36 / 88.86	71.33 / 68.03	73.49 / 92.78	84.38
❄ DICE (arXiv'21)	N/A	66.32	79.27	82.01	74.27	78.23 / 93.89	81.05 / 85.16	79.61 / 78.97	73.80 / 95.70	80.69
❄ GradNorm (NeurIPS'21)	N/A	94.51	64.75	68.34	71.73	76.55 / 96.39	54.78 / 53.44	70.44 / 67.20	75.74 / 95.81	74.14
❄ ReAct (NeurIPS'21)	N/A	82.91	85.87	80.49	74.61	90.29 / 97.38	87.62 / 89.03	79.47 / 80.53	79.26 / 95.18	85.22
❄ MLS (ICML'22)	N/A	98.00	84.82	82.67	75.49	92.49 / 99.08	86.11 / 88.82	**80.95** / 78.64	73.60 / 92.27	86.08
❄ KLM (ICML'22)	N/A	85.39	73.72	77.44	69.36	80.31 / 96.13	78.90 / 82.72	75.48 / 74.73	74.20 / 93.14	79.42
❄ VIM (CVPR'22)	N/A	88.59	83.86	73.41	**77.73**	93.65 / 99.61	85.75 / **92.96**	70.22 / 80.60	79.91 / **98.36**	85.39
❄ KNN (ICML'22)	N/A	97.50	**86.85**	**83.35**	74.12	96.52 / 96.66	**90.48** / 92.83	79.94 / **82.23**	**80.81** / 98.01	**88.28**
☼ ConfBranch (arXiv'18)	N/A	N/A	N/A	N/A	N/A	59.79 / 60.84	88.84 / 90.76	68.93 / 70.65	- / -	73.30
☼ G-ODIN (CVPR'20)	N/A	N/A	N/A	N/A	N/A	81.00 / 79.16	88.96 / **95.83**	**76.41 / 86.01**	- / -	84.56
☼ CSI (NeurIPS'20)	N/A	N/A	N/A	N/A	N/A	75.81 / 91.56	**89.13** / 92.48	70.78 / 66.32	- / -	81.01
☼ ARPL (TPAMI'21)	N/A	N/A	N/A	N/A	N/A	**93.89 / 98.96**	87.19 / 88.00	74.89 / 73.99	- / -	**86.15**
☼ MOS (CVPR'21)	N/A	N/A	N/A	N/A	N/A	93.19 / 94.29	60.79 / 61.17	62.77 / 55.41	- / -	71.27
☼ VOS (ICLR'22)	N/A	N/A	N/A	N/A	N/A	52.09 / 63.50	87.49 / 90.91	71.91 / 71.92	- / -	72.97
- Open Set Recognition & Out-of-Distribution Detection (w/ Extra Data)										
+ OE (ICLR'19)	N/A	N/A	N/A	N/A	N/A	N/A	76.36 / 75.17	63.69 / 70.98	N/A	71.55
+ MCD (ICCV'19)	N/A	N/A	N/A	N/A	N/A	N/A	25.70 / 25.38	49.70 / 33.84	N/A	33.66
+ UDG (ICCV'21)	N/A	N/A	N/A	N/A	N/A	N/A	**91.86 / 93.36**	**75.83** / 67.69	N/A	**82.19**
+ OpenGAN (ICCV'21)	N/A	N/A	N/A	N/A	N/A	N/A	36.60 / 43.22	69.90 / **75.98**	N/A	56.43
- Model Robustness and Uncertainty										
MCDropout (ICML'16)	N/A	96.22	84.52	81.13	73.58	91.53 / 97.07	88.15 / 90.37	80.09 / 79.40	N/A	86.21
DeepEnsemble (NeurIPS'17)	N/A	**97.24**	87.83	**83.12**	76.02	**96.07** / 99.35	90.55 / 93.24	**82.72** / 80.68	N/A	**88.68**
TempScale (ICML'17)	N/A	96.47	85.63	81.95	73.86	91.70 / 98.67	87.92 / 90.96	80.45 / 81.36	N/A	86.90
Mixup (ICLR'18)	N/A	95.67	80.90	81.86	**76.15**	86.05 / 94.23	85.28 / 86.41	80.49 / 78.56	N/A	84.56
CutMix (ICCV'19)	N/A	96.27	81.40	79.89	71.87	93.97 / 91.44	87.78 / 90.20	80.68 / 79.18	N/A	85.27
PixMix (CVPR'21)	N/A	93.85	**90.92**	77.95	73.46	93.65 / **99.49**	**93.06 / 95.66**	79.61 / **85.48**	N/A	88.32

Outline

DEpendable & Explainable Learning

1. Introduction: Post-hoc OOD detection
2. **Existing OOD detection libraries and positioning of OODEEL**
   - OpenOOD
   - PyOD
   - Pytorch-OOD
3. OODEEL in practice

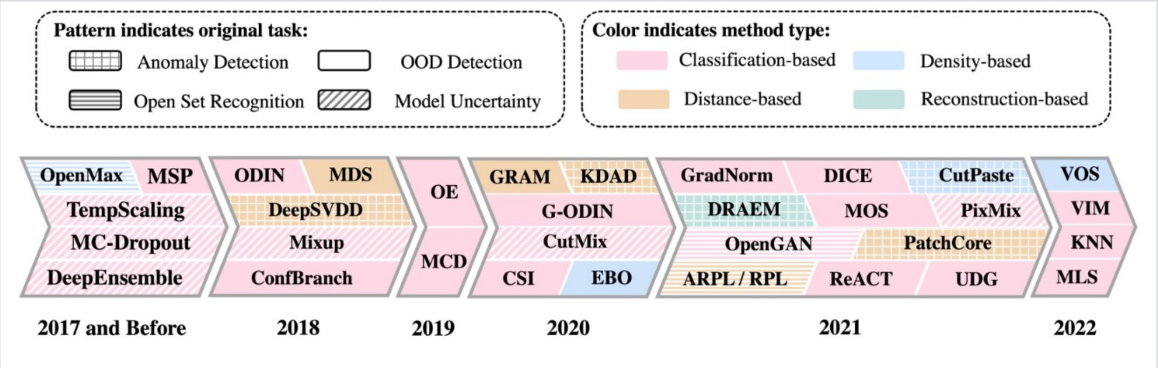
Existing OOD libraries and positioning of OODEEL

OpenOOD

[Jingkang50/OpenOOD: Benchmarking Generalized Out-of-Distribution Detection (github.com)]

- Focused on deep methods
- Extensive, encompasses all the existing deep approaches, and many algorithms
- As a result, less flexible, coherent and more difficult to use.
- More of a benchmarking software than a library
- Pytorch Only

This repository reproduces representative methods within the `Generalized Out-of-Distribution Detection Framework`, aiming to make a fair comparison across methods that initially developed for anomaly detection, novelty detection, open set recognition, and out-of-distribution detection. This codebase is still under construction. Comments, issues, contributions, and collaborations are all welcomed!



