Novelty and Uncertainty Estimation

AI602: Recent Advances in Deep Learning

Lecture 6

Slide made by

Kimin Lee and Jihoon Tack

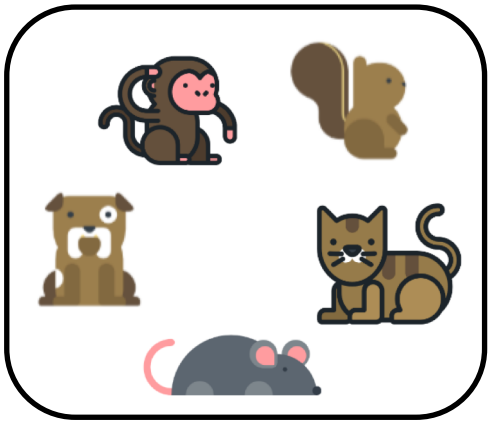
KAIST EE & AI

Table of Contents

Table of Contents

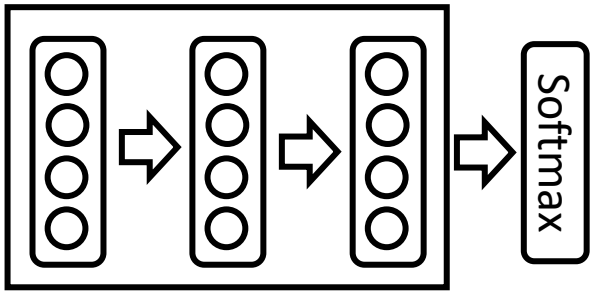
What is Novelty Detection?

- Deep neural networks (DNNs) can be generalized well when the test samples are from similar distribution (i.e., in-distribution)
  - E.g., image classifier

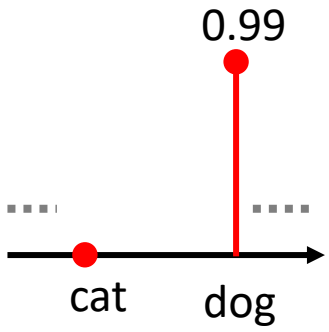


Training data = animal

Test sample

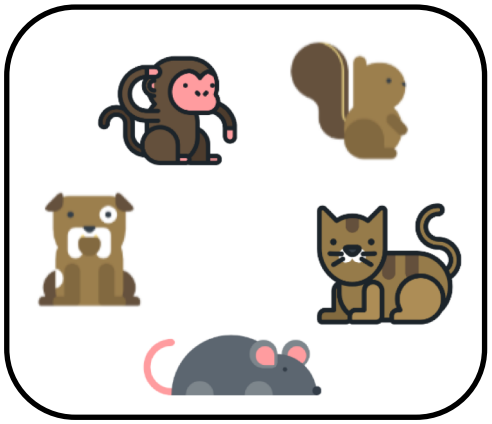


DNNs



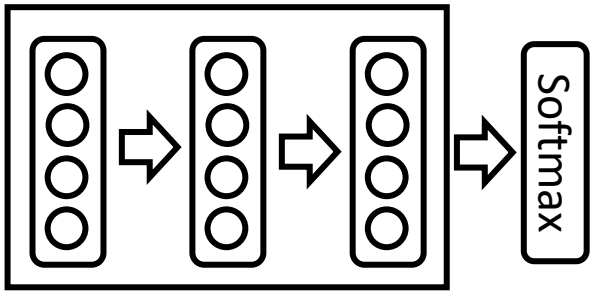
What is Novelty Detection?

- Deep neural networks (DNNs) can be generalized well when the test samples are from similar distribution (i.e., in-distribution)
  - E.g., image classifier

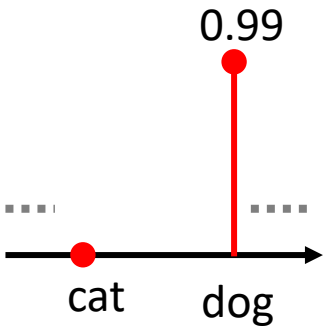


Training data = animal

Test sample



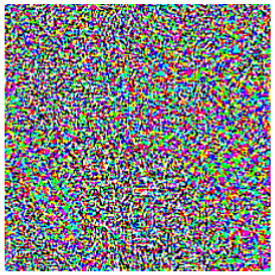
DNNs



- However, in the real world, there are many unknown and unseen samples

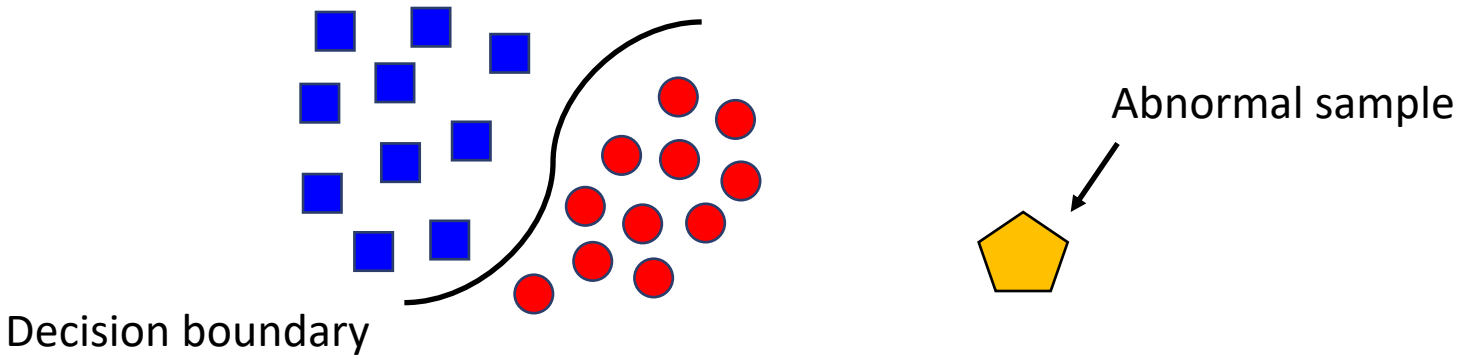
Unseen sample, i.e., out-of-distribution (not animal)

Unknown sample



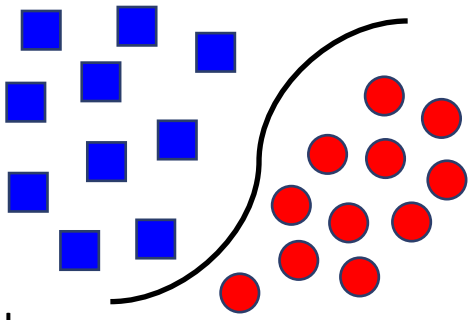
Adversarial samples [Goodfellow et al., 2015]

- **Novelty detection**
  - Detect whether a test sample is from in-distribution (i.e., training distribution by classifier) or not (e.g., out-of-distribution / adversarial samples)

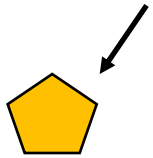


What is Novelty Detection?

- **Novelty detection**
  - Detect whether a test sample is from in-distribution (i.e., training distribution by classifier) or not (e.g., out-of-distribution / adversarial samples)

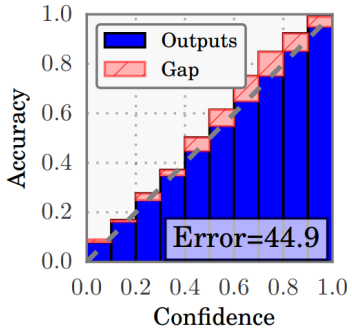


Decision boundary



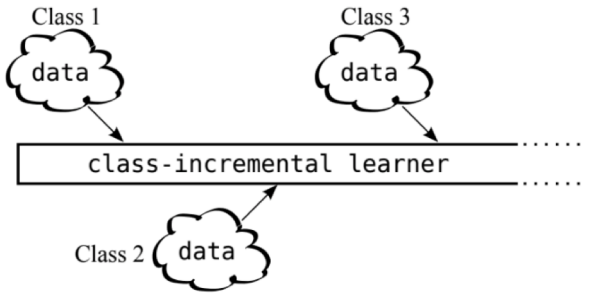
Abnormal sample

- It can be useful for many machine learning problems:



Calibration
[Guo et al., 2017]

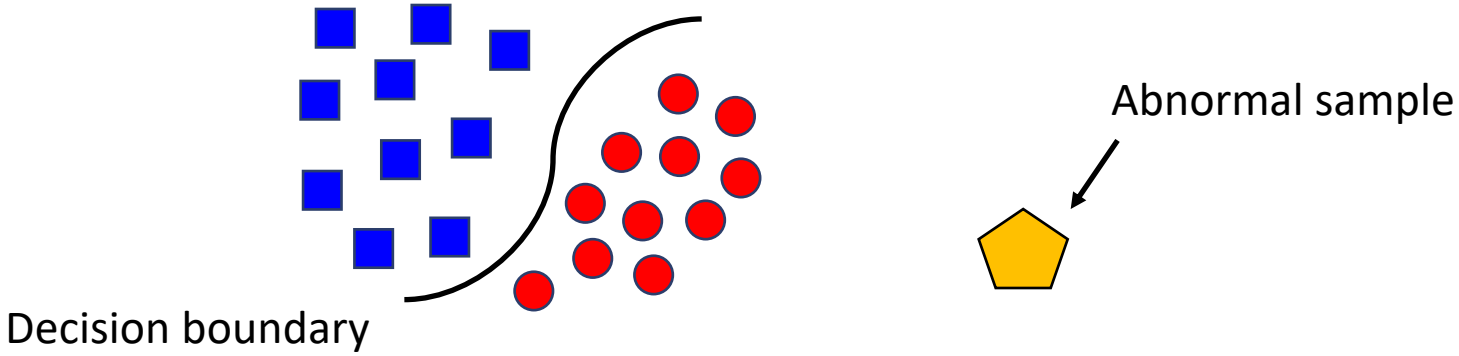
Ensemble learning
[Lee et al., 2017]



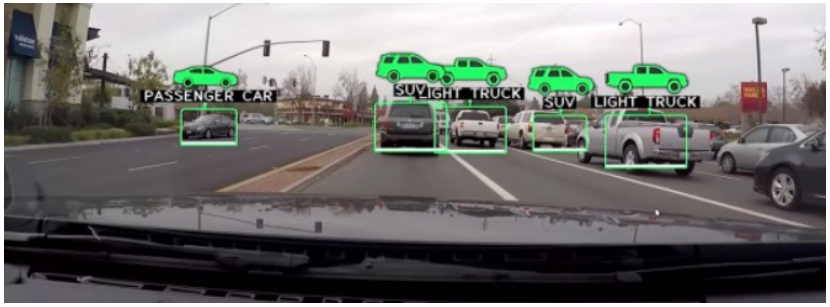
Incremental learning
[Rebuff et al., 2017]

What is Novelty Detection?

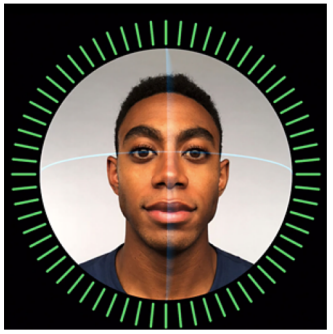
- **Novelty detection**
  - Detect whether a test sample is from in-distribution (i.e., training distribution by classifier) or not (e.g., out-of-distribution / adversarial samples)



- It is also indispensable when deploying DNNs in real-world systems [Amodei et al., 2016]



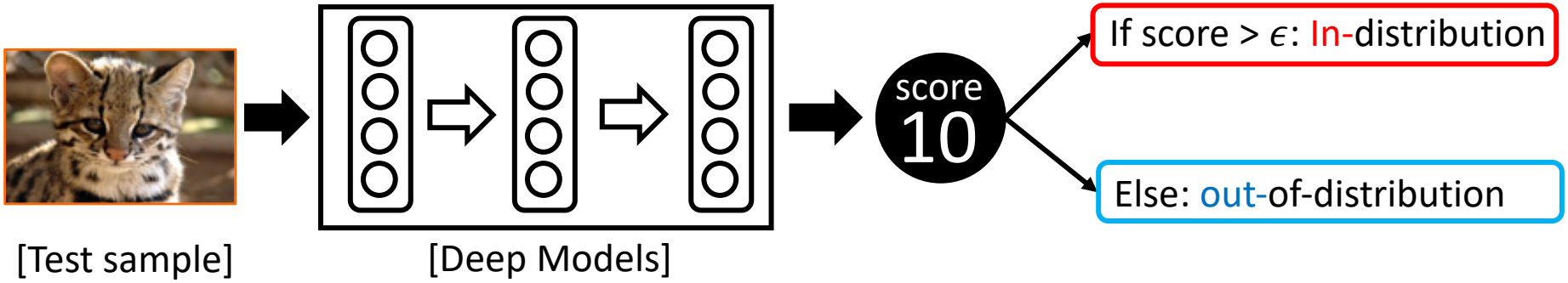
Autonomous drive



Secure authentication system

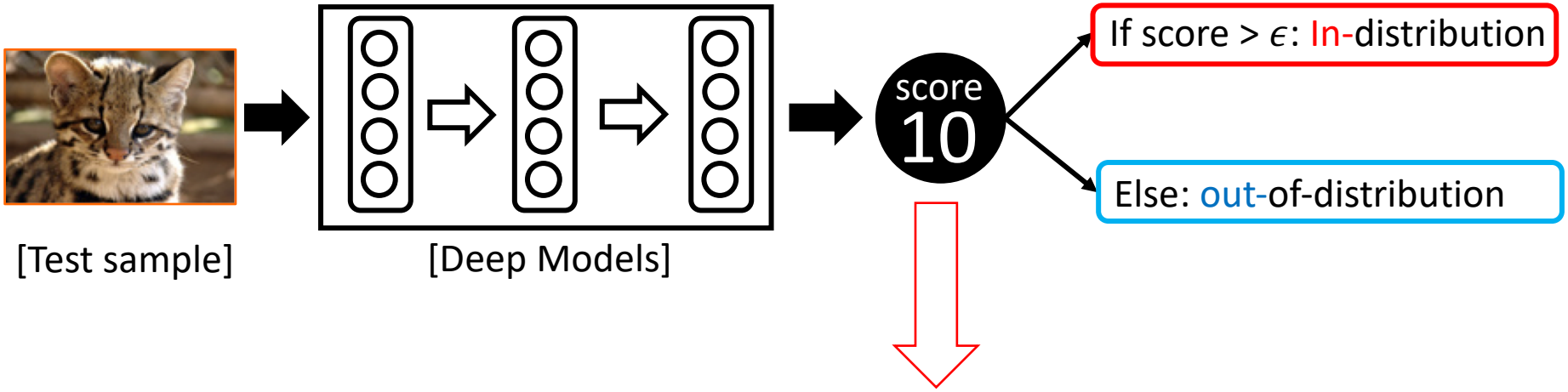
What is Novelty Detection?

- How to solve this problem?
  - **Threshold-based Detector** [Hendrycks et al., 2017, Liang et al., 2018]



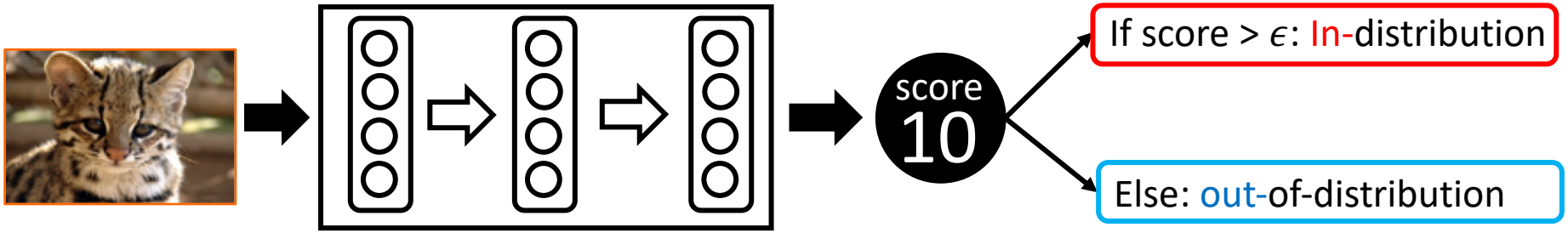
What is Novelty Detection?

- How to solve this problem?
  - **Threshold-based Detector** [Hendrycks et al., 2017, Liang et al., 2018]

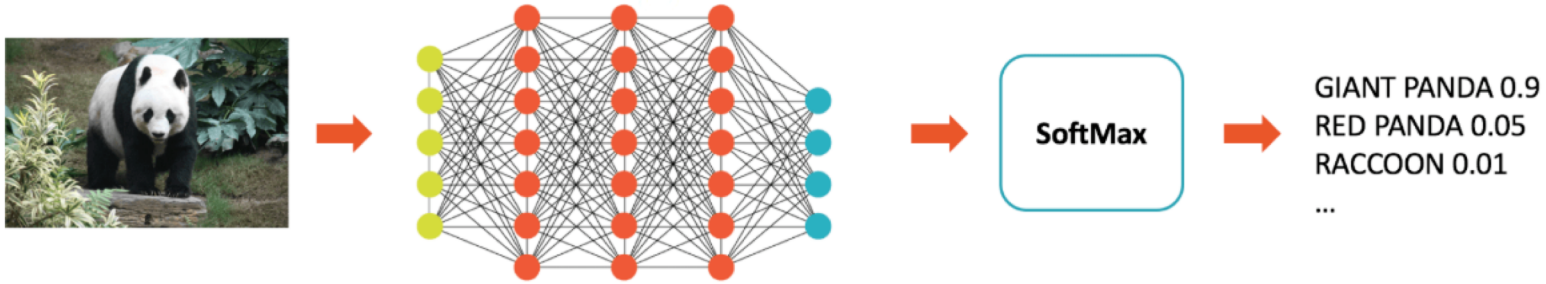


What is Novelty Detection?

- How to solve this problem?
  - **Threshold-based Detector** [Hendrycks et al., 2017, Liang et al., 2018]

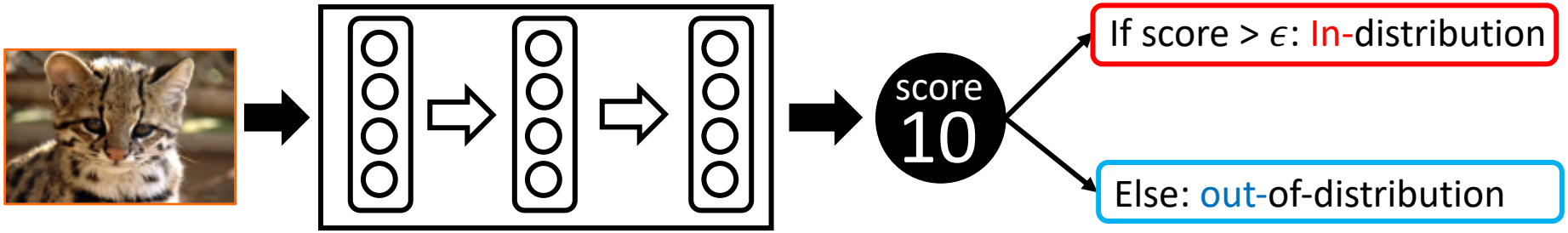


- Part 1. utilizing image classifiers (+ self-supervised learning)

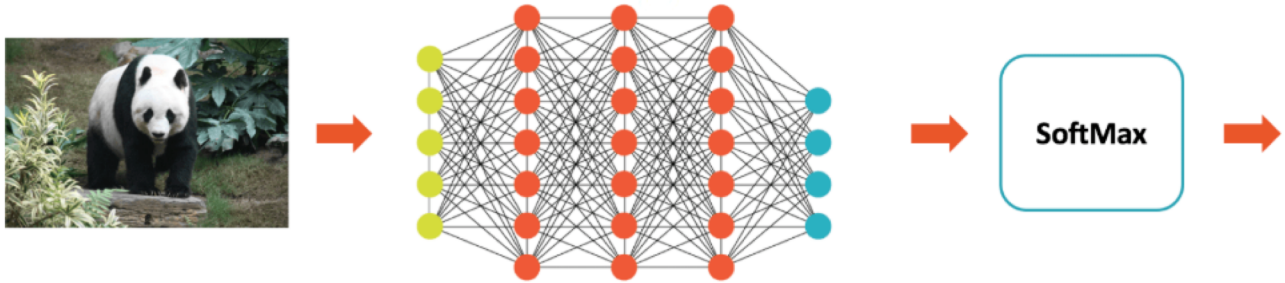


What is Novelty Detection?

