Block Selection Method for Using Feature Norm in Out-of-distribution Detection (Supplementary Material)

1 Details of the training and in-distribution accuracy

We utilize the ResNet18(He, Zhang, Ren, & Sun, 2016), WRN28(Zagoruyko & Komodakis, 2016), VGG11(Simonyan & Zisserman, 2014). Here, we provide details of training those architectures on CIFAR10(Krizhevsky, Hinton, et al., 2009).

1.1 Training for CIFAR10

To train the model for cifar10, we use the following setting for each architecture:

- **ResNet18**: We use the same hyperparameter setting as in Sun, Guo, and Li (2021). The model is trained for 100 epochs using the SGD optimizer with a momentum of 0.9 and weight decay of 5×10^{-4}. The initial learning rate is 0.1 and decays by a factor of 10 at epochs 50, 75, and 90. Also, we use batch size 128.
- **WRN28**: We use the same hyperparameter setting as in Hsu, Shen, Jin, and Kira (2020). The model is trained for 200 epochs using the SGD optimizer with a momentum of 0.9, weight decay of 5×10^{-4}, and 128 batch size. The initial learning rate is 0.1 and decays by a factor of 10 at epochs 100, 150.
- **VGG11**: The model is trained for 100 epochs using the SGD optimizer with a momentum of 0.9 and weight decay of 5×10^{-4}. The initial learning rate is 0.05 and decays by a factor of 10 at epochs 50, 75, and 90. Also, we use batch size 128.

The in-distribution accuracy of the models trained for CIFAR10 is tabulated in Table 1:

Iteration	ResNet18	WRN28	VGG11
1	94.62	95.73	91.08
2	94.54	96.07	90.96
3	94.96	95.84	90.72
4	94.29	95.98	90.89
5	94.37	96.03	91.15
Average	94.556	95.93	90.96

Table 1: In-distribution accuracy of models trained for CIFAR10.

1.2 Training for ImageNet

We utilize the model for ImageNet provided by torchvision of pytorch. For ResNet50(He et al., 2016), the model is trained for 90 epochs using the SGD optimizer with a momentum of 0.9 and weight decay of 1×10^{-4}. The initial learning rate is 0.1 and decays by a factor of 10 at epochs 30, 60. Batch size of 32 is used. Also, VGG16(Simonyan & Zisserman, 2014) use the same setting except the initial learning rate is 0.01. Finally, the MobileNetV3(Howard et al., 2019) is trained for 600 epochs using the RMSprop optimizer with a weight decay of 0.00001. The initial learning rate is 0.064 and decays by 0.973 at every 2 epochs. The more training details can be found at: `https://pytorch.org/vision/stable/index.html`. In particular, we use the torchvision version of 0.12.0

2 Details of the baseline

We compare our method with OOD detection methods: MSP(Hendrycks & Gimpel, 2017), ODIN(Liang, Li, & Srikant, 2018), Energy(Liu, Wang, Owens, & Li, 2020), Energy+ReAct(Sun et al., 2021), Energy+DICE(Macêdo, Ren, Zanchettin, Oliveira, & Ludermir, 2021), The details of the methods are as follows:

- **MSP**. Hendrycks and Gimpel (2017) propose to use maximum softmax probability to detect OOD samples. Then, the in-distribution (ID) score s is calculated as follows:

$$s = \max_i \frac{exp(v_i)}{\sum_k^K exp(v_k)},$$

  where v_i, K refer to the i-th logit, the number of classes, respectively.

- **ODIN**. Lian Liang et al. (2018) propose to use maximum softmax probability to detect OOD samples with temperature scaling and input perturbation. In all experiments, we set the temperature scaling parameter $T = 1000$ and the perturbation parameter $\epsilon = -0.0004$. Therefore, the ID score s is calculated as follows:

$$s = \max_i \frac{exp(v_i/T)}{\sum_k^K exp(v_k/T)},$$

  where v_i refers to the i-th logit that calculated with perturbated input.

- **Energy**. Liu et al. (2020) propose to use energy score to detect OOD samples. The energy function calculate energy from logit as follows:

$$s = -\log \sum_k^K exp(v_i(x)).$$

  In the experiments, Energy is utilized using the standard network.

