# Novelty and Uncertainty Estimation

AI602: Recent Advances in Deep Learning

Lecture 10

Slide made by

Kimin Lee

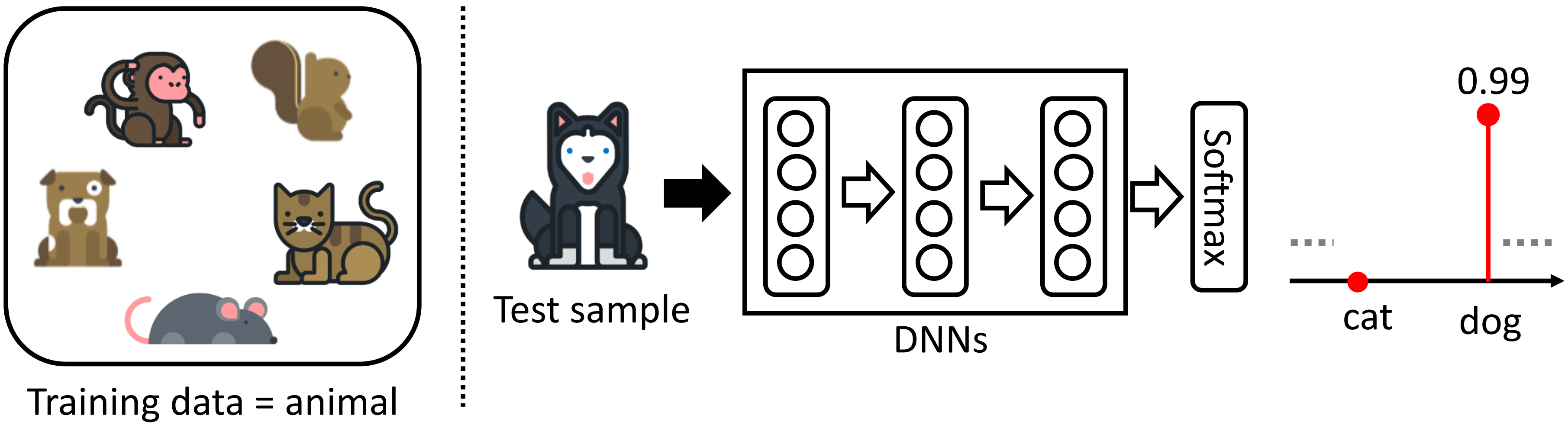
KAIST EE

# Table of Contents

## Table of Contents

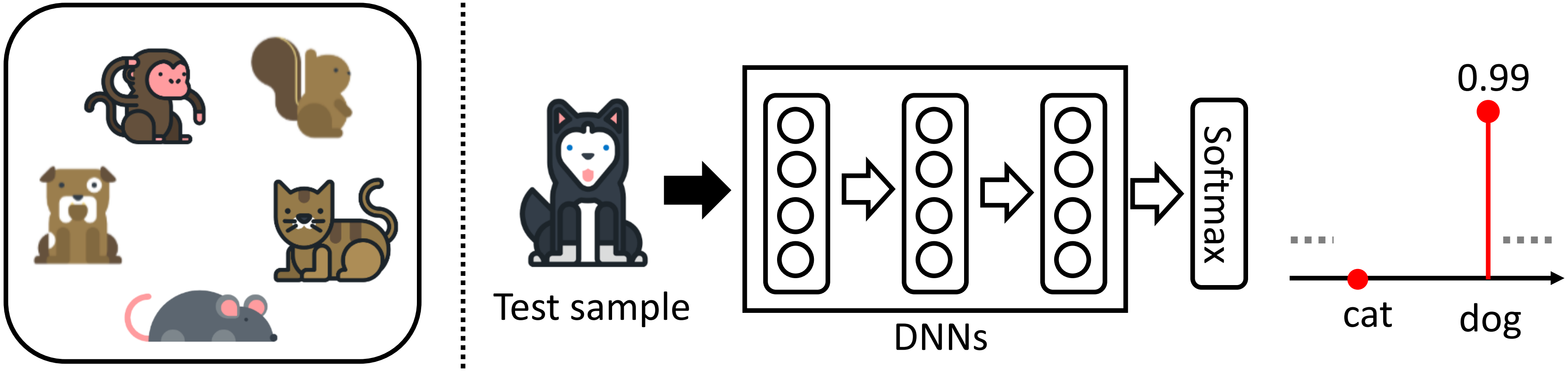
## What is Novelty Detection?

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



## 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

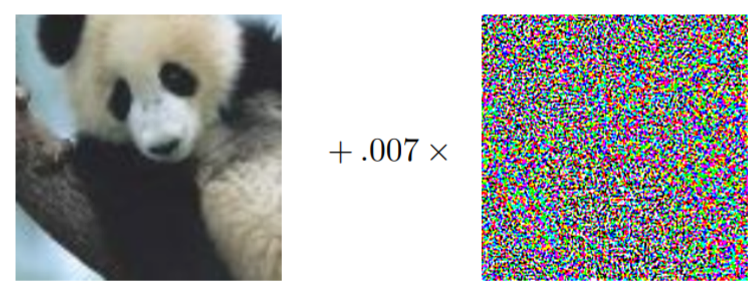



Training data = animal

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

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

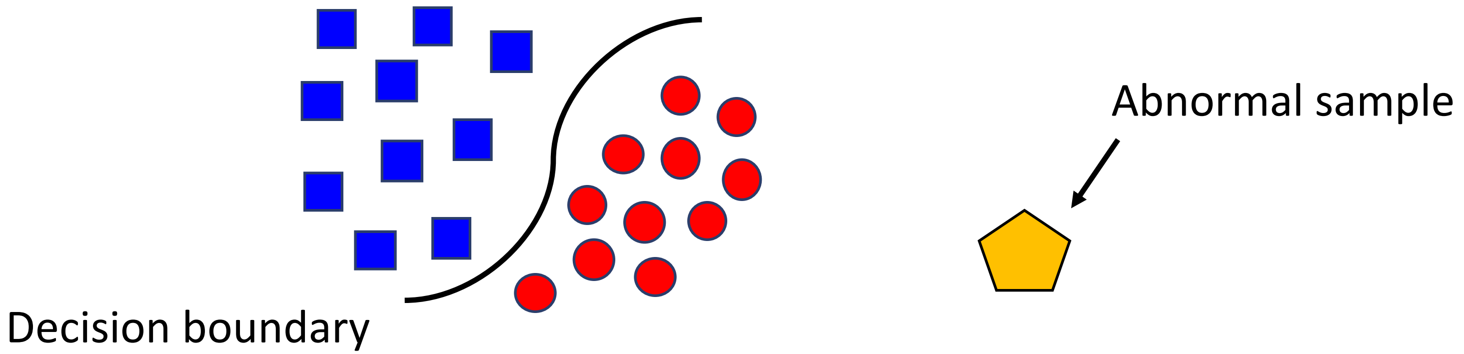
Unknown sample



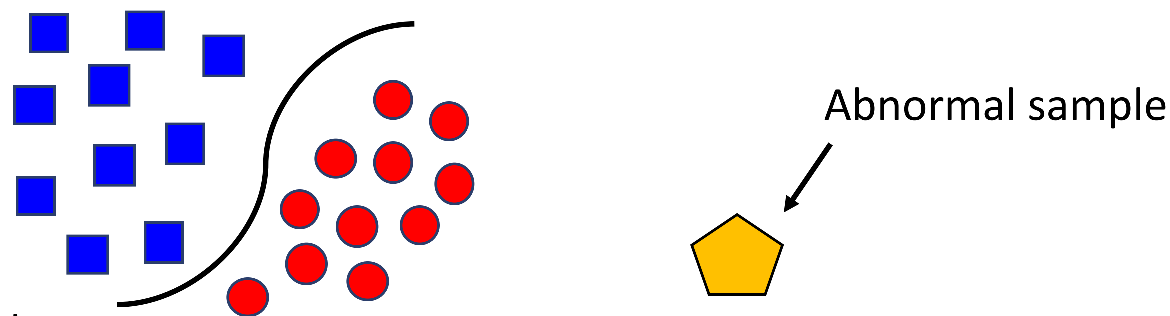

Adversarial samples [Goodfellow et al., 2015]

## 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)

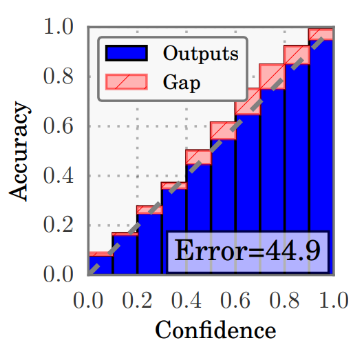


- **Novelty detection**
  - Detect whether a test sample is from in-distribution (i.e., training distribution by classifier) or not (e.g., out-of-distribution / adversarial samples)



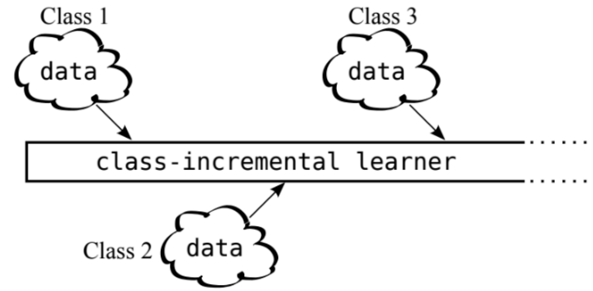

Decision boundary

- It can be useful for many machine learning problems:




Calibration
[Guo et al., 2017]

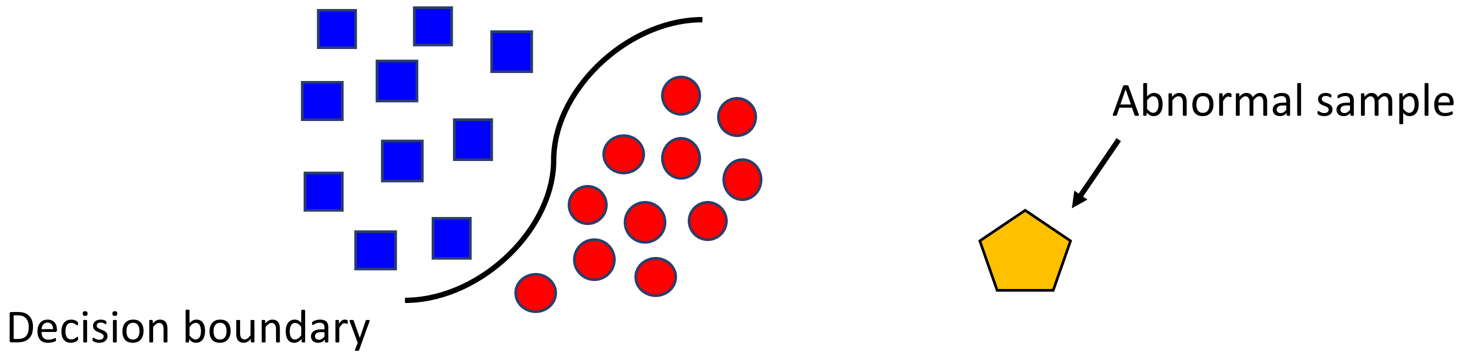

Ensemble learning
[Lee et al., 2017]



