On the evaluation of post hoc Out-Of-Distribution detectors

Lorenzo Rossi
Politecnico di Milano
lorenzo17.rossi@mail.polimi.it

Abstract

Out-Of-Distribution (OOD) detection is becoming more and more relevant as machine learning models are deployed in the real world. This is even more important with deep neural networks as their performance get worse when used outside their training distribution. In this work, we analyze a particular scenario called post hoc, where the task is to detect OOD data using a pre-trained model trained on the In-Distribution data (ID) for a different task (in our case image classification). We consider the effect of a novel method called `ReAct`*[22] on some of the most common scoring functions, which are* `softmax` *score,* `ODIN`*, and* `energy`*-based function. We evaluate these techniques with a wide range of models and datasets to discover their limitations and strengths. Our empirical investigation shows that* `ODIN` *with $\epsilon = 0$ is the best method in the vast majority of the cases. Additionally, we discover some failure cases and that* `ReAct` *could help fix them. The source code is available at https://github.com/tpoppo/ood-post-hoc.*

1. Introduction

Nowadays, deep learning models are used in many different places and situations, even safety-critical cases, therefore, these models need to be reliable and trustworthy. For this reason, detecting Out-Of-Distribution data is a quite important problem to tackle, as most deep learning methods fail outside of their training distribution and the prediction score becomes way more unreliable[18, 14, 24]. Out-Of-Distribution detection is quite a challenging problem as it requires distinguishing whether the data comes from the training distribution or not. This problem can be divided in various subgroups[25], such as open set recognition and one-class anomaly detection. A wide range of recent and old works tried to solve these problems in many different ways[12, 6, 3] and with different assumptions and constraints. In particular, in many situations, it is not possible to train a model to specifically solve this task, as training a model to detect OOD samples can be quite complex[5]. Therefore, some works focus on post hoc OOD detection methods, which means using a pre-trained model (e.g. on image classification) and utilizing the class scores to detect whether the data comes for the training distribution or not. Additionally, another important challenge of OOD detection is how to properly choose and evaluate the hyperparameters, for this reason, many OOD detectors are hyperparameters free or are not particularly important to tune.

Our contributions are:

1. We implemented all the techniques considered in TensorFlow[1].
2. We analyzed and compared the performance of various methods on a wide range of datasets.
3. We found some general and practical guidelines for choosing the post hoc OOD detector.

2. Background

Firstly, we define formally the problem and then we define the mathematical notation used.

2.1. Problem

The problem of post hoc Out-Of-Distribution detection for classification is defined in the following way. Let $f : \mathcal{X} \to \mathbb{R}^K$ be the classifier (trained on the $\mathcal{D}_{in}$ distribution), where $\mathcal{X}$ is the sample space (in our case, it is the image space) and K is the number of output classes. Therefore, the task is to discover whether an input sample $x_0 \in \mathcal{X}$ has been generated from the $\mathcal{D}_{in}$ In-Distribution data, which is known, or the $\mathcal{D}_{out}$ Out-Of-Distribution data, which is unknown, by using a decision function $G(x_0; f)$ (define as in Equation 1).

$$G(x_0; f) = \begin{cases} 0 & \text{if } x_0 \sim \mathcal{D}_{out}, \\ 1 & \text{if } x_0 \sim \mathcal{D}_{in}. \end{cases} \tag{1}$$

The difficulty in detecting OOD samples is strongly correlated with the separation between $\mathcal{D}_{out}$ and $\mathcal{D}_{in}$, the less the distributions are separated, the harder the classification becomes. However, in the vast majority of the practical cases the separation between $\mathcal{D}_{out}$ and $\mathcal{D}_{in}$ is quite large.

In all the cases considered, the methods have a scoring function $S(x_0; f)$, which returns a large value if the sample is ID, otherwise if it is an OOD sample a small one. Therefore, to solve the initial problem we could select a threshold t and define $G(x_0; f)$ as in Equation 2.

$$G(x_0; f) = \begin{cases} 0 & \text{if } S(x_0; f) \leq t, \\ 1 & \text{if } S(x_0; f) > t. \end{cases} \tag{2}$$

2.2. Notation

We will introduce some useful notations. We define $\texttt{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$, $\texttt{sign}(x)$ is the sign function element-wise and `clip(x)` clips all the values between 0 and 1.

3. Related work

In this section, we analyze various scoring functions, which are computed from the class scores of the neural network (the output), and also we consider a novel method that injects an additional layer between the features extraction layer and the logits extraction one.