- How to solve this problem?
  - **Threshold-based Detector** [Hendrycks et al., 2017, Liang et al., 2018]



- Part 1. utilizing image classifiers (+ self-supervised learning)



- Part 2. utilizing generative models

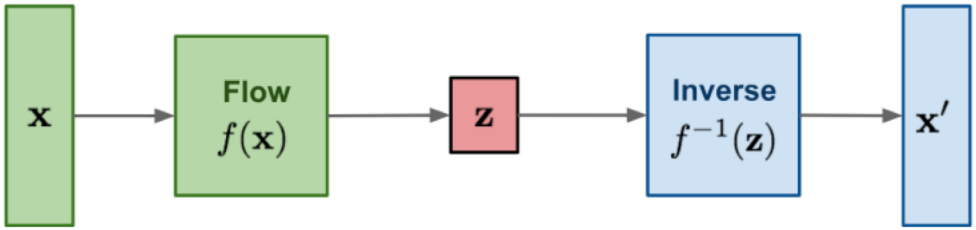


Table of Contents

Utilizing the Classifier: Preliminaries

- Remind that classification is finding an unknown posterior distribution, i.e., P(Y|X)

- How to model our posterior distribution: Softmax classifier with DNNs

$$P(y=c|\mathbf{x}) = \frac{\exp(\mathbf{w}_c^\top f(\mathbf{x}) + b_c)}{\sum_{c'} \exp(\mathbf{w}_{c'}^\top f(\mathbf{x}) + b_{c'})}$$

  - Where $f(\cdot)$ is hidden features from DNNs

Utilizing the Classifier: Preliminaries

- Remind that classification is finding an unknown posterior distribution, i.e., P(Y|X)

- How to model our posterior distribution: Softmax classifier with DNNs

$$P(y=c|\mathbf{x}) = \frac{\exp(\mathbf{w}_c^\top f(\mathbf{x}) + b_c)}{\sum_{c'} \exp(\mathbf{w}_{c'}^\top f(\mathbf{x}) + b_{c'})}$$

  - Where $f(\cdot)$ is hidden features from DNNs

- Natural choice for confidence score
  - 1. maximum value of posterior distribution

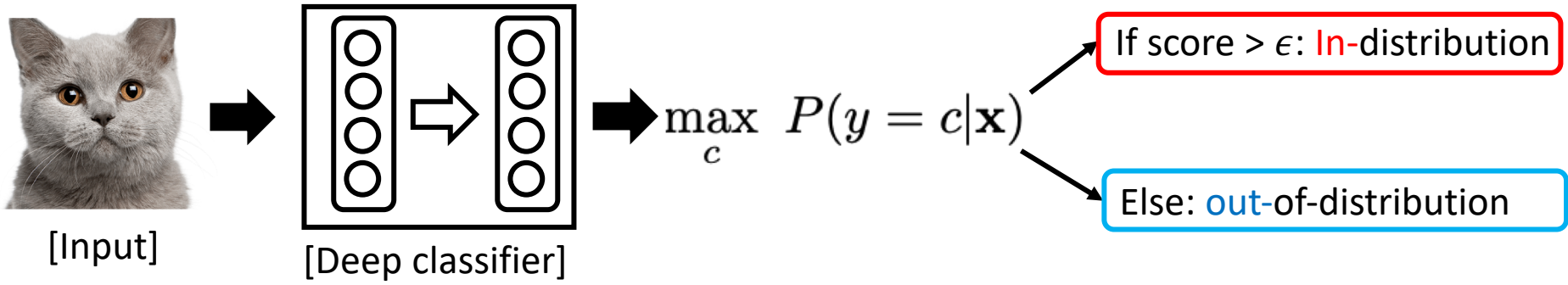
$$\max_c \; P(y=c|\mathbf{x})$$

  - 2. entropy of posterior distribution

$$H = \sum_y -P(y|\mathbf{x}) \log P(y|\mathbf{x})$$

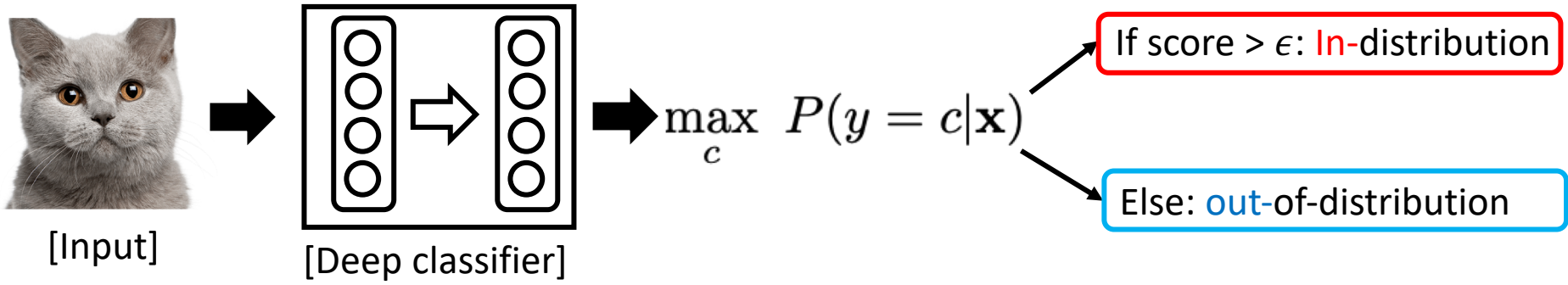
Utilizing the Classifier: Posterior Distribution

- **Baseline detector** [Hendrycks et al., 2017]
  - Confidence score = maximum value of predictive distribution

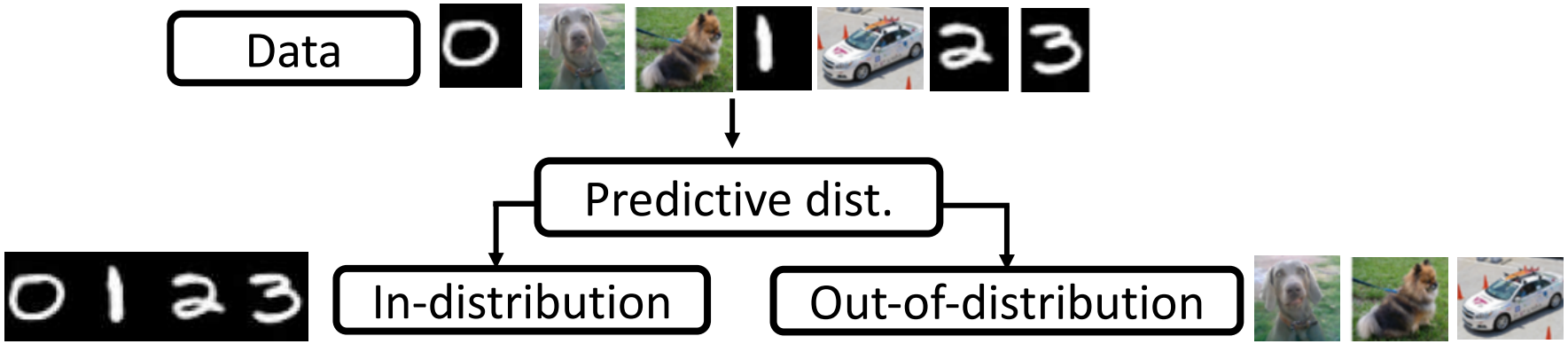


Utilizing the Classifier: Posterior Distribution

- **Baseline detector** [Hendrycks et al., 2017]
  - Confidence score = maximum value of predictive distribution

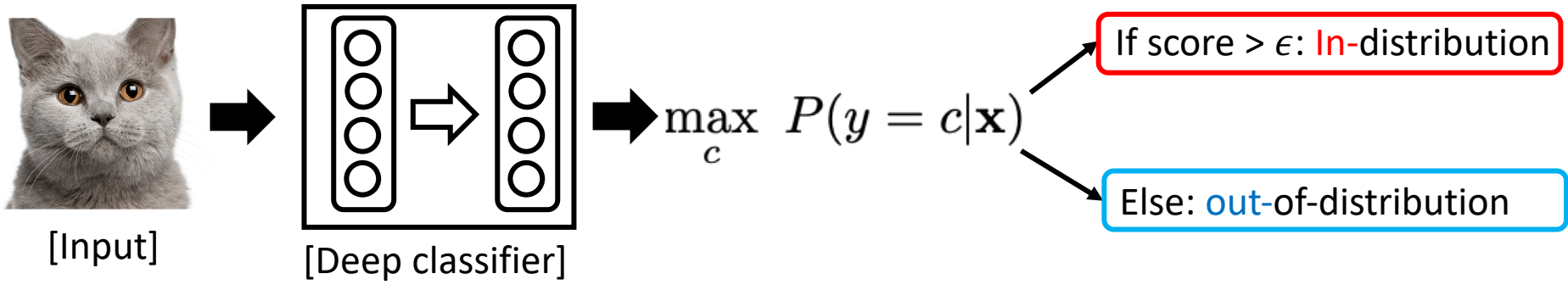


- Evaluation: detecting out-of-distribution
  - Assume that we have classifier trained on MNIST dataset
  - Detecting out-of-distribution for this classifier

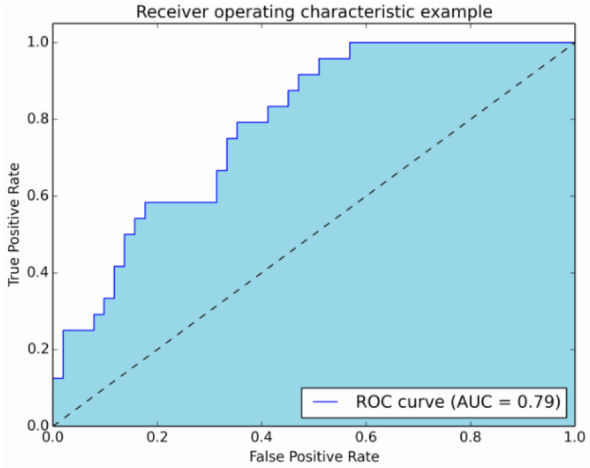


Utilizing the Classifier: Posterior Distribution

- **Baseline detector** [Hendrycks et al., 2017]
  - Confidence score = maximum value of predictive distribution

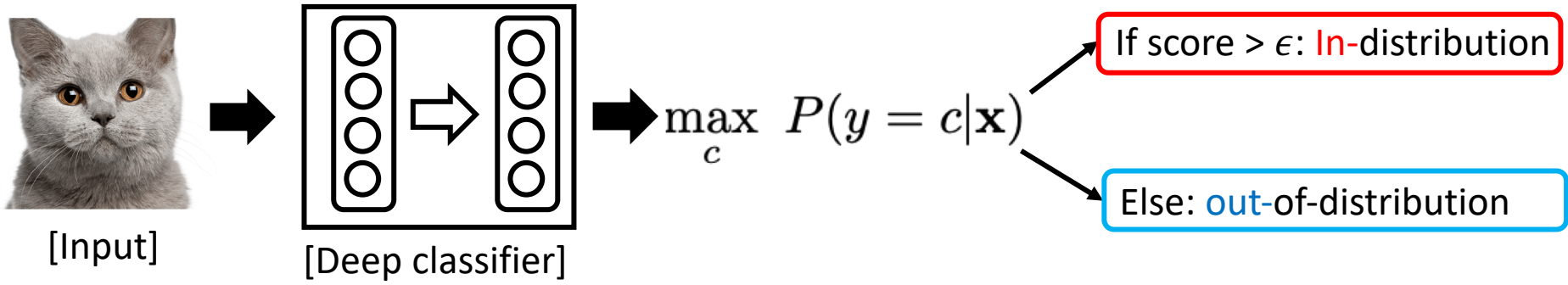


- Evaluation: detecting out-of-distribution
  - TP = true positive / FN = false negative / TN = true negative / FP = false positive
  - AUROC
    - Area under ROC curve
    - ROC curve = relationship between TPR and FPR
  - AUPR (Area under the Precision-Recall curve)
    - Area under PR curve
    - PR curve = relationship between precision and recall



Utilizing the Classifier: Posterior Distribution

- **Baseline detector** [Hendrycks et al., 2017]
  - Confidence score = maximum value of predictive distribution



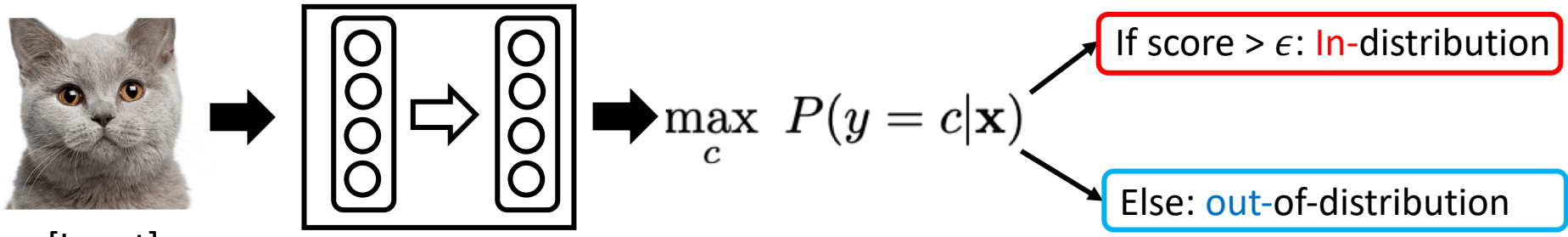
- Evaluation: detecting out-of-distribution
  - **Image classification (computer vision)**

In-Distribution / Out-of-Distribution	AUROC /random	AUPR In /random	AUPR Out/random
CIFAR-10/SUN	95/50	89/33	97/67
CIFAR-10/Gaussian	97/50	98/49	95/51
CIFAR-10/All	96/50	88/24	98/76
CIFAR-100/SUN	91/50	83/27	96/73
CIFAR-100/Gaussian	88/50	92/43	80/57
CIFAR-100/All	90/50	81/21	96/79
MNIST/Omniglot	96/50	97/52	96/48
MNIST/notMNIST	85/50	86/50	88/50
MNIST/CIFAR-10bw	95/50	95/50	95/50
MNIST/Gaussian	90/50	90/50	91/50
MNIST/Uniform	99/50	99/50	98/50
MNIST/All	91/50	76/20	98/80

Baseline method is better than random detector

Utilizing the Classifier: Posterior Distribution

- **Baseline detector** [Hendrycks et al., 2017]
  - Confidence score = maximum value of predictive distribution



- Evaluation: detecting out-of-distribution
  - **Text categorization (NLP)**

Dataset	AUROC /random	AUPR Succ/random	AUPR Err/random
15 Newsgroups	89/50	99/93	42/7.3
Reuters 6	89/50	100/98	35/2.5
Reuters 40	91/50	99/92	45/7.6

    - Out-of-distribution
      - 5 Newsgroups for 15 Newsgroups
      - 2 Reuters for Reuters 6
      - 12 Reuters for 40 Reuters

- **ODIN detector** [Liang et al., 2018]
  - Calibrating the posterior distribution using post-processing

- Two techniques
  - Temperature scaling

Temperature scaling parameter

$$P(y = \widehat{y}|\mathbf{x}; T) = \frac{\exp\left(f_{\widehat{y}}(\mathbf{x})/T\right)}{\sum_y \exp\left(f_y(\mathbf{x})/T\right)},$$

$\mathbf{f} = (f_1, \ldots, f_K)$ is final feature vector of deep neural networks

  - Relaxing the overconfidence by smoothing the posterior distribution

- **ODIN detector** [Liang et al., 2018]
  - Calibrating the posterior distribution using post-processing

- Two techniques
  - Temperature scaling

$$P(y = \widehat{y}|\mathbf{x}; T) = \frac{\exp\left(f_{\widehat{y}}(\mathbf{x})/T\right)}{\sum_y \exp\left(f_y(\mathbf{x})/T\right)},$$

  - Input preprocessing

$$\mathbf{x}' = \mathbf{x} - \varepsilon \text{sign}\left(-\bigtriangledown_{\mathbf{x}} \log P_\theta(y = \widehat{y}|\mathbf{x}; T)\right),$$

Magnitude of noise

$\widehat{y}$ is the predicted label

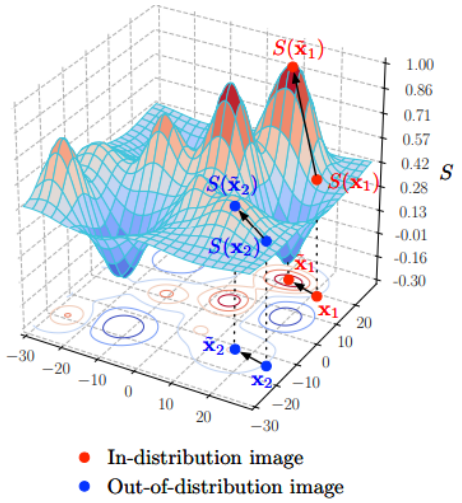


Figure 6: Illustration of effects of the input preprocessing.

