A. Extra Results Using Different In-Distribution Datasets

We further evaluate our method on various in-distribution datasets including SVHN and CIFAR-100 in Tab. 3 using the same setting as Fig. 4. According to Fig. 4, our method achieves state-of-the-art results with various in-distribution datasets.

In-dist. (model)	OOD	Baseline [37]	ODIN [55]	Maha. [51]	Ours-LR
CIFAR-100 (ResNet-34)	SVHN	79.5	70.7	92.4	**94.2**
	LSUN-C	75.8	85.6	98.2	**99.9**
	ImageNet-C	77.2	87.8	98.0	**99.9**
SVHN (ResNet-34)	CIFAR-10	92.9	92.1	99.3	**99.8**
	LSUN-R	91.6	89.4	**99.9**	**99.9**
	ImageNet-R	93.5	92.0	**99.9**	**99.9**
CIFAR-100 (DenseNet-100)	SVHN	82.7	85.2	90.3	**93.3**
	LSUN-C	70.8	85.5	98.0	**99.9**
	ImageNet-C	71.6	84.8	94.1	**99.6**

Table 3. AUROC comparison of detection methods on various in-distribution datasets.

B. Lightweight OOD Detector

- **Logistic regression (LR)** is a kind of classic machine learning model for binary classification. Given an input vector, The LR model performs a dot product on the input with the learned coefficient vector and outputs the prediction score after applying the sigmoid function. A LR model can be trained efficiently by several solvers like LBFGS. [52] We use the default hyper-parameters in sklearn [1] for training of LR detector.

- The **multilayer perceptron (MLP)** we use consists of three full-connected layers following by non-linear activation function ReLU. We adapt dropout after the second full-connected layer and train the MLP with SGD optimizer. As a non-linear model, MLP is able to learn more complex correlation among elements in the input than LR. In practise, we find that MLP has slightly better detection performance than LR, while, with higher overfitting possibility when the number of training examples is limited. We training the MLP using SGD with 0.001 learning rate and 0.9 momentum.

The adapted OOD detector is lightweight. For instance, Our LR model has 8k parameters and 16k FLOPs, significantly smaller than the pre-trained model (*e.g.*, ResNet-34 with $\sim 2\times10^4$k parameters and $\sim 2\times10^6$k FLOPs).

C. Architecture of ConvNet

Consistent to the setting of [44], we use a simple ConvNet in Sec. 5.1 and Tab. 1. ConvNet's architecture is summarized in Tab. 4.

Layer	Configuration
Conv1	(3, 300, kernel size=4, stride=1)
Conv2	(300, 300, kernel size=4, stride=2)
Conv3	(300, 300, kernel size=4, stride=2)
Conv4	(300, 300, kernel size=3, stride=2)
AvgPool	(kernel size=2)
FC	(300, 10)

Table 4. Architecture of ConvNet following [44]. After each convolutional layer, batch normalization and ReLU layers are applied.

D. Pre-trained or Un-trained Models?

In Fig. 3, we show that the average of elements' magnitude in NMD vector from a pre-trained ResNet-34 can be used as OOD score to reliably distinguish OOD batches. Such a proof-of-concept example validates that the off-shelf-shelf pre-trained model can be used as a qualified witness function. Based on this interesting and supervising finding, we believe the off-the-shelf model itself should contain sufficient information about the training data distribution because it was trained to capture training data's features.

To further validate our hypothesis, we replace the pre-trained ResNet-34 with an un-trained ResNet-34 and re-run the experiment. As shown in Fig. 8, an un-trained ResNet-34 cannot act as a qualified witness function to detect OOD batches even the batch size is 8.

E. Neural Variance Discrepancy

As mentioned in Sec. 5.7, one can define Neural Variance Discrepancy (NVD) by computing the activation's second-order statistics in a similar manner as NMD,

$$\texttt{NVD}_c^l(\mathcal{I}) = \sqrt{\boldsymbol{\sigma}^2[f_c^l(\mathcal{I})]} - \sqrt{\boldsymbol{\sigma}^2[f_c^l(\mathcal{D}_{\texttt{tr}})]}\,, \qquad (9)$$

where the second term can be approximated by BN's running average variance. Interestingly, NVD-based detection (*i.e.*, NVD-MLP) achieves a comparable detection performance as NMD.

We further combine NVD and NMD via concatenating them together. Since elements in NVD and NMD may have different magnitude, we adopt the `standardizer` from `sklearn` to remove the mean and scale to unit variance for each dimension of NVM and NMD vectors before concatenating. Combining NMD and NVD obtains a slightly better detection result although extra computation overhead is introduced.