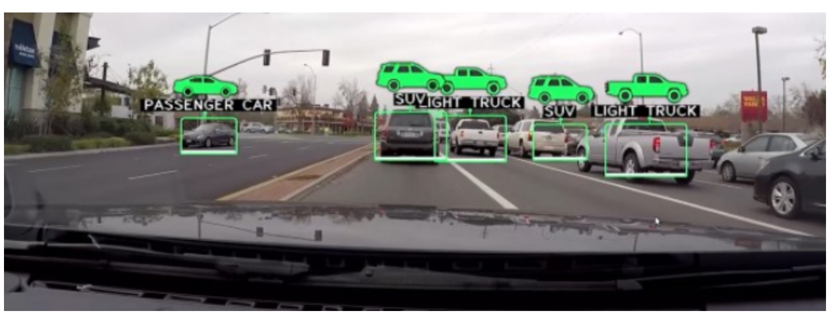

Incremental learning
[Rebuff et al., 2017]

## What is Novelty Detection?

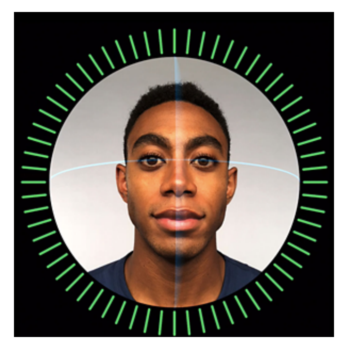
- **Novelty detection**
  - Detect whether a test sample is from in-distribution (i.e., training distribution by classifier) or not (e.g., out-of-distribution / adversarial samples)




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

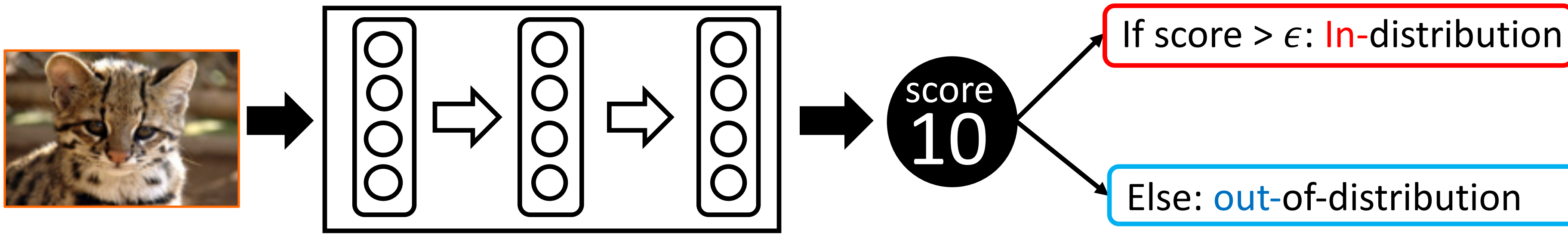



Autonomous drive



Secure authentication system

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

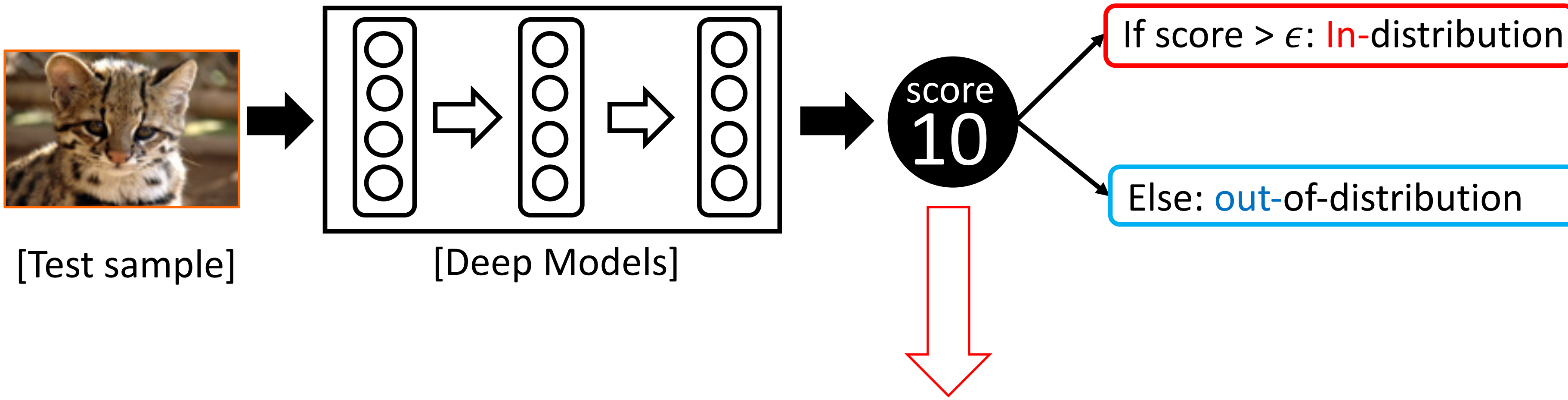



[Test sample]

[Deep Models]

## What is Novelty Detection?

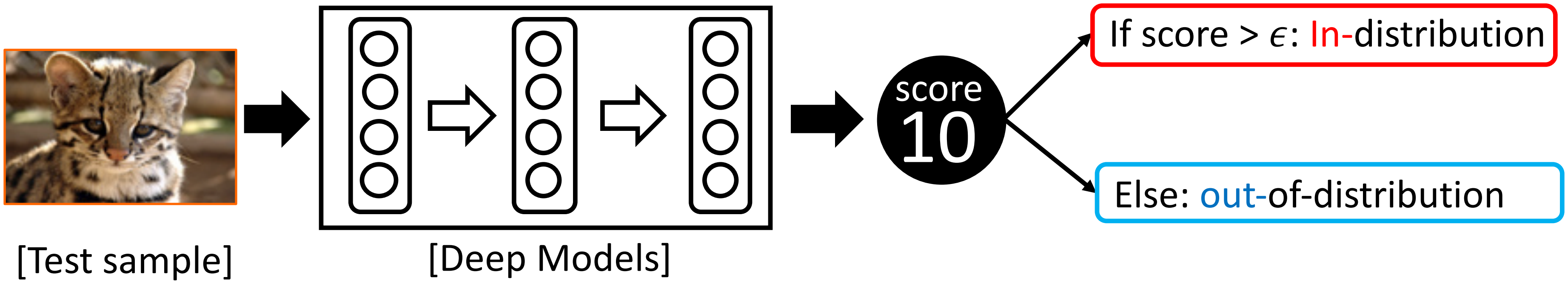
- How to solve this problem?
  - **Threshold-based Detector** [Hendrycks et al., 2017, Liang et al., 2018]



