Variational- and Metric-based Deep Latent Space for Out-of-Distribution Detection – Supplementary Material

Or Dinari[1] **Oren Freifeld**[1]

[1]The Department of Computer Science, Ben-Gurion University of the Negev, Be'er Sheva, Israel

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

Abstract

This document contains the following: 1) the mathematical formulation of the KL loss; 2) the technical details of our experiments, including the values of the hyperparameters that we used, and the details for the training of the deep nets; 3) results on each benchmark in the OOD-detection task (in the paper we reported only the average results across those benchmarks); 4) empirical evaluation of different features for the reconstruction weighting; 5) details regarding the datasets used in the paper; 6) implementation and hardware details; 7) an intuitive explanation for the effect of the overall proposed loss; 8) details of the auto-encoders used for the reconstruction step.

1 THE GAUSSIAN CLASS-CONDITIONED KL DIVERGENCE LOSS

Let $q = \mathcal{N}(\mu_1, \sigma_1^2)$ and $p = \mathcal{N}(\mu_2, \sigma_2^2)$ be two univariate Gaussian distributions and let $D_{\mathrm{KL}}(\cdot||\cdot)$ denote the KL divergence. Then:

$$
\begin{aligned}
D_{\mathrm{KL}}(q||p) &= \int q(z) \log\left(\frac{q(z)}{p(z)}\right) dz && (1)\\
&= \int \left[\log(q(z)) - \log(p(z))\right] q(z)\, dz && (2)\\
&= \int \left[-\frac{1}{2}\log(2\pi) - \log(\sigma_1) - \frac{1}{2}\left(\frac{z-\mu_1}{\sigma_1}\right)^2 + \frac{1}{2}\log(2\pi) + \log(\sigma_2) + \frac{1}{2}\left(\frac{z-\mu_2}{\sigma_2}\right)^2\right] \\
&\quad \times \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left[-\frac{1}{2}\left(\frac{z-\mu_1}{\sigma_1}\right)^2\right] dz && (3)\\
&= \int \left\{\log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{1}{2}\left[\left(\frac{z-\mu_2}{\sigma_2}\right)^2 - \left(\frac{z-\mu_1}{\sigma_1}\right)^2\right]\right\} \times \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left[-\frac{1}{2}\left(\frac{z-\mu_1}{\sigma_1}\right)^2\right] dz && (4)\\
&= E\left\{\log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{1}{2}\left[\left(\frac{z-\mu_2}{\sigma_2}\right)^2 - \left(\frac{z-\mu_1}{\sigma_1}\right)^2\right]\right\} \qquad \text{(the expectation is w.r.t. } q \text{)} && (5)\\
&= \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{1}{2\sigma_2^2} E\left\{(X-\mu_2)^2\right\} - \frac{1}{2\sigma_1^2} E\left\{(X-\mu_1)^2\right\} && (6)\\
&= \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{1}{2\sigma_2^2} E\left\{(X-\mu_2)^2\right\} - \frac{1}{2\sigma_1^2}\sigma_1^2 && (7)\\
&= \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{1}{2\sigma_2^2} E\left\{(X-\mu_2)^2\right\} - \frac{1}{2} && (8)\\
&= \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{1}{2\sigma_2^2}\left[E\left\{(X-\mu_1)^2\right\} + 2(\mu_1-\mu_2)\underbrace{E\left\{X-\mu_1\right\}}_{0} + (\mu_1-\mu_2)^2\right] - \frac{1}{2} && (9)\\
&= \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{\sigma_1^2 + (\mu_1-\mu_2)^2}{2\sigma_2^2} - \frac{1}{2}. && (10)
\end{aligned}
$$

