Title

Unsupervised Out-of-Distribution Detection by Maximum Classifier Discrepancy
Qing Yu, Kiyoharu Aizawa; ICCV, 2019

Citation

Yu, Q., & Aizawa, K. (2019). Unsupervised Out-of-Distribution Detection by Maximum Classifier Discrepancy. In Proceedings of the IEEE International Conference on Computer Vision (pp. 9518-9526).

Summary

Main Points

In this paper, the authors propose a two-headed neural network and maximize the discrepancy between the two classifiers to detect out-of-distribution samples while correctly classifying in-distribution inputs. Also, they propose a new problem setting where they utilize unlabeled data for unsupervised training whereas previous works only exploit labeled in-distribution samples.

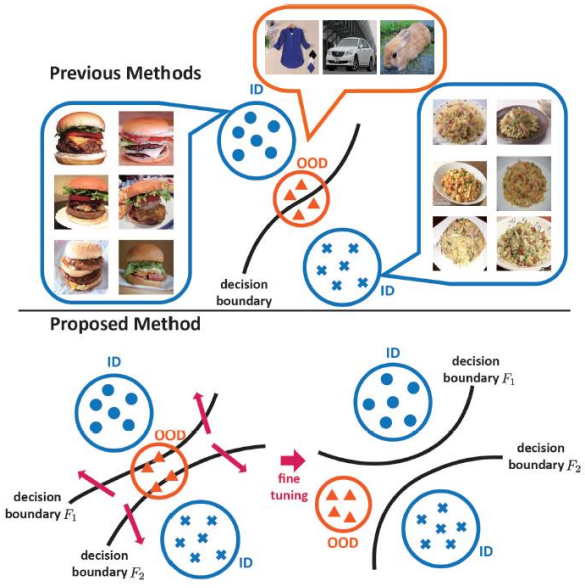


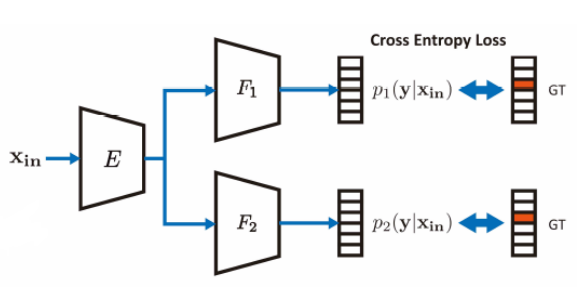
Figure 1: Comparison of previous and the proposed OOD detection methods.

By some empirical experiments, authors figure out that two classifiers involve different decision boundaries. During the unsupervised fine-tuning procedure, these two decision boundaries push OOD inputs outside the manifold of the in-distribution samples; see Figure 1.

Two-headed Network

Typical classification neural networks can be divided into two parts: a feature extraction network and a classifier. In a two-headed network, two fully-connected classifiers share a feature extraction network. Yu and Aizawa implement the feature extractor of their two-headed

network based on DenseNet-BC and WideResNet (WRN).




```
def forward(self, x):
    out = self.conv1(x)
    out = self.trans1(self.dense1(out))
    out = self.trans2(self.dense2(out))
    out = self.trans3(self.dense3(out))
    out = self.dense4(out)
    out = F.avg_pool2d(F.relu(self.bn(out)), 4)
    out = out.view(out.size(0), -1)
    out_1 = self.linear1(out)
    out_2 = self.linear2(out)
    return out_1, out_2
```

Figure 2: *Overview and PyTorch implementation of the proposed two-headed network.*

Discrepancy Loss

Yu and Aizawa define the discrepancy loss as the following equation:

$$d(p_1(\mathbf{y}|\mathbf{x}), p_2(\mathbf{y}|\mathbf{x})) = H(p_1(\mathbf{y}|\mathbf{x})) - H(p_2(\mathbf{y}|\mathbf{x}))$$

, where H is the entropy over the SoftMax probability density function. Maximizing this loss, the entropy of the first classifier's output encourages the first classifier to predict similar probability values for all classes. On the other hand, the entropy of the second classifier's output lead the second classifier to predict high probability for a single class. This is demonstrated empirically in Figure 3. In this contradictory situation, both classifiers will achieve the entropy difference by transforming the manifold surrounding the OOD as much as possible.