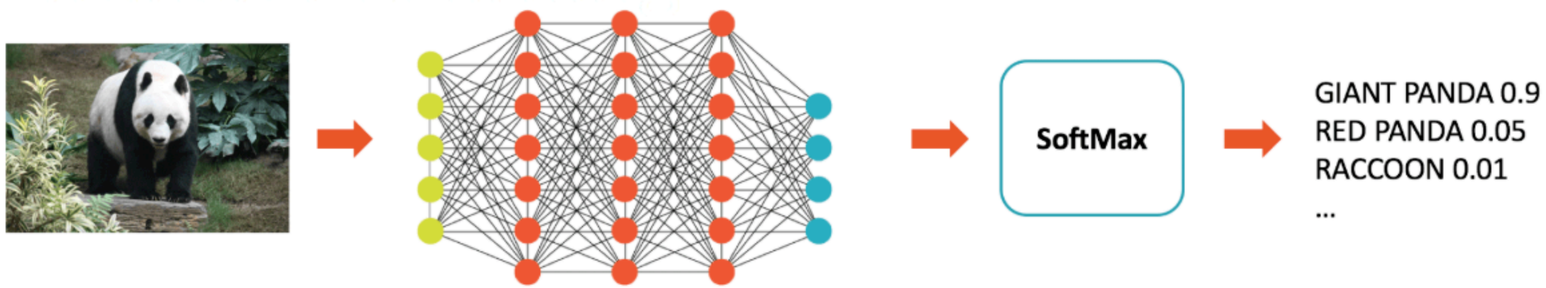

How to get confidence score

## What is Novelty Detection?

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

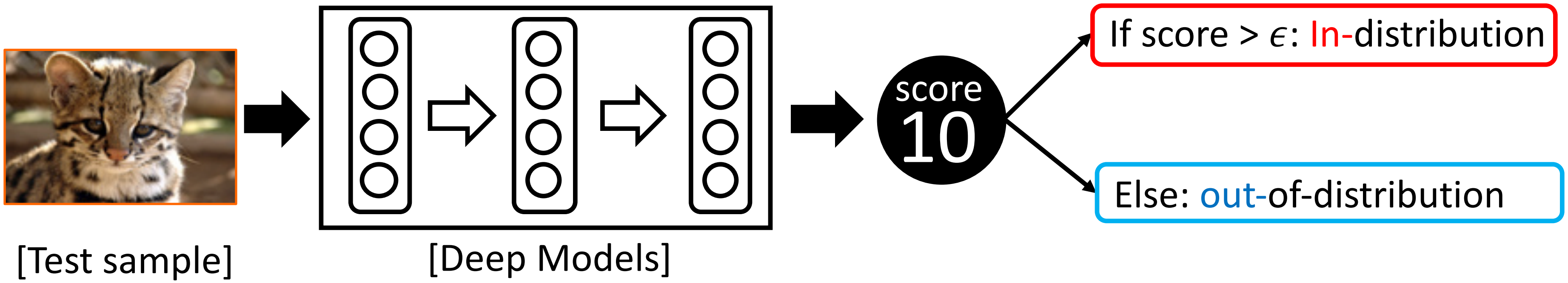



- Part 1. utilizing image classifiers

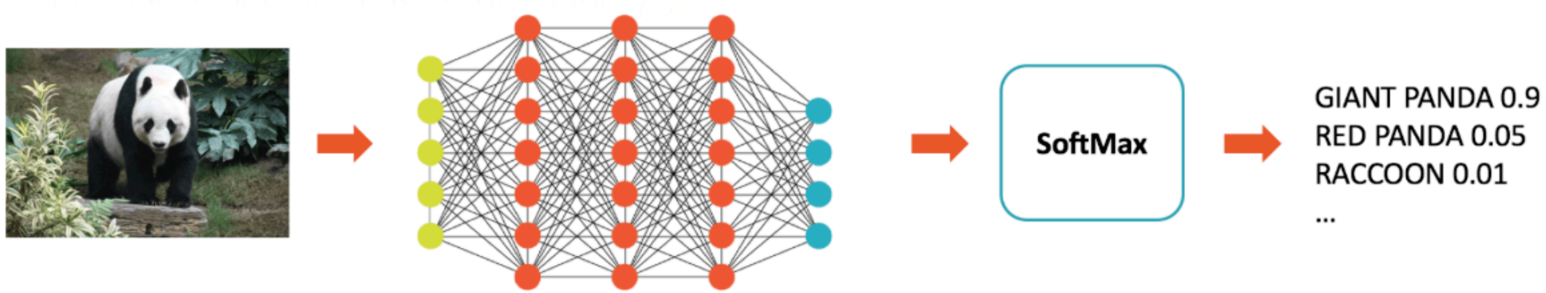


# What is Novelty Detection?

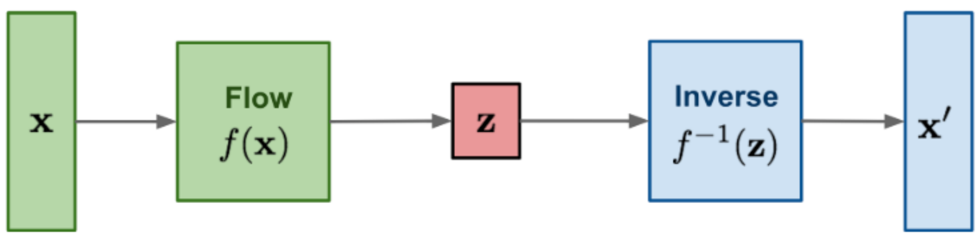
- How to solve this problem?
  - **Threshold-based Detector** [Hendrycks et al., 2017, Liang et al., 2018]




- Part 1. utilizing image classifiers




- Part 2. utilizing generative models



# Table of Contents

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

Input space $\mathbf{X} \xrightarrow{P} \mathbf{Y}$ Output space

- 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)

Input space $\mathbf{X} \xrightarrow{P} \mathbf{Y}$ Output space

- 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

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

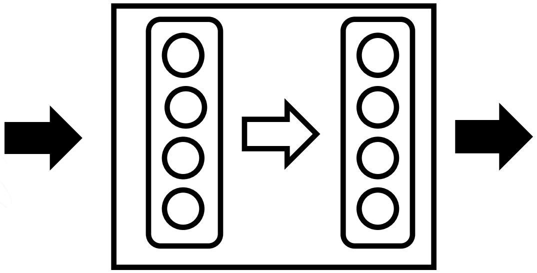
  - 2. entropy of posterior distribution

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

# Utilizing the Classifier: Posterior Distribution

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

[Input]



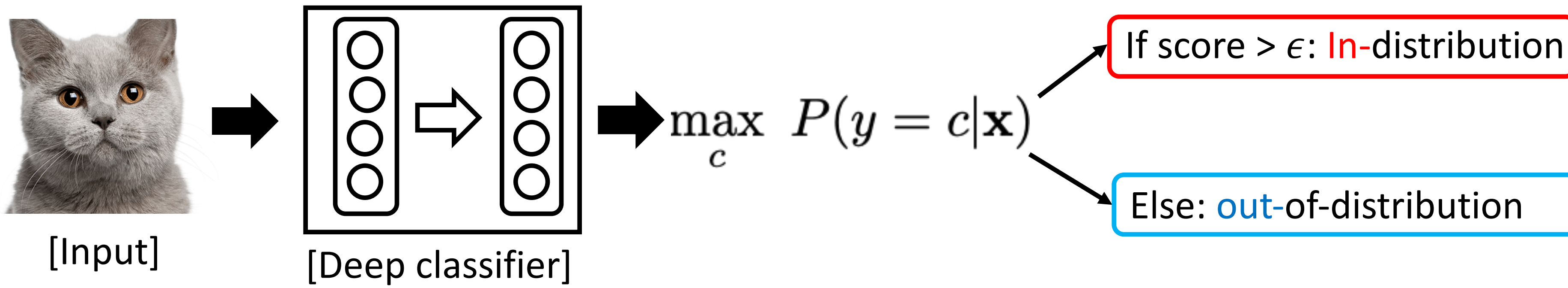
[Deep classifier]

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

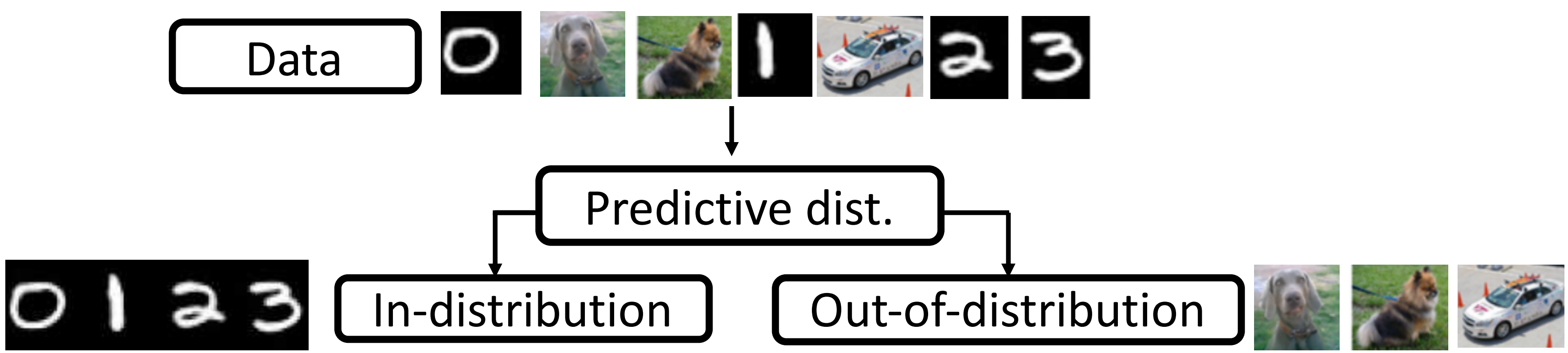
If score > $\epsilon$: In-distribution

Else: out-of-distribution

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



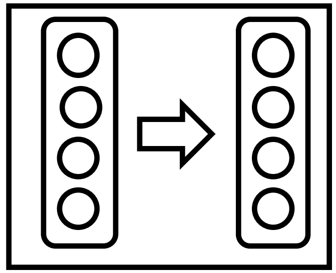

- Evaluation: detecting out-of-distribution
  - Assume that we have classifier trained on MNIST dataset
  - Detecting out-of-distribution for this classifier



## Utilizing the Classifier: Posterior Distribution

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

[Input]



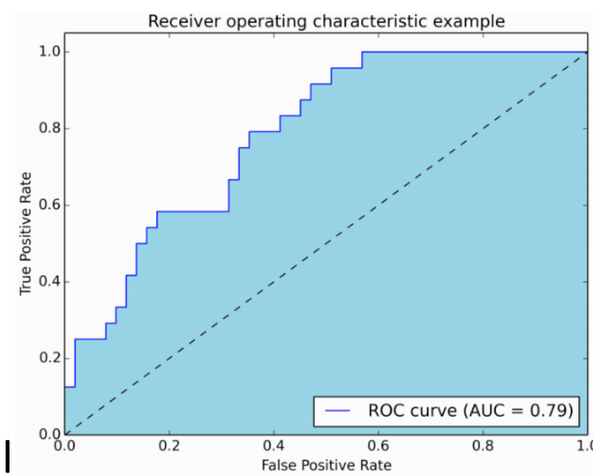
[Deep classifier]

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

If score > $\epsilon$: In-distribution

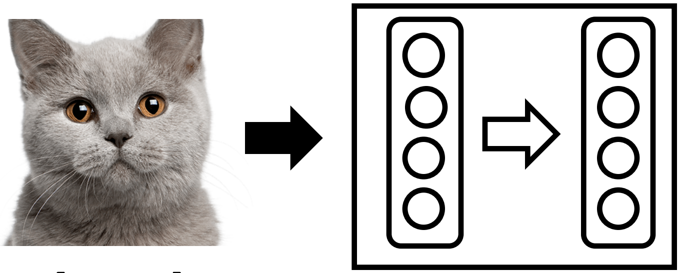
Else: out-of-distribution

- 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



[Input] [Deep classifier]

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

If score > $\epsilon$: In-distribution

Else: out-of-distribution

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

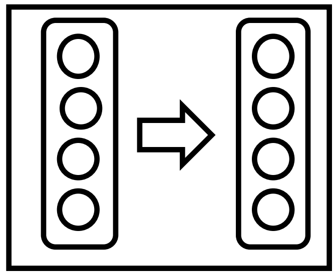
| In-Distribution / Out-of-Distribution | AUROC /random | AUPR In /random | AUPR Out/random |
|---|---|---|---|
| **CIFAR-10/SUN** | 95/50 | 89/33 | 97/67 |
| **CIFAR-10/Gaussian** | 97/50 | 98/49 | 95/51 |
| **CIFAR-10/All** | 96/50 | 88/24 | 98/76 |
| **CIFAR-100/SUN** | 91/50 | 83/27 | 96/73 |
| **CIFAR-100/Gaussian** | 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

[Input]



[Deep classifier]

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

If score > $\epsilon$: In-distribution

Else: out-of-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

## Utilizing the Classifier: Posterior Distribution

- **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

# 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

$$\mathbf{x}'=\mathbf{x}-\underbrace{\varepsilon}_{\text{Magnitude of noise}}\operatorname{sign}\left(-\bigtriangledown_{\mathbf{x}}\log P_{\theta}(y=\underbrace{\widehat{y}}_{\widehat{y}\text{ is the predicted label}}|\mathbf{x};T)\right),$$

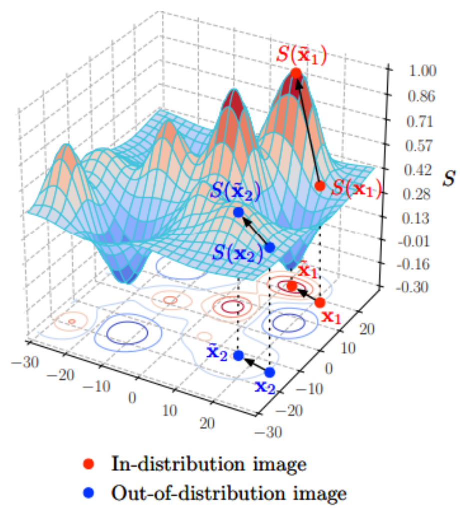



Figure 6: Illustration of effects of the input preprocessing.