3.1. `softmax` score

One of the first OOD methods that tackle this problem is often called `softmax` score[9]. Their main claim is that correctly classified samples have a higher maximum `softmax` probability than the Out-Of-Distribution samples. Thus, the resulting scoring function is shown below (Equation 3).

$$S_{MSP}(x) = \max_k \texttt{softmax}(f(x))_k \tag{3}$$

3.2. `ODIN` score

The `ODIN` score[15] was introduced as an improvement of the `softmax` score. They propose a temperature scaling factor T and add a small adversarial perturbation with factor ϵ (Equation 4).

$$\begin{gathered} S_{\texttt{ODIN}}(\tilde{x}; T) = \max_k \texttt{softmax}(\frac{f(\tilde{x})}{T})_k \\ \tilde{x} = x - \epsilon \texttt{sign}(-\nabla \log S_{\texttt{ODIN}}(x)) \end{gathered} \tag{4}$$

The temperature scaling has been used by other methods[11] to soften the logits prediction and to better separate the in- and Out-Of-Distribution data.

3.3. `energy` score

This method was proposed in [16], and it is based on the idea of an `energy`-based model. Additionally, the authors proposed not only an inference time scoring function (Equation 6), but also a regularization term in the loss (Equation 5) to increase the effectiveness of the `energy` score. The scoring function is hyper-parameters free, while the loss term has two hyper-parameters m_{in} and m_{out}, which are the margin parameters.

$$S_{\texttt{energy}}(x) = -\ln \sum_{i=1}^{K} e^{f(i)_i} \tag{5}$$

$$\begin{aligned} \mathcal{L}_{\texttt{energy}} = & \mathbb{E}[\max(0, S_{\texttt{energy}}(x_{in}) - m_{in})] \\ & + \mathbb{E}[\max(0, m_{out}) - S_{\texttt{energy}}(x_{out})] \end{aligned} \tag{6}$$

3.4. `ReAct`

In [22], they proposed `ReAct` an additional layer added before the last layer (before the last fully connected layer) of the neural network at inference time. The Equation 7 `ReAct` shows how the layer behave.

$$\texttt{ReAct}(x; \lambda) = \min(x, \lambda), \tag{7}$$

where min is the element-wise minimum of the two elements, $x \in \mathcal{X}$ is the input vector (the feature vector) and $\lambda \in \mathbb{R}$ is the truncation value. This value should be chosen to preserve the ID distribution, thus in practice, it is equal to the p-th percentile (generally 90%) of the ID distribution. The idea behind this new layer is tailored to the hypothesis that the OOD distribution has a positively skewed feature vector distribution, and this leads to many scoring methods to miss-classify the data from OOD to ID. The original paper dives into both a theoretical and empirical analysis of the consequences of this phenomenon.

4. Proposed approach

The main aim of our analysis is to find some useful guidelines to decide which method one should use. Overall, our discoveries suggest that:

- `ODIN` with $\epsilon = 0$ and $T = 1000$ seems a strong and simple starting choice, but it is not always optimal.
- `ReAct` might sometimes be useful, thus, we suggest testing with and without it. In particular, we recommend using it when the expected OOD samples are very far from the ID data.
- There is no strong correlation between the architecture and the size of the model, and the OOD detection performance.
- All the methods show good dataset transferability.

5. Methodology and Datasets

Datasets & Augmentations In order to compare the various approaches, we used a total of 10 datasets. For the whole work, we consider the ImageNet dataset without any augmentation as the In-Distribution one, while all the other ones as Out-Of-Distribution. The main group is of 64 dataset and augmentation pairs (see Appendix A for the full list) and it is used in all the cases where the tested model is only ResNet101, while when we compare all the models we used a reduced subset of 10 pairs (see Appendix A). The datasets and augmentation are chosen to test a wide and diverse range of situations (see Appendix C for some examples).

Datasets The ImageNet dataset[4] is our In-Distribution dataset. We use a subset of iNaturalist, SUN, Places used in [22], which only includes non-overlapping categories with ImageNet. Moreover, we also consider ImageNette, ImageNet a[10], Rock Paper Scissors[17], ImageNet v2[20] datasets, and noise randomly generated from the uniform and normal distributions. ImageNette and ImageNet v2 are arguably not OOD, as the concept and categories are the same, but we prefer to consider also these less reasonable and extreme cases.

Augmentations on ImageNet Moreover, we also use a wide range of augmentations on ImageNet: Gaussian noise, blur, pixelization, perspective transformation, and JPEG encoding quality level. Most of the augmentations were done with AugLy[19]. In our work, we consider ImageNet with augmentation as OOD, this assumption is not always useful, as a really small augmentation might be reasonable and not harmful for the model.

