PyTorch-OOD: A Library for Out-of-Distribution Detection based on PyTorch

K. Kirchheim, M. Filax, F. Ortmeier

Faculty of Computer Science
Otto-von-Guericke University Magdeburg
Germany

August 13, 2022

Motivation

- DNNs SOTA for extremely high dimensional data
- Models only work well if input sufficiently similar to training data (in-distribution)
- Prevent errors by rejecting Out-of-Distribution (OOD) inputs
  $\rightarrow$ Out-of-Distribution Detection

Related Fields:

- Anomaly Detection, Novelty Detection
- Open Set Recognition
- Confidence Estimation

PyTorch-OOD Goals

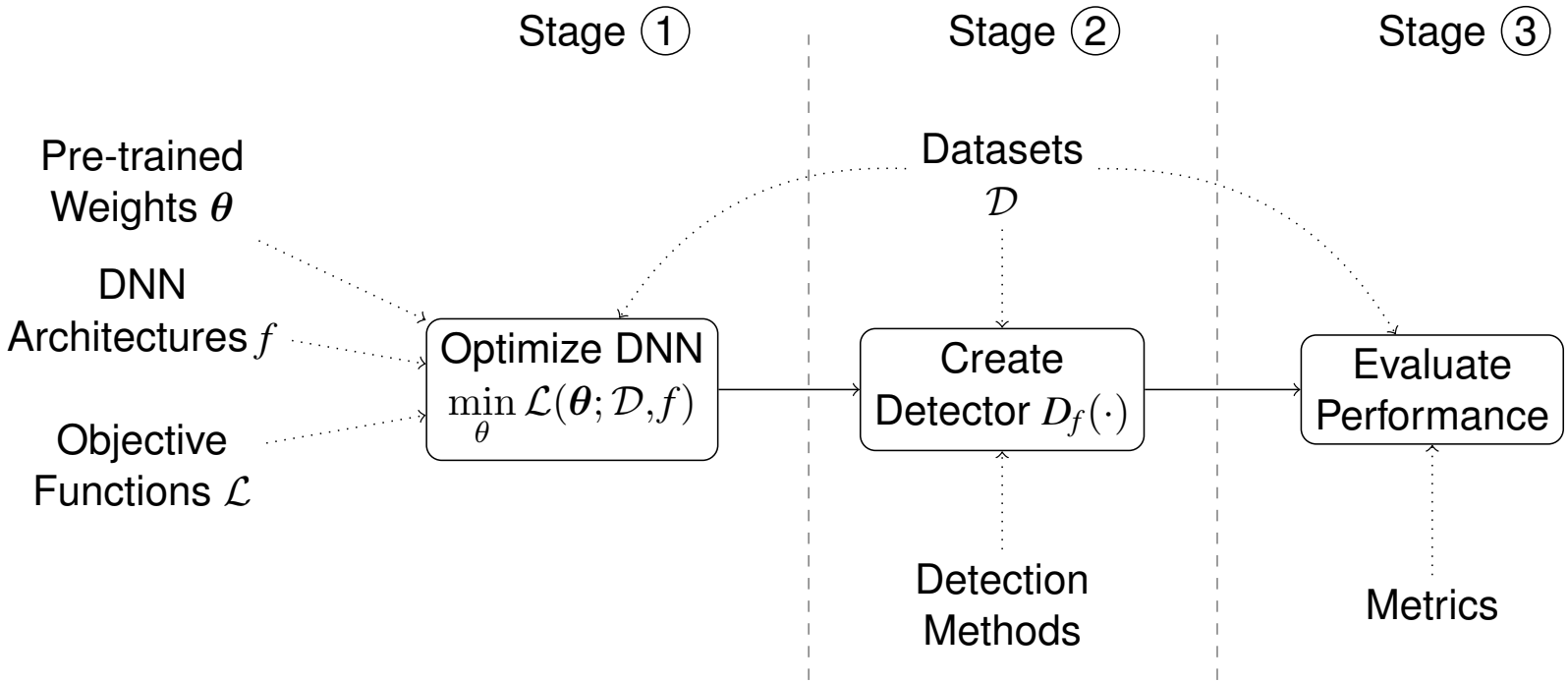
Promote Reproducibility

- Results can be influenced by implementation details [1]
- Open Source: (hopefully) fewer bugs
- Implemented algorithms well tested and documented

Accelerate Research

- no reimplementation of baseline methods
- no dataset setup
- less training (pre-trained models)
- integrates with frameworks like `pytorch-lightning`

Common OOD Detection Workflow



DNN Architectures

- Some architectures in prominent publications used, but implementation not available via package [2, 3, 4, 5]
- Pre-Trained weights available, but must be downloaded manually [5, 3]

```
from pytorch_ood.model import WideResNet

# create Pre-Trained Neural Network
model = WideResNet(pretrained="er-cifar10-tune")
```