Utilizing the Classifier: Posterior Distribution

- **ODIN detector** [Liang et al., 2018]
  - Calibrating the posterior distribution using post-processing

- Two techniques
  - Temperature scaling

$$P(y = \widehat{y}|\mathbf{x}; T) = \frac{\exp\left(f_{\widehat{y}}(\mathbf{x})/T\right)}{\sum_y \exp\left(f_y(\mathbf{x})/T\right)},$$

  - Input preprocessing

$$\mathbf{x}' = \mathbf{x} - \varepsilon \text{sign}\left(-\bigtriangledown_{\mathbf{x}} \log P_\theta(y = \widehat{y}|\mathbf{x}; T)\right),$$

  - Using two methods, the authors define confidence score as follows:

$$\text{Confidence score} = \max_y P(y|\mathbf{x}'; T)$$

Utilizing the Classifier: Posterior Distribution

- **ODIN detector** [Liang et al., 2018]
  - Calibrating the posterior distribution using post-processing

- Two techniques
  - Temperature scaling

$$P(y = \widehat{y}|\mathbf{x}; T) = \frac{\exp\left(f_{\widehat{y}}(\mathbf{x})/T\right)}{\sum_y \exp\left(f_y(\mathbf{x})/T\right)},$$

  - Input preprocessing

$$\mathbf{x}' = \mathbf{x} - \varepsilon \text{sign}\left(-\bigtriangledown_{\mathbf{x}} \log P_\theta(y = \widehat{y}|\mathbf{x}; T)\right),$$

  - Using two methods, the authors define confidence score as follows:

$$\text{Confidence score} = \max_y P(y|\mathbf{x}'; T)$$

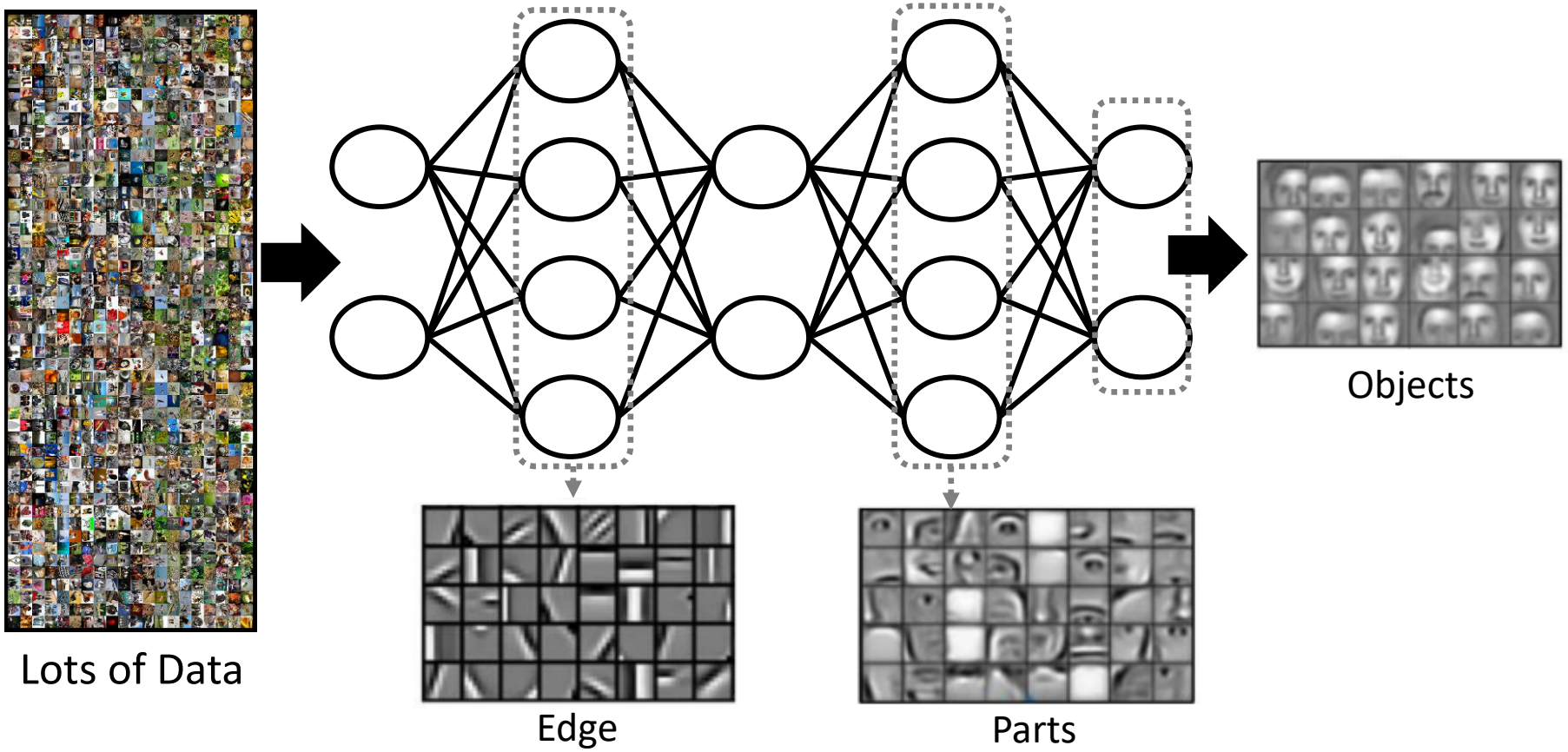
- How to select hyper-parameters
  - Validation
    - 1000 images from in-distribution (positive)
    - 1000 images from out-of-distribution (negative)

- Experimental results

	Out-of-distribution dataset	FPR (95% TPR) ↓	Detection Error ↓	AUROC ↑	AUPR In ↑	AUPR Out ↑
		Baseline (Hendrycks & Gimpel, 2017) / ODIN				
Dense-BC CIFAR-10	TinyImageNet (crop)	34.7/4.3	19.9/4.7	95.3/99.1	96.4/99.1	93.8/99.1
	TinyImageNet (resize)	40.8/7.5	22.9/6.3	94.1/98.5	95.1/98.6	92.4/98.5
	LSUN (crop)	39.3/8.7	22.2/6.9	94.8/98.2	96.0/98.5	93.1/97.8
	LSUN (resize)	33.6/3.8	19.3/4.4	95.4/99.2	96.4/99.3	94.0/99.2
	iSUN	37.2/6.3	21.1/5.7	94.8/98.8	95.9/98.9	93.1/98.8
	Uniform	23.5/0.0	14.3/2.5	96.5/99.9	97.8/100.0	93.0/99.9
	Gaussian	12.3/0.0	8.7/2.5	97.5/100.0	98.3/100.0	95.9/100.0
Dense-BC CIFAR-100	TinyImageNet (crop)	67.8/17.3	36.4/11.2	83.0/97.1	85.3/97.4	80.8/96.8
	TinyImageNet (resize)	82.2/44.3	43.6/24.6	70.4/90.7	71.4/91.4	68.6/90.1
	LSUN (crop)	69.4/17.6	37.2/11.3	83.7/96.8	86.2/97.1	80.9/96.5
	LSUN (resize)	83.3/44.0	44.1/24.5	70.6/91.5	72.5/92.4	68.0/90.6
	iSUN	84.8/49.5	44.7/27.2	69.9/90.1	71.9/91.1	67.0/88.9
	Uniform	88.3/0.5	46.6/2.8	83.2/99.5	88.1/99.6	73.1/99.0
	Gaussian	95.4/0.2	50.2/2.6	81.8/99.6	87.6/99.7	70.1/99.1
WRN-28-10 CIFAR-10	TinyImageNet (crop)	38.9/23.4	21.9/14.2	92.9/94.2	92.5/92.8	91.9/94.7
	TinyImageNet (resize)	45.6/25.5	25.3/15.2	91.0/92.1	89.7/89.0	89.9/93.6
	LSUN (crop)	35.0/21.8	20.0/13.4	94.5/95.9	95.1/95.8	93.1/95.5
	LSUN (resize)	35.0/17.6	20.0/11.3	93.9/95.4	93.8/93.8	92.8/96.1
	iSUN	40.6/21.3	22.8/13.2	92.5/93.7	91.7/91.2	91.5/94.9
	Uniform	1.6/0.0	3.3/2.5	99.2/100.0	99.3/100.0	98.9/100.0
	Gaussian	0.3/0.0	2.6/2.5	99.5/100.0	99.6/100.0	99.3/100.0
WRN-28-10 CIFAR-100	TinyImageNet (crop)	66.6/43.9	35.8/24.4	82.0/90.8	83.3/91.4	80.2/90.0
	TinyImageNet (resize)	79.2/55.9	42.1/30.4	72.2/84.0	70.4/82.8	70.8/84.4
	LSUN (crop)	74.0/39.6	39.5/22.3	80.3/92.0	83.4/92.4	77.0/91.6
	LSUN (resize)	82.2/56.5	43.6/30.8	73.9/86.0	75.7/86.2	70.1/84.9
	iSUN	82.7/57.3	43.9/31.1	72.8/85.6	74.2/85.9	69.2/84.8
	Uniform	98.2/0.1	51.6/2.5	84.1/99.1	89.9/99.4	71.0/97.5
	Gaussian	99.2/1.0	52.1/3.0	84.3/98.5	90.2/99.1	70.9/95.9

Table of Contents

- **Motivation**
  - Hidden features from DNNs contain meaningful features from training data



  - They can be useful for detecting abnormal samples!

- **Local Intrinsic Dimensionality (LID)** [Ma et al., 2018]
  - Expansion dimension
    - Rate of growth in the number of data encountered as the distance from the re ference sample increases (V is volume)

$$\frac{V_2}{V_1} = \left(\frac{r_2}{r_1}\right)^m \Rightarrow m = \frac{\ln(V_2/V_1)}{\ln(r_2/r_1)}. \quad (1)$$

- **Local Intrinsic Dimensionality (LID)** [Ma et al., 2018]
  - Expansion dimension
    - Rate of growth in the number of data encountered as the distance from the re ference sample increases (V is volume)

$$\frac{V_2}{V_1} = \left(\frac{r_2}{r_1}\right)^m \Rightarrow m = \frac{\ln(V_2/V_1)}{\ln(r_2/r_1)}. \tag{1}$$

  - LID = expansion dimension in the statistical setting

Definition 1 (Local Intrinsic Dimensionality)**.**
Given a data sample $x \in X$, let $R > 0$ be a random variable denoting the distance from x to other data samples. If the cumulative distribution function $F(r)$ of R is positive and continuously differentiable at distance $r > 0$, the LID of x at distance r is given by:

$$\mathrm{LID}_F(r) \triangleq \lim_{\epsilon \to 0} \frac{\ln\left(F((1+\epsilon)\cdot r)/F(r)\right)}{\ln(1+\epsilon)} = \frac{r \cdot F'(r)}{F(r)}, \tag{2}$$

whenever the limit exists.

  - Where F is analogous to the volume in equation (1)

- **Local Intrinsic Dimensionality (LID)** [Ma et al., 2018]
  - Expansion dimension
    - Rate of growth in the number of data encountered as the distance from the re ference sample increases (V is volume)

$$\frac{V_2}{V_1} = \left(\frac{r_2}{r_1}\right)^m \Rightarrow m = \frac{\ln(V_2/V_1)}{\ln(r_2/r_1)}. \tag{1}$$

  - LID = expansion dimension in the statistical setting

Definition 1 (Local Intrinsic Dimensionality)**.**
Given a data sample $x \in X$, let $R > 0$ be a random variable denoting the distance from x to other data samples. If the cumulative distribution function $F(r)$ of R is positive and continuously differentiable at distance $r > 0$, the LID of x at distance r is given by:

$$\mathrm{LID}_F(r) \triangleq \lim_{\epsilon \to 0} \frac{\ln\left(F((1+\epsilon)\cdot r)/F(r)\right)}{\ln(1+\epsilon)} = \frac{r \cdot F'(r)}{F(r)}, \tag{2}$$

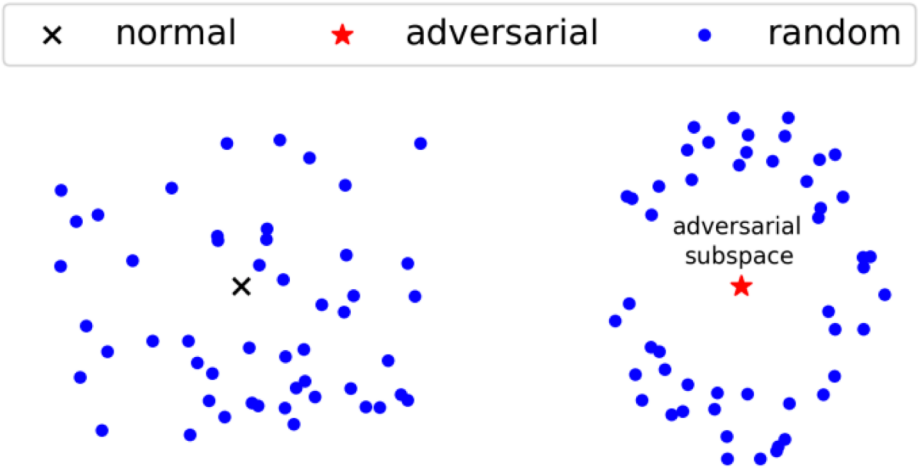
whenever the limit exists.

    - Where F is analogous to the volume in equation (1)
  - Estimation of LID [Amsaleg et al., 2015]

$$\widehat{\mathrm{LID}}(\mathbf{x}) = -\left(\frac{1}{k}\sum_{i=1}^{k} \log \frac{d_i(\mathbf{x})}{d_k(\mathbf{x})}\right)^{-1}$$

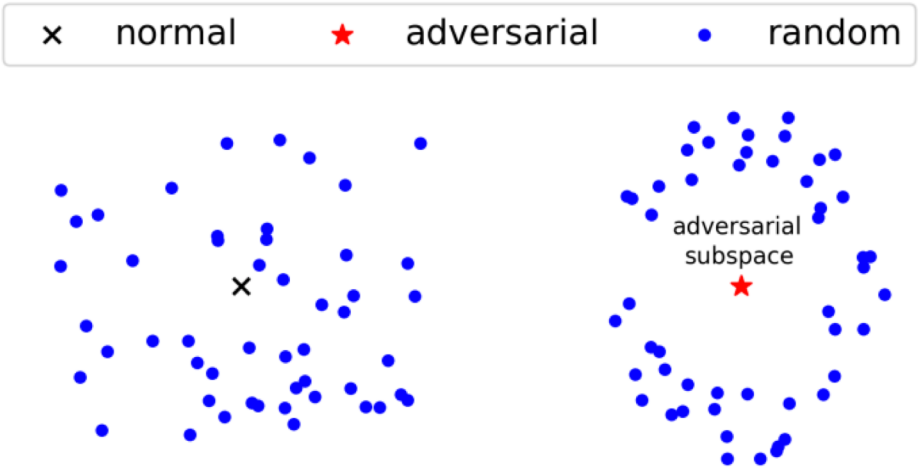
distance between sample and its k-th nearest neighbor

- **Motivation of LID**
  - Abnormal sample might be scattered compared to normal samples

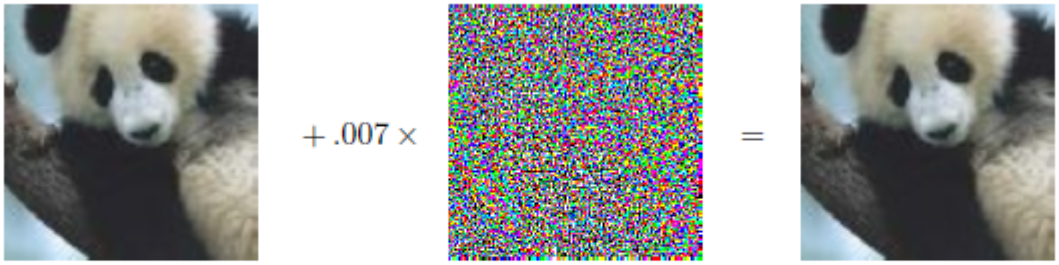


  - This implies that LID can be useful for detecting abnormal samples!

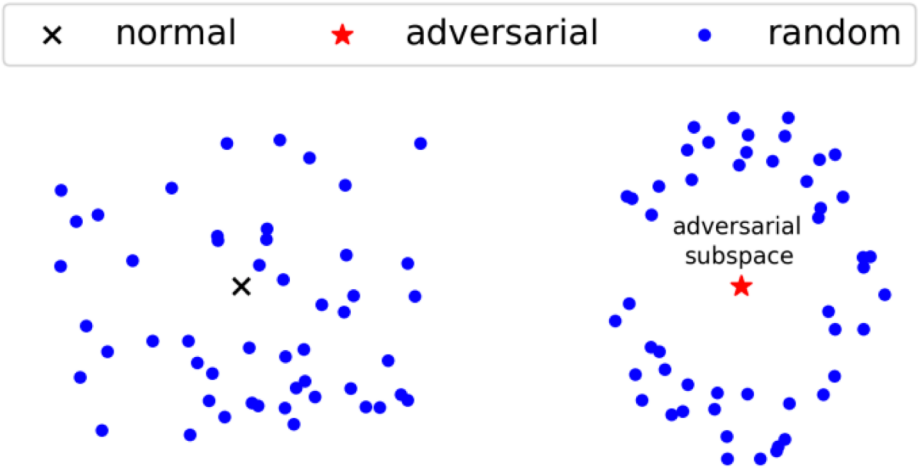
- **Motivation of LID**
  - Abnormal sample might be scattered compared to normal samples



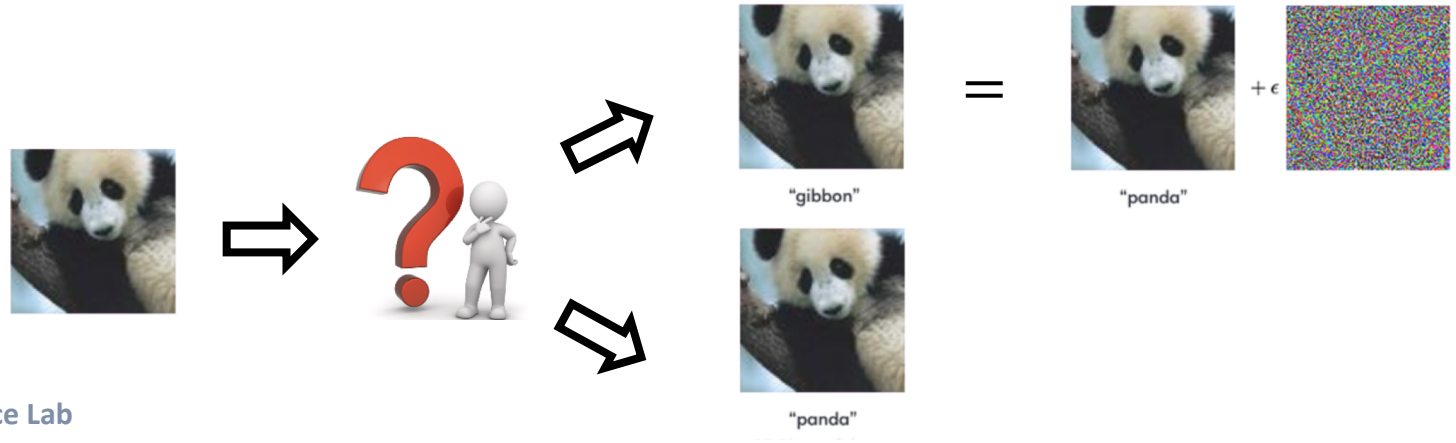
  - This implies that LID can be useful for detecting abnormal samples!
- Evaluation: detecting adversarial samples [Szegedy, et al., 2013]
  - Misclassified examples that are only slightly different from original examples



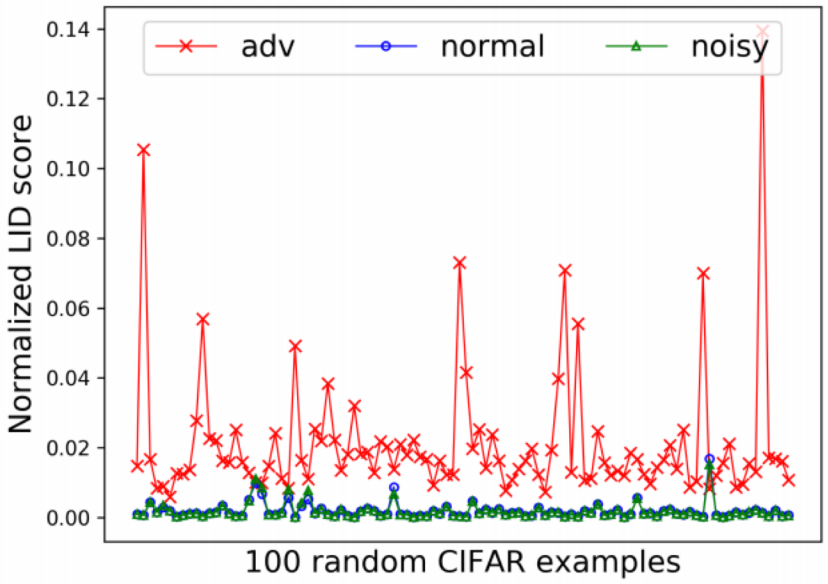
- **Motivation of LID**
  - Abnormal sample might be scattered compared to normal samples

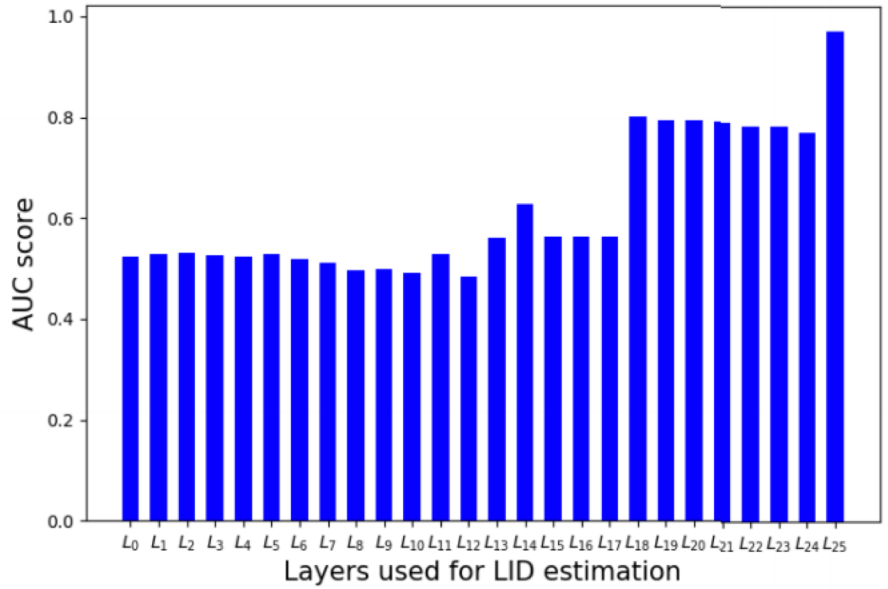


  - This implies that LID can be useful for detecting abnormal samples!
- Evaluation: detecting adversarial samples [Szegedy, et al., 2013]



- **Empirical justification**





- Adversarial samples (generated by OPT attack [Carlini et al., 2017]) can be distinguis hed using LID

- LIDs from low-level layers are also useful in detection

- **Main results on detecting adversarial attacks**
  - Tested method
    - Bayesian uncertainty (BU) and Density estimator (DE) [Feinman et al., 2017]

Table 1: A comparison of the discrimination power (AUC score (%) of a logistic regression classifier) among LID, KD, BU, and KD+BU. The AUC score is computed for each attack strategy on each dataset, and the best results are highlighted in **bold**.