Augmentations on other datasets We use the Fast Gradient Sign Method (`FGSM`)[7] to create adversarial noise, a small perturbation in the input space that can completely change the resulting feature vector. In this way, these experiments try to discover whether starting from an OOD sample is possible to fool the OOD detector to classify the sample as ID. Thus, to discover whether these OOD scorers are robust to weak adversarial attacks. This approach has several limitations, firstly, the adversarial noise is generally outside the scope of many OOD scorers, and secondly, the `FGSM` is one of the simplest and weakest adversarial attack methods. Additionally, the gradient is computed only using the model without `ReAct`.

$$\mathcal{L}_{adv}(x) = \ln \sum_{i=0}^{N} e^{\texttt{softmax}(f(x))_k} \tag{8}$$

We decided to use the loss function of Equation 8, which is the differentiable version of the `softmax` scorer. The resulting transformation is shown in Equation 9

$$x_{adv} = \texttt{clip}(x + \epsilon \texttt{sign}(\nabla \mathcal{L}_{adv}(x))) \tag{9}$$

OOD detectors In our analysis, we used the following scoring functions: `softmax`, `ODIN`, and `energy`. We used two `ODIN` hyperparameter configurations, the first case is with $\epsilon = 0$ and $T = 1000$, while the second one is with $\epsilon = 0.0014$ and $T = 1000$. The other scoring functions are hyperparameters free. For the `energy` scorer, we only used the inference component and we did not use the specific loss function, as we only compare the methods in the post hoc setting. For `ReAct`, we used $p = 90\%$, which is the suggested value in the original paper. In all our experiments, we consider all the possible scoring functions with and without the `ReAct` layer, therefore, for a total of 8 methods. All the methods are implemented in TensorFlow and are compatible with almost all the pre-trained models available in `tensorflow.keras.applications`.

Models used We mainly focus our case study on ResNet101[8], which is used with the large group of dataset and augmentation pairs (if not differently stated we used this configuration). However, we also consider many other models EfficientNetB0[23], DensetNet121[13], DenseNet169[13], DenseNet201[13], ResNet50[8], ResNet152[8], VGG16[21], VGG19[21]. These models were chosen to cover a wide range of CNN architecture and size, and because they are available on TensorFlow.

6. Experiments

In this section, we evaluate `ReAct` in combination with four scoring functions on a suite of OOD detection tasks.

6.1. Main results

Table 1 depicts the main datasets and augmentations of our analysis. In particular, `ODIN` ($\epsilon = 0$) shines in the most important cases, however, it is worse in the toughest and arguably not even considerable OOD cases.

Overall, from both Table 2 and Table 1, `ODIN` with $\epsilon = 0.0$ without the `ReAct` layer achieves the best results. However, the use of the `ReAct` layer is beneficial with `softmax` and `ODIN`.

6.2. Comparison with and without `ReAct`

In this subsection, we highlight the main advantages and disadvantages of adding the `ReAct` layer[22] with ResNet101, thus, using the set of 64 model and scoring function pairs. Overall, the `ReAct` method is quite useful with some scoring functions, while harmful for others. The results remain quite consistent across datasets (Table 3 and Figure 1). In particular, we can notice that while the version without `ReAct` completely fails (with `AUC` < 0.5) in many cases with the `ODIN` and $\epsilon > 0$, the version with `ReAct` achieves good results (see appendix D for more details). An

OOD dataset	Augmentation	`softmax`	`ODIN`($\epsilon = 0$)	`ODIN`($\epsilon = 0.0014$)	`energy`
RockPaperScissors	-	98.54/99.75	**99.98**/99.82	0.40/78.12	92.46/52.22
iNaturalist	-	88.42/94.17	92.37/**94.18**	0.58/78.52	78.13/56.67
SUN	-	86.19/89.78	89.51/**89.58**	1.26/73.51	81.07/50.09
Places	-	82.86/84.14	**87.09**/83.39	0.71/71.10	77.79/47.96
Gaussian Noise	-	99.84/**100.00**	**100.00/100.00**	0.21/90.88	98.03/97.56
Uniform Noise	-	99.85/**100.00**	**100.00/100.00**	0.28/95.04	95.57/98.25
ImageNet v2*	-	55.26/55.55	57.38/55.28	**99.64**/54.98	57.19/48.12
ImageNette*	-	**83.79**/75.92	52.58/53.93	1.06/52.70	55.34/53.00
ImageNet	Normal($\sigma = 0.002$)	51.31/52.73	52.35/52.56	24.77/52.08	**52.91**/48.98
ImageNet	Normal($\sigma = 0.25$)	79.69/87.30	**88.81**/86.97	3.77/79.54	84.82/60.31
ImageNet	Normal($\sigma = 1.25$)	95.63/98.31	**99.50**/98.77	0.49/88.71	97.77/86.15
ImageNet	Blur(r= 1.0)	55.40/54.93	**59.80**/54.85	40.10/53.57	58.06/46.68
ImageNet	Pixelation(r= 0.5)	56.16/56.14	**61.40**/56.30	52.21/54.30	59.16/48.24
ImageNet	JPEG(q= 25)	54.37/55.26	**58.53**/55.46	15.52/53.80	59.02/49.51

