Outline

• Introduction
  • Predictive uncertainty of deep neural networks
  • Summary of contributions

• How to train confident neural networks
  • Training Confidence-Calibrated Classifiers for Detecting Out-of-Distribution Samples [Lee’ 18a]

• Applications
  • Hierarchical novelty detection [Lee’ 18b]

• Conclusion
  • Future work

Introduction: Predictive uncertainty of deep neural networks (DNNs)

- Supervised learning (e.g., regression and classification)
  - Objective: finding an unknown target distribution, i.e., \( P(Y|X) \)

  \[
  \begin{array}{c}
  \text{Input space } X \\
  P \\
  \text{Output space } Y
  \end{array}
  \]

- Recent advances in deep learning have dramatically improved accuracy on several supervised learning tasks


Introduction: Predictive uncertainty of deep neural networks (DNNs)

- Uncertainty of predictive distribution is important in DNN’s applications
  - What is predictive uncertainty?
    - As a example, consider classification task

- It represents a confidence about prediction!
- For example, it can be measured as follows:
  - Entropy of predictive distribution [Lakshminarayanan’ 17]
    \[
    \sum_{y} -P(y|x) \log P(y|x)
    \]
  - Maximum value of predictive distribution [Hendrycks’ 17]
    \[
    \max_{y} P(y|x)
    \]

Introduction: Predictive uncertainty of deep neural networks (DNNs)

- Predictive uncertainty is related to many machine learning problems:
  - Novelty detection [Hendrycks’ 17]
  - Adversarial detection [Song’ 18]
  - Ensemble learning [Lee’ 17]

- Predictive uncertainty is also indispensable when deploying DNNs in real-world systems [Dario’ 16]

Introduction: Predictive uncertainty of deep neural networks (DNNs)

- However, DNNs do not capture their predictive uncertainty

- E.g., DNNs trained to classify MNIST images often produce high confident probability 91% even for random noise [Henderycks’ 17]

- Challenge arises in improving the quality of the predictive uncertainty!

Main topic of this presentation

- How to train confident neural networks?
  - Training confidence-calibrated classifiers for detecting out-of-distribution samples [Lee’ 18a]

Applications

- Confident multiple choice learning [Lee’ 17]
- Hierarchical novelty detection [Lee’ 18b]

Outline

• Introduction
  • Predictive uncertainty of deep neural networks
  • Summary of contributions

• How to train confident neural networks
  • Training Confidence-Calibrated Classifiers for Detecting Out-of-Distribution Samples [Lee’ 18a]

• Applications
  • Confident Multiple Choice Learning [Lee’ 17]
  • Hierarchical novelty detection [Lee’ 18b]

• Conclusion
  • Future work

How to Train Confident Neural Networks?

- Related problem
  - Detecting out-of-distribution [Hendrycks’ 17, Liang’ 18]
    - Detect whether a test sample is from in-distribution (i.e., training distribution by classifier) or out-of-distribution

How to Train Confident Neural Networks?

• Related problem
  • Detecting out-of-distribution [Hendrycks’ 17, Liang’ 18]
    • Detect whether a test sample is from in-distribution (i.e., training distribution by classifier) or out-of-distribution

• E.g., image classification
  • Assume a classifier trains handwritten digits (denoted as in-distribution)
  • Detecting out-of-distribution

• Performance of detector reflects confidence of predictive distribution!