Objective Functions

- Unsupervised, supervised
- **Assumption**: OOD samples have label $y < 0$
  $\rightarrow$ can be handled automatically

```python
from pytorch_ood.loss import EnergyRegularizedLoss
from pytorch_ood.loss import CrossEntropyLoss
# ...

xent = CrossEntropyLoss()
regu = EnergyRegularizedLoss(alpha=0.1)

for x, y in data_loader:
    output = model(x.cuda())
    loss = xent(output, y) + regu(output, y)
    loss.backward()
    # ...
```

OOD Detectors

- $D_f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ constructed from DNN
- **Assumption**: Detector predicts outlier scores

```python
from pytorch_ood.detector import NegativeEnergy

# create detector
detector = NegativeEnergy(model)
outlier_scores = detector(x.cuda())
```

$$\text{outlier}(\mathbf{x}) = \begin{cases} 1 & \text{if } D_f(\mathbf{x}) > \tau \\ 0 & \text{else} \end{cases} \tag{1}$$

Datasets

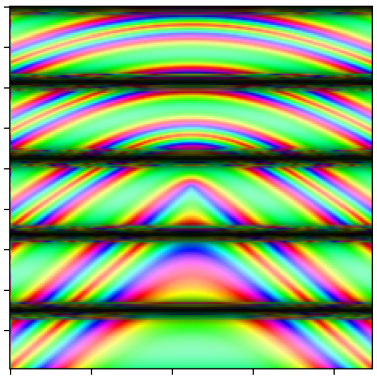
- OOD Datasets for training (Supervised Objective Functions)
- OOD Datasets for testing

Types:

- Images (Classification, Segmentation, Unlabeled, Noise)
- Text (Classification, Unlabeled)
- Audio (Classification)

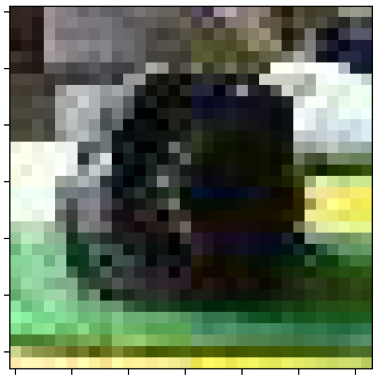
OOD Image Dataset Examples

Fooling Images



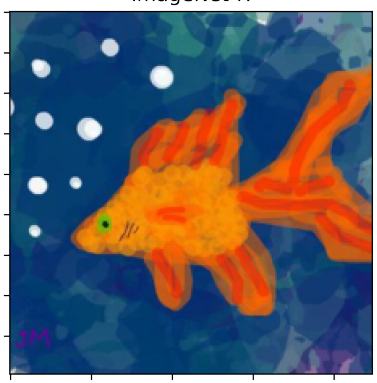
ImageNet-A

TinyImageNet Resize



Textures

ImageNet-R



ImageNet-O

LSUN Crop

LSUN Resize

Putting it all together

```python
from pytorch_ood.model import VisionTransformer
from pytorch_ood.detectors import NegativeEnergy
from pytorch_ood.utils import OODMetrics

# Stage 1: create DNN
model = VisionTransformer(pretrained="b16-cifar100-tune")
model.eval().cuda()

# Stage 2: create detector
detector = NegativeEnergy(model)

# Stage 3: evaluate detector
metrics = OODMetrics()

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

print(metrics.compute())
```

Benchmark: Key Takeaways

- Energy-Based OOD [5] works also well for (sequential) text data
- ViT [6] + Pre-Training [7] + Energy-Based OOD works extremely well
- Supervision (i.e. training with example anomalies) [8, 9, 5] increases performance across all tasks

Install & Contribute

- Git:
  `https://gitlab.com/kkirchheim/pytorch-ood`
- Documentation:
  `https://pytorch-ood.readthedocs.io/`
- PyPI:
  `https://pypi.org/project/pytorch-ood/`

```
> pip install pytorch-ood
```

Thank You

References I


Xavier Bouthillier, César Laurent, and Pascal Vincent.
Unreproducible research is reproducible.
In *International Conference on Machine Learning*, pages 725–734, 2019.


Shiyu Liang, Yixuan Li, and R Srikant.
Enhancing the reliability of out-of-distribution image detection in neural networks.
In *6th International Conference on Learning Representations, ICLR 2018*, 2018.


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

References II


Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin.
A simple unified framework for detecting out-of-distribution samples and adversarial attacks.
Advances in Neural Information Processing Systems, 31, 2018.


Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li.
Energy-based out-of-distribution detection.
Advances in Neural Information Processing Systems, 33, 2020.


Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al.
An image is worth 16x16 words: Transformers for image recognition at scale.

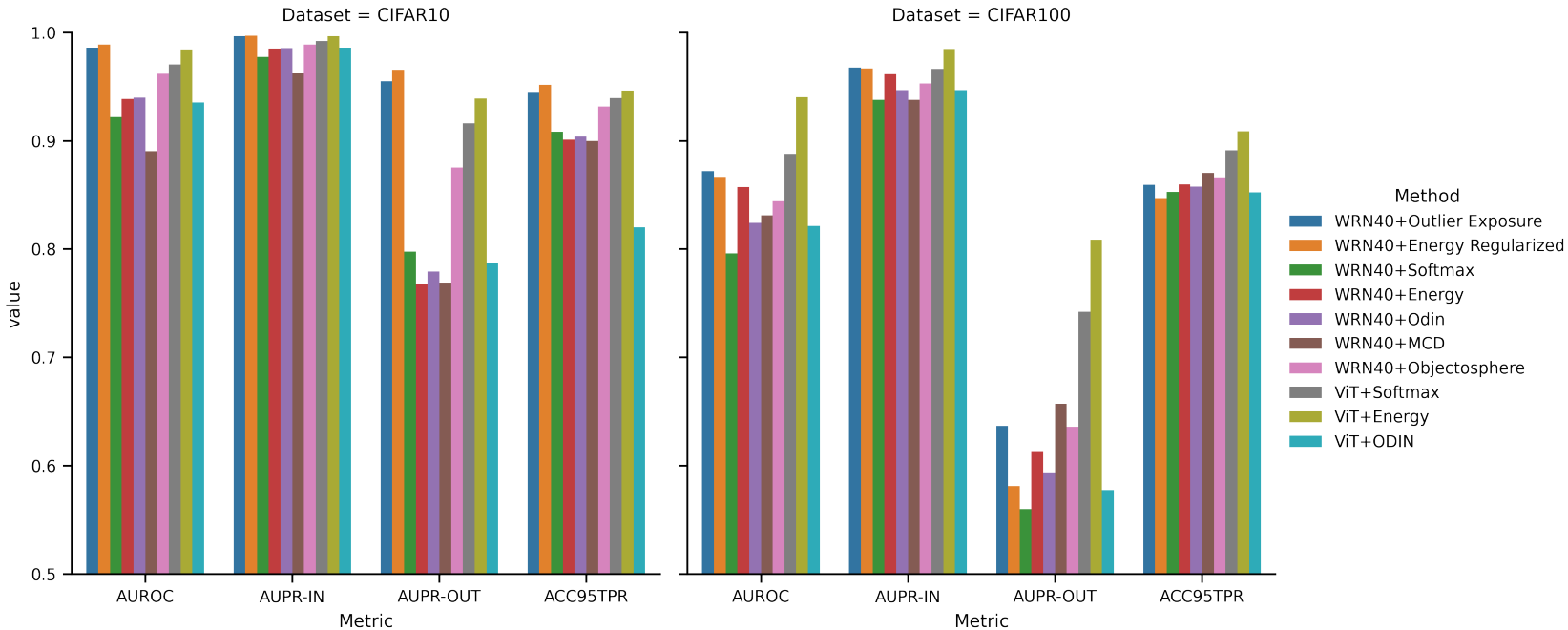
References III

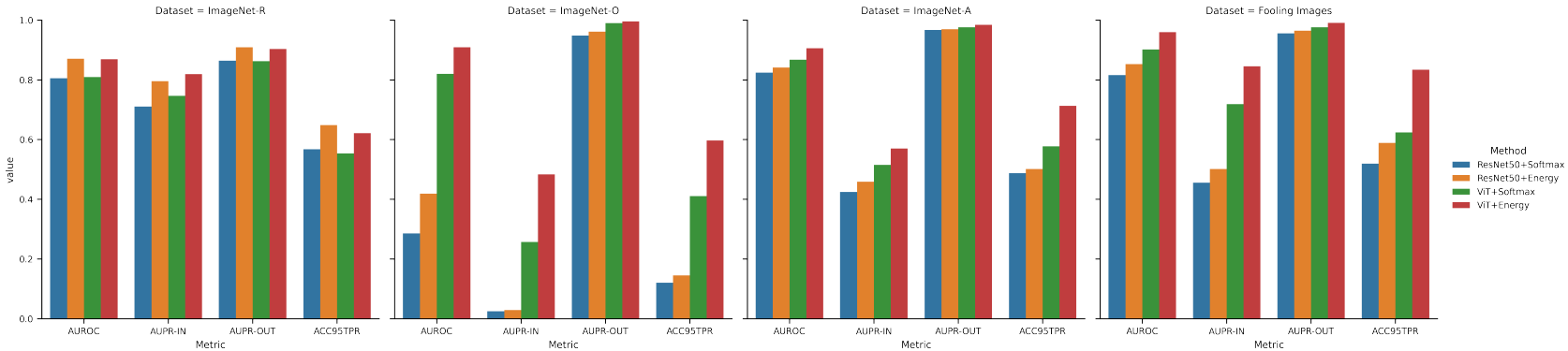
International Conference on Learning Representations, 2021.