Table 1: `AUC` score without `ReAct` and with `ReAct` in various settings using the ResNet101 model. We consider ImageNet (without augmentations) as the ID dataset. In **bold** the best results. *ImageNet v2 and ImageNette have the same concept of ImageNet.

Scoring	`ReAct`	Avg Rank
`ODIN`($\epsilon = 0$)	no	2.27
`softmax`	yes	2.71
`ODIN`($\epsilon = 0$)	yes	2.81
`energy`	no	4.06
`softmax`	no	4.52
`ODIN`($\epsilon = 0.0014$)	yes	5.57
`energy`	yes	6.76
`ODIN`($\epsilon = 0.0014$)	no	7.29

Table 2: Average ranking based on `AUC` score (ranked from 1 to 8, all the possible combination of scoring methods possible) for ResNet101. The lower the better.

additional analysis of the effect of `ReAct` in the scores is present in the Appendix B.

Scoring	Better w/o	Better w/
`softmax`	10	54
`energy`	59	5
`ODIN`($\epsilon = 0$)	41	23
`ODIN`($\epsilon = 0.0014$)	7	57

Table 3: It counts how many times the `AUC` score is better with and without `ReAct`.

The results divided per dataset show that the scoring function plays a more important role than the datasets, which means that there is good transferability across different datasets and augmentations.

6.3. The effect of augmentation

In this subsection, we analyze the effect of various augmentations and, in particular, how the OOD detectors consider the images as we increase the distortion.

6.3.1 Gaussian Noise

Overall, all the methods follow a similar trend and can distinguish even a very small amount of noise, with the only exception of `ODIN`($\epsilon = 0.0014$) without `ReAct` which completely fails.

6.3.2 Other Natural Augmentations

The other natural augmentations (Figure 3) show a similar picture. In particular, the `ODIN`($\epsilon = 0.0014$) without `ReAct` and the `energy` function with `ReAct` do not work properly, as both achieve an `AUC` smaller than 0.5 (random guessing) even in trivial cases.

6.3.3 Adversarial noise

While adversarial noise is outside the scope of the vast majority of the OOD detectors, it might still be an important case study. We used `FGSM`, which is one of the weakest and simplest adversarial attack techniques, and we were able to reduce the `AUC` score of nearly all the methods (Figure 4). The gradient is computed using the model without `ReAct`, thus, the reduction is more marked in these scorers. We do not show the results when computing the gradient with `ReAct`, as a similar situation is depicted just with slightly better scores, maybe because the resulting gradient is less