Now recall our KL loss term is

$$\mathcal{L}_{\mathrm{KL}}(\boldsymbol{x}_i) = -D_{\mathrm{KL}}(q(\cdot|\boldsymbol{g}(\boldsymbol{x}_i))||p(\cdot|y_i)) \tag{11}$$

where $q(\cdot|\boldsymbol{g}(\boldsymbol{x}_i))$ is a d-dimensional Gaussian probability density function (pdf) with a mean vector $\boldsymbol{\mu}(\boldsymbol{g}(\boldsymbol{x}_i))$ and a diagonal covariance matrix whose (j, j) entry is $\sigma_j^2(\boldsymbol{g}(\boldsymbol{x}_i))$ while $p(\cdot|y_i)$ is an isotropic d-dimensional Gaussian pdf, associated with class y_i, with a mean vector $\boldsymbol{m}(y_i) = (m_1(y_i), \ldots, m_d(y_i))$ and variance s^2. Note that both $\boldsymbol{q}$ and $\boldsymbol{p}$ are d-dimensional multivariate Gaussians and each of them has a diagonal covariance matrix. It follows that

$$D_{\mathrm{KL}}(q(\cdot|\boldsymbol{g}(\boldsymbol{x}_i))||p(\cdot|y_i)) = \sum\nolimits_{j=1}^{d} D_{\mathrm{KL}}(q_j(\cdot|\boldsymbol{g}(\boldsymbol{x}_i))||p_j(\cdot|y_i)) \tag{12}$$

Table 1: Experiments hyperparameters.

Dataset	Backbone	d	k	M_d	M_t	Optimizer	B_c	B_n	LR	Finetune LR
CIFAR-10	VGG [Yoshihashi et al., 2019]	32	1024	32	0.1	Adam	8	40	0.001	N/A
CIFAR-10	ResNet18 [He et al., 2016]	32	512	32	0.1	SGD	10	40	0.01	N/A
CIFAR-10	ResNet34 [He et al., 2016]	32	512	32	0.1	SGD	10	8	0.005	N/A
CIFAR-10	DenseNet-BC100 [Huang et al., 2017]	32	342	32	0.1	SGD	10	40	0.1	N/A
CIFAR-10	WideResnet28 [Zagoruyko and Komodakis, 2016]	32	640	32	0.1	SGD	10	20	0.001	0.01
CIFAR-100	DenseNet-BC100 [Huang et al., 2017]	64	342	64	0.1	SGD	20	20	0.0001	0.01
MNIST	CNN [Yoshihashi et al., 2019]	32	500	32	0.1	Adam	10	64	0.0001	N/A

where $q_j(\cdot|\boldsymbol{g}(\boldsymbol{x}_i))$ and $p_j(\cdot|y_i)$ are the univariate Gaussians $\mathcal{N}(\mu_j(\boldsymbol{g}(\boldsymbol{x}_i)), \sigma_j^2(\boldsymbol{g}(\boldsymbol{x}_i)))$ and $\mathcal{N}(m_j(y_i), s^2)$, respectively. Putting it altogether, we obtain that:

$$\begin{aligned} &D_{\mathrm{KL}}(q(\cdot|\boldsymbol{g}(\boldsymbol{x}_i))||p(\cdot|y_i)) \\ &= \sum_{j=1}^{d} \log\left(\frac{s}{\sigma_j(\boldsymbol{g}(\boldsymbol{x}_i))}\right) + \frac{\sigma_j^2(\boldsymbol{g}(\boldsymbol{x}_i)) + (\mu_j(\boldsymbol{g}(\boldsymbol{x}_i)) - m_j(y_i))^2}{2s^2} - \frac{1}{2}\,. \end{aligned} \tag{13}$$

Finally, using the above term in the loss function results in the following expression:

$$\begin{aligned} \mathcal{L}_{\mathrm{KL}}(\boldsymbol{x}_i) &= -D_{\mathrm{KL}}(q(\cdot|\boldsymbol{g}(\boldsymbol{x}_i))||p(\cdot|y_i)) \\ &= \sum_{j=1}^{d} \left[-\log\left(\frac{s}{\sigma_j(\boldsymbol{g}(\boldsymbol{x}_i))}\right) - \frac{\sigma_j^2(\boldsymbol{g}(\boldsymbol{x}_i)) + (\mu_j(\boldsymbol{g}(\boldsymbol{x}_i)) - m_j(y_i))^2}{2s^2} + 0.5\right]. \end{aligned} \tag{14}$$