Timeline of the methods that OpenOOD supports.

Existing OOD libraries and positioning of OODEEL

DEEL
DEpendable & Explainable Learning

PyOD

[yzhao062/pyod: A Comprehensive and Scalable Python Library for Outlier Detection (Anomaly Detection) (github.com)]

- Very coherent and easy to use (see example)
- Broader than deep learning based methods
- As a result, lacks deep training based and post-hoc methods

PyOD is featured for:

- **Unified APIs, detailed documentation, and interactive examples** across various algorithms.
- **Advanced models**, including **classical distance and density estimation**, **latest deep learning methods**, and **emerging algorithms like ECOD**.
- **Optimized performance with JIT and parallelization** using numba and joblib.
- **Fast training & prediction with SUOD** [46].

Outlier Detection with 5 Lines of Code:

```
# train an ECOD detector
from pyod.models.ecod import ECOD
clf = ECOD()
clf.fit(X_train)

# get outlier scores
y_train_scores = clf.decision_scores_  # raw outlier scores on the train data
y_test_scores = clf.decision_function(X_test)  # predict raw outlier scores on test
```

Existing OOD libraries and positioning of OODEEL

DEEL
DEpendable & Explainable Learning

PytorchOOD

[GitHub - kkirchheim/pytorch-ood: PyTorch Out-of-Distribution Detection]

- Very coherent and easy to use (see example)
- Focus on deep training based and post-hoc methods
- Pytorch Only

Out-of-Distribution (OOD) Detection with Deep Neural Networks based on PyTorch.

The library provides:

- Out-of-Distribution Detection Methods
- Loss Functions
- Datasets
- Neural Network Architectures as well as pretrained weights
- Useful Utilities

and is designed such that it should be compatible with frameworks like pytorch-lightning and pytorch-segmentation-models. The library also covers some methods from closely related fields such as Open-Set Recognition, Novelty Detection, Confidence Estimation and Anomaly Detection.

```
from pytorch_ood.model import WideResNet
from pytorch_ood.detector import EnergyBased
from pytorch_ood.utils import OODMetrics

# Create Neural Network
model = WideResNet(pretrained="er-cifar10-tune").eval().cuda()

# Create detector
detector = EnergyBased(model)

# Evaluate
metrics = OODMetrics()

for x, y in data_loader:
    metrics.update(detector(x.cuda()), y)

print(metrics.compute())
```

Existing OOD libraries and positioning of OODEEL

Lib	Post-hoc	Training	Simple API	tensorflow	pytorch
PyOD (not deep)	no	no	yes	no	no
OpenOOD	yes	yes	no	no	yes
Pytorch-OOD	yes	yes	yes	no	yes
OODEEL	yes	no	yes	yes	yes

Outline

DEpendable & Explainable Learning

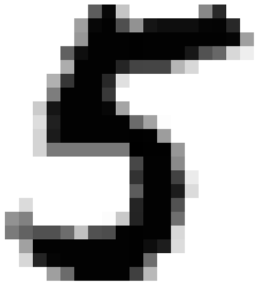
1. Introduction: Post-hoc OOD detection
2. Existing OOD detection libraries and positioning of OODEEL
3. **OODEEL in practice**
   - Benchmarking methodologies
   - Dataset wise tuto: MNIST vs Fashion MNIST
   - Class wise tuto: MNIST [0-4] vs MNIST [5-9]
   - Elecboard Components
   - Eurosat
   - Going further: LARD

OODEEL in practice – benchmarking methodologies

Test case: benchmarking

Two common ways of benchmarking baselines:

1. **Dataset wise: Consider one dataset as ID, and another as OOD**
2. Class wise: For a classification dataset with n classes, consider k classes of as ID and $n - k$ classes as OOD



ID: MNIST

OOD: CIFAR

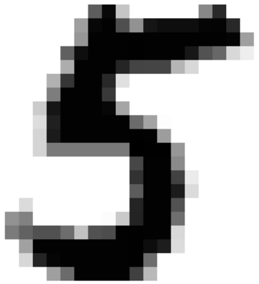
OODEEL in practice – benchmarking methodologies

DEEL
DEpendable & Explainable Learning

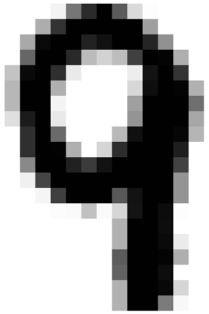
Test case: benchmarking

Two common ways of benchmarking baselines:

1. Dataset wise: Consider one dataset as ID, and another as OOD
2. **Class wise: For a classification dataset with n classes, consider k classes of as ID and $n - k$ classes as OOD**