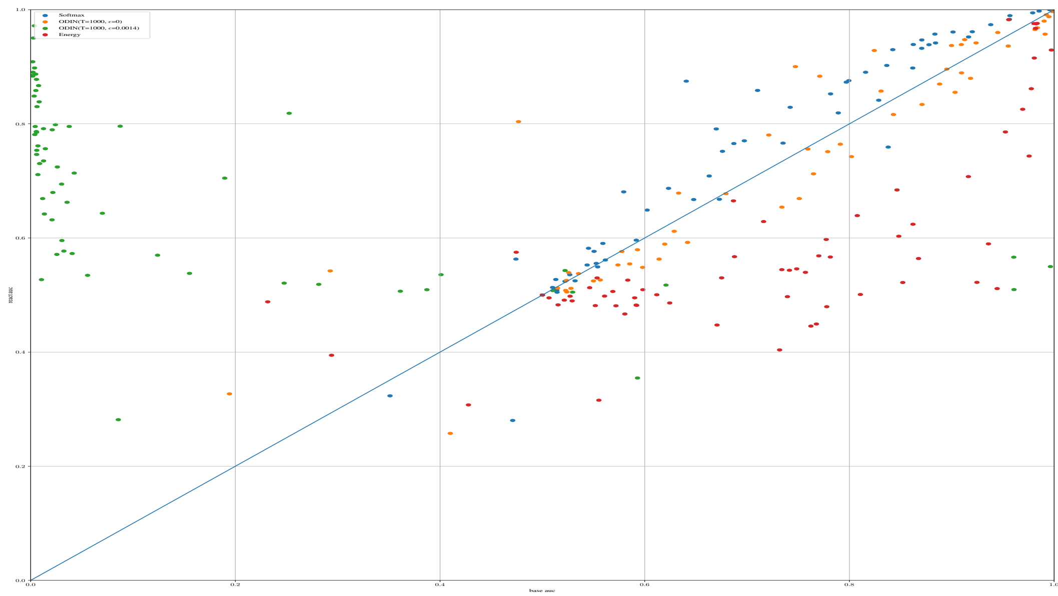


Figure 1: `AUC` score with and without the `ReAct` method across the various scoring functions. There scoring functions used are `softmax` (blue), `ODIN`($\epsilon = 0$) (orange), `ODIN`($\epsilon = 0.0014$) (green), `energy` (red).

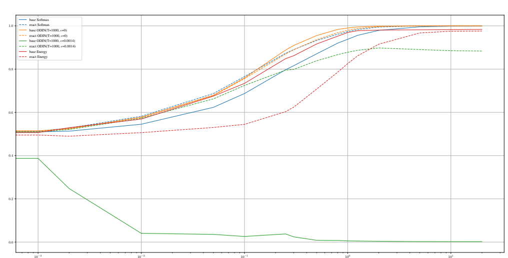


Figure 2: `AUC` score as the σ of the Gaussian noises changes. The ID dataset is ImageNet, while the OOD dataset is ImageNet with Gaussian noise.

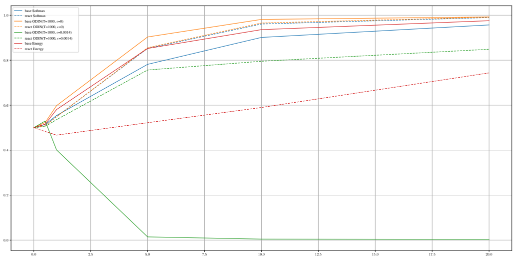
useful[2]. Consequently, when using a weak adversarial attack (as strong adversarial attacks are able to bypass this gradient masking) the `ReAct` method could bring some small benefits.

6.4. Comparison among models

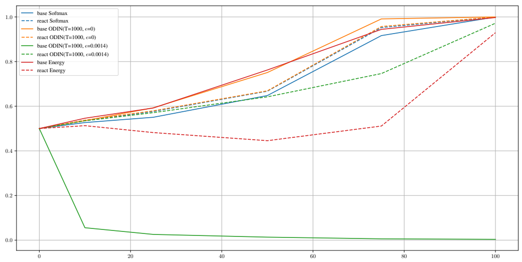
We consider all the possible combinations of model and scoring function and for every OOD dataset we compute the `AUC` score between ImageNet (the ID dataset) and the OOD dataset, then we rank the model and scoring function pairs and compute the mean ranking position for each model. Figure 4 shows that `ODIN` ($\epsilon = 0$) is overall the best method, while it is not clear whether `ReAct` is always better. Additionally, we notice no clear correlation between the OOD detection ability of the model and other model-specific proprieties, such as the number of parameters, kind of architecture, test accuracy, etc.

Model	Scorer	`ReAct`	Avg Rank
EfficientNetB0	`ODIN`($\epsilon = 0$)	no	1.22
ResNet50	`ODIN`($\epsilon = 0$)	no	1.33
DenseNet121	`ODIN`($\epsilon = 0$)	yes	1.44
DenseNet201	`ODIN`($\epsilon = 0$)	yes	1.56
VGG19	`ODIN`($\epsilon = 0$)	no	2.0
ResNet101	`softmax`	yes	2.0
ResNet101	`ODIN`($\epsilon = 0$)	no	2.11
ResNet151	`ODIN`($\epsilon = 0$)	yes	2.22
VGG16	`ODIN`($\epsilon = 0$)	no	2.22
VGG19	`ODIN`($\epsilon = 0$)	yes	2.22

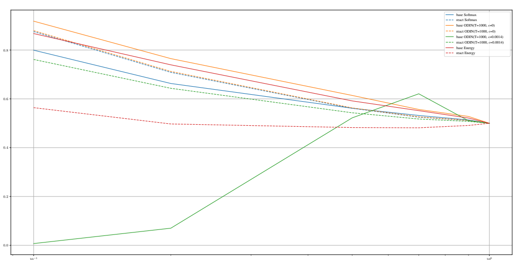
Table 4: These are the 10 best model-scoring function pairs (ranks are from 1 to 64, all the combinations of the 10 models and the 8 scoring methods)



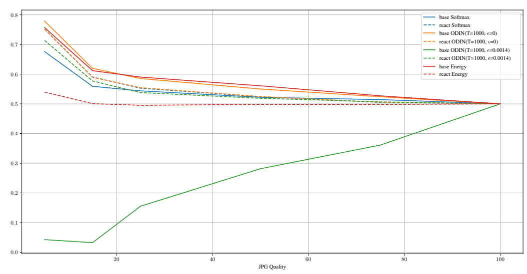
(a) Blur



(b) Prospective Trasformation



(c) pixelization



(d) JPEG quality

Figure 3: `AUC` score as a particular augmentation is used on ImageNet.