Related Work

• Threshold-based Detector [Guo’ 17, Hendrycks’17, Liang’ 18]

[Input] → [Classifier] → score →
If score > $\epsilon$: In-distribution
Else: out-of-distribution

Related Work

- Threshold-based Detector [Guo’ 17, Hendrycks’17, Liang’ 18]

  ![Diagram of threshold-based detector]

- How to define the score?
  - Baseline detector [Hendrycks’17]
    - Confidence score = maximum value of predictive distribution: \( \max_y P(y|x) \)
  - Temperature scaling [Guo’ 17]
    - Confidence score = maximum value of scaled predictive distribution
      \[
      p_i(x; T) = \frac{\exp( f_i(x)/T )}{\sum_{j=1}^{N} \exp( f_j(x)/T )}
      \]

  ![Output of neural networks]

---


Related Work

• Threshold-based Detector [Guo’ 17, Hendrycks’17, Liang’ 18]

In-distribution

Out-distribution

• Limitations
  • Performance of prior works highly depends on how to train the classifiers


Our Contributions

• One can consider
  • Bayesian neural networks [Yingzhen’ 17]
  • Ensemble of classifiers [Balaji’ 17]

  ![Diagram of Bayesian neural networks]

• Training or inferring those models are computationally expensive


Our Contributions

• One can consider
  • Bayesian neural networks [Yingzhen’ 17]
  • Ensemble of classifiers [Balaji’ 17]

• Training or inferring those models are computationally expensive

• Our contribution

  Confidence loss for training more plausible simple DNNs

  GAN for generating out-of-distribution samples

  Joint training method of classifier and GAN
Our Contributions

• One can consider
  • Bayesian neural networks [Yingzhen’ 17]  • Ensemble of classifiers [Balaji’ 17]

• Training or inferring those models are computationally expensive

• Our contribution

  Confidence loss for training more plausible simple DNNs  
  GAN for generating out-of-distribution samples  
  Joint training method of classifier and GAN

• Experimental results
  • Our method drastically improves the detection performance
  • E.g., VGGNet trained by our method improves TPR compared to the baseline: 
    14.0% → 39.1% and 46.3% → 98.9% on CIFAR-10 and SVHN
**Contribution 1: Confident Loss**

- **Confident loss**
  - Minimize the KL divergence on data from out-of-distribution

\[
\min_{\theta} \mathbb{E}_{P_{\text{in}}(\hat{x}, y)} \left[ - \log P_{\theta} (y = \hat{y} | \hat{x}) \right] + \beta \mathbb{E}_{P_{\text{out}}(x)} \left[ KL (\mathcal{U}(y) \parallel P_{\theta} (y | x)) \right],
\]

- **Data from in-dist**
- **Data from out-of-dist**

- **Interpretation**
  - Assigning higher maximum prediction values to in-distribution samples than out-of-distribution ones

\[
P_{\theta}(y | x) \rightarrow P(y | x) \quad \text{Data distribution}
\]

\[
P_{\theta}(y | x) \rightarrow \mathcal{U}(y) \quad \text{Uniform distribution}
\]

[In-distribution data] [Out-of-distribution data] "Zero confidence"
Contribution 1: Confident Loss

- Confident loss
  - Minimize the KL divergence on data from out-of-distribution:
    \[
    \min_{\theta} \mathbb{E}_{P_{\text{in}}(\mathbf{x}, \mathbf{y})} \left[ -\log P_{\theta} (y = \hat{y} | \mathbf{x}) \right] + \beta \mathbb{E}_{P_{\text{out}}(\mathbf{x})} \left[ KL (\mathcal{U}(y) \parallel P_{\theta}(y | \mathbf{x})) \right],
    \]
  - Interpretation
    - Assigning higher maximum prediction values to in-distribution samples than out-of-distribution ones
  - Effects of confidence loss
    - Fraction of the maximum prediction value from simple CNNs (2 Conv + 3 FC)

Dataset Examples:
- SVHN
- CIFAR-10
- TinyImageNet
- LSUN
**Contribution 1: Confident Loss**

- **Confident loss**
  - Minimize the KL divergence on data from out-of-distribution

\[
\min_{\theta} \mathbb{E}_{P_{\text{in}}(x, y)} \left[ -\log P_{\theta}(y = \hat{y} | x) \right] + \beta \mathbb{E}_{P_{\text{out}}(x)} \left[ KL (U(y) || P_{\theta}(y | x)) \right].
\]