## 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

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

  - 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

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

  - 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)

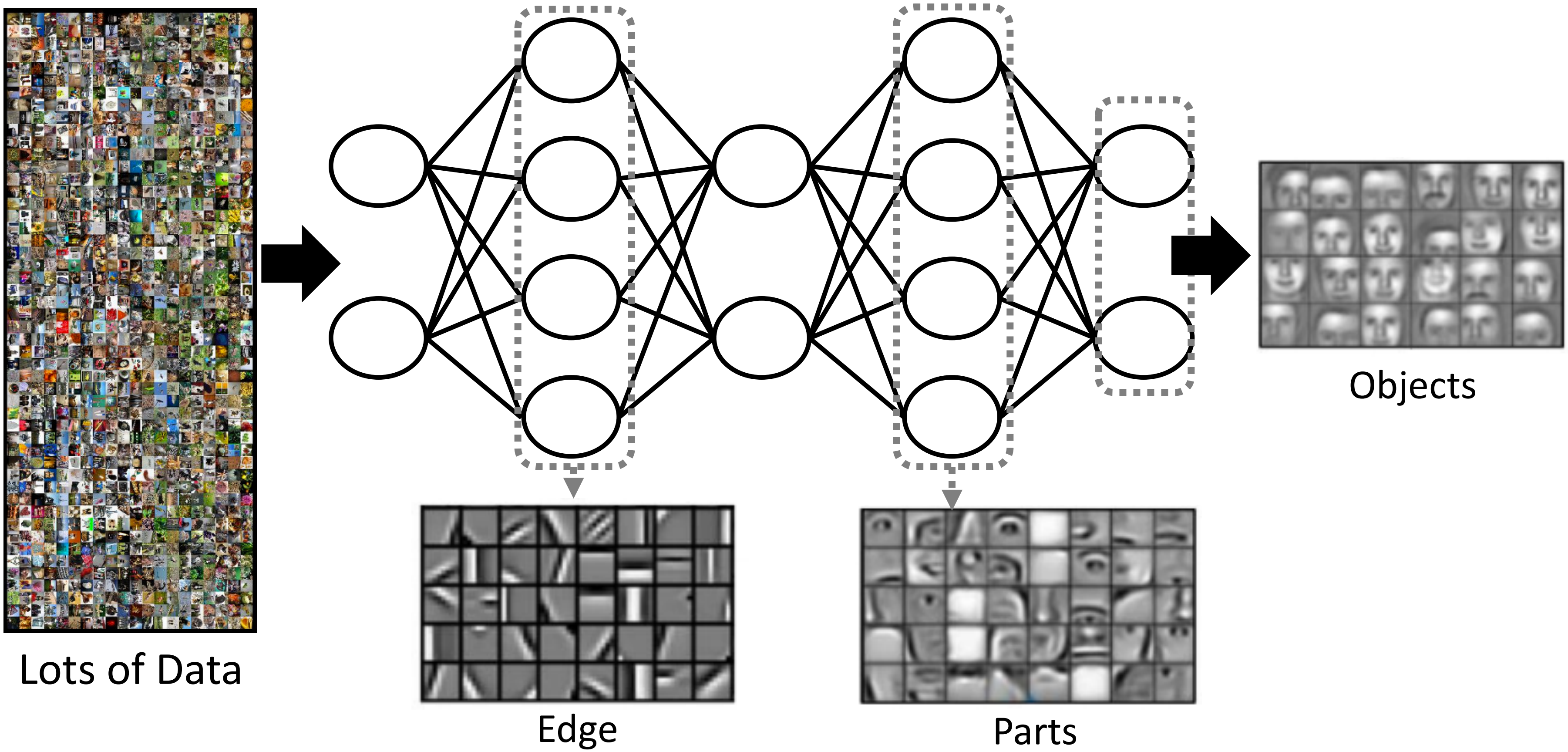
# Utilizing the Classifier: Posterior Distribution

- 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 reference sample increases ($V$ is volume)

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

- **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:*

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

*whenever the limit exists.*

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

## Utilizing the Classifier: Hidden Features

- **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:*

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

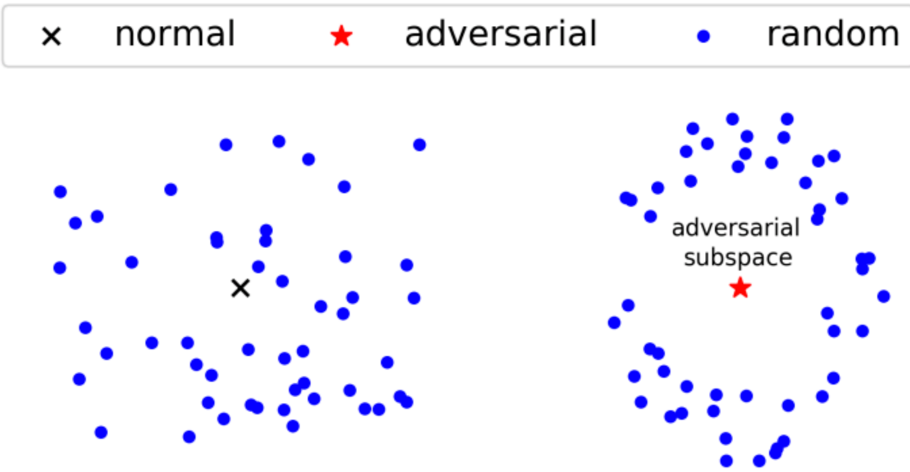
*whenever the limit exists.*

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

distance between sample and its k-th nearest neighbor

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

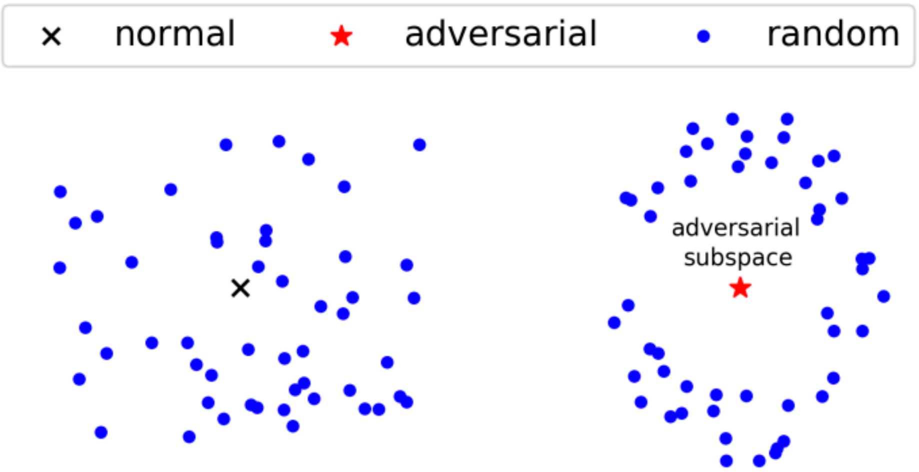
- **Motivation of LID**
  - Abnormal sample might be scattered compared to normal samples



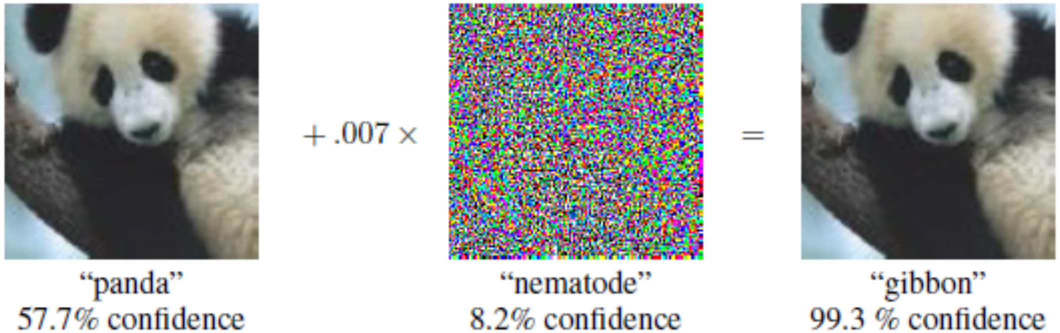

  - This implies that LID can be useful for detecting abnormal samples!

# Utilizing the Classifier: Hidden Features

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

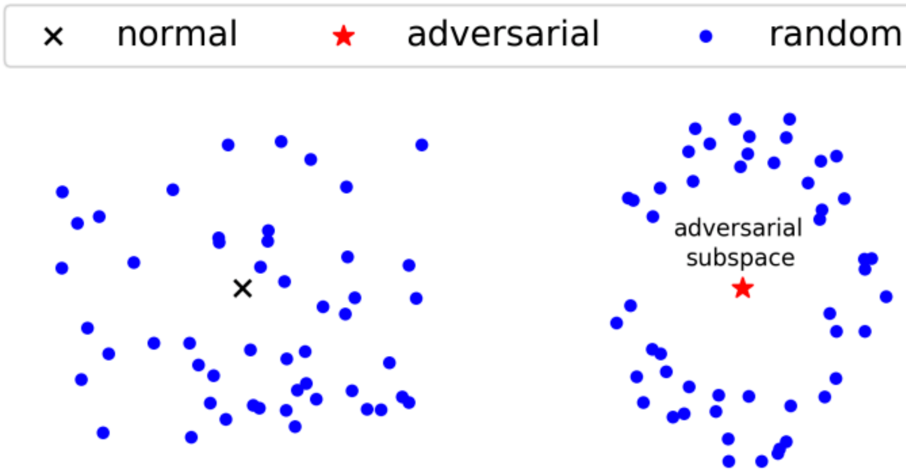



  - This implies that LID can be useful for detecting abnormal samples!
- Evaluation: detecting adversarial samples [Szegedy, et al., 2013]
  - Misclassified examples that are only slightly different from original examples