2 EXPERIMENTS DETAILS

Each of the well-known datasets, CIFAR10 and MNIST, comes with a partition to training data and test data. For our training we have only used the training data. The test data was used for only evaluation.

hyperparameters. All the training hyperparameters appear in Table 1. Recall, from the paper, that we trained our model on each of the following pairs of dataset/backbone:

1. MNIST & CNN;
2. CIFAR10 & VGG;
3. CIFAR10 & ResNet18;
4. CIFAR10 & ResNet34;
5. CIFAR10 & DenseNet-BC100;
6. CIFAR10 & WideResnet28.
7. CIFAR100 & DenseNet-BC100.

Moreover, and as was also stated in the paper, in five of these cases we trained the model from scratch, while only in the (CIFAR100, DenseNet-BC100) and (CIFAR10, WideResnet28) pairs we merely fine-tuned a pre-trained Deep Neural Net (DNN). Thus, in Table 1, the learning rate (LR) for the fine tuning appears only in the row that corresponds to such a pair.

We emphasize, and as was mentioned in the main text, that in our method (regardless if we train a model from scratch or if take a pre-trained model and fine tune it for our tasks) there are no parameters that are learned/tuned using OOD data. Particularly, the only hyperparameters are standard training parameters, as described in Table 1, while the weights of the DNN are learned using only data from the known classes, not OOD data.

2.1 TRAINING DETAILS FOR THE FIVE CASES WHERE THE TRAINING WAS DONE FROM SCRATCH

In these experiments we used M_d warm-up rounds (where the value of M_d appears in Table 1), where we gradually increased the distancing margin from 0.15 to M_d. While this warm-up stage is optional, we found that it increases the stability of the training. In addition, we have started with $B'_s = B_s/5$ for the first 150 epochs. This step too is optional, but it improves the convergence speed. We reduced the LR by a factor of $\frac{1}{10}$ every 100 epochs. The total number of epochs was 500 epochs.

Table 2: ODIN [Liang et al., 2018] OOD Benchmarks

	CIFAR-10		CIFAR-100	
	AUROC	*TNR@TPR95*	*AUROC*	*TNR@TPR95*
ImageNet-Crop	0.991	0.957	0.945	0.731
ImageNet-Resize	0.985	0.925	0.855	0.430
LSUN-Crop	0.979	0.962	0.968	0.814
LSUN-Resize	0.992	0.937	0.871	0.420

Table 3: ODIN* [Hsu et al., 2020] OOD Benchmarks

In-distribution	CIFAR-10		CIFAR-100	
OOD	*AUROC*	*TNR@TPR95*	*AUROC*	*TNR@TPR95*
ImageNet-Crop	0.882	0.478	0.905	0.560
ImageNet-Resize	0.901	0.519	0.911	0.594
LSUN-Crop	0.913	0.635	0.899	0.530
LSUN-Resize	0.923	0.592	0.930	0.640

2.2 TRAINING DETAILS FOR THE TWO CASES WHERE WE FINE-TUNED A PRE-TRAINED DNN

For the fine-tuning experiments (*i.e.*, CIFAR-100 with the DenseNet backbone and CIFAR-10 with the WideResnet28 backbone), we have loaded the pre-trained weights from [Liang et al., 2018] for CIFAR100, or first trained a SoftMax-based classifier for CIFAR10 and then loaded its weights. Then, using the 'Finetune LR' (Table 1) we have trained the $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ layers for (only) 20 epochs. Next, we unfroze the entire network, and continued training for additional 130 epochs, with two LR decreases, each after 64 epochs, starting from 'LR' (Table 1).

2.3 DATA AUGMENTATION

We have used standard random augmentations during training. Specifically, for the MNIST experiments we have used random rotations and random resized crops. For the CIFAR experiments we have used random crops and random horizontal flips.