- **Energy+ReAct**. Sun et al. (2021) propose to use energy score or maximum softmax probability to detect OOD samples with clipped activation using ReAct operation. The ReAct operation calculate the clipped activation f^{ReAct} as follows:

$$f^{ReAct} = min(f_i, c),$$

  where f_i, c refer to the i-th element of original feature vector f and activation truncation threshold. After the ReAct operation is calculated, energy score is calculated with clipped activation f^{ReAct}. In the experiments, Performance of the ReAct is chosen as the best result between the 70% clip or 90% clip.

- **Energy+DICE** Sun and Li (2022) propose to use energy score with output that is derived by selectively using the most salient weight. They measure the saliency of the feature vector by define a contribution matrix $\mathbf{V} \in \mathbb{R}^{m \times C}$, where each column $\mathbf{v}_c$ is given by:

$$\mathbf{v}_c = \mathbb{E}_{x \in D}[\mathbf{w}_c \odot h(x)],$$

  where $\odot$ refers to the element-wise multiplication, and $\mathbf{w}_c$ refers to the c-th class weight vector. Also, m and (x) refer to the feature dimension and feature vector. Then, an element of V intuitively measures the contribution of the elements of the feature vector to each class. Finally, top-k weights can be selected based on the k-largest elements in V. Then, the model output is calculated by

$$f^{DICE}(x; \theta) = (M \odot W)^\top h(x),$$

  where M is a masking matrix, which returns a matrix by setting 1 for entries corresponding to the k largest elements in $\mathbf{V}$ and setting other elements to 0. There is hyperparameter p which is used for how much elements is remained for calculating output. As well as done in Energy+ReAct, we select the best result between results of $p = 70\%$ and $p = 90\%$.

3 Details of the block selection

In this section, we provide the details about the block selection method using the NormRatio calculated by the training images and the pseudo images generated from training images. First, we provide the description and the NormRatio of the block for each architecture in Table 2, 3, 4, 5, 6, and 7. The NormRatio we provided is calculated using the Jigsaw puzzle images as pseudo OOD.

3.1 Summary of the block for each architecture

Block Num	Channel dimension	Output size	Structure	NormRatio
1	64	32x32	[Conv-BN-ReLU] x 2	0.83
2			[Conv-BN-ReLU] x 2	0.85
3	128	16x16	Shortcut+[Conv-BN-ReLU] x 2	0.81
4			[Conv-BN-ReLU] x 2	0.85
5	256	8X8	Shortcut+[Conv-BN-ReLU] x 2	0.93
6			[Conv-BN-ReLU] x 2	1.02
7	512	4x4	Shortcut+[Conv-BN-ReLU] x 2	**1.11**
8			[Conv-BN-ReLU] x 2	0.97

Table 2: Summary of the block in ResNet18 that trained for CIFAR10. The NormRatio is averaged over 5 runs.

Block Num	Channel dimension	Output size	Structure	NormRatio
1	160	32x32	[BN-ReLU-Conv]x2	0.83
2			[BN-ReLU-Conv]x2	0.87
3			[BN-ReLU-Conv]x2	0.89
4			[BN-ReLU-Conv]x2	0.90
5	320	16x16	Shortcut + [BN-ReLU-Conv]x2	0.95
6			[BN-ReLU-Conv]x2	0.94
7			[BN-ReLU-Conv]x2	0.98
8			[BN-ReLU-Conv]x2	0.99
9	640	8X8	Shortcut + [BN-ReLU-Conv]x2	1.04
10			[BN-ReLU-Conv]x2	1.16
11			[BN-ReLU-Conv]x2	**1.36**
12			[BN-ReLU-Conv]x2	1.28

Table 3: Summary of the block in WRN28 that trained for CIFAR10. The NormRatio is averaged over 5 runs.

Block Num	Channel dimension	Output size	Structure	NormRatio
1	64	32x32	Conv-ReLU	0.87
2	128	16x16	MaxPool-Conv-ReLU	0.85
3	256	8x8	MaxPool-Conv-ReLU	0.98
4			Conv-ReLU	0.94
5	512	4x4	MaxPool-Conv-ReLU	1.47
6			Conv-ReLU	**1.77**
7	512	2x2	MaxPool-Conv-ReLU	1.56
8			Conv-ReLU	1.48

Table 4: Summary of the block in VGG11 that trained for CIFAR10. The NormRatio is averaged over 5 runs.