ID: MNIST = 5



OOD: MNIST ≠ 5

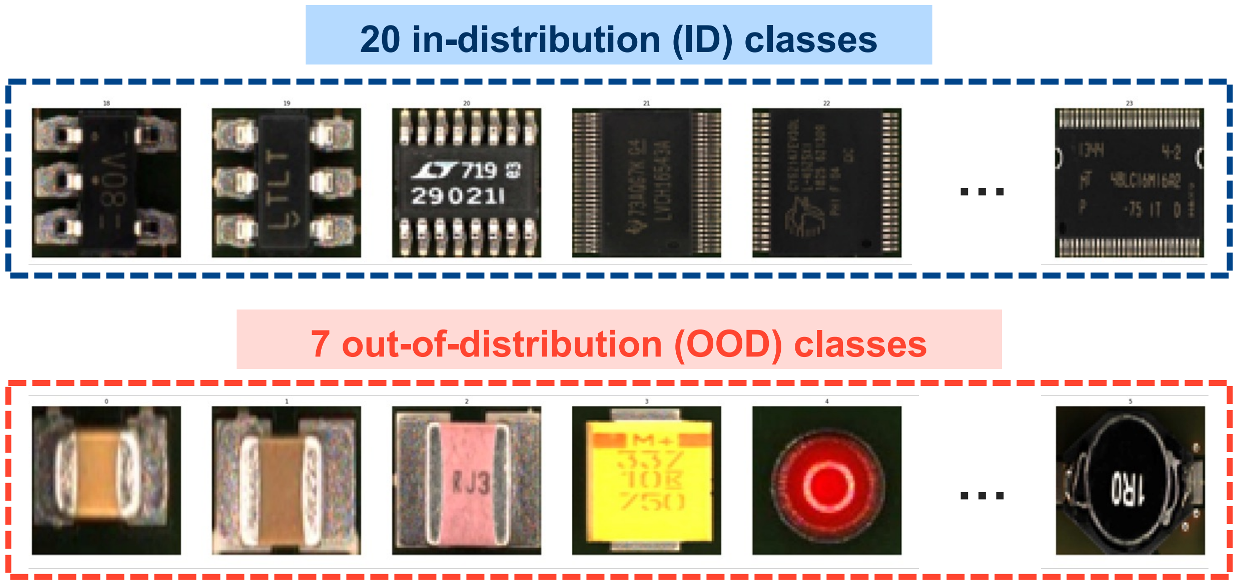
OODEEL in practice – Elecboard Components

Elecboard Components

Dataset DEEL

- 10,000 images of electronic components on circuit boards (64x64x3, RGB)
- 27 classes

Experiment: OOD detection to discover unknown components



Samples of **Elecboard Components dataset**.
We arbitrary choose 20 ID and 7 OOD classes for our experiment

OODEEL in practice – Elecboard Components

ID / OOD data

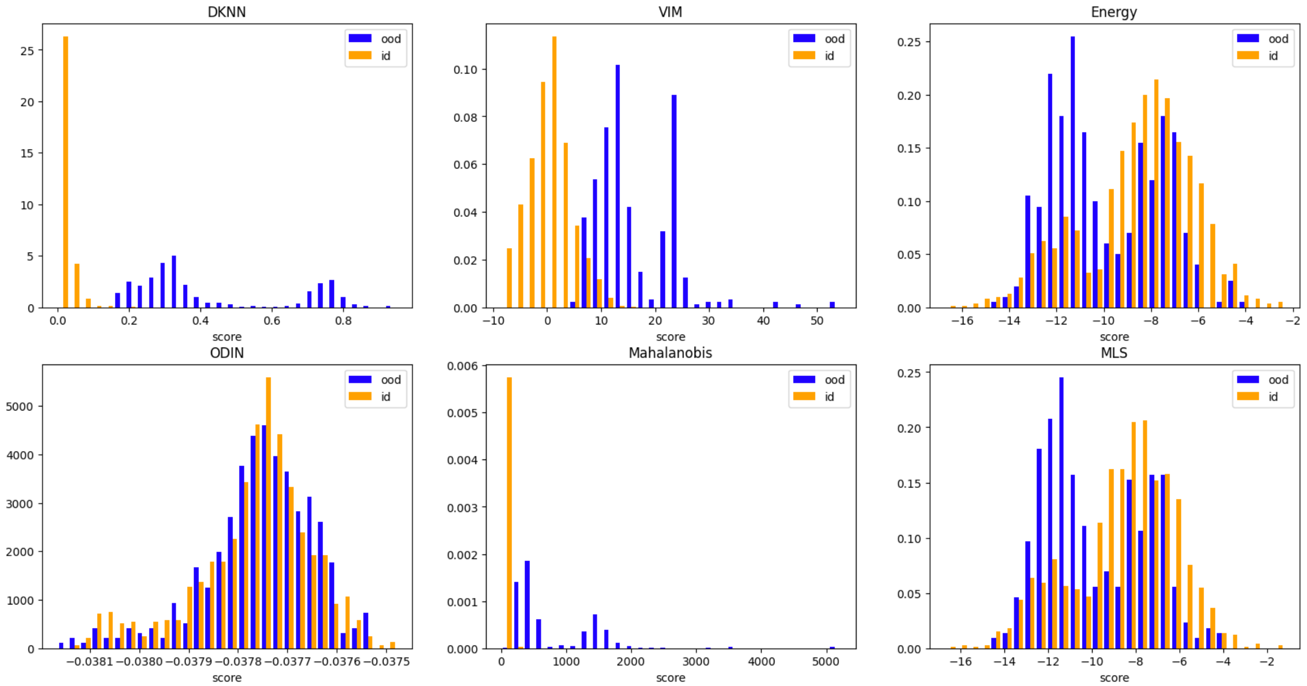
- **ID (train distribution)**: 20 components classes with similar shape, size and color
- **OOD**: 7 other components classes

Model

- ResNet20 (suited for small images)
- Trained on **ID data** (**97.8%** of test accuracy)
- No data augmentation
- Pretrained on CIFAR-10

OODEEL in practice – Elecboard Components

OOD Scores



Histograms of the OOD scores for **ID** and **OOD** data (Elecboard Component test set)

	DKNN	VIM	Energy	ODIN	Mahalanobis	MLS
AUROC (↑)	0.9999	0.992	0.345	0.538	1.0	0.356
FPR95 (↓)	0.0008	0.042	0.954	0.919	0.0	0.948