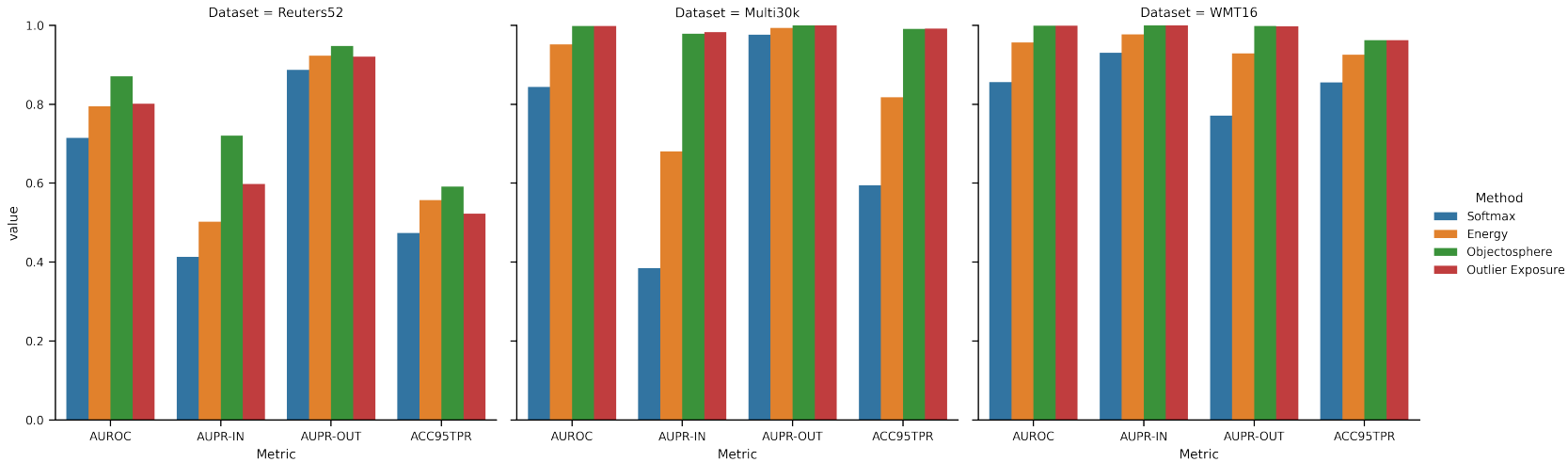

Dan Hendrycks, Kimin Lee, and Mantas Mazeika.
Using pre-training can improve model robustness and uncertainty.
In *International Conference on Machine Learning*, pages 2712–2721. PMLR, 2019.


Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich.
Deep anomaly detection with outlier exposure.
In *International Conference on Learning Representations*, 2018.


Akshay Raj Dhamija, Manuel Günther, and Terrance Boult.
Reducing network agnostophobia.
In *Advances in Neural Information Processing Systems*, pages 9157–9168, 2018.


Dataset = CIFAR10
Dataset = CIFAR100
value
1.0
0.9
0.8
0.7
0.6
0.5
AUROC
AUPR-IN
AUPR-OUT
ACC95TPR
Metric
Method
WRN40+Outlier Exposure
WRN40+Energy Regularized
WRN40+Softmax
WRN40+Energy
WRN40+Odin
WRN40+MCD
WRN40+Objectosphere
ViT+Softmax
ViT+Energy
ViT+ODIN


Dataset = ImageNet-R
Dataset = ImageNet-O
Dataset = ImageNet-A
Dataset = Fooling Images
value
1.0
0.8
0.6
0.4
0.2
0.0
AUROC
AUPR-IN
AUPR-OUT
ACC95TPR
Metric
Method
ResNet50+Softmax
ResNet50+Energy
ViT+Softmax
ViT+Energy


Dataset = Reuters52
Dataset = Multi30k
Dataset = WMT16
value
1.0
0.8
0.6
0.4
0.2
0.0
AUROC
AUPR-IN
AUPR-OUT
ACC95TPR
Metric
Method
Softmax
Energy
Objectosphere
Outlier Exposure