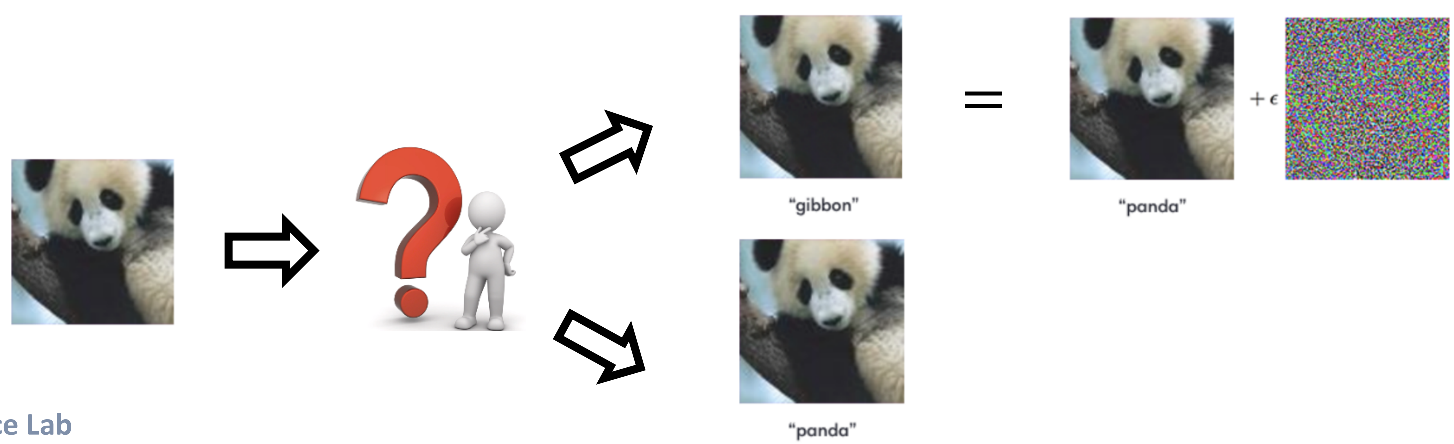


## Utilizing the Classifier: Hidden Features

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

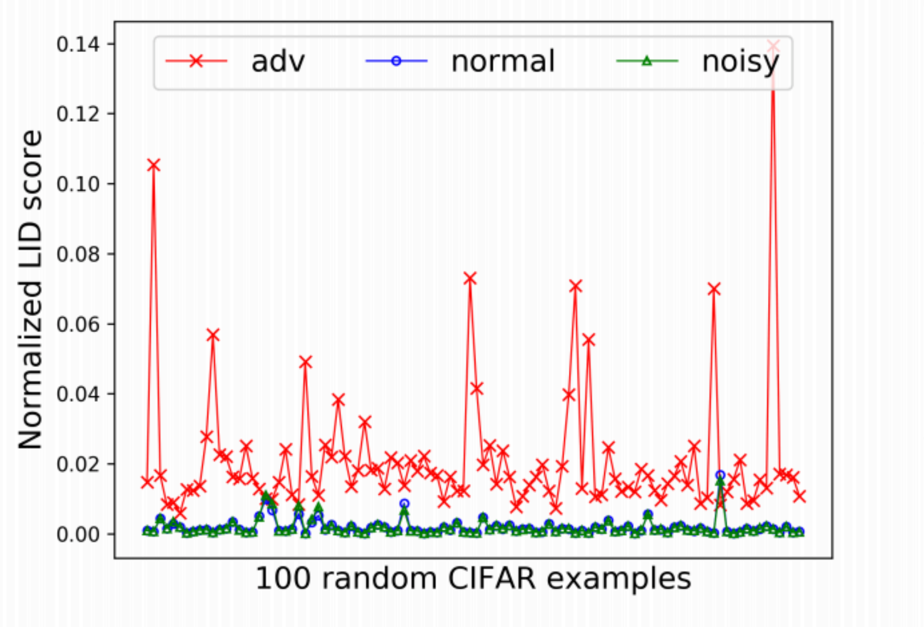


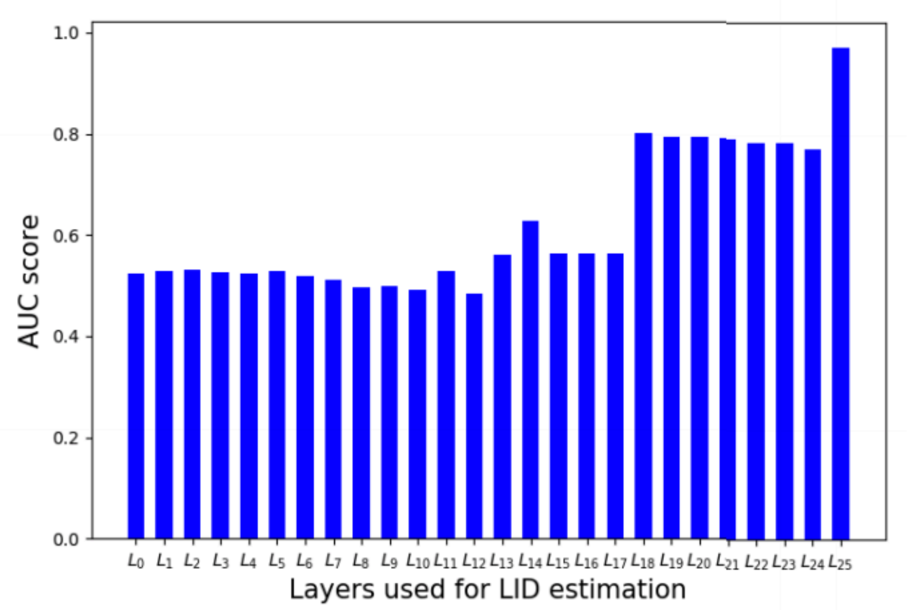

  - This implies that LID can be useful for detecting abnormal samples!
- Evaluation: detecting adversarial samples [Szegedy, et al., 2013]



# Utilizing the Classifier: Hidden Features

- **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]

# Utilizing the Classifier: Hidden Features

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

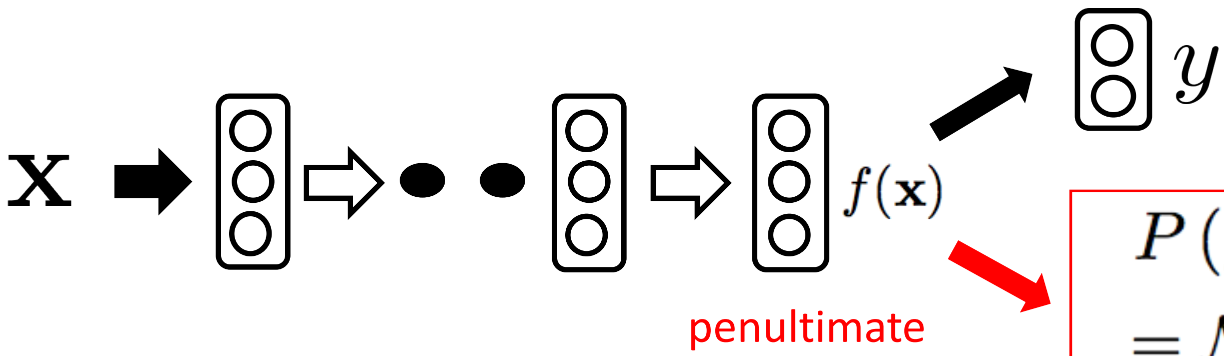
$$\boxed{P_\theta\left(y=c|\mathbf{x}\right)=\frac{\exp\left(\mathbf{w}_c^T f_\phi\left(\mathbf{x}\right)+b_c\right)}{\sum_{c'}\exp\left(\mathbf{w}_{c'}^T f_\phi\left(\mathbf{x}\right)+b_{c'}\right)},}$$

## Utilizing the Classifier: Hidden Features

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

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

  - Inducing a generative classifier on hidden feature space




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

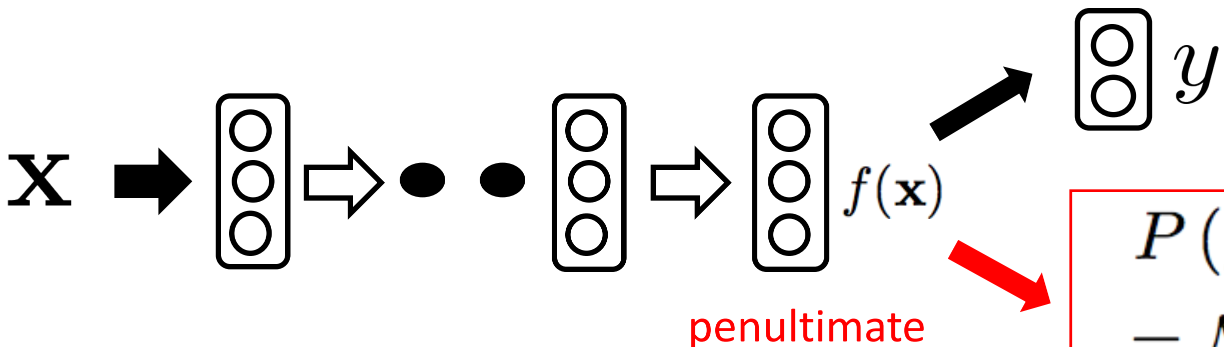
$$\boxed{\begin{aligned}&P\left(f(\mathbf{x})|y=c\right)\\&=\mathcal{N}\left(f(\mathbf{x})|\mu_c,\mathbf{\Sigma}\right)\end{aligned}}$$

## Utilizing the Classifier: Hidden Features

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

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

  - Inducing a generative classifier on hidden feature space




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

$$\boxed{\begin{aligned}&P\left(f(\mathbf{x})|y=c\right)\\&=\mathcal{N}\left(f(\mathbf{x})|\mu_c,\mathbf{\Sigma}\right)\end{aligned}}$$

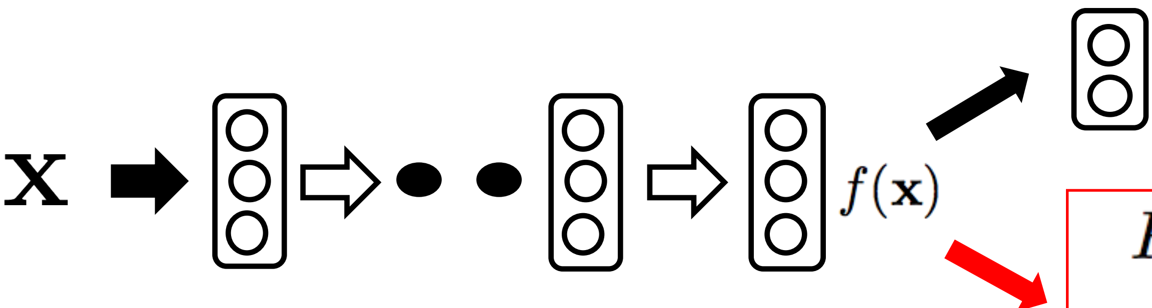
  - Motivation: connection between softamx and generative classifier (LDA)

$$\boxed{\begin{aligned}&P_\theta(y=c|\mathbf{x})=\frac{\exp(\mathbf{w}_c\mathbf{x}+b_c)}{\sum_{c'}\exp(\mathbf{w}_{c'}\mathbf{x}+b_{c'})}\\&\mathbf{w}_c=\mathbf{\Sigma}^{-1}\mu_c\quad b_c=-0.5\mu_c^T\mathbf{\Sigma}^{-1}\mu_c+\log\pi_c\end{aligned}} \sim = \boxed{\begin{aligned}&P_\theta(y=c|\mathbf{x})=\frac{P_\theta(\mathbf{x}|y=c)P_\theta(y=c)}{\sum_{c'}P_\theta(\mathbf{x}|y=c')P_\theta(y=c')}\\&P_\theta(\mathbf{x}|y=c)=\mathcal{N}(\mathbf{x}|\mu_c,\mathbf{\Sigma}),\quad P_\theta(y=c)=\frac{\pi_c}{\sum_{c'}\pi_{c'}}\end{aligned}}$$