AUROC and FPR95 metrics to evaluate OOD detectors

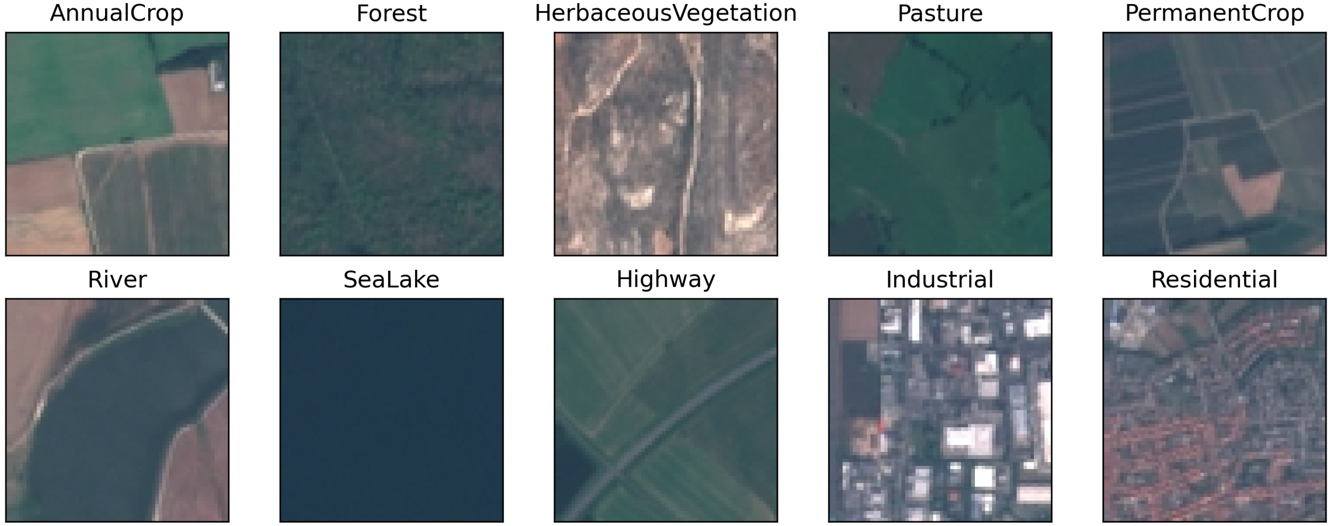
OODEEL in practice – EuroSat dataset

DEpendable & Explainable Learning

EuroSat

https://github.com/phelber/EuroSAT

- 27,000 satellite images of shape 64x64x3 (Sentinel 2, RGB)
- 10 classes



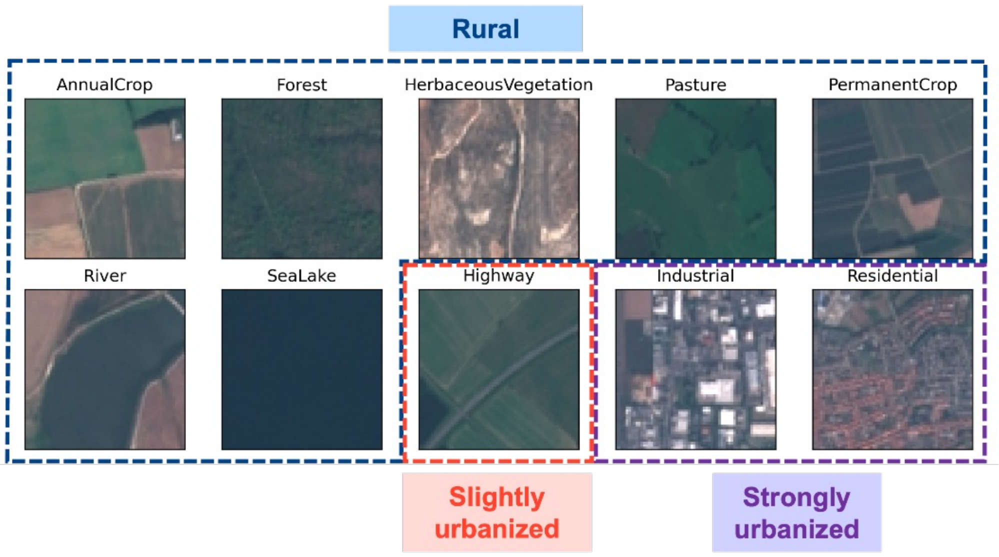
A few samples on EuroSat dataset.

OODEEL in practice – EuroSat dataset

First scenario: OOD detection to discover new types of land

ID / OOD data

- **ID (train distribution)**: Rural classes (Forest, Herbaceous vegetation, River etc.)
- **Near OOD**: Slightly urbanized class (Highway)
- **Far OOD**: Strongly urbanized classes (Industrial, Residential)



Model

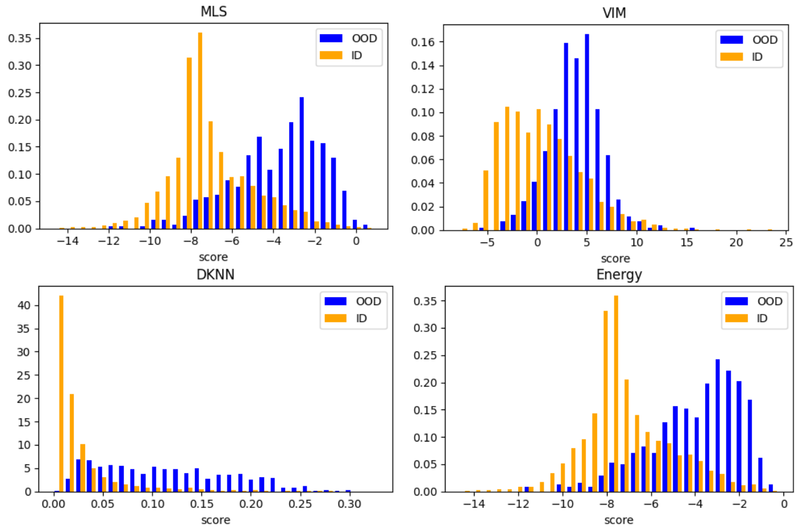
- ResNet20 (suited for small images)
- Trained on **Rural classes** (**97.8%** of test accuracy)
- No data augmentation
- Pretrained on CIFAR-10