Dataset	Feature	FGM	BIM-a	BIM-b	JSMA	Opt
MNIST	KD	78.12	98.14	98.61	68.77	95.15
	BU	32.37	91.55	25.46	88.74	71.30
	KD+BU	82.43	99.20	98.81	90.12	95.35
	LID	**96.89**	**99.60**	**99.83**	**92.24**	**99.24**
CIFAR-10	KD	64.92	68.38	98.70	85.77	91.35
	BU	70.53	81.60	97.32	87.36	91.39
	KD+BU	70.40	81.33	98.90	88.91	93.77
	LID	**82.38**	**82.51**	**99.78**	**95.87**	**98.94**
SVHN	KD	70.39	77.18	99.57	86.46	87.41
	BU	86.78	84.07	86.93	91.33	87.13
	KD+BU	86.86	83.63	99.52	93.19	90.66
	LID	**97.61**	**87.55**	**99.72**	**95.07**	**97.60**

- LID outperforms all baseline methods

Utilizing the Classifier: Hidden Features

- **Mahalanobis distance-based confidence score** [Lee et al., 2018b]

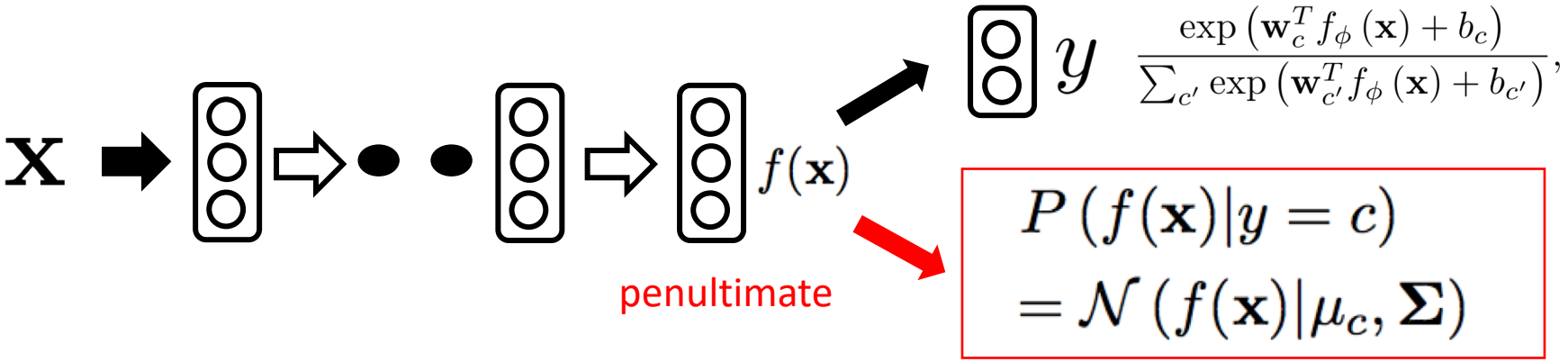
- **Mahalanobis distance-based confidence score** [Lee et al., 2018b]
  - Given pre-trained Softmax classifier with DNNs

$$P_\theta\left(y=c|\mathbf{x}\right)=\frac{\exp\left(\mathbf{w}_c^T f_\phi\left(\mathbf{x}\right)+b_c\right)}{\sum_{c'}\exp\left(\mathbf{w}_{c'}^T f_\phi\left(\mathbf{x}\right)+b_{c'}\right)},$$

- **Mahalanobis distance-based confidence score** [Lee et al., 2018b]
  - Given pre-trained Softmax classifier with DNNs

$$P_\theta\left(y=c|\mathbf{x}\right)=\frac{\exp\left(\mathbf{w}_c^T f_\phi\left(\mathbf{x}\right)+b_c\right)}{\sum_{c'}\exp\left(\mathbf{w}_{c'}^T f_\phi\left(\mathbf{x}\right)+b_{c'}\right)},$$

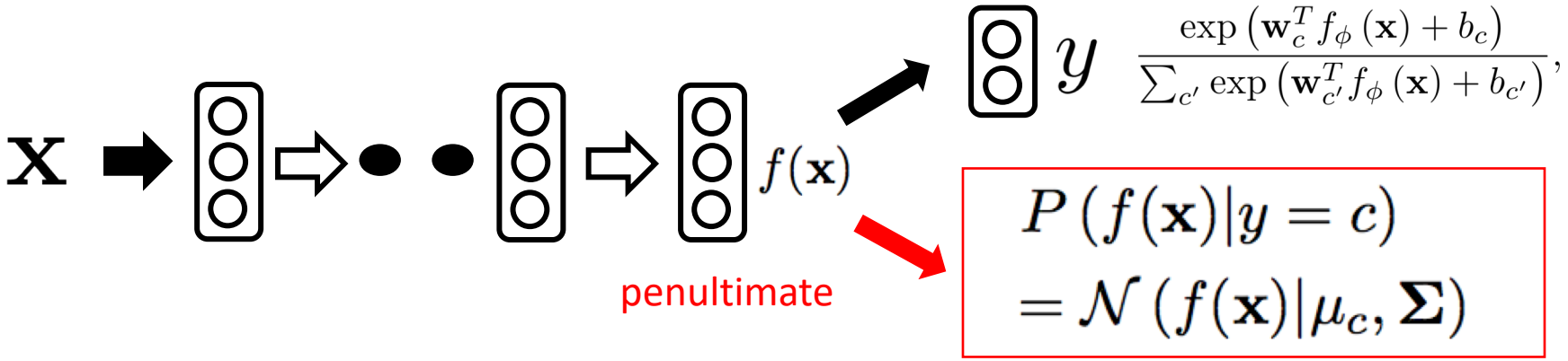
  - Inducing a generative classifier on hidden feature space



- **Mahalanobis distance-based confidence score** [Lee et al., 2018b]
  - Given pre-trained Softmax classifier with DNNs

$$P_\theta\left(y=c|\mathbf{x}\right)=\frac{\exp\left(\mathbf{w}_c^T f_\phi\left(\mathbf{x}\right)+b_c\right)}{\sum_{c'}\exp\left(\mathbf{w}_{c'}^T f_\phi\left(\mathbf{x}\right)+b_{c'}\right)},$$

  - Inducing a generative classifier on hidden feature space



  - Motivation: connection between Softmax and generative classifier (LDA)

$$P_\theta(y=c|\mathbf{x})=\frac{\exp(\mathbf{w}_c\mathbf{x}+b_c)}{\sum_{c'}\exp(\mathbf{w}_{c'}\mathbf{x}+b_{c'})}$$

$$\mathbf{w}_c=\mathbf{\Sigma}^{-1}\mu_c \quad b_c=-0.5\mu_c^T\mathbf{\Sigma}^{-1}\mu_c+\log\pi_c$$

$$\sim =$$

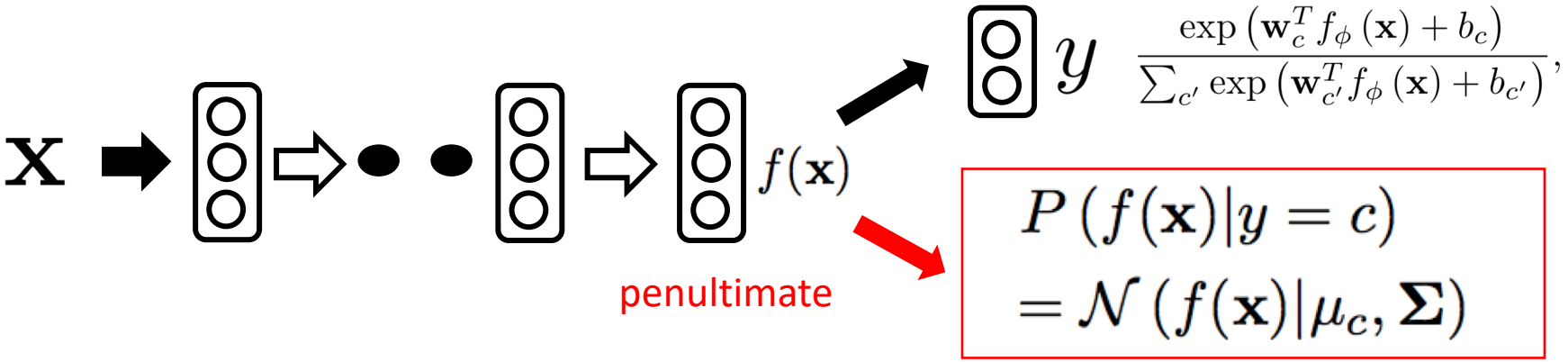
$$P_\theta(y=c|\mathbf{x})=\frac{P_\theta(\mathbf{x}|y=c)P_\theta(y=c)}{\sum_{c'}P_\theta(\mathbf{x}|y=c')P_\theta(y=c')}$$

$$P_\theta(\mathbf{x}|y=c)=\mathcal{N}(\mathbf{x}|\mu_c,\mathbf{\Sigma}), \quad P_\theta(y=c)=\frac{\pi_c}{\sum_{c'}\pi_{c'}}$$

- **Mahalanobis distance-based confidence score** [Lee et al., 2018b]
  - Given pre-trained Softmax classifier with DNNs

$$P_\theta\left(y=c|\mathbf{x}\right)=\frac{\exp\left(\mathbf{w}_c^T f_\phi\left(\mathbf{x}\right)+b_c\right)}{\sum_{c'}\exp\left(\mathbf{w}_{c'}^T f_\phi\left(\mathbf{x}\right)+b_{c'}\right)},$$

  - Inducing a generative classifier on hidden feature space



  - The parameters of generative classifier = sample means and covariance
    - Given training data $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

$$\widehat{\mu}_c=\frac{1}{N_c}\sum_{i:y_i=c} f(\mathbf{x}_i),\quad \widehat{\mathbf{\Sigma}}=\frac{1}{N}\sum_c\sum_{i:y_i=c}\left(f(\mathbf{x}_i)-\widehat{\mu}_c\right)\left(f(\mathbf{x}_i)-\widehat{\mu}_c\right)^\top,$$

Utilizing the Classifier: Hidden Features

- Using generative classifier, we define new confidence score:

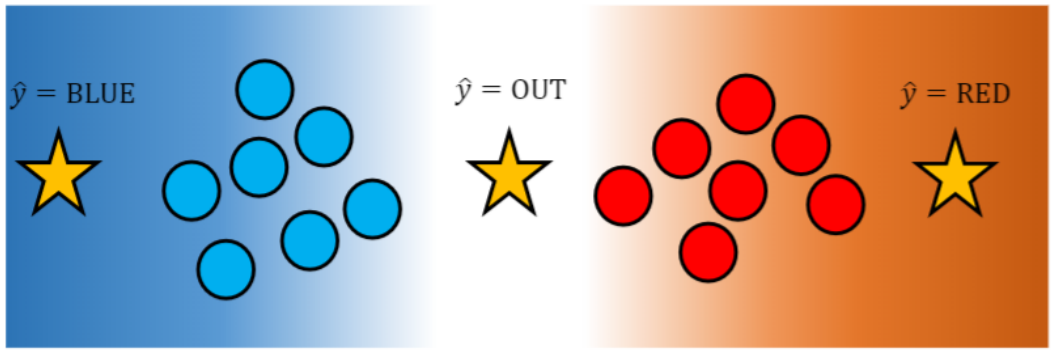
$$M(\mathbf{x}) = \max_c \; -(f(\mathbf{x}) - \widehat{\mu}_c)^\top \widehat{\mathbf{\Sigma}}^{-1} (f(\mathbf{x}) - \widehat{\mu}_c)$$

  - Measuring the log of the probability densities of the test sample

- Using generative classifier, we define new confidence score:

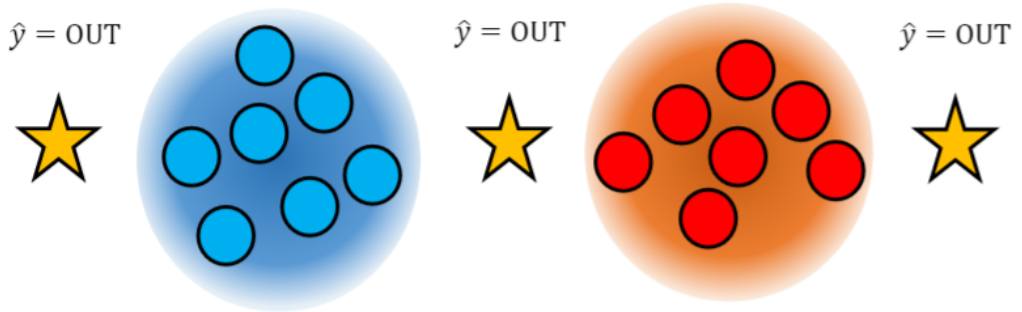
$$M(\mathbf{x}) = \max_c \; -(f(\mathbf{x}) - \hat{\mu}_c)^\top \hat{\mathbf{\Sigma}}^{-1} (f(\mathbf{x}) - \hat{\mu}_c)$$

  - Measuring the log of the probability densities of the test sample
- Intuition



(a) $\max_y P(y|\boldsymbol{x})$

$P(y|\mathbf{x})$



(b) $P(\boldsymbol{x})$

$P(\mathbf{x}|y)$

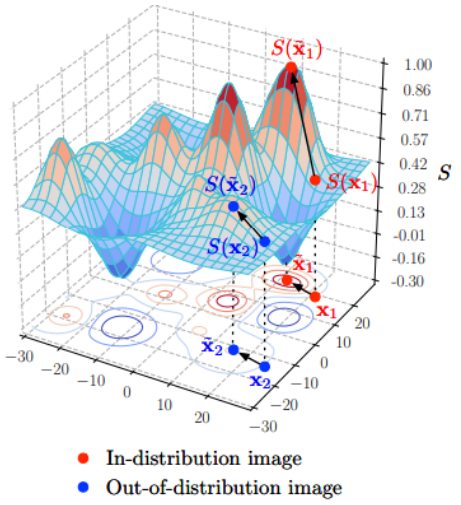
- Using generative classifier, we define new confidence score:

$$M(\mathbf{x}) = \max_c \; -(f(\mathbf{x}) - \widehat{\mu}_c)^\top \widehat{\mathbf{\Sigma}}^{-1} (f(\mathbf{x}) - \widehat{\mu}_c)$$

  - Measuring the log of the probability densities of the test sample
- Boosting the performance
  - Input pre-processing

$$\widehat{\mathbf{x}} = \mathbf{x} + \varepsilon \text{sign}\left(\nabla_{\mathbf{x}} M(\mathbf{x})\right) = \mathbf{x} - \varepsilon \text{sign}\left(\nabla_{\mathbf{x}} (f(\mathbf{x}) - \widehat{\mu}_{\widehat{c}})^\top \widehat{\mathbf{\Sigma}}^{-1} (f(\mathbf{x}) - \widehat{\mu}_{\widehat{c}})\right)$$

    - Motivated by ODIN [Liang et al., 2018]



Utilizing the Classifier: Hidden Features

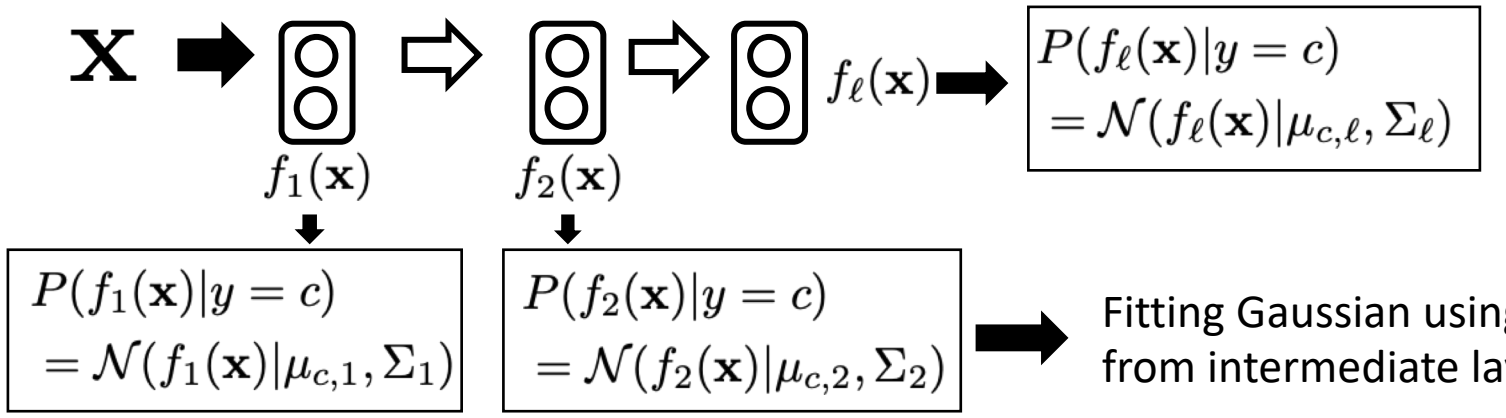
- Using generative classifier, we define new confidence score:

$$M(\mathbf{x}) = \max_c \; -(f(\mathbf{x}) - \widehat{\mu}_c)^\top \widehat{\mathbf{\Sigma}}^{-1} (f(\mathbf{x}) - \widehat{\mu}_c)$$

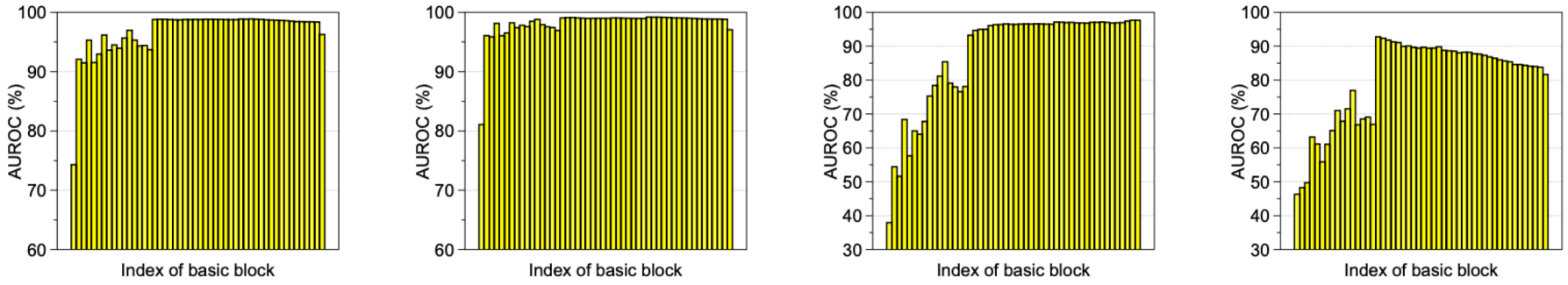
  - Measuring the log of the probability densities of the test sample
- Boosting the performance
  - Input pre-processing

$$\widehat{\mathbf{x}} = \mathbf{x} + \varepsilon \text{sign}\left(\nabla_{\mathbf{x}} M(\mathbf{x})\right) = \mathbf{x} - \varepsilon \text{sign}\left(\nabla_{\mathbf{x}} (f(\mathbf{x}) - \widehat{\mu}_{\widehat{c}})^\top \widehat{\mathbf{\Sigma}}^{-1} (f(\mathbf{x}) - \widehat{\mu}_{\widehat{c}})\right)$$

  - Feature ensemble

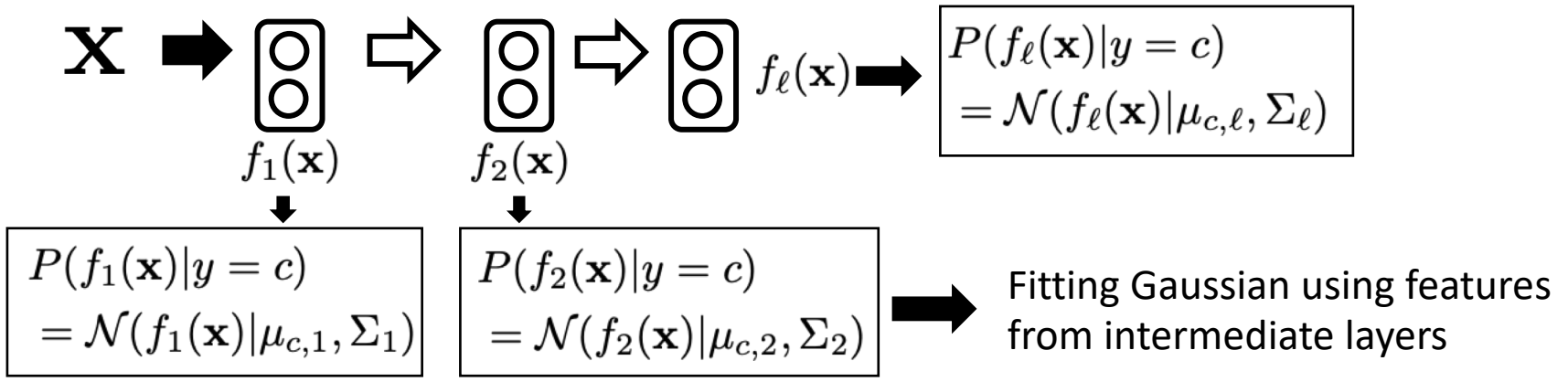


Utilizing the Classifier: Hidden Features



(a) TinyImageNet (b) LSUN (c) SVHN (d) DeepFool

Figure 2: AUROC (%) of threshold-based detector using the confidence score in (2) computed at different basic blocks of DenseNet trained on CIFAR-10 dataset. We measure the detection performance using (a) TinyImageNet, (b) LSUN, (c) SVHN and (d) adversarial (DeepFool) samples.