2.4 FEATURE ENSEMBLES

As noted in the main text, for block-based networks (such as DenseNet or ResNet) we used the output of the first three blocks for (t_1, t_2, t_3). For the non-block based networks that we used in the experiments (*i.e.*, VGG and the plain CNN), we have chosen the following features: For VGG, we have taken the outputs of the second, fourth, and eight convolution layer, each after Batch Norm and ReLU. For the plain CNN, we have used the outputs of the second, third and fourth convolution layers.

3 OOD BENCHMARKS

In this section we provide the full results of the OOD benchmarks. The results for each method are shown in its own table: ODIN in Table 2; ODIN* in Table 3; Mahalanobis in Table 4; Mahalanobis* in Table 5; DeConf-C in Table 6; CSI in Table 7; SubSpaces in Table 8; **VDMLS**$_b$ (ours) in Table 9; **VDMLS** (ours) in Table 10; Each such table shows the results, where the in-distribution data is either CIFAR10 or CIFAR100, using one of four different OOD datasets. The reported numbers in these seven tables stand for the macro-average results across those four OOD datasets.

Table 4: Mahalanobis [Lee et al., 2018] OOD Benchmarks

	CIFAR-10		CIFAR-100	
	AUROC	*TNR@TPR95*	*AUROC*	*TNR@TPR95*
ImageNet-Crop	0.998	0.997	0.996	0.998
ImageNet-Resize	0.988	0.952	0.974	0.866
LSUN-Crop	0.996	0.996	0.993	0.982
LSUN-Resize	0.992	0.973	0.982	0.914

Table 5: Mahalanobis* [Hsu et al., 2020] OOD Benchmarks

In-distribution	CIFAR-10		CIFAR-100	
OOD	*AUROC*	*TNR@TPR95*	*AUROC*	*TNR@TPR95*
ImageNet-Crop	0.963	0.812	0.924	0.635
ImageNet-Resize	0.982	0.909	0.964	0.820
LSUN-Crop	0.922	0.642	0.812	0.316
LSUN-Resize	0.982	0.917	0.966	0.826

Table 6: DeConf-C [Hsu et al., 2020] OOD Benchmarks

	CIFAR-10		CIFAR-100	
	AUROC	*TNR@TPR95*	*AUROC*	*TNR@TPR95*
ImageNet-Crop	0.987	0.934	0.976	0.878
ImageNet-Resize	0.991	0.958	0.986	0.933
LSUN-Crop	0.983	0.915	0.953	0.750
LSUN-Resize	0.994	0.976	0.987	0.938

4 EMPIRICAL EVALUATION OF THE RECONSTRUCTION FEATURES

deep-level features. In Figure 1, we can see not only that the weighting based on reconstruction of low-level features (b) is better (*i.e.*, achieves better separation) using either no weighting (a) or weighting based on reconstruction of deep-level features (c) but also that the weighting based on reconstruction of deeper features deteriorates the performance (*i.e.*, is even worse than using no weighting). Thus the choice of low-level features is the optimal for our use case.

5 DATASETS

In the paper we have used several datasets:

- ImageNet [Deng et al., 2009], which contains a large set of images of various categories. The curators of ImageNet do not hold the copyright of all the images, and the usage of that dataset is governed by the terms of the ImageNet license https://www.image-net.org/download.php.
- CIFAR10 and CIFAR100 [Krizhevsky, 2009] (datesets of 32 x 32 RGB images).
- LSUN [Yu et al., 2015].
- MNIST [LeCun, 1998] (an image dataset of handwritten digits in which each instance consists of a 28x28 gray-scale image).
- Omniglot [Lake et al., 2015], under the MIT license.

.