OODEEL in practice – EuroSat dataset

DEEL
DEpendable & Explainable Learning

OOD scores

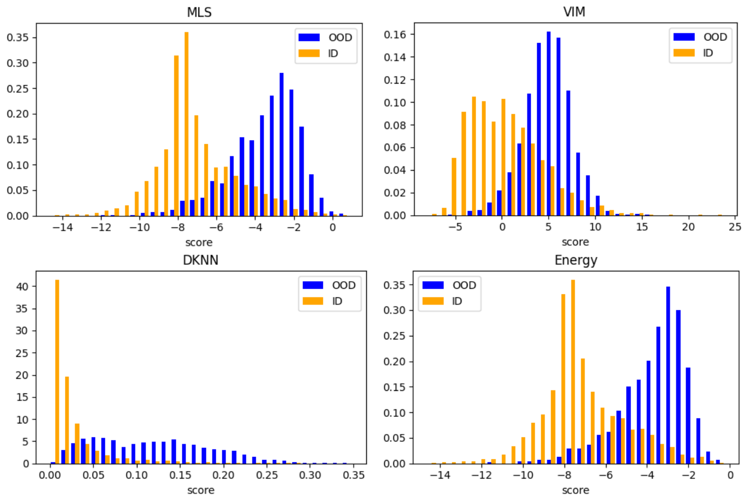
Near OOD: Highway



Histograms of the OOD scores for ID vs OOD data

	MLS	VIM	DKNN	Energy
AUROC (↑)	0.872	0.795	0.918	0.872
FPR95 (↓)	0.511	0.536	0.296	0.511

Far OOD: Industrial, Residential



Histograms of the OOD scores for ID vs OOD data

	MLS	VIM	DKNN	Energy
AUROC (↑)	0.906	0.850	0.926	0.906
FPR95 (↓)	0.331	0.409	0.263	0.331

For GOAD, ICLR2020, an Anomaly Detection method:
0.73 AUROC while involving another training

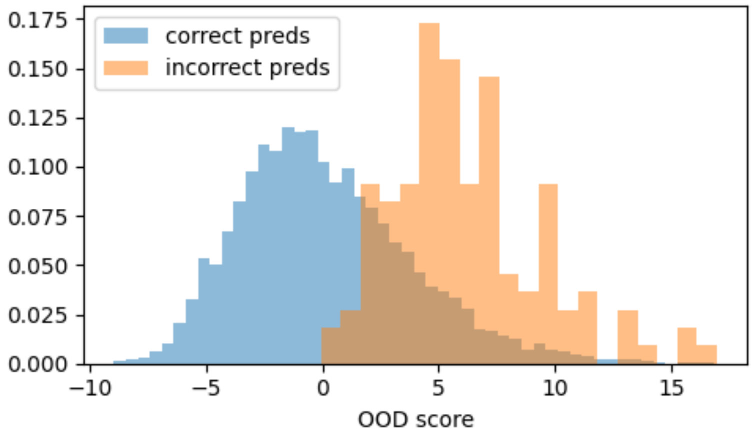
OODEEL in practice – EuroSat dataset

DEEL
DEpendable & Explainable Learning

Second scenario: OOD score as a proxy for classification error

Model

- ResNet20 (suited for small images)
- Trained on **the whole dataset** (**97.6%** of test accuracy)
- No data augmentation
- Pretrained on CIFAR-10



OOD scores on VIM for correctly vs incorrectly predicted samples (**EuroSat test set**)

With correctly predicted samples taken as ID data and incorrectly predicted samples taken as OOD data:

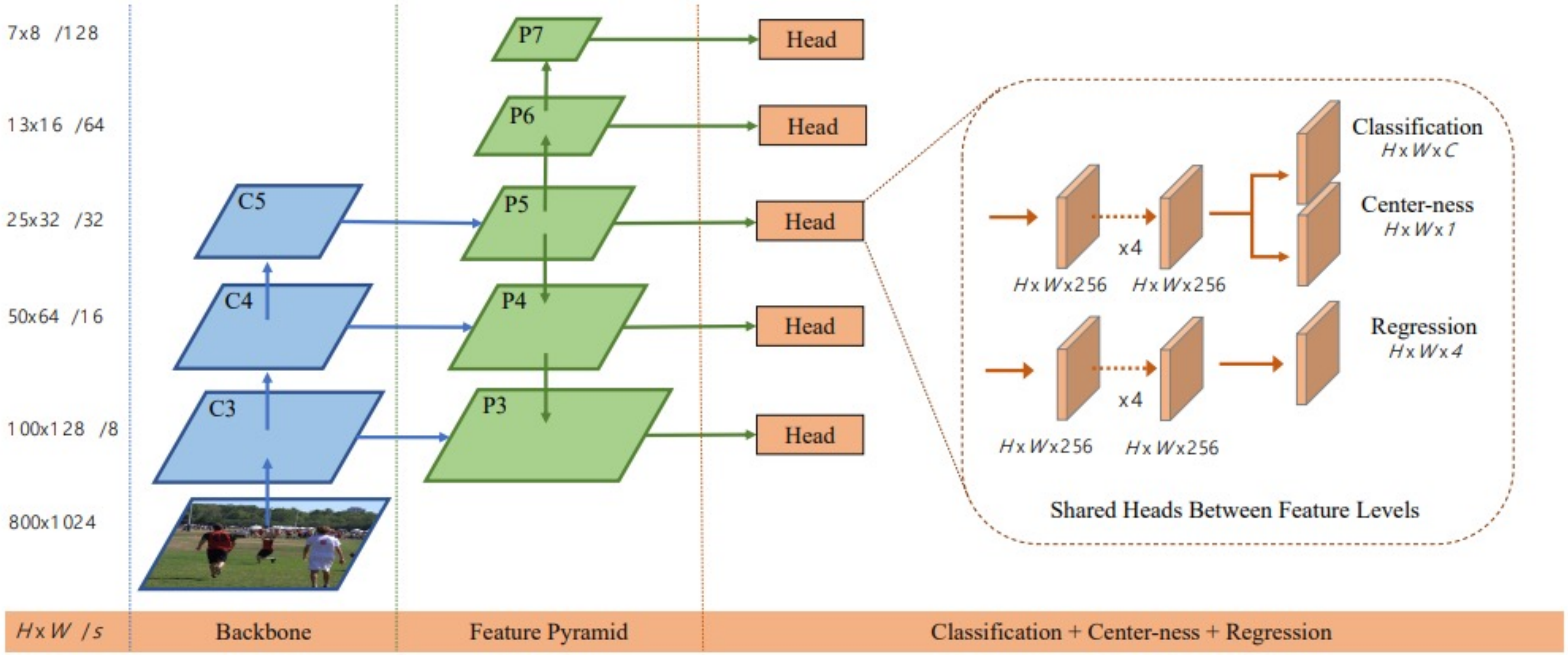
	VIM
AUROC (↑)	0.893
TPR5FPR (↓)	0.349

=> Means that if we raise alarms for 5% of workable data (on which the classifier is accurate), we avoid 34.9% of the remaining errors