- Intuition: low-level feature also can be useful for detecting abnormal samples

- Main algorithm

Algorithm 1 Computing the Mahalanobis distance-based confidence score.

Input: Test sample $\mathbf{x}$, weights of logistic regression detector α_ℓ, noise ε and parameters of Gaussian distributions $\{\widehat{\mu}_{\ell,c}, \widehat{\mathbf{\Sigma}}_\ell : \forall \ell, c\}$

Initialize score vectors: $\mathbf{M}(\mathbf{x}) = [M_\ell : \forall \ell]$
for each layer $\ell \in 1, \ldots, L$ **do**
Find the closest class: $\widehat{c} = \arg\min_c \ (f_\ell(\mathbf{x}) - \widehat{\mu}_{\ell,c})^\top \widehat{\mathbf{\Sigma}}_\ell^{-1} (f_\ell(\mathbf{x}) - \widehat{\mu}_{\ell,c})$
Add small noise to test sample: $\widehat{\mathbf{x}} = \mathbf{x} - \varepsilon \text{sign}\left(\nabla_{\mathbf{x}} \left(f_\ell(\mathbf{x}) - \widehat{\mu}_{\ell,\widehat{c}}\right)^\top \widehat{\mathbf{\Sigma}}_\ell^{-1} \left(f_\ell(\mathbf{x}) - \widehat{\mu}_{\ell,\widehat{c}}\right)\right)$
Computing confidence score: $M_\ell = \max_c - \left(f_\ell(\widehat{\mathbf{x}}) - \widehat{\mu}_{\ell,c}\right)^\top \widehat{\mathbf{\Sigma}}_\ell^{-1} \left(f_\ell(\widehat{\mathbf{x}}) - \widehat{\mu}_{\ell,c}\right)$
end for
return Confidence score for test sample $\sum_\ell \alpha_\ell M_\ell$

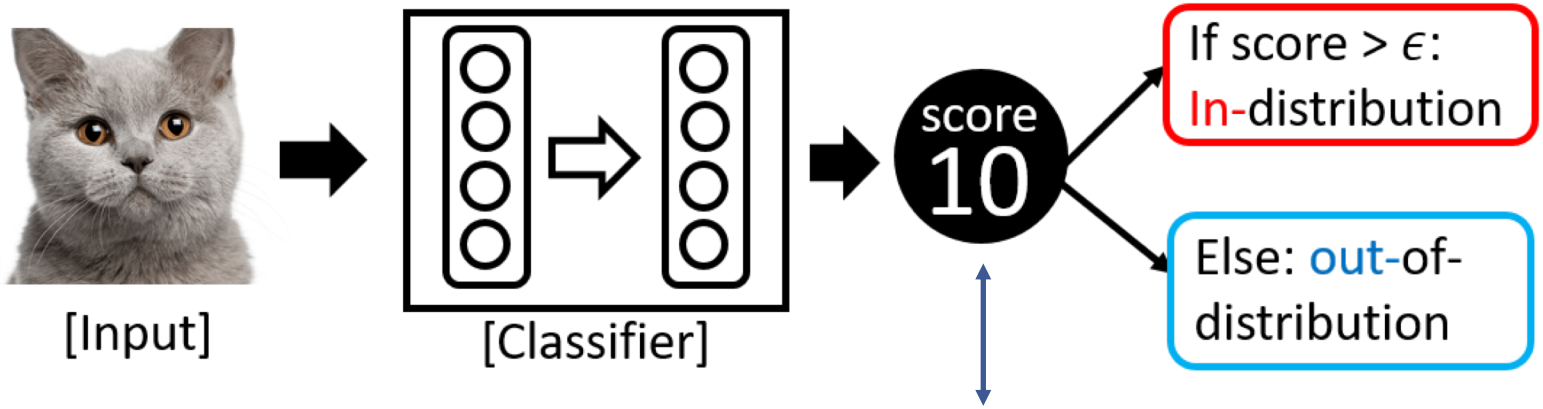
- Remark that
  - We combine the confidence scores from multiple layers using weighted ensemble

$$\sum_\ell \alpha^\ell M^\ell$$

  - Ensemble weights are selected by utilizing the validation set

- Experimental results on detecting out-of-distribution
  - Contribution by each technique

Method	Feature ensemble	Input pre-processing	TNR at TPR 95%	AUROC	Detection accuracy	AUPR in	AUPR out
Baseline [13]	-	-	32.47	89.88	85.06	85.40	93.96
ODIN [21]	-	-	86.55	96.65	91.08	92.54	98.52
Mahalanobis (ours)	-	-	54.51	93.92	89.13	91.56	95.95
	-	✓	92.26	98.30	93.72	96.01	99.28
	✓	-	91.45	98.37	93.55	96.43	99.35
	✓	✓	**96.42**	**99.14**	**95.75**	**98.26**	**99.60**



Baseline [13]: maximum value of posterior distribution
ODIN [21]: maximum value of posterior distribution after post-processing
Ours: the proposed Mahalanobis distance-based score

Utilizing the Classifier: Hidden Features

- Experimental results on detecting out-of-distribution
  - Main results

In-dist (model)	OOD	Validation on OOD samples			Validation on adversarial samples		
		TNR at TPR 95%	AUROC	Detection acc.	TNR at TPR 95%	AUROC	Detection acc.
		Baseline [13] / ODIN [21] / Mahalanobis (ours)			Baseline [13] / ODIN [21] / Mahalanobis (ours)		
CIFAR-10 (DenseNet)	SVHN	40.4 / 77.0 / **91.2**	89.9 / 94.6 / **98.2**	83.2 / 88.1 / **93.5**	40.4 / 49.3 / **79.1**	89.9 / 89.8 / **94.6**	83.2 / 81.7 / **88.9**
	TinyImageNet	59.4 / 92.5 / **95.3**	94.1 / 98.5 / **99.0**	88.5 / 94.0 / **95.3**	59.4 / 92.5 / **94.1**	94.1 / 98.5 / **98.4**	88.5 / 93.9 / **94.6**
	LSUN	66.9 / 96.2 / **97.5**	95.5 / 99.2 / **99.3**	90.2 / 95.6 / **96.5**	66.9 / 96.2 / **96.9**	95.5 / 99.2 / **99.1**	90.2 / 95.6 / **96.1**
CIFAR-100 (DenseNet)	SVHN	26.2 / 56.8 / **82.1**	82.6 / 92.5 / **97.2**	75.5 / 86.0 / **91.4**	26.2 / 39.5 / **50.8**	82.6 / 88.2 / **90.7**	75.5 / 80.7 / **83.8**
	TinyImageNet	17.3 / 43.1 / **86.6**	71.6 / 85.5 / **97.3**	65.7 / 77.3 / **92.0**	17.3 / 43.1 / **86.3**	71.6 / 85.3 / **97.3**	65.7 / 77.2 / **91.5**
	LSUN	16.4 / 41.5 / **91.2**	70.8 / 85.8 / **97.8**	65.0 / 77.5 / **93.8**	16.4 / 41.5 / **89.6**	70.8 / 85.7 / **97.8**	65.0 / 77.4 / **93.1**
SVHN (DenseNet)	CIFAR-10	69.1 / 69.1 / **97.9**	91.8 / 91.8 / **99.1**	86.5 / 86.5 / **96.5**	69.1 / 53.0 / **91.1**	91.8 / 82.0 / **97.4**	86.5 / 76.4 / **93.7**
	TinyImageNet	79.7 / 84.0 / **99.9**	94.8 / 95.1 / **99.9**	90.2 / 90.3 / **99.0**	79.7 / 74.4 / **99.7**	94.8 / 90.7 / **99.7**	90.2 / 85.3 / **98.6**
	LSUN	77.1 / 81.2 / **99.9**	94.1 / 94.5 / **99.9**	89.2 / 89.2 / **99.3**	77.1 / 73.4 / **99.9**	94.1 / 90.5 / **99.9**	89.2 / 84.8 / **99.1**
CIFAR-10 (ResNet)	SVHN	32.2 / 81.9 / **97.4**	89.9 / 95.8 / **99.2**	85.1 / 89.1 / **96.2**	32.2 / 40.4 / **87.8**	89.9 / 86.5 / **97.7**	85.1 / 77.8 / **92.6**
	TinyImageNet	44.1 / 71.9 / **97.8**	91.0 / 93.9 / **99.5**	84.9 / 86.3 / **96.8**	44.1 / 69.5 / **97.1**	91.0 / 93.8 / **99.4**	84.9 / 85.9 / **96.3**
	LSUN	45.1 / 73.8 / **99.3**	91.1 / 94.1 / **99.8**	85.3 / 86.6 / **98.2**	45.1 / 70.1 / **98.8**	91.1 / 93.7 / **99.7**	85.3 / 85.7 / **97.5**
CIFAR-100 (ResNet)	SVHN	19.9 / 68.1 / **92.5**	79.3 / 92.1 / **98.5**	73.2 / 85.1 / **93.9**	19.9 / 18.3 / **80.1**	79.3 / 72.0 / **96.2**	73.2 / 66.7 / **90.3**
	TinyImageNet	20.2 / 49.3 / **90.9**	77.1 / 87.6 / **98.2**	70.8 / 80.0 / **93.4**	20.2 / 46.5 / **88.0**	77.1 / 86.8 / **96.5**	70.8 / 78.9 / **91.9**
	LSUN	18.4 / 45.3 / **91.9**	75.6 / 85.0 / **98.3**	69.8 / 77.8 / **93.9**	18.4 / 43.2 / **85.1**	75.6 / 84.4 / **95.4**	69.8 / 77.0 / **91.0**
SVHN (ResNet)	CIFAR-10	78.3 / 78.3 / **98.6**	92.9 / 92.9 / **99.3**	90.1 / 90.1 / **97.0**	78.3 / 78.3 / **96.0**	92.9 / 92.9 / **98.3**	90.1 / 90.1 / **95.6**
	TinyImageNet	79.1 / 79.1 / **99.9**	93.5 / 93.5 / **99.9**	90.4 / 90.4 / **99.1**	79.1 / 79.1 / **99.3**	93.5 / 93.5 / **99.3**	90.4 / 90.4 / **98.9**
	LSUN	74.5 / 74.5 / **99.9**	91.5 / 91.5 / **99.9**	88.9 / 88.9 / **99.5**	74.5 / 74.5 / **99.9**	91.5 / 91.5 / **99.9**	88.9 / 88.9 / **99.5**

  - For all cases, ours outperforms ODIN and baseline method
  - Validation consists of 1K data from each in- and out-of-distribution pair
  - Validation consists of 1K data from each in- and corresponding FGSM data
    - No information about out-of-distribution

Utilizing the Classifier: Hidden Features

- Experimental results on detecting adversarial attacks
  - Main results

Model	Dataset (model)	Score	Detection of known attack				Detection of unknown attack			
			FGSM	BIM	DeepFool	CW	FGSM (seen)	BIM	DeepFool	CW
DenseNet	CIFAR-10	KD+PU [7]	84.30	98.08	77.23	74.92	84.30	75.69	76.95	72.48
		LID [22]	98.48	100.0	83.36	79.23	98.48	99.50	68.96	65.85
		Mahalanobis (ours)	**99.97**	**100.0**	**83.73**	**85.28**	**99.97**	**99.57**	**83.58**	**84.18**
	CIFAR-100	KD+PU [7]	68.24	84.80	67.60	47.80	68.24	14.91	67.58	52.08
		LID [22]	99.67	99.88	88.37	68.52	99.67	52.38	86.95	64.98
		Mahalanobis (ours)	**99.89**	**100.0**	**91.47**	**80.31**	**99.89**	**100.0**	**90.24**	**76.38**
	SVHN	KD+PU [7]	89.57	98.33	90.94	90.20	89.57	92.08	91.05	90.22
		LID [22]	99.48	99.37	93.42	93.75	99.48	98.50	88.60	84.90
		Mahalanobis (ours)	**99.91**	**99.95**	**96.36**	**96.19**	**99.91**	**99.82**	**94.43**	**95.07**
ResNet	CIFAR-10	KD+PU [7]	84.67	99.66	80.92	70.38	84.67	82.37	80.85	70.41
		LID [22]	99.77	99.88	88.94	80.74	99.77	98.65	87.48	73.12
		Mahalanobis (ours)	**99.99**	**99.99**	**94.21**	**93.33**	**99.99**	**99.95**	**93.58**	**92.58**
	CIFAR-100	KD+PU [7]	73.41	90.55	78.41	67.32	73.41	50.36	78.85	67.36
		LID [22]	99.01	**99.80**	88.88	74.96	99.01	36.46	**87.06**	69.83
		Mahalanobis (ours)	**99.85**	99.48	**93.84**	**86.24**	**99.85**	**99.16**	60.25	**82.87**
	SVHN	KD+PU [7]	86.76	96.16	91.45	84.22	86.76	93.38	91.44	84.37
		LID [22]	97.18	96.39	95.88	86.81	97.18	93.45	93.05	71.92
		Mahalanobis (ours)	**99.24**	**99.40**	**97.17**	**91.06**	**99.24**	**99.10**	**95.60**	**86.09**

- For all tested cases, our method outperforms LID and KD estimator
- For unseen attacks, our method is still working well
  - FGSM samples denoted by “seen” are used for validation

Utilizing the Classifier: Hidden Features

- Detecting OOD samples with the **Gram matrix** [Sastry et al., 2020]
  - Use the Gram matrices to compute the (**hidden**) feature correlations
  - The detect OOD samples that have *dis-similar Gram matrix value*

Utilizing the Classifier: Hidden Features

- Detecting OOD samples with the **Gram matrix** [Sastry et al., 2020]
  - Use the Gram matrices to compute the (**hidden**) feature correlations
  - The detect OOD samples that have *dis-similar Gram matrix value*

- **Gram matrix:** feature correlation
  - Often used for encoding the style information [Gatys et al., 2016]
  - For a given l^{th} layer activation F_l, the Gram matrix is as follows:

$$G_l^p = \left(F_l^p F_l^{p\top} \right)^{\frac{1}{p}}$$

- Detecting OOD samples with the **Gram matrix** [Sastry et al., 2020]
  - Use the Gram matrices to compute the (**hidden**) feature correlations
  - The detect OOD samples that have *dis-similar Gram matrix value*

- **Gram matrix:** feature correlation
  - Often used for encoding the style information [Gatys et al., 2016]
  - For a given l^{th} layer activation F_l, the Gram matrix is as follows:

$$G_l^p = \left(F_l^p F_l^{p\top} \right)^{\frac{1}{p}}$$

- **Detection score**
  - Detect sample that has different Gram matrix value from the training data

$$\delta_l(D) = \sum_{p=1}^{P} \sum_{i=1}^{\frac{1}{2} n_l (n_l+1)} \delta\left(\text{Mins}[D_c][l][p][i], \text{Maxs}[D_c][l][p][i], G_l^p(D)[i] \right)$$

Distance function

Min/Maximum Gram matrix value (of the training dataset)

Test sample's Gram matrix

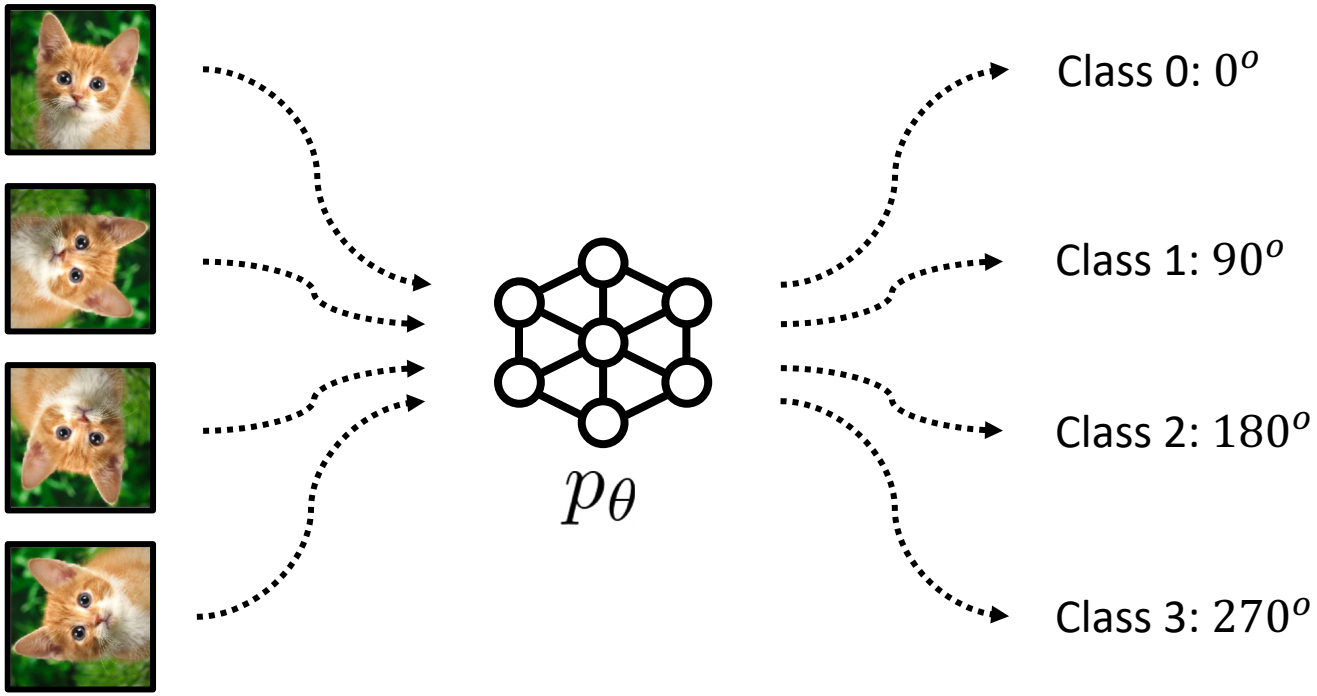
- Experimental results on detecting out-of-distribution
  - Main results