Inference

At inference time, in order to distinguish between in-distribution samples and out-of-distribution samples, Yu and Aizawa consider the L1 distance between the two classifiers' outputs:

$$\sum_{i=1}^{K} |p_1(y_i|\mathbf{x}) - p_2(y_i|\mathbf{x})| > \delta$$

The one whose L1 distance is higher than the threshold is considered as an out-of-distribution sample.

Evaluation (Main benefits and drawbacks)

Benefits

With very little change to the underlying deep neural network, this approach surpasses other state-of-the-art methods.

Drawbacks

Better is the enemy of good: due to its unsupervised nature, increasing the learning time do not guarantee a better performance. It is crucial to decide an early stopping point with validation sets since the learning is unstable.

Results

Reproduced Results

Disclaimer
Since the description of the fine-tuning procedure is ambiguous and not sufficient to write code that reproduces the reported results, I repeated experiments several times with a variety of tweaks on my own. The main difference is that in the fine-tuning step, I set the learning rate to 0.001 instead of the reported value (0.1). Besides, since it was unclear whether the loss formula (3) used in the fine-tuning step is the aggregated loss of step A and step B or the loss used only in step B, I just chose the former empirically (see the fine_tune function in utils.py).

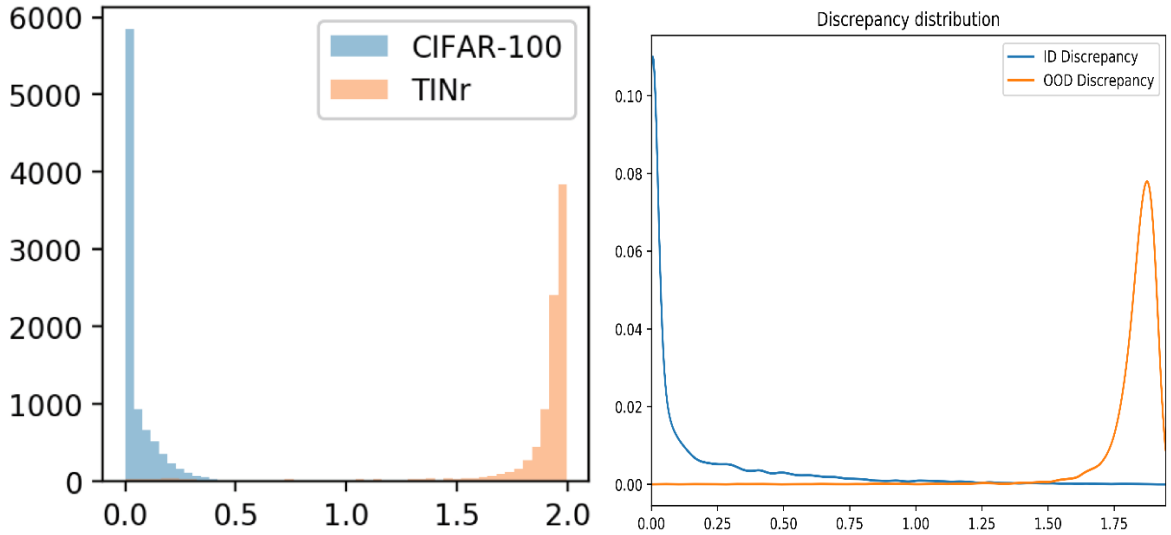


Figure 3: *Histogram of ID (CIFAR-100 or CIFAR-10) and OOD (resized Tiny ImageNet dataset) detection scores of the proposed method (**left**) and that of the reproduced model (**right**). The reproduced model follows the results reported in the paper fairly well. Noticeably, OOD samples are distributed near 1.8, which is* SUM([1.0, 0.0, …, 0.0] – [0.1, 0.1, …, 0.1])*, due to the maximum discrepancy training.*

	OOD dataset	FPR (95% TPR) ↓			Detection Error ↓			AUROC ↑		
		ODIN	ELOC	Ours	ODIN	ELOC	Ours	ODIN	ELOC	Ours
Dense-BC CIFAR-10	TINc	3.6	1.5	**0.1**	4.2	3.0	**0.7**	99.2	99.6	**99.9**
	TINr	10.1	3.2	**1.7**	6.9	4.0	**2.3**	98.2	99.3	**99.6**
	LSUNc	6.0	3.4	**0.2**	5.3	4.1	**0.7**	98.7	99.3	**99.9**
	LSUNr	3.5	1.4	**0.4**	4.2	2.7	**1.1**	99.2	99.6	**99.9**
	iSUN	5.9	-	**0.6**	5.3	-	**1.3**	98.9	-	**99.9**