OODEEL in practice – Going further with LARD

LARD
LANDING APPROACH RUNWAY DETECTION

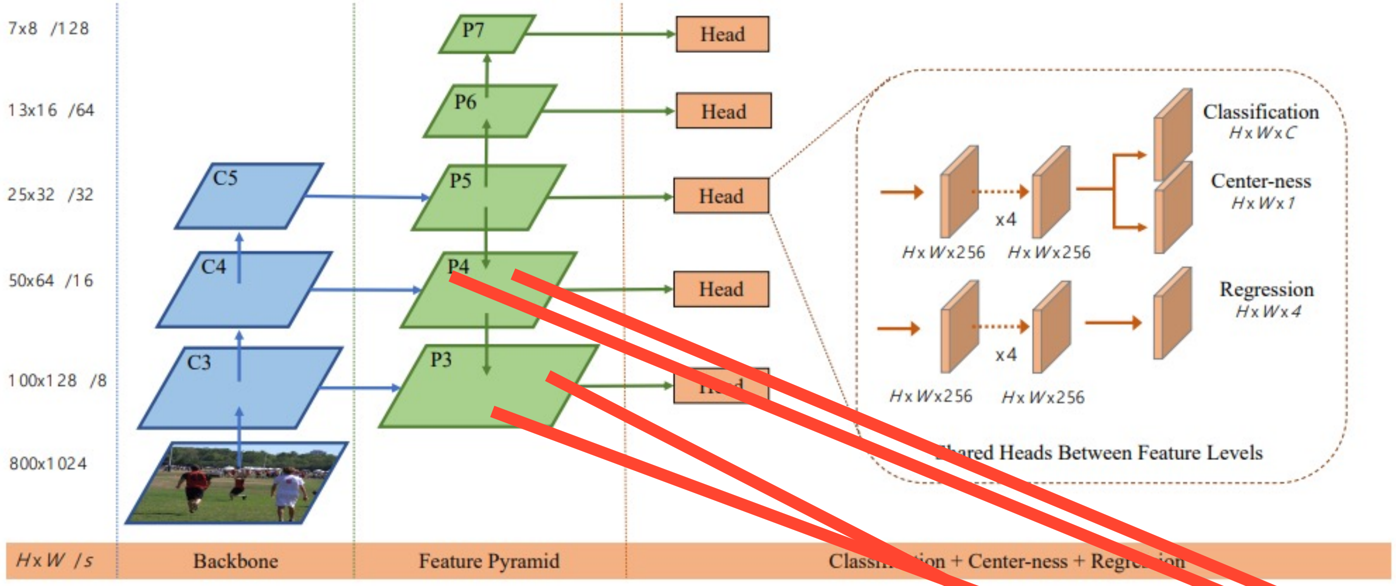
Reminder: FCOS architecture



What is OOD in object detection

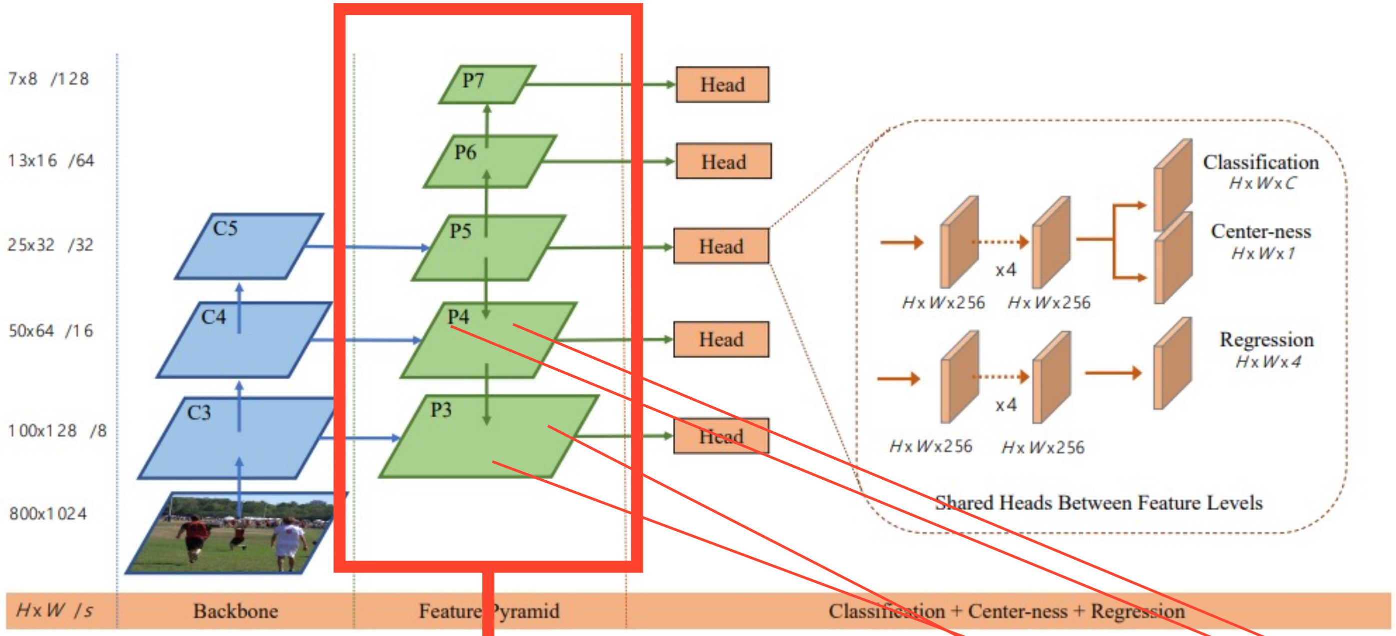
DEEL
DEpendable & Explainable Learning

Back to bounding boxes embedings



Using OODEEL for OOD in object detection

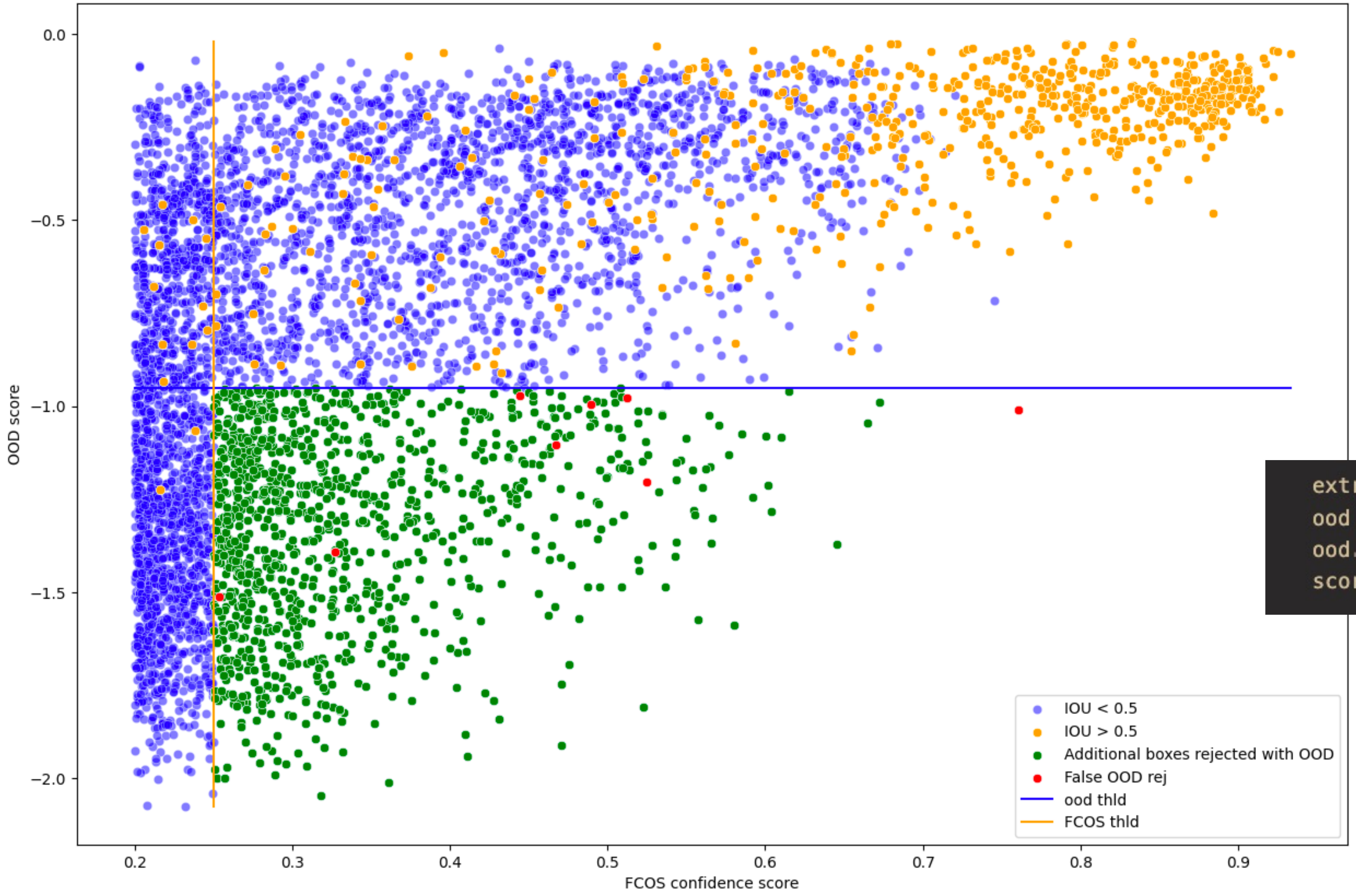
DEEL
DEpendable & Explainable Learning



DEEL
OODEEL
A post-hoc deep OOD detection toolbox

Using OODEEL for OOD in object detection

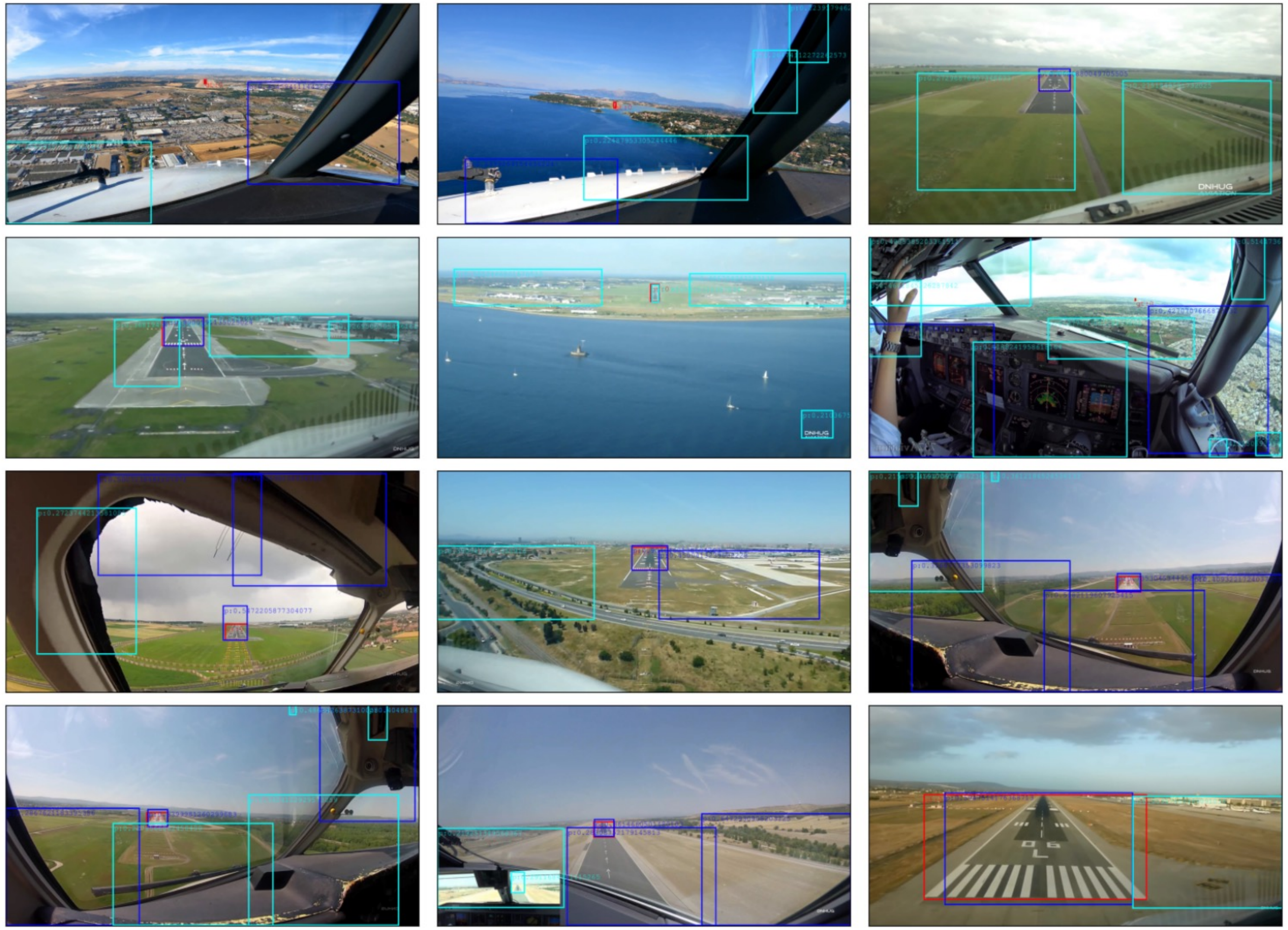
DEpendable & Explainable Learning



Custom class inherited from buit in with only few method overrides

```
extractor = FcosFeatureExtractor(model)
ood = DKNN()
ood.fit(extractor, data_loader_fit, backend="torch")
scores_calib, _ = ood.score(data_loader_test)
```

OOD in object detection: OOD bbox filtering



Conclusion and Takeaway

DEEL
DEpendable & Explainable Learning

- Post-hoc OOD detection is a convenient and efficient way of performing OOD detection on already trained models

Conclusion and Takeaway

DEEL
DEpendable & Explainable Learning

- Post-hoc OOD detection is a convenient and efficient way of performing OOD detection on already trained models
- **OODEEL implements various recent Post-Hoc OOD detection methods**

Conclusion and Takeaway

DEEL
DEpendable & Explainable Learning

- Post-hoc OOD detection is a convenient and efficient way of performing OOD detection on already trained models
- OODEEL implements various recent Post-Hoc OOD detection methods
- **OODEEL works for pytorch and tensorflow already-trained models**

Conclusion and Takeaway

DEEL
DEpendable & Explainable Learning

- Post-hoc OOD detection is a convenient and efficient way of performing OOD detection on already trained models
- OODEEL implements various recent Post-Hoc OOD detection methods
- OODEEL works for pytorch and tensorflow already-trained models
- **Many development efforts have been put into making OODEEL simple to use for various test cases (dataset wise and class wise benchmarking)**

Conclusion and Takeaway

DEEL
DEpendable & Explainable Learning