Table 7: CSI [Tack et al., 2020] OOD Benchmarks

	CIFAR-10		CIFAR-100	
	AUROC	*TNR@TPR95*	*AUROC*	*TNR@TPR95*
ImageNet-Crop	0.982	0.900	0.966	0.805
ImageNet-Resize	0.978	0.875	0.839	0.284
LSUN-Crop	0.987	0.944	0.974	0.849
LSUN-Resize	0.978	0.870	0.871	0.388

Table 8: SubSpaces [Zaeemzadeh et al., 2021] OOD Benchmarks

	CIFAR-10		CIFAR-100	
	AUROC	*TNR@TPR95*	*AUROC*	*TNR@TPR95*
ImageNet-Crop	0.981	0.910	0.891	0.583
ImageNet-Resize	0.985	0.924	0.940	0.706
LSUN-Crop	0.994	0.972	0.936	0.612
LSUN-Resize	0.993	0.966	0.960	0.797

6 IMPLEMENTATION AND HARDWARE DETAILS

6.1 IMPLEMENTATION DETAILS

We have implemented our model using PyTorch [Paszke et al., 2019], with the aid of several useful frameworks: PyTorch-Lighting [Falcon, 2019], PyTorch Metric Learning [Musgrave et al., 2020]. For the DenseNet [Huang et al., 2017] implementation, we have modified the model implementation from https://github.com/andreasveit/densenet-pytorch. For the WideResnet28 [Zagoruyko and Komodakis, 2016] implementation, we have modified the model implementation from [Zaeemzadeh et al., 2021].

6.2 HARDWARE DETAILS

All our experiments were done using a single NVIDIA *Tesla P100* card, with the exception of the WideResNet experiments, which were done using a single NVIDIA *RTX3090* card.

7 AN INTUITIVE EXPLANATION FOR THE EFFECT OF THE OVERALL PROPOSED LOSS

Recall, *e.g.*, from Figure 4d in the paper, that the empirical effect of the overall loss is not only creating small isotropic Gaussians that are far from each other but also forming a large empty area between them. In this section we provide some intuition behind this behavior which was consistent throughout our experiments.

For simplicity, suppose that there are only 5 classes and that each class consists of a single example. Let us denote the examples, as represented in a 2D latent space, by $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, $\boldsymbol{\mu}_3$, $\boldsymbol{\mu}_4$ and $\boldsymbol{\mu}_5$. Now consider the following configuration. $\boldsymbol{\mu}_1 = (-1, 0)$, $\boldsymbol{\mu}_2 = (0, -1)$, $\boldsymbol{\mu}_3 = (1, 0)$, $\boldsymbol{\mu}_4 = (0, 1)$, and $\boldsymbol{\mu}_5 = (0, 0)$. Here, $\boldsymbol{\mu}_5$ is at a relatively-small distance from each of the other 4 points. Moving $\boldsymbol{\mu}_5$ elsewhere would thus yield an improvement in terms of the distancing loss. In fact, a simple calculation (together with the fact that the hypotenuse is the longest side of a right-angled triangle) will show that moving $\boldsymbol{\mu}_5$ by some small epsilon in *any* direction from $(0, 0)$ will improve (in effect, decrease) the distancing loss. Thus, the central region between the clusters remains effectively empty and the points will move away from it.

Note however, that in our case all clusters are initialized in the center of the dataset (assuming random intialization of the latent space), thus they move simultaneously outwards, pushing away from the center of the latent space and in different directions to maximize the inter-cluster distances (if two clusters were to move together in the same direction, they

Table 9: **VMDLS**$_b$ OOD Benchmarks

	CIFAR-10		CIFAR-100	
	AUROC	*TNR@TPR95*	*AUROC*	*TNR@TPR95*
ImageNet-Crop	0.989	0.999	0.997	0.998
ImageNet-Resize	0.961	0.953	0.949	0.748
LSUN-Crop	0.991	0.999	0.996	0.999
LSUN-Resize	0.967	0.972	0.961	0.768

Table 10: **VMDLS** OOD Benchmarks

In-distribution	CIFAR-10		CIFAR-100	
OOD	*AUROC*	*TNR@TPR95*	*AUROC*	*TNR@TPR95*
ImageNet-Crop	0.997	0.999	0.999	0.999
ImageNet-Resize	0.985	0.959	0.987	0.946
LSUN-Crop	0.991	0.999	0.999	0.999
LSUN-Resize	0.988	0.973	0.994	0.973

would be too close to each other, incurring a penalty). Effectively, this creates a sphere-like shape (empirically, the center of that sphere tends to coincide with the origin of the latent space). For example, when the five points are placed on a circle in some radius at the angles (in degrees) $\{0, 72, 144, 216, 288\}$ then each point is close to only two other points, while its distances from the others are larger. Thus, in terms of the distancing loss the penalty is smaller when compared to the aforementioned original configuration.