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

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

  - Inducing a generative classifier on hidden feature space




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

$$\boxed{\begin{aligned}&P\left(f(\mathbf{x})|y=c\right)\\&=\mathcal{N}\left(f(\mathbf{x})|\mu_c,\mathbf{\Sigma}\right)\end{aligned}}$$

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

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

## Utilizing the Classifier: Hidden Features

- Using generative classifier, we define new confidence score:

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

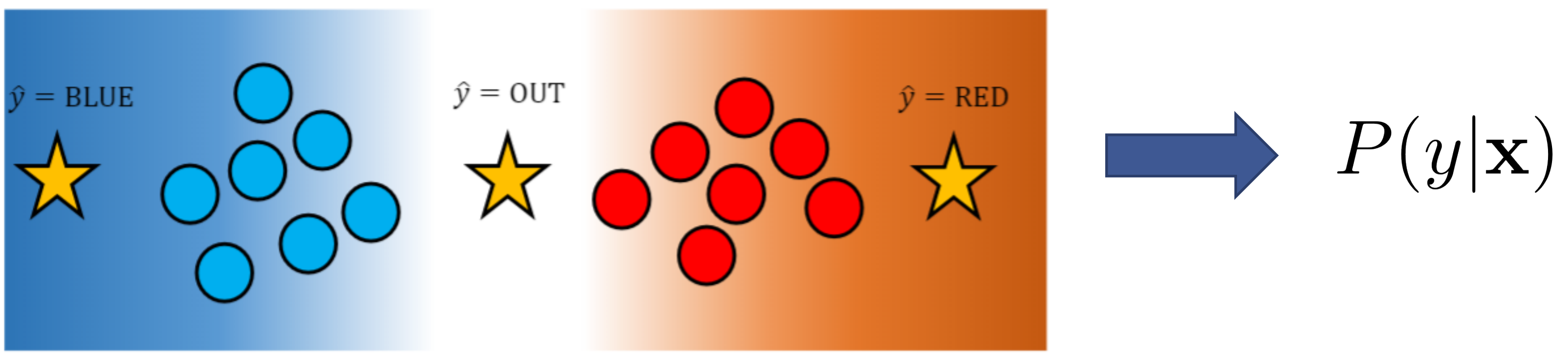
  - Measuring the log of the probability densities of the test sample

## Utilizing the Classifier: Hidden Features

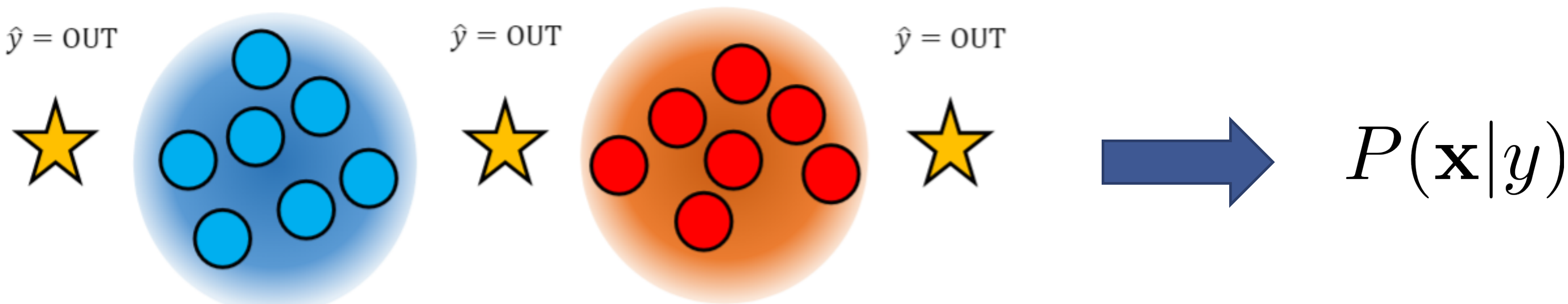
- Using generative classifier, we define new confidence score:

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

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




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




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

## Utilizing the Classifier: Hidden Features

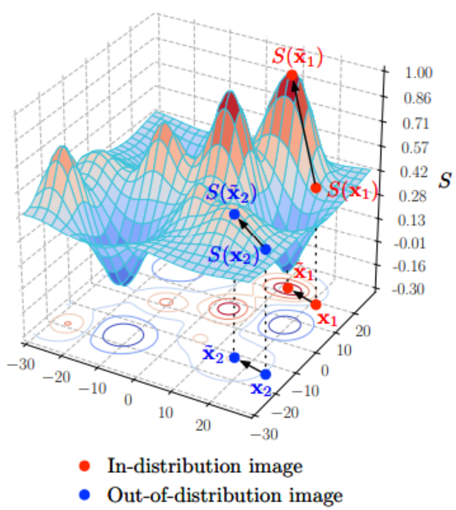
- Using generative classifier, we define new confidence score:

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

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

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

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



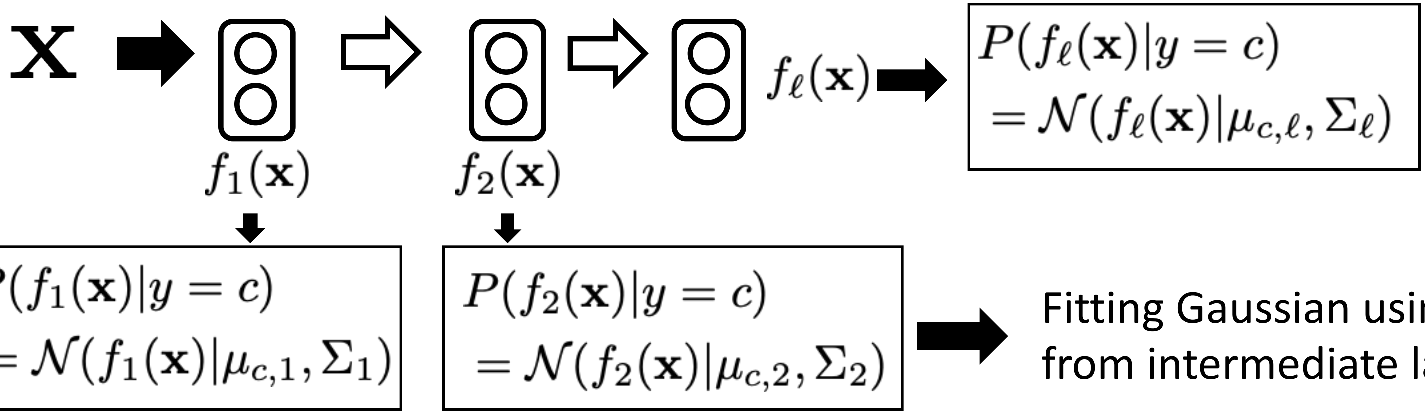
- Using generative classifier, we define new confidence score:

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

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

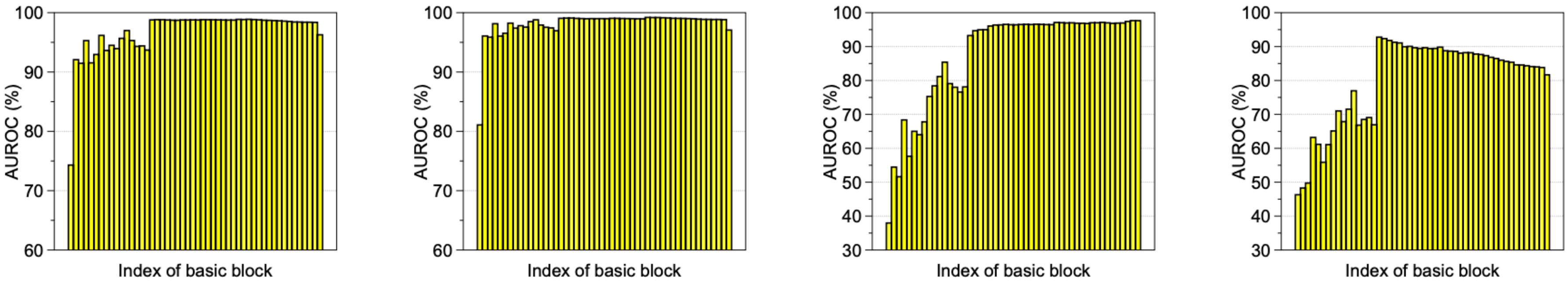
$$\widehat{\mathbf{x}} = \mathbf{x} + \varepsilon \mathrm{sign}\left(\nabla_{\mathbf{x}} M(\mathbf{x})\right) = \mathbf{x} - \varepsilon \mathrm{sign}\left(\nabla_{\mathbf{x}} \left(f(\mathbf{x}) - \widehat{\mu}_{\widehat{c}}\right)^\top \widehat{\mathbf{\Sigma}}^{-1} \left(f(\mathbf{x}) - \widehat{\mu}_{\widehat{c}}\right)\right)$$