Block Num	Channel dimension	Output size	Structure	NormRatio
1	256	56x56	Shortcut+[Conv-BN-ReLU] x 3	0.98
2			[Conv-BN-ReLU] x 3	0.98
3			[Conv-BN-ReLU] x 3	0.98
4	512	28x28	Shortcut+[Conv-BN-ReLU] x 3	0.95
5			[Conv-BN-ReLU] x 3	0.95
6			[Conv-BN-ReLU] x 3	0.94
7			[Conv-BN-ReLU] x 3	0.95
8	1024	14x14	Shortcut+[Conv-BN-ReLU] x 3	0.96
9			[Conv-BN-ReLU] x 3	0.96
10			[Conv-BN-ReLU] x 3	0.97
11			[Conv-BN-ReLU] x 3	0.98
12			[Conv-BN-ReLU] x 3	0.98
13			[Conv-BN-ReLU] x 3	0.95
14	2048	7x7	Shortcut+[Conv-BN-ReLU] x 3	0.98
15			[Conv-BN-ReLU] x 3	**1.04**
16			[Conv-BN-ReLU] x 3	0.93

Table 5: Summary of the block in ResNet50 that trained for ImageNet and provided by PyTorch.

Block Num	Channel dimension	Output size	Structure	NormRatio
1	64	224x224	Conv-ReLU	0.95
2			Conv-ReLU	0.91
3	128	112x112	MaxPool-Conv-ReLU	0.89
4			Conv-ReLU	0.88
5	256	56x56	MaxPool-Conv-ReLU	0.87
6			Conv-ReLU	0.87
7			Conv-ReLU	0.86
8	512	28x28	MaxPool-Conv-ReLU	0.84
9			Conv-ReLU	0.86
10			Conv-ReLU	0.88
11	512	14x14	MaxPool-Conv-ReLU	0.96
12			Conv-ReLU	1.00
13			Conv-ReLU	**1.13**

Table 6: Summary of the block in VGG16 that trained for ImageNet and provided by PyTorch.

Block Num	Channel dimension	Output size	Structure	NormRatio
1	16	112x112	Conv-BN-HS	1.15
2	16	112x112	[Conv-BN-ReLU]+Conv-BN	1.15
3	24	56x56	[Conv-BN-ReLU]x2+Conv-BN	1.21
4			[Conv-BN-ReLU]x2+Conv-BN	1.14
5	40	28x28	[Conv-BN-ReLU]x2+SE+Conv-BN	1.00
6			[Conv-BN-ReLU]x2+SE+Conv-BN	1.00
7	40	28x28	[Conv-BN-ReLU]x2+SE+Conv-BN	1.06
8			[Conv-BN-HS]x2+Conv-BN	1.09
9	80	14x14	[Conv-BN-HS]x2+Conv-BN	1.06
10	80	14x14	[Conv-BN-HS]x2+SE+Conv-BN	1.05
11	80	14x14	[Conv-BN-HS]x2+SE+Conv-BN	1.04
12	112	14x14	[Conv-BN-HS]x2+SE+Conv-BN	1.04
13	112	14x14	[Conv-BN-HS]x2+SE+Conv-BN	0.99
14	160	7x7	[Conv-BN-HS]x2+SE+Conv-BN	1.02
15	160	7x7	[Conv-BN-HS]x2+SE+Conv-BN	1.02
16	160	7x7	[Conv-BN-HS]x2+SE+Conv-BN	1.18
17	960	7x7	Conv-BN-HS	**1.37**

Table 7: Summary of the block in MobilNetV3_large that trained for ImageNet and provided by PyTorch.

4 Ablation of pseudo OOD generation

We use Jigsaw images as pseudo OOD for calculating the NormRatio. However, we may use different pseudo OOD images that can be directly generated from in-distribution images (e.g., blurred images or vertical flipped images) or can be easily obtained (e.g., gaussian noise images). Thus, we show NormRatio for each pseudo OOD image. In Table 8, we show the NormRatio of each block in WRN28 for given various pseudo OOD: Blurred training samples (Blur), Vertically flipped training samples (Vertical flip), 2x2 Jigsaw puzzle images (Jigsaw 2x2), 3x3 Jigsaw puzzle images (Jigsaw 3x3), 9x9 Jigsaw puzzle images (Jigsaw 9x9), and Gaussian noise. Also, the similar pseudo OOD generation methods: ATOM(Chen, Li, Wu, Liang, and Jha (2021)) and CNC(Hebbalaguppe, Goshal, Prakash, Khadilkar, and Arora (2022)) are utilized.