In-dist (model)	OOD	TNR at TPR 95%	AUROC	Detection Acc.
		Baseline / ODIN / Mahalanobis / Ours		
CIFAR-10 (ResNet)	iSUN	44.6 / 73.2 / 97.8 / **99.3**	91.0 / 94.0 / 99.5 / **99.8**	85.0 / 86.5 / 96.7 / **98.1**
	LSUN (R)	49.8 / 82.1 / 98.8 / **99.6**	91.0 / 94.1 / 99.7 / **99.9**	85.3 / 86.7 / 97.7 / **98.6**
	LSUN (C)	48.6 / 62.0 / 81.3 / **89.8**	91.9 / 91.2 / 96.7 / **97.8**	86.3 / 82.4 / 90.5 / **92.6**
	TinyImgNet (R)	41.0 / 67.9 / 97.1 / **98.7**	91.0 / 94.0 / 99.5 / **99.7**	85.1 / 86.5 / 96.3 / **97.8**
	TinyImgNet (C)	46.4 / 68.7 / 92.0 / **96.7**	91.4 / 93.1 / 98.6 / **99.2**	85.4 / 85.2 / 93.9 / **96.1**
	SVHN	50.5 / 70.3 / 87.8 / **97.6**	89.9 / 96.7 / 99.1 / **99.5**	85.1 / 91.1 / 95.8 / **96.7**
	CIFAR-100	33.3 / **42.0** / 41.6 / 32.9	86.4 / 85.8 / **88.2** / 79.0	80.4 / 78.6 / **81.2** / 71.7
CIFAR-100 (ResNet)	iSUN	16.9 / 45.2 / 89.9 / **94.8**	75.8 / 85.5 / 97.9 / **98.8**	70.1 / 78.5 / 93.1 / **95.6**
	LSUN (R)	18.8 / 23.2 / 90.9 / **96.6**	75.8 / 85.6 / 98.2 / **99.2**	69.9 / 78.3 / 93.5 / **96.7**
	LSUN (C)	18.7 / 44.1 / **64.8** / **64.8**	75.5 / 82.7 / 92.0 / **92.1**	69.2 / 75.9 / 84.0 / **84.2**
	TinyImgNet (R)	20.4 / 36.1 / 90.9 / **94.8**	77.2 / 87.6 / 98.2 / **98.9**	70.8 / 80.1 / 93.3 / **95.0**
	TinyImgNet (C)	24.3 / 44.3 / 80.9 / **88.5**	79.7 / 85.4 / 96.3 / **97.7**	72.5 / 78.3 / 89.9 / **92.2**
	SVHN	20.3 / 62.7 / **91.9** / 80.8	79.5 / 93.9 / **98.4** / 96.0	73.2 / 88.0 / **93.7** / 89.6
	CIFAR-10	19.1 / 18.7 / **20.2** / 12.2	77.1 / 77.2 / **77.5** / 67.9	71.0 / 71.2 / **72.1** / 63.4
CIFAR-10 (DenseNet)	iSUN	62.5 / 93.2 / 95.3 / **99.0**	94.7 / 98.7 / 98.9 / **99.8**	89.2 / 94.3 / 95.2 / **97.9**
	LSUN (R)	66.6 / 96.2 / 97.2 / **99.5**	95.4 / 99.2 / 99.3 / **99.9**	90.3 / 95.7 / 96.3 / **98.6**
	LSUN (C)	51.8 / 70.6 / 48.2 / **88.4**	92.9 / 93.6 / 80.2 / **97.5**	86.9 / 86.4 / 75.6 / **92.0**
	TinyImgNet (R)	58.9 / 92.4 / 95.0 / **98.8**	94.1 / 98.5 / 98.8 / **99.7**	88.5 / 93.9 / 95.0 / **97.9**
	TinyImgNet (C)	56.7 / 87.0 / 84.2 / **96.7**	93.8 / 97.6 / 95.3 / **99.3**	88.1 / 92.3 / 89.9 / **96.1**
	SVHN	40.2 / 86.2 / 90.8 / **96.1**	89.9 / 95.5 / 98.1 / **99.1**	83.2 / 91.4 / 93.9 / **95.9**
	CIFAR-100	40.3 / **53.1** / 14.5 / 26.7	89.3 / **90.2** / 58.5 / 72.0	**82.9** / 82.7 / 57.2 / 67.3

- **Gram matrix** achieves the state-of-the-art performance in all tested scenarios **without any OOD validation set**
- Other baselines (e.g., ODIN, Mahalanobis) results are tuned with OOD validation

Table of Contents

- **Self-supervised learning (SSL)** [Doersch et al., 2015]
  - Supervised learning with automatically generated labels (*class label not required*)



- **SSL for uncertainty estimation** [Hendrycks et al., 2019c]

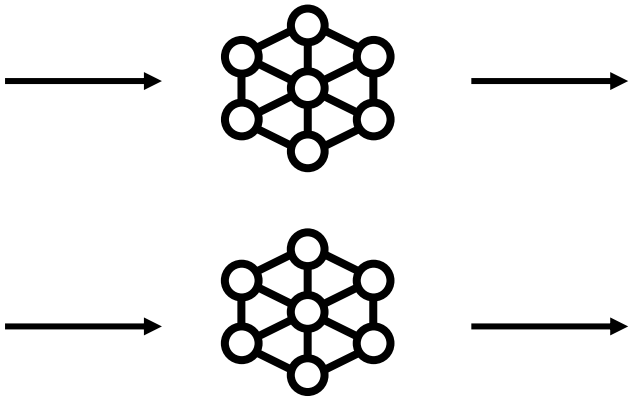
$$\mathcal{L}_{\text{SS}}(x;\theta) = \frac{1}{4}\left[\sum_{r\in\{0°,90°,180°,270°\}} \mathcal{L}_{\text{CE}}(\texttt{one\_hot}(r), p_{\texttt{rot\_head}}(r \mid R_r(x)); \theta)\right]$$

  - Learns to predict the applied transformation (rotation angle)
  - Detect samples that *fail to predict the applied transformation*

- **Intuition**
  - Hard to predict the correct self-supervision label for OOD samples

Seen distribution sample

Unseen distribution sample



- One-class OOD detection (unsupervised OOD detection)
  - Given a dataset consisting in k classes, train a model on one class and use the remaining K-1 classes as out-of-distribution

- Experimental results

		OC-SVM	DeepSVDD	Geometric	RotNet	DIM	IIC	Supervised (OE)	Ours
CIFAR-10 classes	Airplane	65.6	61.7	76.2	71.9	72.6	68.4	87.6	77.5
	Automobile	40.9	65.9	84.8	94.5	52.3	89.4	93.9	96.9
	Bird	65.3	50.8	77.1	78.4	60.5	49.8	78.6	87.3
	Cat	50.1	59.1	73.2	70.0	53.9	65.3	79.9	80.9
	Deer	75.2	60.9	82.8	77.2	66.7	60.5	81.7	92.7
	Dog	51.2	65.7	84.8	86.6	51.0	59.1	85.6	90.2
	Frog	71.8	67.7	82.0	81.6	62.7	49.3	93.3	90.9
	Horse	51.2	67.3	88.7	93.7	59.2	74.8	87.9	96.5
	Ship	67.9	75.9	89.5	90.7	52.8	81.8	92.6	95.2
	Truck	48.5	73.1	83.4	88.8	47.6	75.7	92.1	93.3
	Mean	58.8	64.8	82.3	83.3	57.9	67.4	87.3	90.1

  - Supervised: one class (IN) vs remaining CIFAR-10 classes (OOD)
  - Ours: predict applied rotation + translation (RotNet only predicts rotation)
  - Self-supervised learning outperforms the baseline methods

- One-class OOD detection (unsupervised OOD detection)
  - Given a dataset consisting in k classes, train a model on one class and use the remaining K-1 classes as out-of-distribution

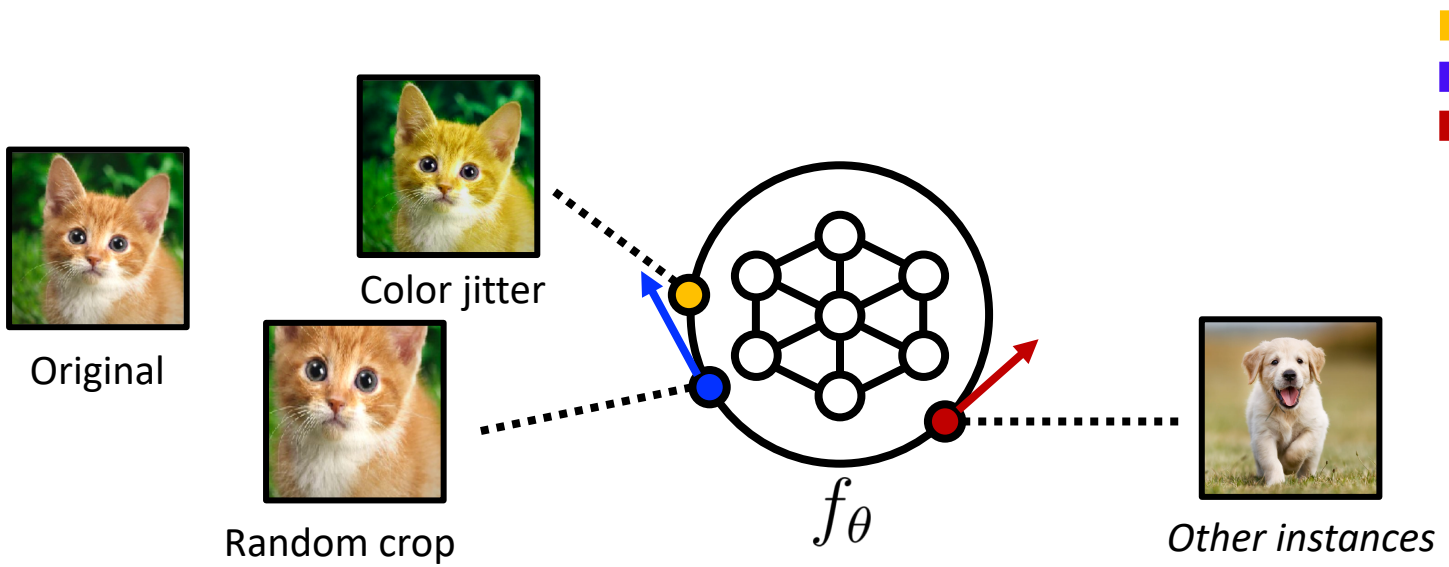
- Experimental results

Method	AUROC
Supervised (OE)	56.1
RotNet	65.3
RotNet + Translation	77.9
RotNet + Self-Attention	81.6
RotNet + Translation + Self-Attention	84.8
RotNet + Translation + Self-Attention + Resize	85.7

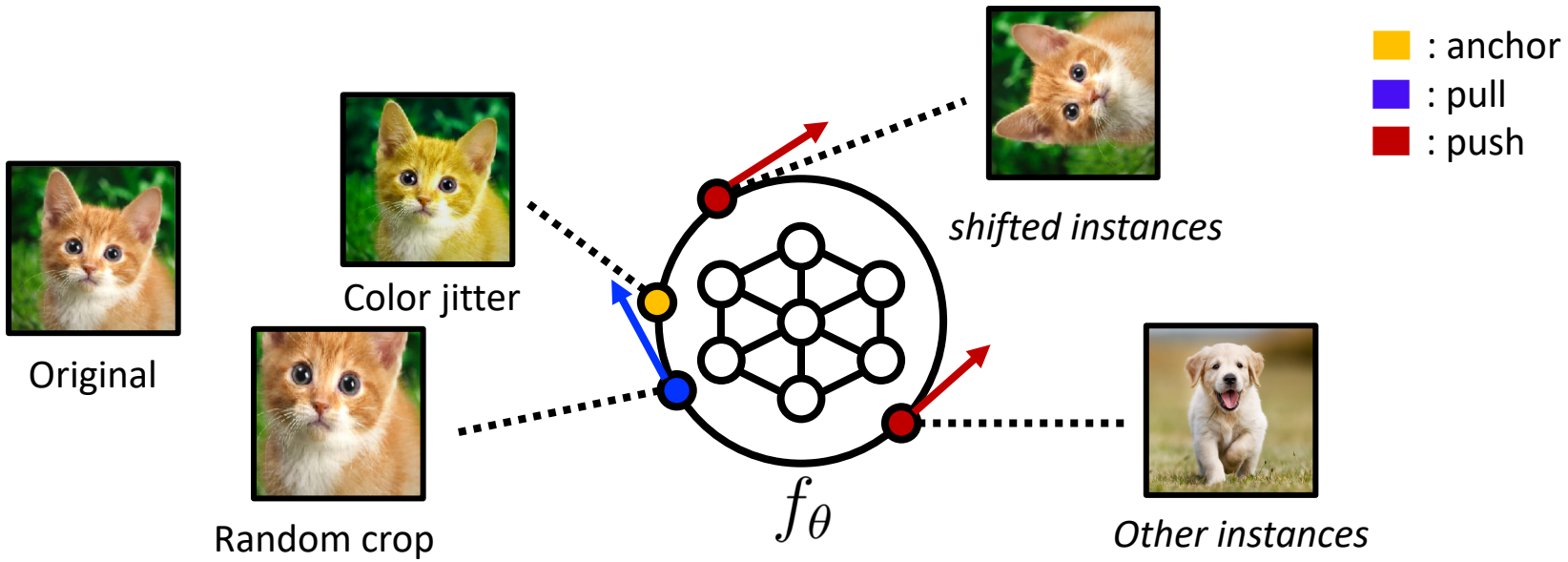
  - Dataset: ImageNet-30 (ImageNet subclass)
  - Supervised: one class (IN) vs ImageNet 22K (OOD)
  - Adding more self-supervision (e.g., self-attention) consistently benefits

- **Contrastive learning**
  - Learn the representation that *encodes the similarity between data points*

- Simple contrastive learning (SimCLR) [Chen et al., 2020]
  - **pull** (i.e., maximize similarity) the same samples of *different augmentations*
  - **push** (i.e., minimize similarity) the different samples



- **Contrasting shifted instances (CSI)** [Tack et al., 2020]
  - Contrasting hard (shifted) augments improve in-vs-out discriminability
  - Found contrastively learned representation is effective at OOD detection
  - CSI further improve OOD detection by
    - **(+) contrasting** (pushing) shifted samples in addition to the different samples
    - **(+) classifying** the shifting transformation



- OOD detection score for CSI
  - Detection score for **contrastively learned representation:**
    - ***cosine similarity*** to the nearest training sample
    - ***norm*** of the representation

$$s_{\text{con}}(x; \{x_m\}) := \max_m \ \text{sim}(z(x_m), z(x)) \cdot \| \ z(x) \ \| \ .$$

*score: cosine similarity * norm*

- OOD detection score for CSI
  - Detection score for **contrastively learned representation:**
    - ***cosine similarity*** to the nearest training sample
    - ***norm*** of the representation

$$s_{\text{con}}(x;\{x_m\}) := \max_m \ \text{sim}(z(x_m), z(x)) \cdot \| \ z(x) \ \| \ .$$

*score: cosine similarity * norm*

  - Further improve the score by **utilizing the shifting transformation:**
    - **(+)** Ensemble the score $s_{con}(x;\{x_m\})$ over all shifting transformation
    - **(+)** Confidence of the shifting transformation classifier

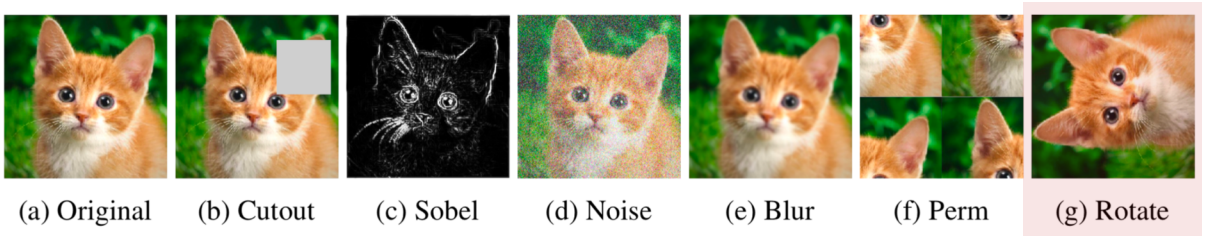
- OOD detection score for CSI
  - Detection score for **contrastively learned representation:**
    - ***cosine similarity*** to the nearest training sample
    - ***norm*** of the representation

$$s_{\text{con}}(x;\{x_m\}) := \max_m \ \text{sim}(z(x_m), z(x)) \cdot \| \ z(x) \ \| \ .$$

*score: cosine similarity * norm*

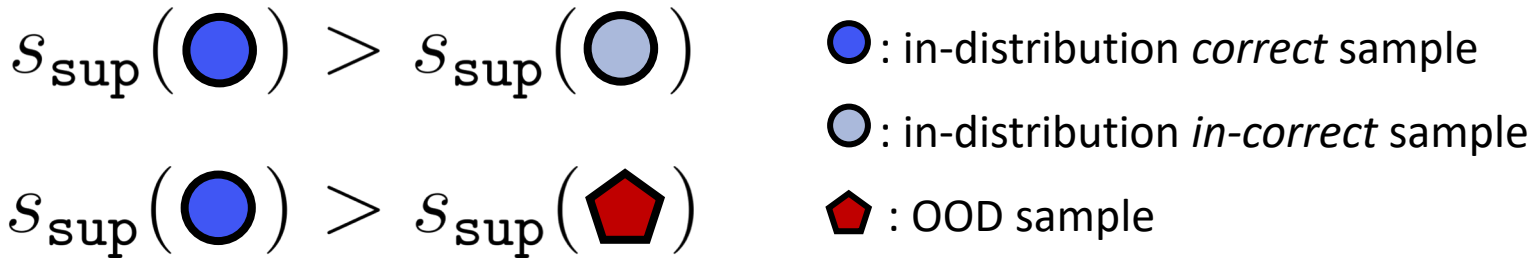
  - Further improve the score by **utilizing the shifting transformation:**
    - **(+)** Ensemble the score $s_{con}(x;\{x_m\})$ over all shifting transformation
    - **(+)** Confidence of the shifting transformation classifier

- Also, provide a method for **choosing the proper shifting transformation**
  - Choose the transformation that generates the most OOD-like samples

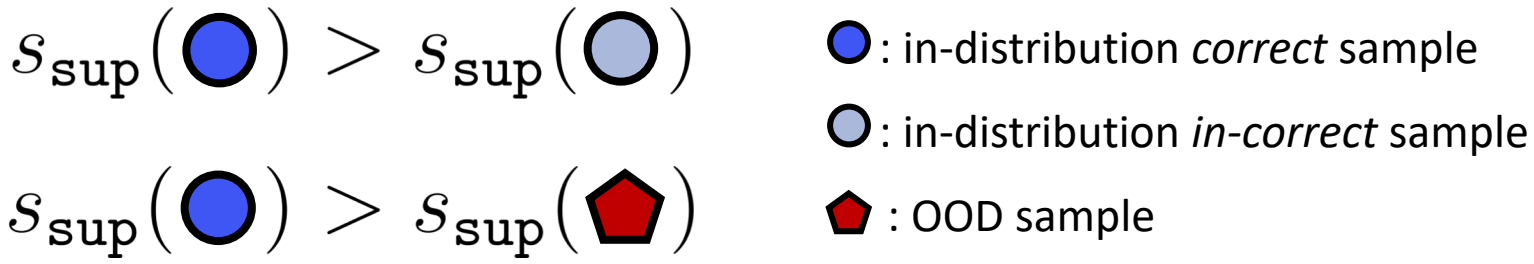


(a) Original (b) Cutout (c) Sobel (d) Noise (e) Blur (f) Perm (g) Rotate

- **CSI can be also extended** for training **confidence-calibrated classifier:**
  - Accurate prediction on label y when input x is in-distribution
  - *Confidence* $s_{\text{sup}}(x) \coloneqq \max_y p(y|x)$ of the classifier is well-calibrated