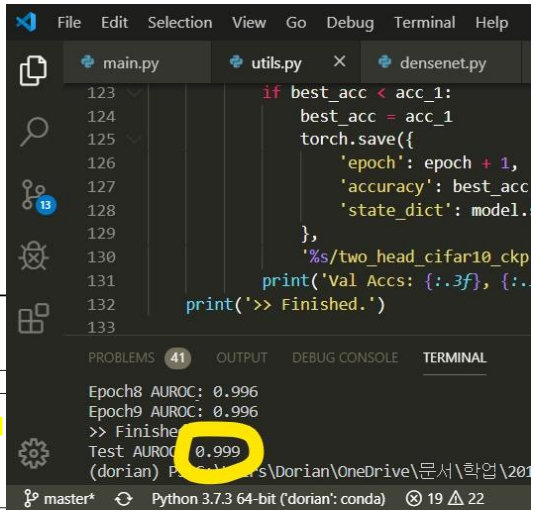


Figure 4: *The result of distinguishing ID and OOD benchmarks (**left**). Authors' method is compared with ODIN and Ensemble of Leave-Out Classifier (ELOC). All values are percentages. The **right** figure shows the reproduced AUROC result of mine (Model: **Dense-BC** (L=100, k=12), ID: **CIFAR-10**, OOD: resized Tiny ImageNet (**TINr**)).*

In Figure 3 and 4, I present the histogram and benchmark results of ID and OOD detection

scores. For a reliable comparison, the AUROC value is obtained by repeating fine-tuning three times with a fixed pretrained Dense-BC (L=100, k=12). It is 99.83 ± 0.03, which is even better than the reported AUROC.

Experiment on Another Dataset (MNIST)

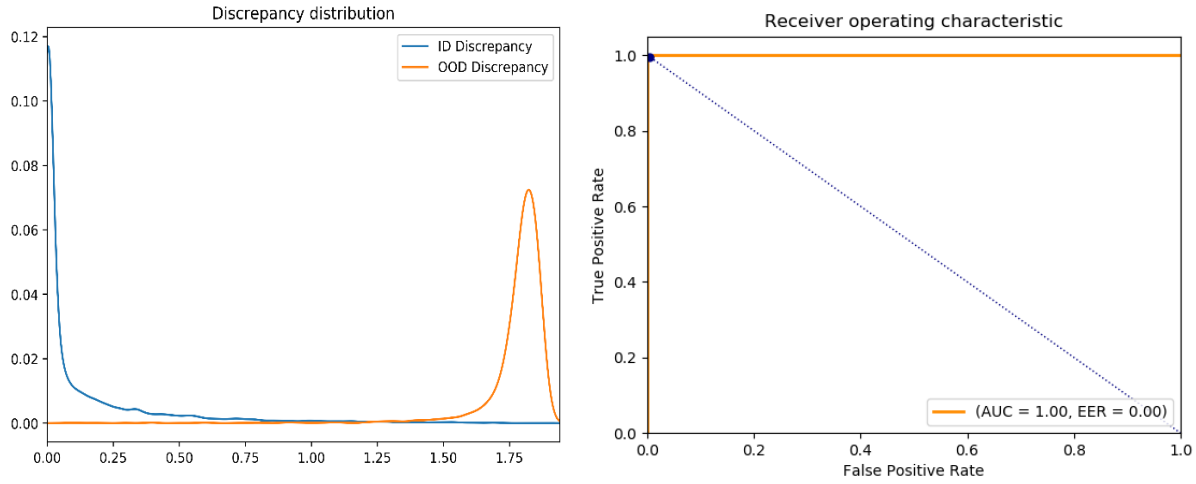


Figure 5: *Histogram of ID (CIFAR-10) and OOD (MNIST) detection scores of the reproduced model (**left**) and the ROC curve of distinguishing ID and OOD (**right**). Except for the OOD data, I use the same ID data, pre-trained model, and hyperparameters (learning rate, etc.) as* **Figure 3** *and* **4**. *Noticeably, OOD samples are distributed near 1.8.*

Figure 5 shows the AUROC is 1, which means the reproduced model perfectly discriminate ID samples and OOD samples. In fact, the images of the MNIST data are very easily distinguished by the human eye.

Source Code Link

https://github.com/Mephisto405/Unsupervised-Out-of-Distribution-Detection-by-Maximum-Classifier-Discrepancy (My repo)