We find that the blurred images can produce the highest NormRatio in the shallow block because blurred images have fewer low-level abstraction (i.e., lowly activate filters in the shallow block) and similar high-level abstraction compared to original image. Also, we find that the vertically flipped images act similarly to Jigsaw puzzle images since they also have similar low-level abstraction and different high-level abstraction compared to training samples. Finally, we find that the Gaussian noise images highly activate the shallow block, while lowly activate the deeper block. However, the standard deviation is very high for the Gaussian noise, and the penultimate block, which is the best for OOD detection, may not be selected.

Block Num	Blur	Vertical filp	Jigsaw 2x2	Jigsaw 3x3	Jigsaw 9x9	Guassian noise	ATOM	CNC
1	1.57 ± 0.09	1.00 ± 0.00	0.96 ± 0.00	0.83 ± 0.01	0.74 ± 0.01	0.49 ± 0.02	1.02 ± 0.01	0.98 ± 0.02
2	1.72 ± 0.13	1.00 ± 0.00	0.96 ± 0.00	0.87 ± 0.01	0.77 ± 0.01	0.42 ± 0.03	1.03 ± 0.01	1.01 ± 0.02
3	2.19 ± 0.25	1.00 ± 0.00	0.95 ± 0.00	0.89 ± 0.00	0.76 ± 0.01	0.43 ± 0.02	1.01 ± 0.00	1.06 ± 0.03
4	**2.51** ± 0.20	1.00 ± 0.00	0.95 ± 0.00	0.90 ± 0.00	0.77 ± 0.01	0.47 ± 0.02	1.01 ± 0.00	1.08 ± 0.02
5	1.71 ± 0.02	1.01 ± 0.00	0.96 ± 0.00	0.95 ± 0.01	0.94 ± 0.02	0.66 ± 0.02	1.04 ± 0.00	1.03 ± 0.01
6	1.79 ± 0.04	1.01 ± 0.00	0.96 ± 0.00	0.94 ± 0.01	0.98 ± 0.02	0.78 ± 0.04	1.05 ± 0.00	1.07 ± 0.01
7	1.17 ± 0.05	1.01 ± 0.00	0.98 ± 0.00	0.98 ± 0.01	1.01 ± 0.02	0.69 ± 0.04	1.04 ± 0.00	1.00 ± 0.01
8	1.21 ± 0.04	1.03 ± 0.04	0.99 ± 0.00	0.99 ± 0.01	1.01 ± 0.03	0.69 ± 0.05	1.05 ± 0.00	1.01 ± 0.01
9	1.78 ± 0.09	1.07 ± 0.00	1.02 ± 0.00	1.04 ± 0.02	1.10 ± 0.05	1.08 ± 0.09	1.13 ± 0.01	1.22 ± 0.02
10	1.79 ± 0.14	1.15 ± 0.01	1.09 ± 0.01	1.16 ± 0.02	1.22 ± 0.05	1.15 ± 0.19	1.19 ± 0.03	1.33 ± 0.02
11	2.03 ± 0.10	**1.30** ± 0.02	**1.24** ± 0.01	**1.36** ± 0.02	**1.39** ± 0.04	**1.25** ± 0.22	**1.39** ± 0.02	**1.52** ± 0.01
12	1.76 ± 0.09	1.19 ± 0.01	1.15 ± 0.01	1.28 ± 0.01	1.34 ± 0.03	1.19 ± 0.19	1.27 ± 0.01	1.39 ± 0.02

Table 8: Ablation on pseudo OOD generation methods using WRN28. The results are average over five runs.