Now, in practice cluster will of course usually contain many points, not just a single one. However, the KL loss pushes each cluster to be small and isotropic, and thus effectively behaving like a single point so the informal analysis above still holds. In contrast, without the KL loss, and due to the varying and elongated shapes of the clusters, the metric losses by themselves do not suffice for obtaining that effect.

8 DETAILS OF THE AUTO-ENCODERS

During the test phase of **VMDLS**, we use a pre-trained Auto-Encoder to reconstruct the images, and use the difference between the shallow features of the reconstructed image and the original one, in order to use it as weighting during the likelihood-based decision. The Auto-Encoders we have used are fairly simple and standard: For CIFAR10 and CIFAR100 experiments we have used a ResNet18-based AE, as implemented by [Falcon, 2019]. For the MNIST experiment we have used a simple convolutional AE, which consists of 5 convolution layers, each followed by a ReLU activation function and a pooling layer. The exact implementation of the AEs is available in our publicly-available code.

References

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

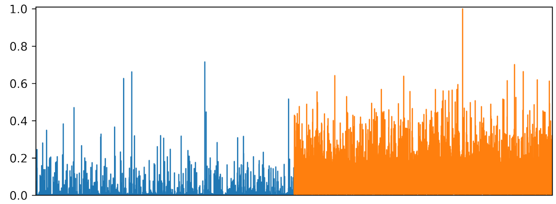
WA Falcon. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

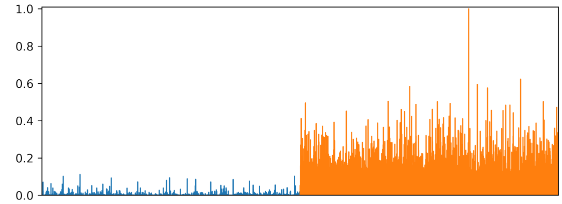
Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. Generalized ODIN: Detecting out-of-distribution image without learning from out-of-distribution data. In *CVPR*, 2020.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

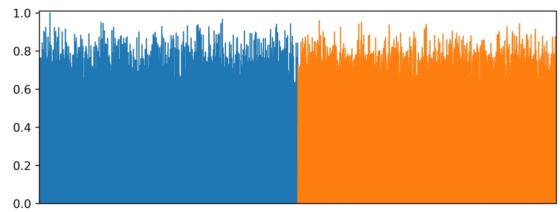
Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.



(a) Unmodified values (**VMDLS**$_b$)



(b) Low-level features (**VMDLS**)



(c) Deep-level features

Figure 1: Negative Log Likelihood (scaled to one) without (a), with (b) weighting based on the low-level features and with (c) weighting based on the deep-level features. In-distribution: CIFAR-10 (blue). OOD: ImageNet-resize (orange). Note the better separation in (b).

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *NeurIPS*, 2018.

Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR*, 2018.

Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. Pytorch metric learning, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. Csi: Novelty detection via contrastive learning on distributionally shifted instances. In *Advances in Neural Information Processing Systems*, 2020.

Ryota Yoshihashi, Wen Shao, Rei Kawakami, Shaodi You, Makoto Iida, and Takeshi Naemura. Classification-reconstruction learning for open-set recognition. In *CVPR*, 2019.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv*, 2015.

Alireza Zaeemzadeh, Niccolò Bisagno, Zeno Sambugaro, Nicola Conci, Nazanin Rahnavard, and Mubarak Shah. Out-of-distribution detection using union of 1-dimensional subspaces. In *CVPR*, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv*, 2016.