- **Interpretation**
  - Assigning higher maximum prediction values to in-distribution samples than out-of-distribution ones

- **Effects of confidence loss**
  - Fraction of the maximum prediction value from simple CNNs (2 Conv + 3 FC)
  - In-distribution: SVHN

![Graph](image)
**Contribution 1: Confident Loss**

- **Confident loss**
  - Minimize the KL divergence on data from out-of-distribution

\[
\min_{\theta} \mathbb{E}_{P_{\text{in}}(x, y)} \left[ -\log P_{\theta}(y = \hat{y} | x) \right] + \beta \mathbb{E}_{P_{\text{out}}(x)} \left[ KL(U(y) \parallel P_{\theta}(y | x)) \right].
\]

- **Interpretation**
  - Assigning higher maximum prediction values to in-distribution samples than out-of-distribution ones

- **Effects of confidence loss**
  - Fraction of the maximum prediction value from simple CNNs (2 Conv + 3 FC)
  - KL divergence term is optimized using CIFAR-10 training data

![Graphs showing comparison between Cross entropy loss and Confidence loss](image-url)
Contribution 2. GAN for Generating Out-of-Distribution Samples

- Main issues of confidence loss
  - How to optimize the KL divergence loss?

\[
\min_{\theta} \mathbb{E}_{P_{in}(x, y)} \left[ -\log P_{\theta}(y = \hat{y} | \hat{x}) \right] + \beta \mathbb{E}_{P_{out}(x)} \left[ KL (\mathcal{U}(y) \parallel P_{\theta}(y | x)) \right],
\]

Data from out-of-dist
Main issues of confidence loss

- How to optimize the KL divergence loss?
  - The number of out-of-distribution samples might be almost infinite to cover the entire space

\[
\min_{\theta} \mathbb{E}_{P_{\text{in}}(\hat{x}, \hat{y})} \left[ -\log P_{\theta} (y = \hat{y} | \hat{x}) \right] + \beta \mathbb{E}_{P_{\text{out}}(x)} \left[ KL (\mathcal{U}(y) \parallel P_{\theta} (y | x)) \right],
\]

Data from out-of-dist
Contribution 2. GAN for Generating Out-of-Distribution Samples

- Main issues of confidence loss
  - How to optimize the KL divergence loss?
    - The number of out-of-distribution samples might be almost infinite to cover the entire space

\[
\min_\theta \mathbb{E}_{P_{in}(\hat{x}, \hat{y})} \left[ - \log P_\theta (y = \hat{y} | \hat{x}) \right] + \beta \mathbb{E}_{P_{out}(x)} \left[ KL (U(y) \| P_\theta (y | x)) \right],
\]

Data from out-of-dist
Contribution 2. GAN for Generating Out-of-Distribution Samples

• Main issues of confidence loss
  • How to optimize the KL divergence loss?
    • The number of out-of-distribution samples might be almost infinite to cover the entire space

• Our intuition
  • Samples close to in-distribution could be more effective in improving the detection performance
Contribution 2. GAN for Generating Out-of-Distribution Samples

• Main issues of confidence loss
  • How to optimize the KL divergence loss?
    • The number of out-of-distribution samples might be almost infinite to cover the entire space

• Our intuition
  • Samples close to in-distribution could be more effective in improving the detection performance

Figure 2: Illustrating the behavior of classifier under different datasets. We generate the out-of-distribution samples from (a) 2D box $[-50, 50]^2$, and show (b) the corresponding decision boundary of classifier. We also generate the out-of-distribution samples from (c) 2D box $[-20, 20]^2$, and show (d) the corresponding decision boundary of classifier.
Contribution 2. GAN for Generating Out-of-Distribution Samples

- Main issues of confidence loss
  - How to optimize the KL divergence loss?
    - The number of out-of-distribution samples might be almost infinite to cover the entire space

- Our intuition
  - Samples close to in-distribution could be more effective in improving the detection performance