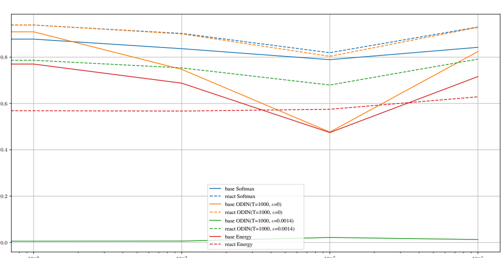


Figure 4: `AUC` score as the ϵ of the `FGSM` increases. The ID dataset is ImageNet, while the OOD dataset is iNaturalist with adversarial noise.

7. Conclusion

Overall, our analysis shows results similar to the ones present in the literature. However, compare with what the `ReAct` work suggested we found no clear performance boost and it heavily depends on many unknown and hard to analyze factors. Most of the techniques analyzed are quite robust and achieves good results even in a complex situation and are able to detect even small shift in the data distribution.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 1

[2] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples, 2018. 5

[3] J. Chen, Y. Li, X. Wu, Y. Liang, and S. Jha. Robust out-of-distribution detection for neural networks, 2020. 1

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 3

[5] X. Du, Z. Wang, M. Cai, and Y. Li. Vos: Learning what you don't know by virtual outlier synthesis, 2022. 1

[6] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, 2015. 1

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples, 2014. 3

[8] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks, 2016. 3

[9] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. 2016. 2
[10] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song. Natural adversarial examples. *arXiv preprint arXiv:1907.07174*, 2019. 3
[11] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2015. 2
[12] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data, 2020. 1
[13] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks, 2016. 3
[14] D. Karimi and A. Gholipour. Improving calibration and out-of-distribution detection in medical image segmentation with convolutional neural networks, 2020. 1
[15] S. Liang, Y. Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks, 2017. 2
[16] W. Liu, X. Wang, J. D. Owens, and Y. Li. Energy-based out-of-distribution detection, 2020. 2
[17] L. Moroney. Rock, paper, scissors dataset, feb 2019. 3
[18] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, 2014. 1
[19] Z. Papakipos and J. Bitton. Augly: Data augmentations for robustness, 2022. 3
[20] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400, 2019. 3
[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. 3
[22] Y. Sun, C. Guo, and Y. Li. React: Out-of-distribution detection with rectified activations, 2021. 1, 2, 3
[23] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2019. 3
[24] R. Taori, A. Dave, V. Shankar, N. Carlini, B. Recht, and L. Schmidt. Measuring robustness to natural distribution shifts in image classification, 2020. 1
[25] J. Yang, K. Zhou, Y. Li, and Z. Liu. Generalized out-of-distribution detection: A survey. *CoRR*, abs/2110.11334, 2021. 1

A. Full list of datasets & augmentations

The full list of datasets & augmentations used with ResNet101 (64 pairs):