- Post-hoc OOD detection is a convenient and efficient way of performing OOD detection on already trained models
- OODEEL implements various recent Post-Hoc OOD detection methods
- OODEEL works for pytorch and tensorflow already-trained models
- Many development efforts have been put into making OODEEL simple to use for various test cases (dataset wise and class wise benchmarking)
- **Specifically, it is very easy to customize OODEEL and to add your own method (not covered in the presentation but see this doc tutorial)**

Conclusion and Takeaway

DEEL
DEpendable & Explainable Learning

- Post-hoc OOD detection is a convenient and efficient way of performing OOD detection on already trained models
- OODEEL implements various recent Post-Hoc OOD detection methods
- OODEEL works for pytorch and tensorflow already-trained models
- Many development efforts have been put into making OODEEL simple to use for various test cases (dataset wise and class wise benchmarking)
- Specifically, it is very easy to customize OODEEL and to add your own method (not covered in the presentation but see this doc tutorial)
- **OODEEL is a library ready for real world applications but it is also very adapted for research (e.g. for object detection or benchmarking new OOD methods)**

Conclusion and Takeaway

DEEL
DEpendable & Explainable Learning

- Post-hoc OOD detection is a convenient and efficient way of performing OOD detection on already trained models
- OODEEL implements various recent Post-Hoc OOD detection methods
- OODEEL works for pytorch and tensorflow already-trained models
- Many development efforts have been put into making OODEEL simple to use for various test cases (dataset wise and class wise benchmarking)
- Specifically, it is very easy to customize OODEEL and to add your own method (not covered in the presentation but see this doc tutorial)
- OODEEL is a library ready for real world applications but it is also very adapted for research (e.g. for object detection or benchmarking new OOD methods)

What to do next?

DEEL
DEpendable & Explainable Learning

DEEL
OODEEL
A post-hoc deep OOD detection toolbox

Thank you for your attention