F. Crafting OOD Data by Pixel Permuting

As discussed in Sec. 5.3, if no OOD example is accessible, we craft artificial OOD examples by randomly permut-

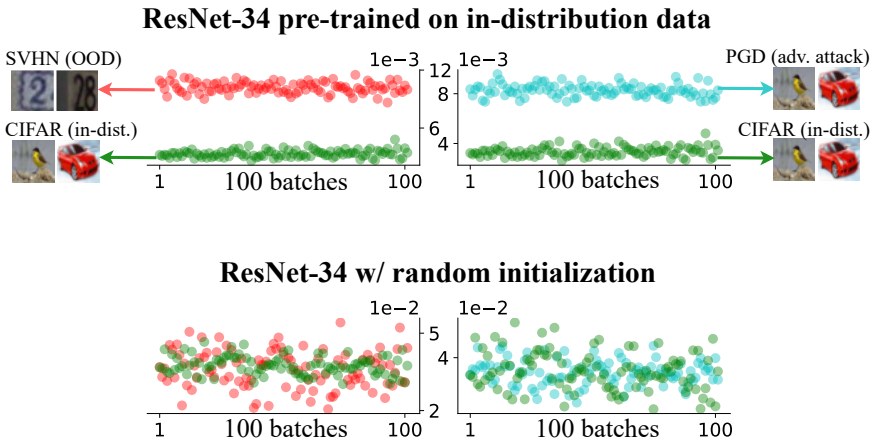

Figure 8. We redo the proof-of-concept experiment in Fig. 3 with an un-trained ResNet-34. The batch size is 8.

ing pixels of in-distribution examples and use the crafted OOD examples to guide our detector for finding the decision boundary. The premise of using crafted OOD example is that the method has high generalizability across datasets (*i.e.*, for unseen OOD data) as validated in Sec. 5.5. Specifically, we do pixel permuting in the block granularity instead of in the pixel granularity [73] to avoid tuning the hyperparameter "mutation rate". Taking CIFAR-10 example as an example, we split an image into 16 non-overlapping (8×8) blocks and randomly permute their positions. Results of detection performance without OOD examples are shown in Fig. 5 and Tab. 6.

G. Training and Inference Efficiency

In Sec. 5.6, we compare the training and inference costs of the proposed Ours-MLP with baselines as shown in Fig. 1. Training and inference time are measured on a machine with one NVIDIA GPU 1080 Ti and a Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz. Some approaches conduct model fine-tuning using MIT 80 Million Tiny Images Dataset which is not available any more. For those methods, we use the target OOD dataset (*i.e.*, CIFAR-100 training set) to do fine-tuning but with the same number of iterations as using MIT 80 Million Tiny Images Dataset. For methods which require repeating experiments for several times to search hyper-parameters, we count all such time into training time. To measure the inference latency, we repeat single example detection for 10,000 times and compute the average inference time for a single example.

A recent study, MOOD [56], achieves state-of-the-art inference efficiency leveraging early exiting [3]. We do not include MOOD in Tab. 5 because it depends a special architecture with dynamic exits. In addition, our method is orthogonal to MOOD and could be combined for a future work as discussed in Secs. 5.7 and 7.

Method	Fine-tuning	Training	Inference
Gram	True	330s	0.37s
Maha	False	1397s	25.7ms
ODIN	False	1270s	16.0ms
G-ODIN	True	1830s	22.1ms
OE	True	560s	6.72ms
GOOD	True	756m	47.4ms
ACET	True	201m	6.89ms
Energy-FT	True	620s	7.24ms
Plain ResNet-34	-	-	6.72ms
Ours-MLP	False	94s	7.54ms

Table 5. Training and inference time comparison with CIFAR-10against CIFAR-100 (OOD) detection on ResNet-34. (Also see Fig. 1)

In-dist (model)	OOD	Energy (w/o FT)	Gram (w/o FT)	G-ODIN (w/ FT)	1D (w/ FT)	Ours-MLP (w/o FT)
		TNR at TPR 95% / AUROC / Detection acc.				
CIFAR-10 (ResNet-34)	iSUN	60.4 / 92.2 / 87.0	99.3 / 99.8 / 98.1	95.3 / 98.9 / 95.6	76.9 / 86.3 / 92.9	**99.7 / 99.9 / 98.6**
	SVHN	58.4 / 90.6 / 85.5	97.6 / 99.5 / **96.7**	89.5 / 97.8 / 92.9	86.2 / 95.1 / 88.9	**97.7** / **99.6** / 96.6
	Texture	41.1 / 85.5 / 80.8	88.0 / 97.5 / 91.9	81.4 / 95.0 / 88.9	72.4 / 91.1 / 84.9	**94.0 / 98.9 / 94.6**
	LSUN-C	89.2 / 98.0 / 93.8	89.8 / 97.8 / 92.6	**93.9 / 98.8** / 94.0	77.1 / 92.9 / 86.5	**93.9 / 98.8 / 94.5**
	ImageNet-C	67.4 / 93.6 / 88.7	**96.7** / **99.2** / **96.1**	90.8 / 98.2 / 94.3	81.9 / 94.6 / 88.5	96.1 / **99.2** / 95.6
	CIFAR-100	43.1 / 87.1 / 80.7	32.9 / 79.0 / 71.7	36.3 / 85.5 / 79.3	57.4 / 87.2 / 80.8	**63.8 / 90.1 / 83.4**

Table 6. Comparison of detection methods when only in-distribution dataset is accessible. (Also see Fig. 5)