ImageNet & adversarial 0.1 softmax base, ImageNet & adversarial 0.1 msoftmax base, ImageNet & adversarial 0.1 softmax100 base, ImageNet & adversarial 0.1 softmax ReAct, ImageNet & adversarial 0.1 msoftmax ReAct, ImageNet & adversarial 0.1 softmax100 ReAct, ImageNet & adversarial 0.01 softmax base, ImageNet & adversarial 0.01 msoftmax base, ImageNet & adversarial 0.01 softmax100 base, ImageNet & adversarial 0.01 softmax ReAct, ImageNet & adversarial 0.01 msoftmax ReAct, ImageNet & adversarial 0.01 softmax100 ReAct, ImageNet a , rock paper scissors , ImageNet v2 , ImageNette , iNaturalist , SUN , Places , iNaturalist & adversarial 0.1 softmax base, iNaturalist & adversarial 0.1 msoftmax base, iNaturalist & adversarial 0.1 softmax ReAct, iNaturalist & adversarial 0.1 msoftmax ReAct, gauss , uniform , ImageNet, ImageNet & gaussian 0.001, ImageNet & gaussian 0.002, ImageNet & gaussian 0.05, ImageNet & gaussian 0.1, ImageNet & gaussian 0.3, ImageNet & gaussian 0.5, ImageNet & gaussian 0.8, ImageNet & gaussian 0.25, ImageNet & gaussian 1.0, ImageNet & gaussian 1.25, ImageNet & gaussian 2.0, ImageNet & gaussian 5.0, ImageNet & gaussian 10.0, ImageNet & gaussian 20.0, ImageNet & blur 0.5, ImageNet & blur 1.0, ImageNet & blur 5.0, ImageNet & blur 10.0, ImageNet & blur 20.0, ImageNet & pixelization 0.9, ImageNet & pixelization 0.7, ImageNet & pixelization 0.5, ImageNet & pixelization 0.2, ImageNet & pixelization 0.1, ImageNet & perspectivetransform 10.0, ImageNet & perspectivetransform 25.0, ImageNet & perspectivetransform 50.0, ImageNet & perspectivetransform 75.0, ImageNet & perspectivetransform 100.0, ImageNet & encodingquality 75, ImageNet & encodingquality 50, ImageNet & encodingquality 25, ImageNet & encodingquality 15, ImageNet & encodingquality 5, iNaturalist & adversarial 0.01 softmax base, iNaturalist & adversarial 0.001 softmax base, iNaturalist & adversarial 0.0001 softmax base

The list of datasets & augmentations used for all the other models (9 pairs, that is a subset of the previous one):

imagenet, iNaturalist, SUN, Places, ImageNet V2, ImageNet & gaussian 0.002, ImageNet & gaussian 0.01, ImageNet & gaussian 0.05, ImageNet & gaussian 0.25, ImageNet & gaussian 1.25, iNaturalist & adversarial 0.01 softmax base, iNaturalist & adversarial 0.001 softmax base, iNaturalist & adversarial 0.0001 softmax base

B. How does the score distribution change?

Figures 5 shows the score distributions in various scenarios using the scores obtained by `ODIN` with $\epsilon = 0$ with and without the `ReAct` layer. The difference is quite marked even for large values of p (in our case 90%), as it reduces the variance of all the four distributions.

C. Sample Images

Figure 7 shows some sample of the images used from various augmentations and datasets.

D. Our hypothesises on why ODIN($\epsilon = 0.0014$) fails

`ODIN`($\epsilon = 0.0014$) without `ReAct` completely fails, however, if we multiply the scores by -1, it becomes one of the best methods. Unfortunately, this is true only for the version without ReAct. We were not able to explain this phenomena, our possible explanation are:

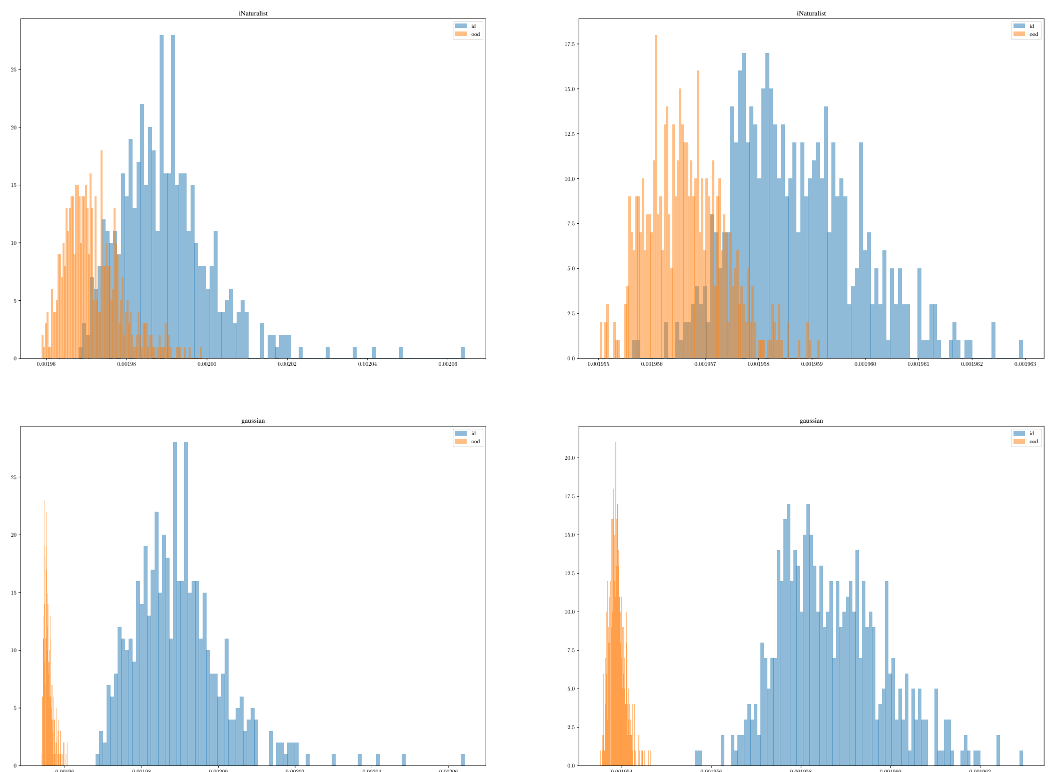
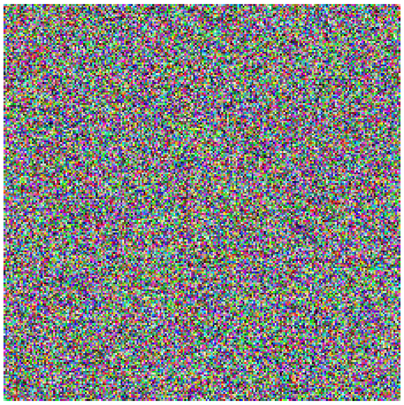