5 Ablation of L_p-norm and fusing score

Architecture	**Method**	**OOD**													
		SVHN		Textures		LSUN©		LSUN®		iSUN		Places365		**Average**	
		FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑
ResNet18	L_2-norm	7.37	98.49	31.30	91.96	5.09	99.13	29.00	95.05	27.05	95.26	66.33	81.57	27.69	93.58
	L_1-norm	9.95	97.99	43.36	87.51	**0.08**	**99.95**	35.21	93.60	34.39	93.63	64.35	83.72	31.22	92.73
	L_{max}-norm	6.90	98.73	23.84	**95.18**	0.25	99.91	24.00	95.91	22.79	96.11	**62.16**	85.21	23.32	95.18
	L_2-norm of two blocks	**4.28**	**99.18**	**22.89**	94.83	0.18	99.90	**19.26**	**96.85**	**17.28**	**97.16**	62.61	**86.06**	**21.08**	**95.66**
	L_2-norm of three blocks	4.80	99.01	24.44	94.05	0.69	99.71	25.84	95.88	22.62	96.37	70.41	82.82	24.80	94.64
WRN28	L_2-norm	3.83	99.18	14.23	97.06	**0.32**	**99.81**	**8.13**	**98.32**	**5.98**	**98.71**	**48.69**	**90.91**	**13.53**	**97.33**
	L_1-norm	26.11	95.68	31.05	91.86	0.42	99.85	10.55	98.00	8.13	98.38	49.30	90.46	20.92	95.71
	L_{max}-norm	**2.93**	**99.42**	**10.38**	**97.89**	1.19	99.59	11.21	97.86	8.21	98.37	52.11	89.83	14.34	97.16
	L_2-norm of two blocks	6.03	98.75	19.33	95.68	1.06	99.57	15.99	97.25	12.29	97.83	61.87	87.11	19.43	96.03
	L_2-norm of three blocks	13.26	97.52	30.92	91.88	1.67	99.54	36.71	94.03	30.15	95.07	73.53	80.58	31.04	93.10

Table 9: Ablation on L_p-norm and fusing score using ResNet18 and WRN28 architectures. The results are average over five runs.

We use L_2-norm in our study, but the other L_p-norm may perform better compared to L_2-norm. In this experiment, we utilize L_1-norm and L_{max}-norm to compare the OOD detection performance. Also, we utilize the highest block using block selection method, but there can be an OOD detection performance improvement when we use fusing score from various blocks. Thus, we select the two or three highest blocks using block selection method, and we sum FeatureNorm from the selected blocks for the OOD indicator. The Table 9 demonstrate the ablation results. We find that L_2-norm achieved the best results compared to L_1-norm and L_{max}-norm on WRN28 architecture. However, we find that L_{max}-norm achieved the best results compared to L_1-norm and L_2-norm on ResNet18 architecture. Also, using two blocks achieved the better results compared to the original one on ResNet18, while it showed degraded performance on WRN28 architecture. Moreover, using three blocks works poorly since it contains the last block, which is overconfident about any inputs. Thus, we argue that using one block is the more proper way to calculate FeatureNorm.

6 Comparison with recent OOD detection methods

Architecture	**Method**	**OOD**													
		SVHN		Textures		LSUNⓒ		LSUNⓡ		iSUN		Places365		**Average**	
		FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑
ResNet18	FeatureNorm (ours)	**7.13**	**98.65**	**31.18**	**92.31**	**0.07**	**99.96**	27.08	95.25	**26.02**	**95.38**	62.54	84.62	**25.67**	**94.36**
	RankFeat	57.16	83.54	55.71	82.94	27.40	93.57	69.07	77.19	70.20	75.96	85.97	64.34	60.92	79.59
	TFEM	30.46	94.06	45.65	90.59	7.23	98.63	**23.69**	**95.92**	27.17	95.32	**43.37**	**90.33**	29.59	94.14
	ViM	70.47	90.46	49.16	90.79	88.69	88.78	59.16	90.79	56.05	90.95	70.62	85.88	65.69	89.61
	GradNorm	100.00	47.47	100.00	46.85	100.00	46.70	100.00	45.64	100.00	45.76	100.00	45.78	100.00	46.37
WRN28	FeatureNorm (ours)	**3.83**	**99.18**	**14.23**	**97.06**	**0.32**	**99.81**	**8.13**	**98.32**	**5.98**	**98.71**	**48.69**	**90.91**	**13.53**	**97.33**
	RankFeat	90.04	80.72	92.79	72.54	97.67	72.19	89.99	76.75	89.77	76.92	83.60	79.74	90.64	76.48
	TFEM	33.08	90.88	45.69	85.78	5.89	98.76	22.75	94.92	25.13	94.18	38.28	88.67	28.47	92.20
	ViM	24.33	96.34	35.15	94.22	76.58	88.88	46.80	92.94	47.34	92.63	66.23	88.02	49.40	92.17
	GradNorm	68.20	66.73	76.47	59.78	41.21	79.59	69.55	67.80	68.62	68.10	80.79	58.49	67.47	66.75

Table 10: Comparison with recent post-hoc OOD detection methods using ResNet18 and WRN28 architectures. The results are average over five runs.