  - Feature ensemble



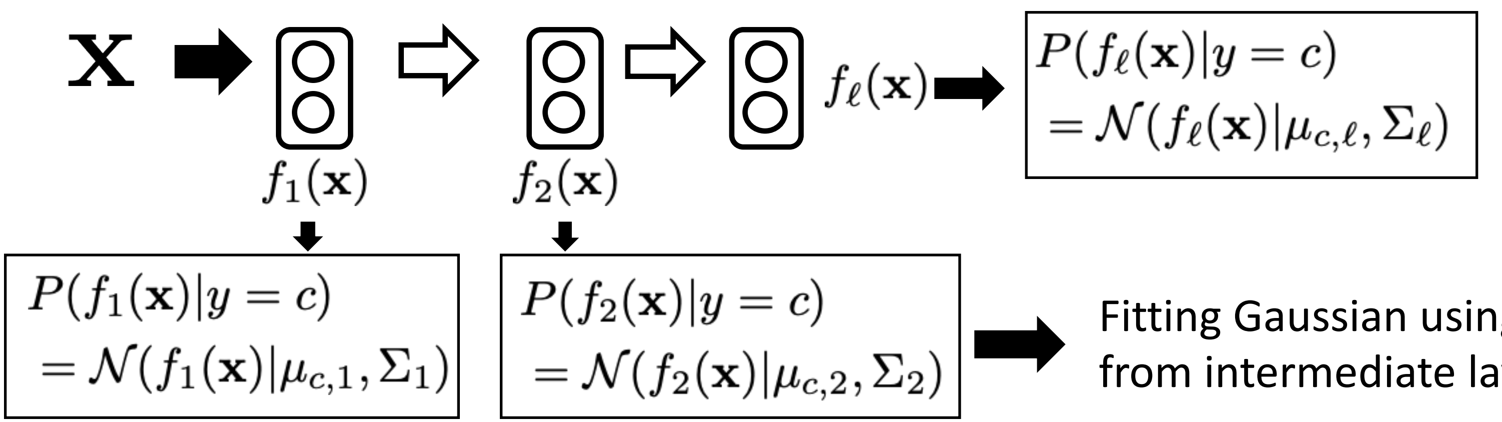

Fitting Gaussian using features from intermediate layers

# 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.




Fitting Gaussian using features from intermediate layers

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

# Utilizing the Classifier: Hidden Features

- Main algorithm

**Algorithm 1** Computing the Mahalanobis distance-based confidence score.

**Input:** Test sample $\mathbf{x}$, weights of logistic regression detector $\alpha_\ell$, noise $\varepsilon$ 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 $\sum_\ell \alpha_\ell M_\ell$

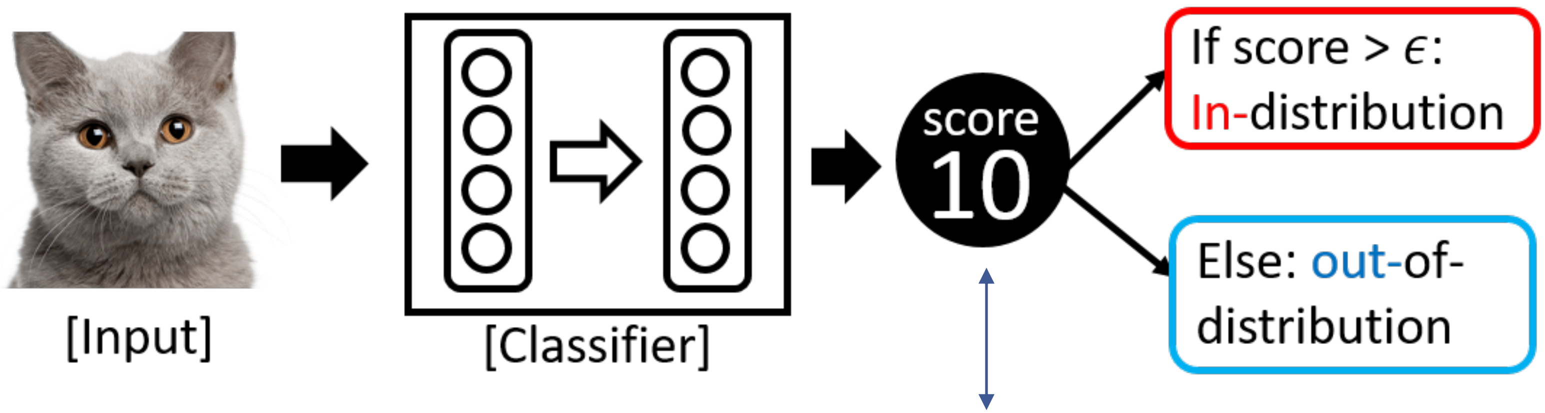
- Remark that
  - We combine the confidence scores from multiple layers using weighted ensemble

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

  - 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: TNR at TPR 95% | AUROC | Detection acc. | Validation on adversarial samples: 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** |

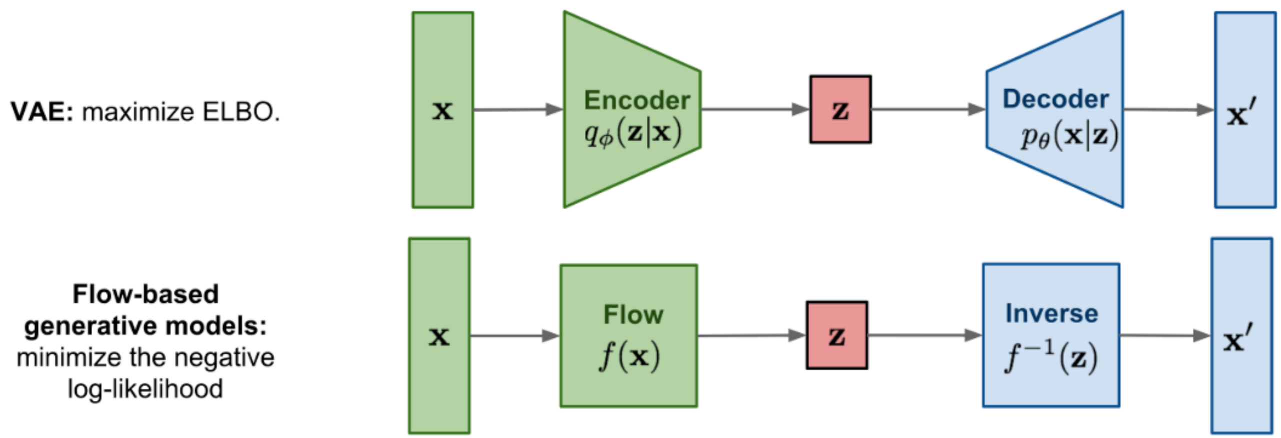
  - For all tested cases, our method outperforms LID and KD estimator
  - For unseen attacks, our method is still working well
    - FGSM samples denoted by "seen" are used for validation

## Table of Contents

- Generative models such as VAE [Kingma et al., 2014] and GLOW [Kingma et al., 2018] directly 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]



## Utilizing the Generative Models: Likelihood

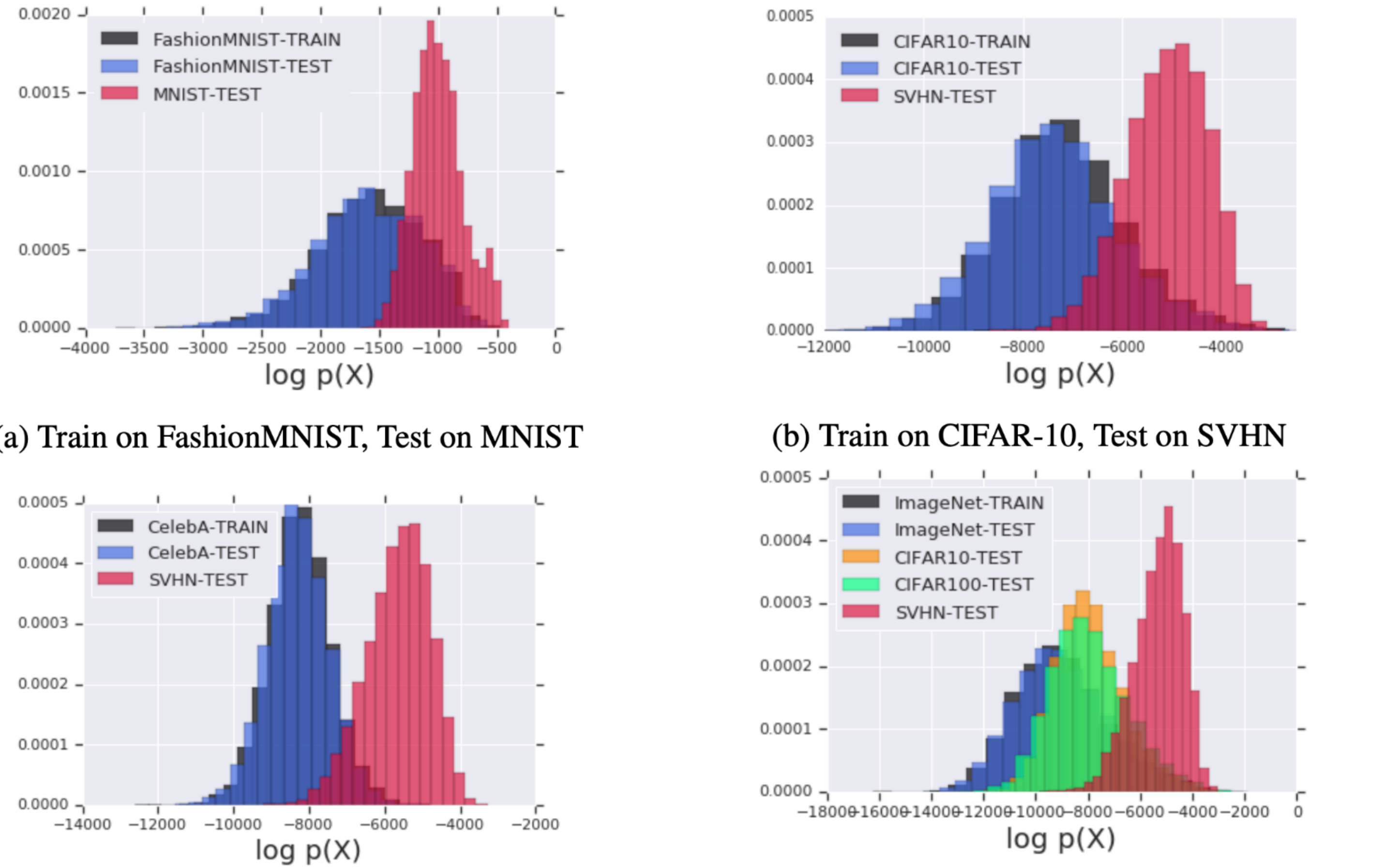
- Generative models such as VAE [Kingma et al., 2014] and GLOW [Kingma et al., 2018] directly model the data distribution
  - They have achieved the state-of-the-art performances on image generation

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?