Figure 5: score distribution of the ID (light blue) and OOD dataset (orange) with the `ODIN`($\epsilon = 0$) scorer. On the left with `ReAct` on the Right without it. In the first row the OOD dataset is iNaturalist, while in the second is the gaussian noise.

- Wrong hyper-parameters choice, generally, the ϵ is tune specifically for each model.
- Implementation bug or a minor, but fundamental, implementation detail missing.
- The use of $\epsilon > 0$ is actually harmful in some situations.



(a) Gaussian noise

(b) ImageNet

(c) ImageNet with Blur (r=1.0)

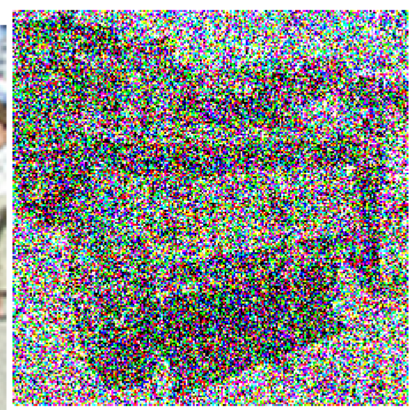
(d) ImageNet with gaussian noise ($\sigma = 0.25$)

(e) Rock Paper Scissors

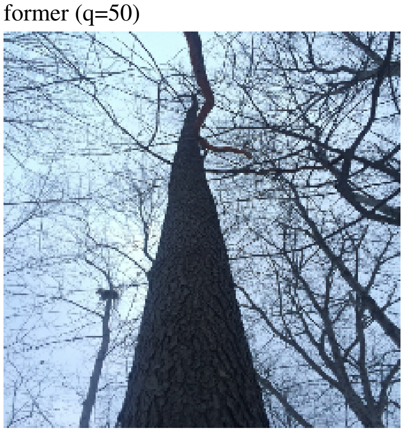
(f) ImageNet with JPEG (q=15)

(g) ImageNet with perspective transformer (q=50)

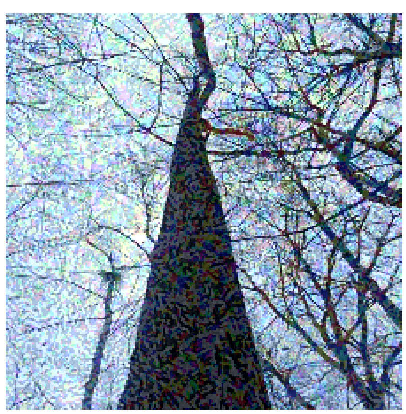
(h) ImageNet with pixelization (q=0.5)



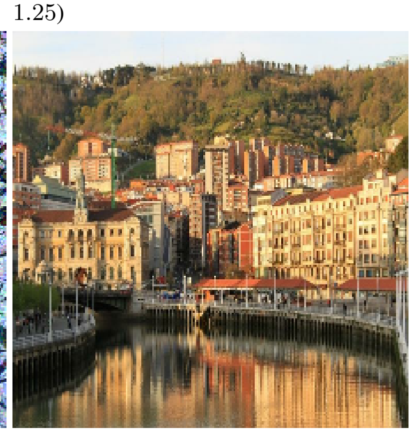
(i) ImageNet with gaussian noise ($\sigma = 1.25$)



(j) iNaturalist



(k) iNaturalist adversarial noise ($\epsilon = 0.1$)



(l) Places

Figure 6: These are the 10 best model-scoring function pairs (ranks are from 1 to 64, all the combinations of the 10 models and the 8 scoring methods)