- **CSI can be also extended** for training **confidence-calibrated classifier:**
  - Accurate prediction on label y when input x is in-distribution
  - *Confidence* $s_{\text{sup}}(x) \coloneqq \max_y p(y|x)$ of the classifier is well-calibrated



- **Adapt CSI** to the supervised contrastive learning (SupCLR) [Khosla et al., 2020]
  - SupCLR contrasts samples in *class-wise*, instead of in instance-wise
  - Similar to CSI, consider the **shifted instance as a different class's sample**

Label: cat

Label: 90^o *rotated cat*

- Experimental results on detecting out-of-distribution
  - Main results

(a) One-class CIFAR-10

Method	Network	Plane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck	Mean
OC-SVM* [64]	-	65.6	40.9	65.3	50.1	75.2	51.2	71.8	51.2	67.9	48.5	58.8
DeepSVDD* [60]	LeNet	61.7	65.9	50.8	59.1	60.9	65.7	67.7	67.3	75.9	73.1	64.8
AnoGAN* [63]	DCGAN	67.1	54.7	52.9	54.5	65.1	60.3	58.5	62.5	75.8	66.5	61.8
OCGAN* [55]	OCGAN	75.7	53.1	64.0	62.0	72.3	62.0	72.3	57.5	82.0	55.4	65.7
Geom* [17]	WRN-16-8	74.7	95.7	78.1	72.4	87.8	87.8	83.4	95.5	93.3	91.3	86.0
Rot* [27]	WRN-16-4	71.9	94.5	78.4	70.0	77.2	86.6	81.6	93.7	90.7	88.8	83.3
Rot+Trans* [27]	WRN-16-4	77.5	96.9	87.3	80.9	92.7	90.2	90.9	96.5	95.2	93.3	90.1
GOAD* [2]	WRN-10-4	77.2	96.7	83.3	77.7	87.8	87.8	90.0	96.1	93.8	92.0	88.2
Rot [27]	ResNet-18	78.3±0.2	94.3±0.3	86.2±0.4	80.8±0.6	89.4±0.5	89.0±0.4	88.9±0.4	95.1±0.2	92.3±0.3	89.7±0.3	88.4
Rot+Trans [27]	ResNet-18	80.4±0.3	96.4±0.2	85.9±0.3	81.1±0.5	91.3±0.3	89.6±0.3	89.9±0.3	95.9±0.1	95.0±0.1	92.6±0.2	89.8
GOAD [2]	ResNet-18	75.5±0.3	94.1±0.3	81.8±0.5	72.0±0.3	83.7±0.9	84.4±0.3	82.9±0.8	93.9±0.3	92.9±0.3	89.5±0.2	85.1
CSI (ours)	ResNet-18	**89.9**±0.1	**99.1**±0.0	**93.1**±0.2	**86.4**±0.2	**93.9**±0.1	**93.2**±0.2	**95.1**±0.1	**98.7**±0.0	**97.9**±0.0	**95.5**±0.1	**94.3**

(b) One-class CIFAR-100 (super-class)

Method	Network	AUROC
OC-SVM* [64]	-	63.1
Geom* [17]	WRN-16-8	78.7
Rot [27]	ResNet-18	77.7
Rot+Trans [27]	ResNet-18	79.8
GOAD [2]	ResNet-18	74.5
CSI (ours)	ResNet-18	**89.6**

(c) One-class ImageNet-30

Method	Network	AUROC
Rot* [27]	ResNet-18	65.3
Rot+Trans* [27]	ResNet-18	77.9
Rot+Attn* [27]	ResNet-18	81.6
Rot+Trans+Attn* [27]	ResNet-18	84.8
Rot+Trans+Attn+Resize* [27]	ResNet-18	85.7
CSI (ours)	ResNet-18	**91.6**

- **CSI** achieves the state-of-the-art performance in **all tested scenarios**
- Unlabeled one-class OOD detection: outperforms prior methods in every classes

- Experimental results on detecting out-of-distribution
  - Main results

(a) Unlabeled CIFAR-10

Method	Network	CIFAR10 → SVHN	LSUN	ImageNet	LSUN (FIX)	ImageNet (FIX)	CIFAR-100	Interp.
Likelihood*	PixelCNN++	8.3	-	64.2	-	-	52.6	52.6
Likelihood*	Glow	8.3	-	66.3	-	-	58.2	58.2
Likelihood*	EBM	63.0	-	-	-	-	-	70.0
Likelihood Ratio* [55]	PixelCNN++	91.2	-	-	-	-	-	-
Input Complexity* [61]	PixelCNN++	92.9	-	58.9	-	-	53.5	-
Input Complexity* [61]	Glow	95.0	-	71.6	-	-	73.6	-
Rot [25]	ResNet-18	97.6±0.2	89.2±0.7	90.5±0.3	77.7±0.3	83.2±0.1	79.0±0.1	64.0±0.3
Rot+Trans [25]	ResNet-18	97.8±0.2	92.8±0.9	94.2±0.7	81.6±0.4	86.7±0.1	82.3±0.2	68.1±0.8
GOAD [2]	ResNet-18	96.3±0.2	89.3±1.5	91.8±1.2	78.8±0.3	83.3±0.1	77.2±0.3	59.4±1.1
CSI (ours)	ResNet-18	**99.8**±0.0	**97.5**±0.3	**97.6**±0.3	**90.3**±0.3	**93.3**±0.1	**89.2**±0.1	**79.3**±0.2

(b) Unlabeled ImageNet-30

Method	Network	ImageNet-30 → CUB-200	Dogs	Pets	Flowers	Food-101	Places-365	Caltech-256	DTD
Rot [25]	ResNet-18	76.5±0.7	77.2±0.5	70.0±0.5	87.2±0.2	72.7±1.5	52.6±1.4	70.9±0.1	89.9±0.5
Rot+Trans [25]	ResNet-18	74.5±0.5	77.8±1.1	70.0±0.8	86.3±0.3	71.6±1.4	53.1±1.7	70.0±0.2	89.4±0.6
GOAD [2]	ResNet-18	71.5±1.4	74.3±1.6	65.5±1.3	82.8±1.4	68.7±0.7	51.0±1.1	67.4±0.8	87.5±0.8
CSI (ours)	ResNet-18	**90.5**±0.1	**97.1**±0.1	**85.2**±0.2	**94.7**±0.4	**89.2**±0.3	**78.3**±0.3	**87.1**±0.1	**96.9**±0.1

- **CSI** achieves the state-of-the-art performance in **all tested scenarios**
- Unlabeled multi-class OOD detection: outperforms prior methods in every OOD datasets

- Experimental results on detecting out-of-distribution
  - Main results

(a) Labeled CIFAR-10

			CIFAR10 →						
Train method	Test acc.	ECE	SVHN	LSUN	ImageNet	LSUN (FIX)	ImageNet (FIX)	CIFAR100	Interp.
Cross Entropy	93.0±0.2	6.44±0.2	88.6±0.9	90.7±0.5	88.3±0.6	87.5±0.3	87.4±0.3	85.8±0.3	75.4±0.7
SupCLR [30]	93.8±0.1	5.56±0.1	97.3±0.1	92.8±0.5	91.4±1.2	91.6±1.5	90.5±0.5	88.6±0.2	75.7±0.1
CSI (ours)	94.8±0.1	4.40±0.1	96.5±0.2	96.3±0.5	96.2±0.4	92.1±0.5	92.4±0.0	90.5±0.1	78.5±0.2
CSI-ens (ours)	**96.1**±0.1	**3.50**±0.1	**97.9**±0.1	**97.7**±0.4	**97.6**±0.3	**93.5**±0.4	**94.0**±0.1	**92.2**±0.1	**80.1**±0.3

(b) Labeled ImageNet-30

			ImageNet-30 →							
Train method	Test acc.	ECE	CUB-200	Dogs	Pets	Flowers	Food-101	Places-365	Caltech-256	DTD
Cross Entropy	94.3	5.08	88.0	96.7	95.0	89.7	79.8	90.5	90.6	90.1
SupCLR [30]	96.9	3.12	86.3	95.6	94.2	92.2	81.2	89.7	90.2	92.1
CSI (ours)	97.0	2.61	93.4	97.7	96.9	96.0	87.0	92.5	91.9	93.7
CSI-ens (ours)	**97.8**	**2.19**	**94.6**	**98.3**	**97.4**	**96.2**	**88.9**	**94.0**	**93.2**	**97.4**

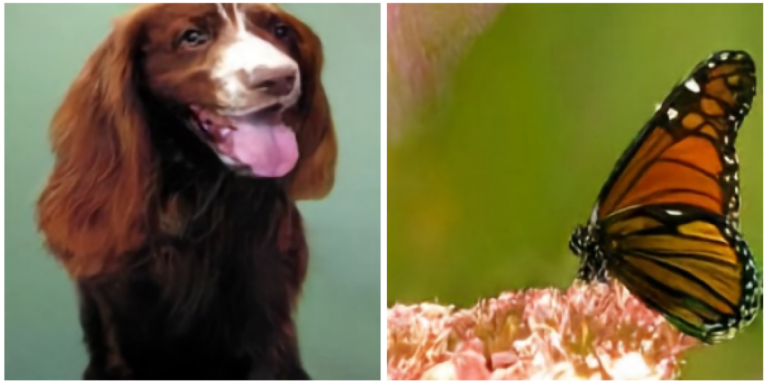
- **CSI** achieves the state-of-the-art performance in **all tested scenarios**
- Labeled multi-class OOD detection: outperforms prior methods in every OOD datasets
- Expected calibration error (ECE) also consistently benefits

Table of Contents

Utilizing the Generative Models: Likelihood

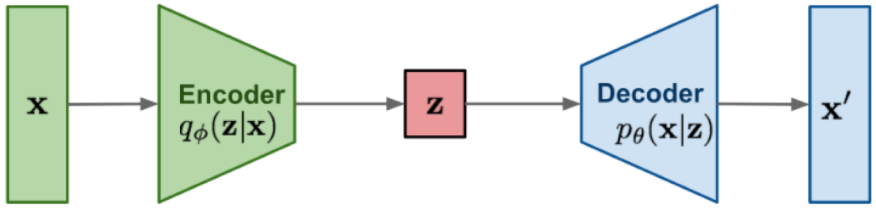
- Generative models such as VAE [Kingma et al., 2014] and GLOW [Kingma et al., 2018] model the data distribution
  - They have achieved the state-of-the-art performances on image generation

GLOW [Kingma et al., 2018]

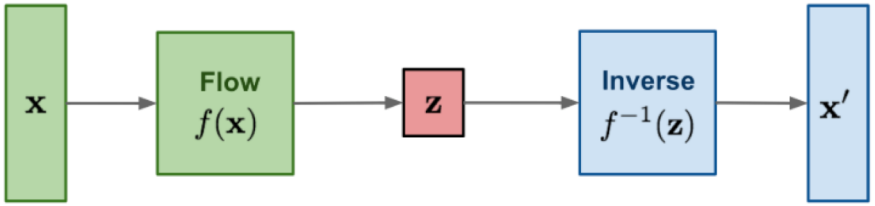


VQ-VAE-2 [Razavi et al., 2019]

VAE: maximize ELBO.



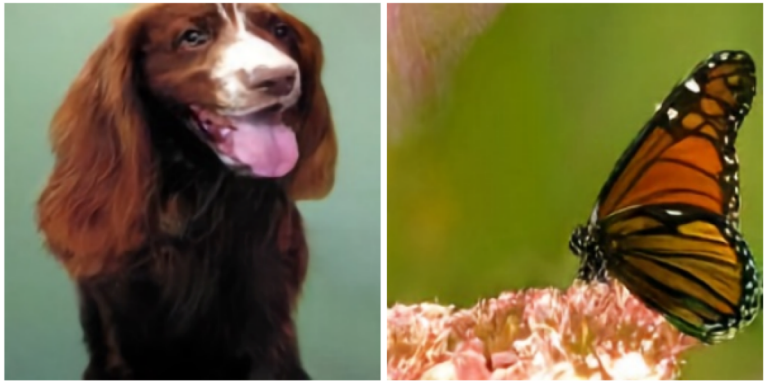
Flow-based generative models: minimize the negative log-likelihood



Utilizing the Generative Models: Likelihood

- Generative models such as VAE [Kingma et al., 2014] and GLOW [Kingma et al., 2018] model the data distribution
  - They have achieved the state-of-the-art performances on image generation

GLOW [Kingma et al., 2018]

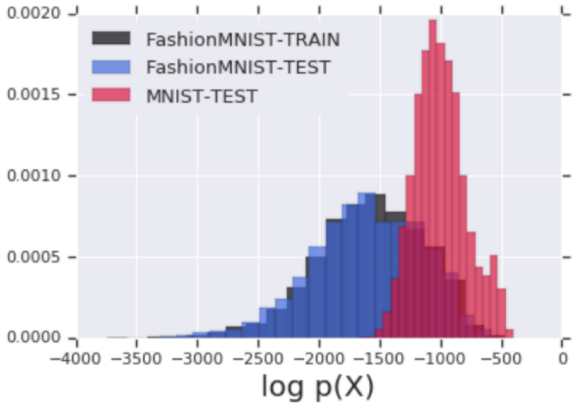


VQ-VAE-2 [Razavi et al., 2019]

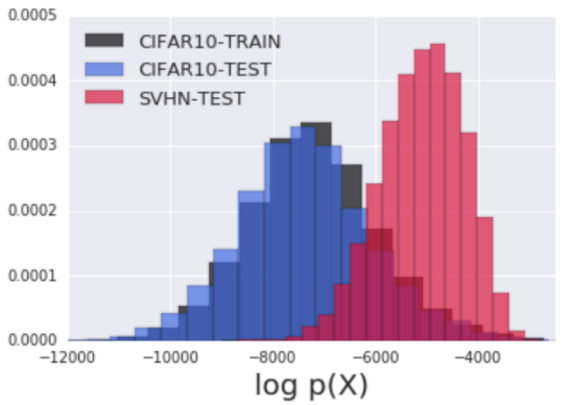
- Questions
  - Are they really capture the data distribution?
  - Are they robust to out-of-distributions?

Utilizing the Generative Models: Likelihood

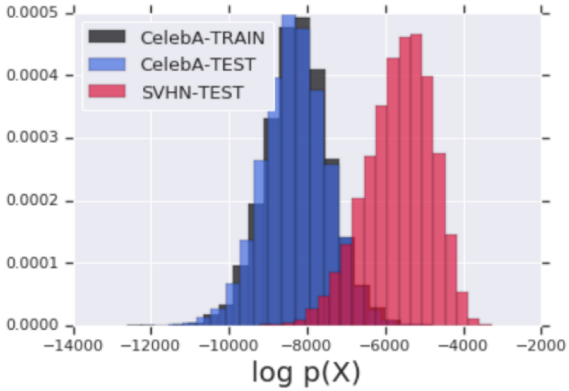
- Generative models are overconfident to out-of-distribution [Nalisnick et al., 2019b]



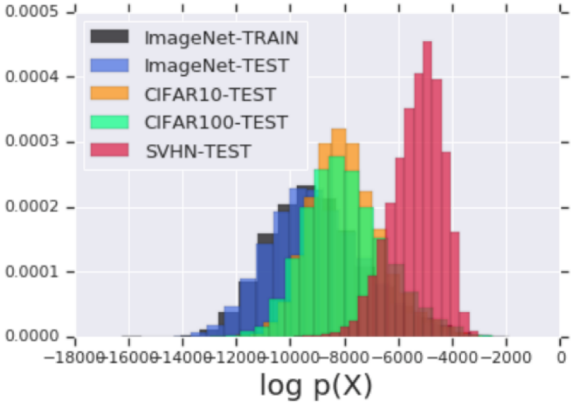
(a) Train on FashionMNIST, Test on MNIST



(b) Train on CIFAR-10, Test on SVHN



(c) Train on CelebA, Test on SVHN



(d) Train on ImageNet,
Test on CIFAR-10 / CIFAR-100 / SVHN

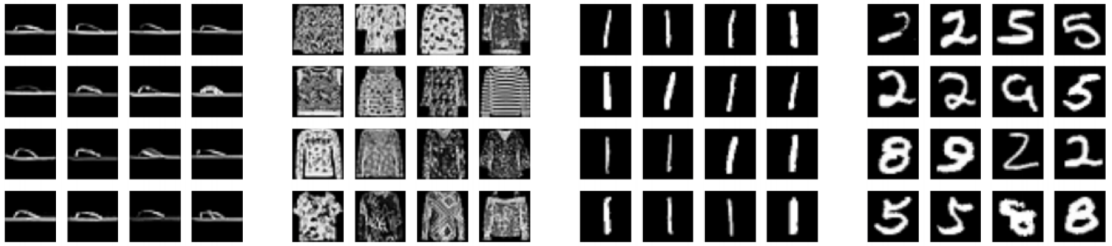
Figure 2: Histogram of Glow log-likelihoods for FashionMNIST vs MNIST (a), CIFAR-10 vs SVHN (b), CelebA vs SVHN (c), and ImageNet vs CIFAR-10 / CIFAR-100 / SVHN (d).

Utilizing the Generative Models: Likelihood

- **Likelihood ratios (LLR)** as a detection score for generative model [Ren et al., 2019]

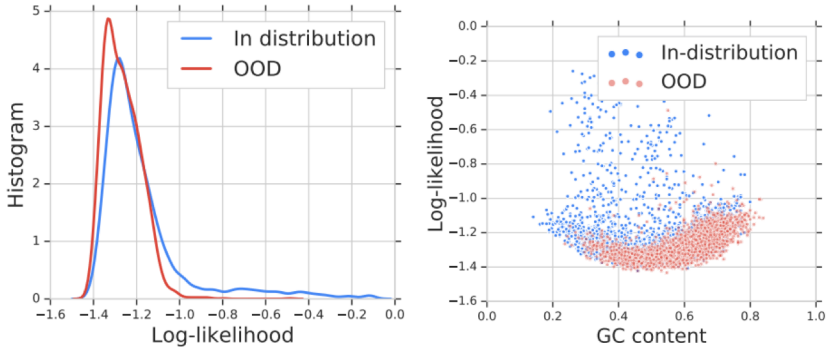
Utilizing the Generative Models: Likelihood

- **Likelihood ratios (LLR)** as a detection score for generative model [Ren et al., 2019]
- *New observation:*
  - likelihood score is heavily affected by **population level background statistics**
  - Example 1) images with a high population of zero pixels have high likelihoods



(a) FashionMNIST: highest log-likelihood. (b) FashionMNIST: lowest log-likelihood. (c) MNIST: highest log-likelihood. (d) MNIST: lowest log-likelihood.