- 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

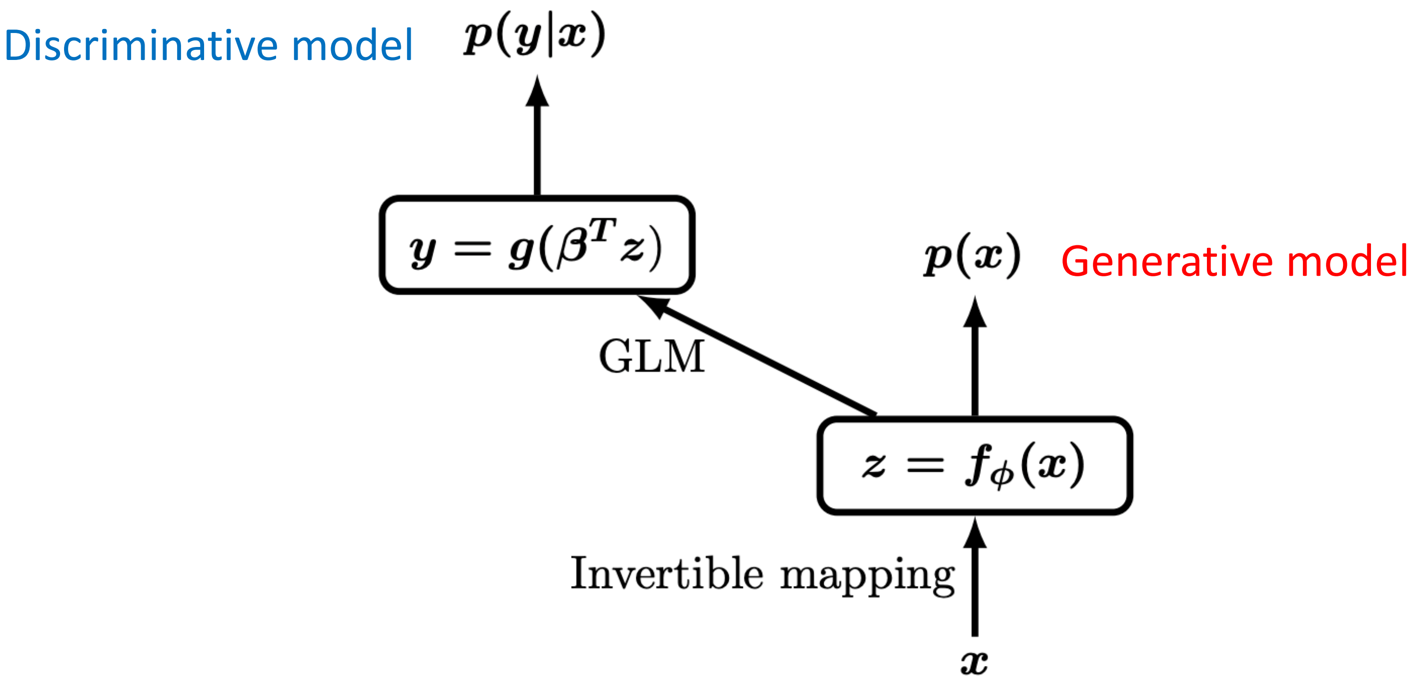
(c) Train on CelebA, Test on SVHN

(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: Hybrid Model

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




  - Weighted objective

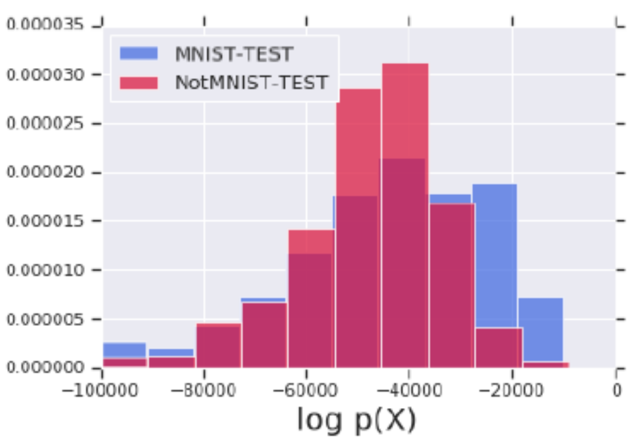
$$\mathcal{J}_\lambda(\boldsymbol{\theta}) = \sum_{n=1}^{N} \left( \log p(y_n | \boldsymbol{x}_n; \boldsymbol{\beta}, \boldsymbol{\phi}) + \lambda \log p(\boldsymbol{x}_n; \boldsymbol{\phi}) \right)$$

# Utilizing the Generative Models: Hybrid Model

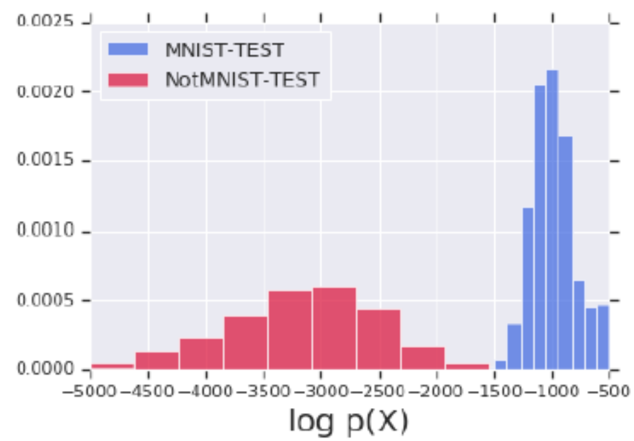
- Bits-per-dimension (BPD), error and negative log likelihood (NLL)

| 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

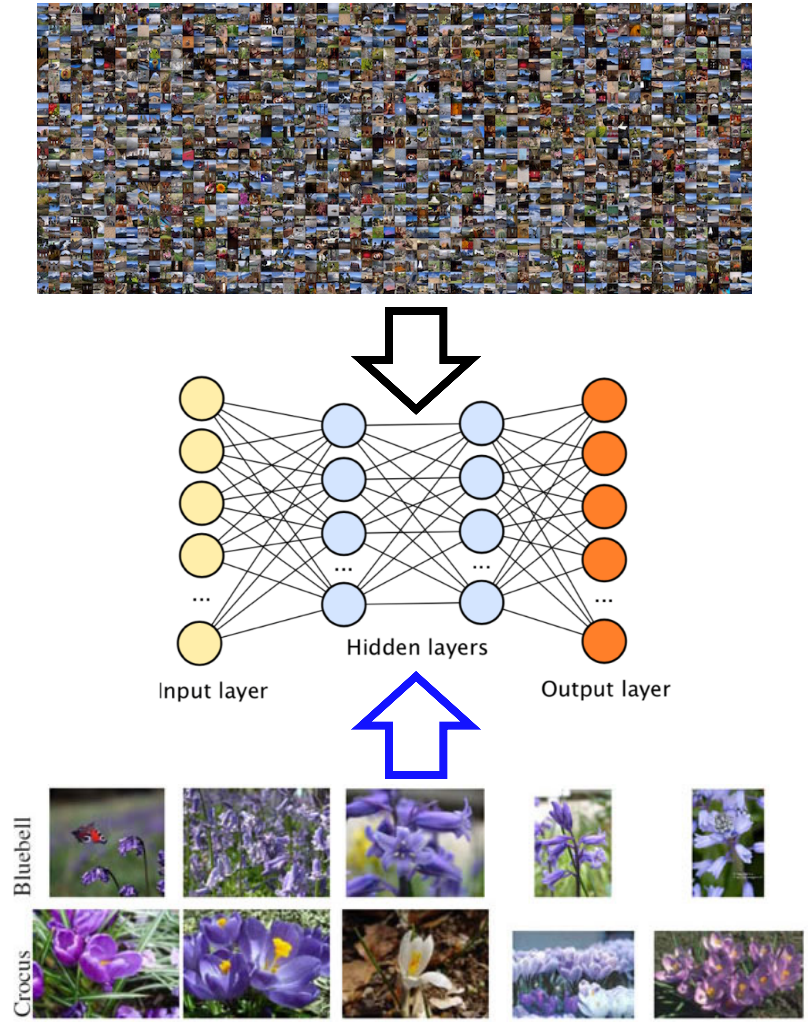
## Table of Contents

- Pre-training
  - Transfer the knowledge from related tasks

1. Pre-train the networks on a large-scale source datasets

2. Use pre-trained weights as initial parameters

3. Fine-tune the networks on a target dataset



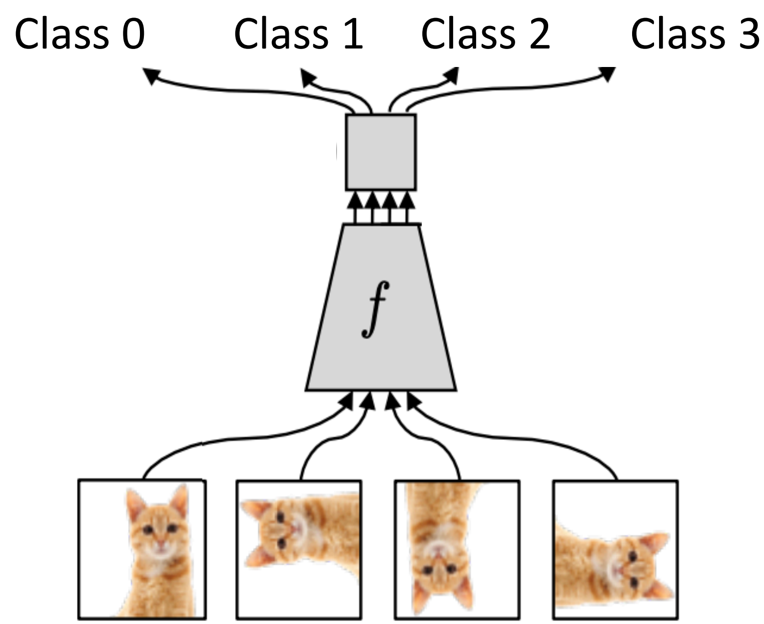
## Other Approaches: Pre-training

- **Question:** Can pre-training provide large benefits to model robustness and uncertainty? [Hendrycks et al., 2019b]
- Experimental setup
  - Pre-trained model
    - Wide ResNets trained on Down-sampled ImageNet
  - In-distribution
    - CIFAR-10, CIFAR-100 and TinyImageNet
  - Out-of-distribution
    - Gaussian noise, textures, Places365 scene images
- Experimental results on a baseline detector

| | AUROC | | AUPR | |
|---|---|---|---|---|
| | Normal | Pre-Train | Normal | Pre-Train |
| CIFAR-10 | 91.5 | 94.5 | 63.4 | 73.5 |
| CIFAR-100 | 69.4 | 83.1 | 29.7 | 52.7 |
| Tiny ImageNet | 71.8 | 73.9 | 30.8 | 31.0 |

## Other Approaches: Self-Supervised Learning

- Self-supervised learning [Doersch et al., 2015]
  - Supervised learning with automatically generated labels




- Self-supervised learning can improve uncertainty estimation [Hendrycks et al., 2019c]

# Other Approaches: Self-Supervised Learning

- Problem setup
  - Given a dataset consisting in k classes, train a model on one class and use remainin g K-1 classes as out-of-distribution
  - 30 classes from ImageNet

- 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 |

  - Supervised: one class (positive) vs ImageNet 22K (negative)

## Summary

- In this lecture, we cover various methods for detecting abnormal samples like out-of-distribution and adversarial samples
  - Posterior distribution-based methods
  - Hidden feature-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

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

# References

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

[Kingma et al., 2014] Auto-Encoding Variational Bayes. In ICLR, 2014.
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

[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