Figure 2: Illustrating the behavior of classifier under different datasets. We generate the out-of-distribution samples from (a) 2D box $[-50, 50]^2$, and show (b) the corresponding decision boundary of classifier. We also generate the out-of-distribution samples from (c) 2D box $[-20, 20]^2$, and show (d) the corresponding decision boundary of classifier.
Contribution 2. GAN for Generating Out-of-Distribution Samples

• Main issues of confidence loss
  • How to optimize the KL divergence loss?
    • The number of out-of-distribution samples might be almost infinite to cover the entire space

• Our intuition
  • Samples close to in-distribution could be more effective in improving the detection performance

Figure 2: Illustrating the behavior of classifier under different datasets. We generate the out-of-distribution samples from (a) 2D box $[-50, 50]^2$, and show (b) the corresponding decision boundary of classifier. We also generate the out-of-distribution samples from (c) 2D box $[-20, 20]^2$, and show (d) the corresponding decision boundary of classifier.
Contribution 2. GAN for Generating Out-of-Distribution Samples

- New GAN objective

\[
\min_G \max_D \left( \mathbb{E}_{P_{\text{in}}(x)} \left[ \log D(x) \right] + \mathbb{E}_{P_G(x)} \left[ \log (1 - D(x)) \right] \right),
\]

- Term (b) corresponds to the original GAN loss
  - Generating out-of-distribution samples close to in-distribution
Contribution 2. GAN for Generating Out-of-Distribution Samples

- New GAN objective

\[
\min_G \max_D \beta \mathbb{E}_{P_G(x)} \left[ KL \left( \mathcal{U}(y) \parallel P_\theta \left( y \mid x \right) \right) \right] \\
\left( a \right) \\
+ \mathbb{E}_{P_{in}(x)} \left[ \log D \left( x \right) \right] + \mathbb{E}_{P_G(x)} \left[ \log \left( 1 - D \left( x \right) \right) \right], \quad \left( b \right)
\]

- Term (a) forces the generator to generate low-density samples
  - (approximately) minimizing the log negative likelihood of in-distribution
- Term (b) corresponds to the original GAN loss
  - Generating out-of-distribution samples close to in-distribution

\[ P_{in}(x) \approx \exp \left( KL \left( \mathcal{U}(y) \parallel P_\theta \left( y \mid x \right) \right) \right) \]
Contribution 2. GAN for Generating Out-of-Distribution Samples

- New GAN objective
  \[
  \min_G \max_D \beta \mathbb{E}_{P_G(x)} \left[ KL \left( U(y) \parallel P_\theta(y|x) \right) \right] \\
  + \mathbb{E}_{P_{\text{in}}(x)} \left[ \log D(x) \right] + \mathbb{E}_{P_G(x)} \left[ \log (1 - D(x)) \right],
  \]

- Term (a) forces the generator to generate low-density samples
  - (approximately) minimizing the log negative likelihood of in-distribution
- Term (b) corresponds to the original GAN loss
  - Generating out-of-distribution samples close to in-distribution
Contribution 2. GAN for Generating Out-of-Distribution Samples

• New GAN objective

\[
\min_G \max_D \beta \mathbb{E}_{P_G(x)} \left[ KL \left( U(y) \ || \ P_\theta(y|x) \right) \right] \\
+ \mathbb{E}_{P_{in}(x)} \left[ \log D(x) \right] + \mathbb{E}_{P_G(x)} \left[ \log (1 - D(x)) \right],
\]

• Term (a) forces the generator to generate low-density samples
  • (approximately) minimizing the log negative likelihood of in-distribution
• Term (b) corresponds to the original GAN loss
  • Generating out-of-distribution samples close to in-distribution

• Experimental results on toy example and MNIST

Figure 3: The generated samples from original GAN (a)/(c) and proposed GAN (b)/(d).
Contribution 3. Joint Confidence Loss

- We suggest training the proposed GAN using a confident classifier
  - Converse is also possible
We suggest training the proposed GAN using a confident classifier
- Converse is also possible