We compare our method with more recent post-hoc OOD detection methods, which are RankFeat, TFEM, ViM, GradNorm. **RankFeat**(Song, Sebe, & Wang, 2022) is a method that applies a refinement strategy to feature vectors, which often have a larger dominant singular value for OOD data than for ID data. The method removes the rank-1 matrix, composed of the largest singular value and its associated singular vectors, from the high-level feature. After refinement, the energy score of the logit is used as the OOD detection score. **TFEM**(Zhu et al., 2022) proposed the method of rectifying the feature into its typical set and calculating the OOD score with the typical features to achieve reliable OOD detection. In the paper, the authors propose a calculation method using batch normalization layer to find typical features. However, we utilize the proposed calculation method without using batch normalization layer to find typical feature(i.e., Typical Feature Estimated Method (TFEM)), which calculates typical feature by training images. **ViM**(Wang, Li, Feng, & Zhang, 2022) is Virtual-logit Matching (ViM), which combines the class-agnostic score from feature space and the ID class-dependent logits for OOD detection. ViM generates an additional logit representing the virtual OOD class from the residual of the feature against the principal space and matches it with the original logits by a constant scaling. The probability of this virtual logit after softmax is the indicator of OOD-ness. **GradNorm**(Huang, Geng, & Li, 2021) is an OOD detection method that uses information extracted from the gradient space of networks. GradNorm directly employs the vector norm of gradients, backpropagated from the KL divergence between the softmax output and a uniform probability distribution. Comparison with above methods are tabulated in Table 10.

References

Chen, J., Li, Y., Wu, X., Liang, Y., & Jha, S. (2021). Atom: Robustifying out-of-distribution detection using outlier mining. In *Machine learning and knowledge discovery in databases. research track: European conference, ecml pkdd 2021, bilbao, spain, september 13–17, 2021, proceedings, part iii 21* (pp. 430–445).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Hebbalaguppe, R. S., Goshal, S. S., Prakash, J., Khadilkar, H., & Arora, C. (2022). A novel data augmentation technique for out-of-distribution sample detection using compounded corruptions. *arXiv preprint arXiv:2207.13916*.

Hendrycks, D., & Gimpel, K. (2017). A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proceedings of international conference on learning representations.*

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., ... others (2019). Searching for mobilenetv3. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 1314–1324).

Hsu, Y.-C., Shen, Y., Jin, H., & Kira, Z. (2020). Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 10951–10960).

Huang, R., Geng, A., & Li, Y. (2021). On the importance of gradients for detecting distributional shifts in the wild. In *Advances in neural information processing systems.*

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Liang, S., Li, Y., & Srikant, R. (2018). Enhancing the reliability of out-of-distribution image detection in neural networks. In *6th international conference on learning representations, iclr 2018.*

Liu, W., Wang, X., Owens, J., & Li, Y. (2020). Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*.

Macêdo, D., Ren, T. I., Zanchettin, C., Oliveira, A. L. I., & Ludermir, T. (2021). Entropic out-of-distribution detection: Seamless detection of unknown examples. *IEEE Transactions on Neural Networks and Learning Systems*, 1-15. DOI: 10.1109/TNNLS.2021.3112897

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Song, Y., Sebe, N., & Wang, W. (2022). Rankfeat: Rank-1 feature removal for out-of-distribution detection. In *Neurips.*

Sun, Y., Guo, C., & Li, Y. (2021). React: Out-of-distribution detection with rectified activations. In *Advances in neural information processing systems.*

Sun, Y., & Li, Y. (2022). Dice: Leveraging sparsification for out-of-distribution detection. In *European conference on computer vision.*

Wang, H., Li, Z., Feng, L., & Zhang, W. (2022). Vim: Out-of-distribution with virtual-logit matching. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition.*

Zagoruyko, S., & Komodakis, N. (2016, September). Wide residual networks. In E. R. H. Richard C. Wilson & W. A. P. Smith (Eds.), *Proceedings of the british machine vision conference (bmvc)* (p. 87.1-87.12). BMVA Press. Retrieved from https://dx.doi.org/10.5244/C.30.87 DOI: 10.5244/C.30.87

Zhu, Y., Chen, Y., Xie, C., Li, X., Zhang, R., Xue', H., . . . Chen, Y. (2022). Boosting out-of-distribution detection with typical features. In A. H. Oh, A. Agarwal, D. Belgrave, & K. Cho (Eds.), *Advances in neural information processing systems.* Retrieved from https://openreview.net/forum?id=4maAiUt0A4