Appendix

In this section, we give more results on *OOD* detection with our proposed BAL together with the detailed parameter settings in training and inference phase.

6.1 Hardware and Software

All experiments are performed with Python 3.6.5 and Pytorch 1.6.0. Eight NVIDIA GeForce Titan X (each with 12GB memory) GPUs are used for training and inference.

6.2 Pre-trained Neural Networks

The pre-trained classification networks and post-processed dataset used in this paper are from ODIN git repository [5], including ResNet, DenseNet and many other resources.

6.3 Quantitative results on *OOD* detection.

We give the average confidence scores of max-softmax baseline and the proposed BAL on *ID* and *OOD* data respectively. From the results shown in Table **??**, we can see that the proposed BAL gives *OOD* data much lower scores compared to max-softmax baseline, while it still gives high confidence scores to *ID* data.

Table 6. Average confidence scores on *ID* and *OOD*. C-10, C-100, TIN are CIFAR-10, CIFAR-100 and TinyImageNet respectively.

Train	Test	ResNet[11]		DenseNet[16]	
		max-softmax[12]	BAL(ours)	max-softmax[12]	BAL(ours)
C-10	C-10	**0.98**	0.87	**0.97**	0.86
	SVHN	0.84	**0.03**	0.85	**0.22**
	LSUN	0.72	**0.40**	0.79	**0.66**
	TIN	0.76	**0.48**	0.80	**0.67**
C-100	C-100	**0.86**	0.75	**0.85**	0.79
	SVHN	0.60	**0.33**	0.60	**0.27**
	LSUN	0.74	**0.38**	0.66	**0.30**
	TIN	0.72	**0.33**	0.64	**0.28**
SVHN	SVHN	**0.98**	0.74	**0.99**	0.69
	C-10	0.75	**0.23**	0.79	**0.17**
	LSUN	0.69	**0.20**	0.80	**0.18**
	TIN	0.68	**0.18**	0.79	**0.17**

6.4 Hyperparameters of ODIN

In ODIN, we search the optimal parameter settings for temperature T and magnitude ϵ. We list them below.

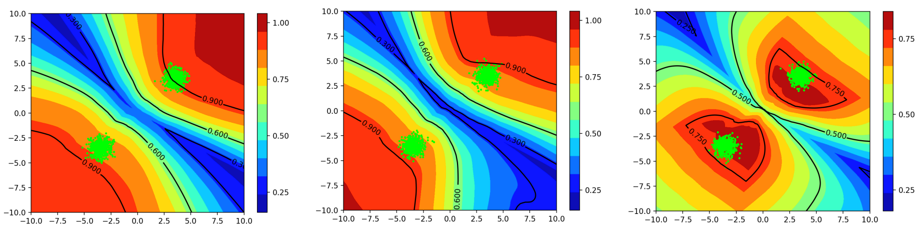
[5] https://github.com/ShiyuLiang/odin-pytorch

Table 7. Hyperparameters settings for ODIN.

ID	CIFAR-10		CIFAR-100		SVHN	
	ResNet	DenseNet	ResNet	DenseNet	ResNet	DenseNet
T	100	10	100	10	10	10
ϵ	0.0014	0.0005	0.0024	0.0020	0.0005	0.0005

6.5 Decision boundary of different schemes

We provide more visualization results of different schemes' decision boundary here.



(a) ODIN [25] (b) G-ODIN [15] (c) AEC [6]

Fig. 7. Decision boundaries of existing schemes. We train a classifier (MLP) to identify two gaussian distribution. The lime points in image above are the original training data. The background color indicates the distribution of confidence score in raw data space. For AEC, we train a reconstruction neural network to rebuild the input. The reconstruction error is mapping to confidence using function e^{-d} where d represents the L_1 distance between the input and rebuilt data. AEC is trained with an AdamW optimizer. ODIN and G-ODIN are post-processing methods without introducing any new module.

6.6 Parameters for image normalization

Since we use the pretrained classification models, it important to use the same normalization method for reproducing the original results. For colour images fed in ResNet, they are normalized with mean $(0.4914, 0.4822, 0.4465)$ and std $(0.2023, 0.1994, 0.2010)$ for RGB channels respectively. For colour images fed in DenseNet, they are normalized with mean $(0.4914, 0.4824, 0.4467)$ and std $(0.2471, 0.2435, 0.2616)$ for RGB channels respectively.

6.7 Optimizers for training

Adam is used with learning rate decay from 1e-4. The learning rate halves every 30 epochs. α and β od Adam are set to (0.5, 0.999).

Table 8. Architectures of conditional GAN.

	Generator	Discriminator
layer1	nn.Linear(feat_dim, 32)	nn.Linear(feat_dim, 32)
layer2	nn.Linear(label_dim, 32)	nn.Linear(label_dim, 32)
layer3	concatenate + nn.Linear(64, 128)	concatenate + nn.Linear(64, 128)
layer4	nn.Linear(128, 64)	nn.Linear(128, 64)
layer5	nn.Linear(64, feat_dim)	nn.Linear(64, 2)

6.8 Architectures of conditional GAN

Both generator and discriminator use four fully connected layer since the input features are vectors. LeakyReLU is used as activation function. We give the architectures of conditional GAN in Pytorch style below.

6.9 More results on mixture dataset

We provide detection results of AEC and G-ODIN on mixture dataset.

Table 9. Detecting *OOD* samples on MNIST, Fashion-MNIST and Omniglot with ResNet18. We use the mixture of two datasets as *OOD* samples.

ID	MNIST						F-MNIST					
OOD	F-MNIST & Omniglot						MNIST & Omniglot					
Methods	Softmax baseline / ODIN / G-ODIN / GCPL / AEC / BAL(ours)											
↑ Cls Acc	**99.43**	**99.43**	99.39	99.23	99.12	**99.43**	**91.51**	**91.51**	91.47	90.93	90.27	**91.51**
↓ Det Err	4.14	5.01	3.76	4.77	3.10	**3.06**	32.42	19.14	12.17	30.73	17.32	**7.10**
↓ FPR 95	3.29	5.03	2.14	4.54	**1.01**	1.11	59.84	33.27	23.01	56.45	21.92	**9.20**
↑ AUROC	97.66	97.94	98.96	97.96	**99.36**	99.32	89.44	93.45	96.70	81.79	94.57	**97.82**
↑ AUPR$_{in}$	97.22	97.42	99.21	98.14	99.42	**99.46**	90.80	94.28	96.82	72.40	95.09	**98.31**
↑ AUPR$_{out}$	97.24	97.64	98.47	97.35	**99.13**	99.09	86.20	91.36	96.33	82.38	92.92	**96.95**

6.10 Evaluation metrics

We report the following metrics to measure the performance of *OOD* detection. The quantity of *ID* and *OOD* examples are strictly kept same in evaluation.
FPR at 95% TPR (FPR95) is the probability of an *OOD* example being misclassified as *ID* examples when the True Positive Rate is 95%. True positive Rate and False Positive Rate are the same as defined in *ROC* curve.
Detection Error measures the misclassification probability when True Positive Rate is 95%. It is defined as $0.5(1 - \text{TPR}) + 0.5\text{FPR}$.
AUROC represents the area under *ROC* curve. Greater AUROC indicates that the neural network is more confident to assign higher score to *ID* data than *OOD* data. An ideal classifier has an AUROC score of 100%.
AUPR represents the area under Precision-Recall curve. AUPR$_{in}$ indicates the ability of detecting *ID* data while AUPR$_{out}$ indicates that of *OOD* data.