  - Example 2) likelihood of genomic sequences are biased toward {G,C} content



genomic sequences are composed by {A,C,G,T}

Utilizing the Generative Models: Likelihood

- **Likelihood ratios (LLR)** as a detection score for generative model [Ren et al., 2019]
- *Assumption*:
  - 1) Every data can be decomposed into **semantic** and **background** component
  - 2) Each components are **independent**

$$p(\boldsymbol{x}) = p(\boldsymbol{x}_B)p(\boldsymbol{x}_S).$$

Utilizing the Generative Models: Likelihood

- **Likelihood ratios (LLR)** as a detection score for generative model [Ren et al., 2019]
- *Assumption*:
  - 1) Every data can be decomposed into **semantic** and **background** component
  - 2) Each components are **independent**

$$p(\boldsymbol{x}) = p(\boldsymbol{x}_B)p(\boldsymbol{x}_S).$$

- **LLR score for removing the background information**

$$\text{LLR}(\boldsymbol{x}) = \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x})}{p_{\boldsymbol{\theta}_0}(\boldsymbol{x})} = \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_B)\, p_{\boldsymbol{\theta}}(\boldsymbol{x}_S)}{p_{\boldsymbol{\theta}_0}(\boldsymbol{x}_B)\, p_{\boldsymbol{\theta}_0}(\boldsymbol{x}_S)}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_B) \approx p_{\boldsymbol{\theta}_0}(\boldsymbol{x}_B)$$

The goal of background model

- p_θ is trained with in-distribution data, and p_{θ_0} is a model for capturing background
- p_{θ_0} is trained with **noise perturbation**, hence *only capture background populations*

Utilizing the Generative Models: Likelihood

- Experimental results on detecting out-of-distribution
  - **(a)** Fashion-MNIST (IN) vs MNIST (OUT), **(b)** genomic dataset detection (new dataset)

	AUROC↑	AUPRC↑	FPR80↓
Likelihood	0.089 (0.002)	0.320 (0.000)	1.000 (0.001)
Likelihood Ratio (ours, μ)	0.973 (0.031)	0.951 (0.063)	0.005 (0.008)
Likelihood Ratio (ours, μ, λ)	**0.994 (0.001)**	**0.993 (0.002)**	**0.001 (0.000)**
$p(\hat{y}\|\boldsymbol{x})$	0.734 (0.028)	0.702 (0.026)	0.506 (0.046)
Entropy of $p(y\|\boldsymbol{x})$	0.746 (0.027)	0.726 (0.026)	0.448 (0.049)
ODIN	0.752 (0.069)	0.763 (0.062)	0.432 (0.116)
Mahalanobis distance	0.942 (0.017)	0.928 (0.021)	0.088 (0.028)
Ensemble, 5 classifiers	0.839 (0.010)	0.833 (0.009)	0.275 (0.019)
Ensemble, 10 classifiers	0.851 (0.007)	0.844 (0.006)	0.241 (0.014)
Ensemble, 20 classifiers	0.857 (0.005)	0.849 (0.004)	0.240 (0.011)
Binary classifier	0.455 (0.105)	0.505 (0.064)	0.886 (0.126)
$p(\hat{y}\|\boldsymbol{x})$ with noise class	0.877 (0.050)	0.871 (0.054)	0.195 (0.101)
$p(\hat{y}\|\boldsymbol{x})$ with calibrations	0.904 (0.023)	0.895 (0.023)	0.139 (0.044)
WAIC, 5 models	0.221 (0.013)	0.401 (0.008)	0.911 (0.008)

(a)

	AUROC↑	AUPRC↑	FPR80↓
Likelihood	0.626 (0.001)	0.613 (0.001)	0.661 (0.002)
Likelihood Ratio (ours, μ)	0.732 (0.015)	0.685 (0.017)	0.534 (0.031)
Likelihood Ratio (ours, μ, λ)	**0.755 (0.005)**	**0.719 (0.006)**	**0.474 (0.011)**
$p(\hat{y}\|\boldsymbol{x})$	0.634 (0.003)	0.599 (0.003)	0.669 (0.007)
Entropy of $p(y\|\boldsymbol{x})$	0.634 (0.003)	0.599 (0.003)	0.617 (0.007)
Adjusted ODIN	0.697 (0.010)	0.671 (0.012)	0.550 (0.021)
Mahalanobis distance	0.525 (0.010)	0.503 (0.007)	0.747 (0.014)
Ensemble, 5 classifiers	0.682 (0.002)	0.647 (0.002)	0.589 (0.004)
Ensemble, 10 classifiers	0.690 (0.001)	0.655 (0.002)	0.574 (0.004)
Ensemble, 20 classifiers	0.695 (0.001)	0.659 (0.001)	0.570 (0.004)
Binary classifier	0.635 (0.016)	0.634 (0.015)	0.619 (0.025)
$p(\hat{y}\|\boldsymbol{x})$ with noise class	0.652 (0.004)	0.627 (0.005)	0.643 (0.008)
$p(\hat{y}\|\boldsymbol{x})$ with calibration	0.669 (0.005)	0.635 (0.004)	0.627 (0.006)
WAIC, 5 models	0.628 (0.001)	0.616 (0.001)	0.657 (0.002)

(b)

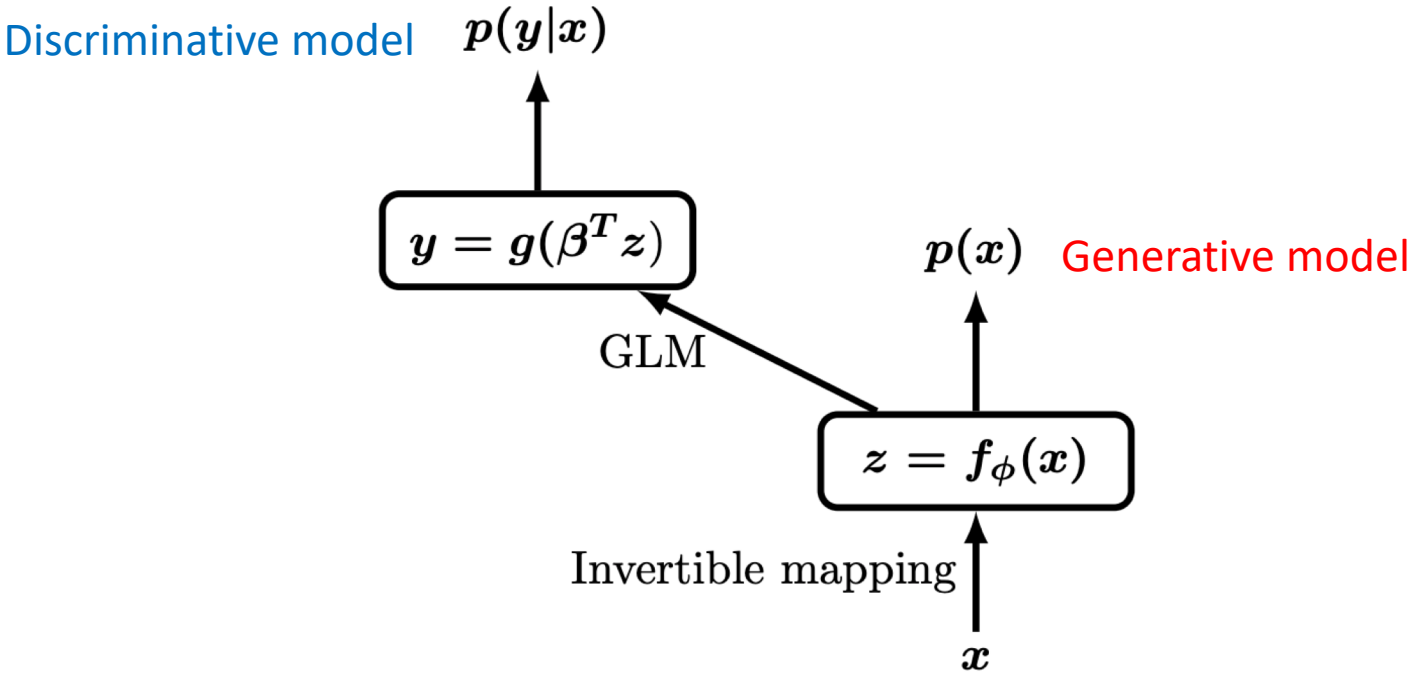
  - Results on CIFAR-10 (IN) vs SVHN (OUT)

	AUROC↑	AUPRC↑	FPR80↓
Likelihood	0.095 (0.003)	0.320 (0.001)	1.000 (0.000)
Likelihood Ratio (ours, μ)	0.931 (0.032)	0.888 (0.049)	0.062 (0.073)
Likelihood Ratio (ours, μ, λ)	0.930 (0.042)	0.881 (0.064)	0.066 (0.123)

*μ: noise strength, λ: l_2 weight decay

- LLR significantly and consistently outperforms the naïve likelihood in all cases
- LLR even shows better result than the classifiers for some datasets (w.o. any label)

- Deep invertible generalized linear model (DIGLM) [Nalisnick et al., 2019a]
  - Hybrid model of generative and discriminative models



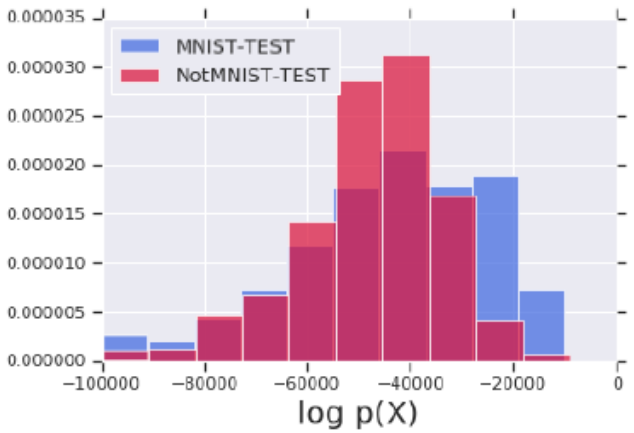
  - Weighted objective

$$\mathcal{J}_\lambda(\boldsymbol{\theta}) = \sum_{n=1}^{N} \Big(\log p(y_n|\boldsymbol{x}_n; \boldsymbol{\beta}, \boldsymbol{\phi}) + \lambda \log p(\boldsymbol{x}_n; \boldsymbol{\phi}) \Big)$$

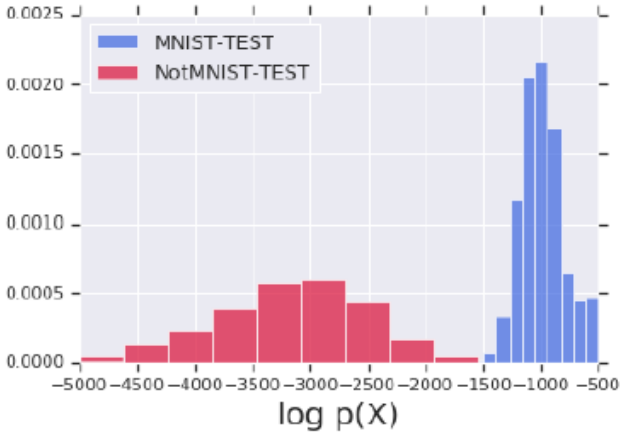
- Bits-per-dimension (BPD), error and negative log likelihood (NLL)

Model	MNIST			NotMNIST		
	BPD ↓	error ↓	NLL ↓	BPD ↑	NLL ↓	Entropy ↑
Discriminative ($\lambda = 0$)	81.80*	**0.67%**	0.082	87.74*	29.27	0.130
Hybrid ($\lambda = 0.01/D$)	1.83	0.73%	**0.035**	5.84	2.36	**2.300**
Hybrid ($\lambda = 1.0/D$)	1.26	2.22%	0.081	6.13	**2.30**	**2.300**
Hybrid ($\lambda = 10.0/D$)	**1.25**	4.01%	0.145	**6.17**	**2.30**	**2.300**

Model	SVHN			CIFAR-10		
	BPD ↓	error ↓	NLL ↓	BPD ↑	NLL ↓	Entropy ↑
Discriminative ($\lambda = 0$)	15.40*	**4.26%**	**0.225**	15.20*	4.60	0.998
Hybrid ($\lambda = 0.1/D$)	3.35	4.86%	0.260	7.06	5.06	1.153
Hybrid ($\lambda = 1.0/D$)	2.40	5.23%	0.253	6.16	4.23	1.677
Hybrid ($\lambda = 10.0/D$)	**2.23**	7.27%	0.268	**7.03**	**2.69**	**2.143**

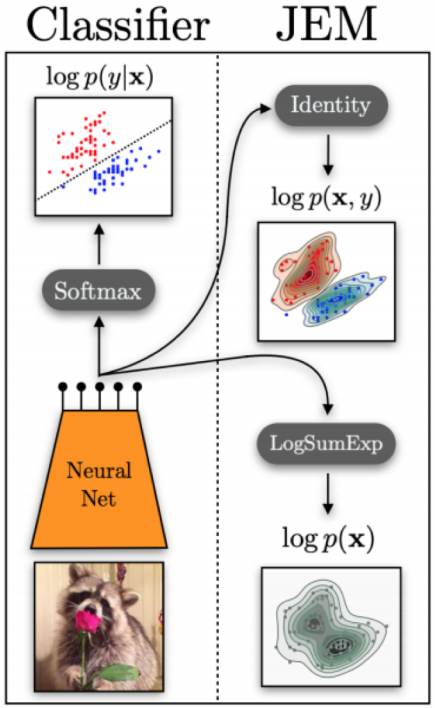


(a) Discriminative Model ($\lambda = 0$)



(b) Hybrid Model

- **Joint energy-based model (JEM)** [Grathwohl et al., 2020]



$$\log p_\theta(\mathbf{x}, y) = \log p_\theta(\mathbf{x}) + \log p_\theta(y|\mathbf{x})$$

Energy-based model (EBM) **Cross-entropy**

$$E_\theta(\mathbf{x}) = -\text{LogSumExp}_y(f_\theta(\mathbf{x})[y]) = -\log \sum_y \exp(f_\theta(\mathbf{x})[y])$$

Assume the energy function as
logit sum exponential of class logits

- **Recap**: jointly models the (unnormalized) likelihood and class probability
- Effective at OOD detection than both generative and discriminative models.

- Experimental results on detecting out-of-distribution
  - Main results

$s_\theta(\mathbf{x})$	Model	SVHN	CIFAR10 Interp	CIFAR100	CelebA
$\log p(\mathbf{x})$	Unconditional Glow	.05	.51	.55	.57
	Class-Conditional Glow	.07	.45	.51	.53
	IGEBM	.63	**.70**	.50	.70
	JEM (Ours)	**.67**	.65	**.67**	**.75**
$\max_y p(y\|\mathbf{x})$	Wide-ResNet	**.93**	**.77**	.85	.62
	Class-Conditional Glow	.64	.61	.65	.54
	IGEBM	.43	.69	.54	.69
	JEM (Ours)	.89	.75	**.87**	**.79**
$\left\|\left\|\frac{\partial \log p(\mathbf{x})}{\partial \mathbf{x}}\right\|\right\|$	Unconditional Glow	**.95**	.27	.46	.29
	Class-Conditional Glow	.47	.01	.52	.59
	IGEBM	.84	.65	.55	.66
	JEM (Ours)	.83	**.78**	**.82**	**.79**

- **JEM** achieves outperforms generative/discriminative/hybrid models in most cases under various score functions
- Interp. denotes the interpolation dataset of CIFAR-10 (i.e., mixup)

Summary

- In this lecture, we cover various methods for detecting abnormal samples like o ut-of-distribution and adversarial samples
  - Posterior distribution-based methods
  - Hidden feature-based methods
  - Self-supervised learning based methods
  - Generative model based methods

- There are also training methods for obtaining more calibrated scores
  - Ensemble of classifier [Balaji et al., 2017]
  - Bayesian deep models [Li et al., 2017]
  - Calibration loss with GAN [Lee et al., 2018a]
  - Calibration loss for generative models [Hendrycks' 19a]

- Such methods can be useful for many machine learning applications
  - Active learning [Gal et al., 2017]
  - Incremental learning [Rebuff et al., 2017]
  - Ensemble learning [Lee et al., 2017]
  - Network calibration [Guo et al., 2017]

References

[Hendrycks et al., 2017] A baseline for detecting misclassified and out-of-distribution examples in neural networks. In ICLR 2017. https://arxiv.org/abs/1610.02136

[Ma et al., 2018] Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality. In ICLR, 2018. https://openreview.net/pdf?id=B1gJ1L2aW

[Feinman et al., 2017] Detecting adversarial samples from artifacts. *arXiv preprint*, 2017. https://arxiv.org/abs/1703.00410

[Lee, et al., 2018a] Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples, In ICLR, 2018. https://arxiv.org/abs/1711.09325

[Lee, et al., 2018b] A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks, In NeurIPS, 2018. https://arxiv.org/abs/1807.03888

[Liang, et al., 2018] Principled Detection of Out-of-Distribution Examples in Neural Networks. In ICLR, 2018. *https://arxiv.org/abs/1706.02690*

[Goodfellow et al., 2015] Explaining and harnessing adversarial examples. In ICLR, *2015.* https://arxiv.org/pdf/1412.6572.pdf

[Amodei, et al., 2016] Concrete problems in ai safety. *arXiv preprint*, 2016. https://arxiv.org/abs/1606.06565

[Guo et al., 2017] On Calibration of Modern Neural Networks. In ICML, 2017. https://arxiv.org/abs/1706.04599

[Lee et al., 2017] Confident Multiple Choice Learning. In ICML, 2017. https://arxiv.org/abs/1706.03475

[Balaji et al., 2017] Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles, In NeurIPS, 2017. https://arxiv.org/pdf/1612.01474.pdf

References

[Rebuff et al., 2017] iCaRL: Incremental Classifier and Representation Learning. In CVPR, 2017.
https://arxiv.org/pdf/1611.07725.pdf

[Huang et al., 2017] Densely connected convolutional networks, In CVPR, 2017.
https://arxiv.org/abs/1608.06993

[Zagoruyko et al., 2016] Wide residual networks, In BMVC 2016.
https://arxiv.org/pdf/1605.07146.pdf

[Amsaleg et al., 2015] Estimating local intrinsic dimensionality. In SIGKDD, 2015.
http://mistis.inrialpes.fr/~girard/Fichiers/p29-amsaleg.pdf

[Szegedy et al., 2013] Intriguing properties of neural networks. *arXiv preprint*, 2013.
https://arxiv.org/abs/1312.6199

[Li et al., 2017] Dropout Inference in Bayesian Neural Networks with Alpha-divergences, In ICML, 2017.
https://arxiv.org/abs/1703.02914

[Gal et al., 2017] Deep Bayesian Active Learning with Image Data, In ICML, 2017.
https://arxiv.org/abs/1703.02910

[Carlini et al., 2017] Towards evaluating the robustness of neural networks. In IEEE SP, 2017.
https://arxiv.org/abs/1608.04644

[Nalisnick et al., 2019a] Hybrid Models with Deep and Invertible Features. In ICML 2019.
https://arxiv.org/pdf/1902.02767

[Nalisnick et al., 2019b] Do Deep Generative Models Know What They Don't Know. In ICLR 2019.
https://arxiv.org/pdf/1906.02994

[Kingma et al., 2014] Auto-Encoding Variational Bayes. In ICLR, 2014.
https://arxiv.org/pdf/1807.03039

References

[Kingma et al., 2018] Glow: Generative Flow with Invertible 1×1 Convolutions. In NeurIPS, 2018. https://arxiv.org/pdf/1807.03039

[Razavi et al., 2019] Generating Diverse High-Fidelity Images with VQ-VAE-2. *arXiv preprint*, 2019. https://arxiv.org/abs/1906.00446

[Ren et al., 2019b] Likelihood Ratios for Out-of-Distribution Detection. In NeurIPS 2019. https://arxiv.org/abs/1906.02845

[Grathwohl et al., 2020] Your Classifier is Secretly an Energy Based Model and You Should Treat it Like One. In ICLR 2020. https://arxiv.org/abs/1912.03263

[Hendrycks' 19a] Deep Anomaly Detection with Outlier Exposure In ICLR, 2019a. https://arxiv.org/pdf/1812.04606

[Hendrycks et al., 2019b] Using Pre-training Can Improve Model Robustness and Uncertainty. In ICML, 2019b. https://arxiv.org/abs/1901.09960

[Hendrycks et al., 2019c] Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty. In NeurIPS, 2019c. https://arxiv.org/abs/1901.09960

[Doersch et al., 2015] Unsupervised visual representation learning by context prediction. In ICCV, 2015. https://arxiv.org/abs/1505.05192

[Gatys et al., 2016] Image Style Transfer Using Convolutional Neural Networks. In CVPR, 2016. https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Gatys_Image_Style_Transfer_CVPR_2016_paper.html

[Sastry et al., 2020] Detecting Out-of-Distribution Examples with Gram Matrices. In ICML, 2020. https://arxiv.org/abs/1912.12510

[Chen et al., 2020] A Simple Framework for Contrastive Learning of Visual Representations. In ICML, 2020. https://arxiv.org/abs/2002.05709

References

[Tack et al., 2020] CSI: Novelty Detection via Contrastive Learning on Distributionally Shifted Instances. In NeurIPS, 2020. https://arxiv.org/abs/2007.08176