We propose a joint confidence loss

$$\min_G \max_D \min_\theta \quad \mathbb{E}_{P_{in}(\hat{x}, \hat{y})} \left[ - \log P_\theta(y = \hat{y} | \hat{x}) \right] + \beta \mathbb{E}_{P_G(x)} \left[ KL (U(y) \parallel P_\theta(y|x)) \right]$$

$$+ \mathbb{E}_{P_{in}(\hat{x})} \left[ \log D(\hat{x}) \right] + \mathbb{E}_{P_G(x)} \left[ \log (1 - D(x)) \right].$$

- Classifier’s confidence loss: (c) + (d)
- GAN loss: (d) + (e)
Contribution 3. Joint Confidence Loss

- We suggest training the proposed GAN using a confident classifier
  - Converse is also possible

- We propose a joint confidence loss

\[
egin{align*}
\min_{G} \max_{D} \min_{\theta} & \quad \mathbb{E}_{P_{in}(\hat{x}, \hat{y})} \left[ -\log P_{\theta} (y = \hat{y}|\hat{x}) \right] + \beta \mathbb{E}_{P_{G}(x)} \left[ KL \left( \mathcal{U} (y) \mid \mid P_{\theta} (y|x) \right) \right] \\
& + \mathbb{E}_{P_{in}(\hat{x})} \left[ \log D (\hat{x}) \right] + \mathbb{E}_{P_{G}(x)} \left[ \log (1 - D(x)) \right].
\end{align*}
\]

- Classifier’s confidence loss: (c) + (d)
- GAN loss: (d) + (e)

- Alternating algorithm for optimizing the joint confidence loss

Step 1. update GAN

\[ \nabla_{G(z)} KL \left( \mathcal{U} (y) \mid \mid P_{\theta} (y|G(z)) \right) \]

Step 2. update classifier

\[ \nabla_{G(z)} KL \left( \mathcal{U} (y) \mid \mid P_{\theta} (y|G(z)) \right) \]
Experimental Results: dataset & model

- Model: VGGNet [Christian’ 15] with 13 layers
- In-distribution: CIFAR-10 or SVHN

<table>
<thead>
<tr>
<th>CIFAR-10 [Krizhevsky’ 09]</th>
<th>SVHN [Netzer’ 11]</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 32×32 RGB</td>
<td>• 32×32 RGB</td>
</tr>
<tr>
<td>• 10 classes</td>
<td>• 10 classes</td>
</tr>
<tr>
<td>• 50,000 training set</td>
<td>• 73,257 training set</td>
</tr>
<tr>
<td>• 10,000 test set</td>
<td>• 26,032 test set</td>
</tr>
</tbody>
</table>

- Out-of-distribution: (resized) TinyImageNet and LSUN

<table>
<thead>
<tr>
<th>TinyImageNet</th>
<th>LSUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 32×32 RGB</td>
<td>• 32×32 RGB</td>
</tr>
<tr>
<td>• 200 classes</td>
<td>• 10 classes</td>
</tr>
<tr>
<td>• 10,000 test set</td>
<td>• 10,000 test set</td>
</tr>
</tbody>
</table>

Experimental Results - Metric

- TP = true positive
- FN = false negative
- TN = true negative
- FP = false positive

[Metrics]

- FPR at 95% TPR
  - FPR = FP/(FP + TN), TPR = TP/(TP + FN)
- AUROC (Area Under the Receiver Operating Characteristic curve)
  - ROC curve = relationship between TPR and FPR
- Detection Error
  - Minimum misclassification probability over all thresholds
    \[
    \min_{\delta} \left\{ H (g (x; \sigma) \neq 1 | z = 1) \cdot H (z = 1) + H (g (x; \sigma) \neq 0 | z = 0) \cdot H (z = 0) \right\}
    \]

- AUPR (Area under the Precision-Recall curve)
  - PR curve = relationship between precision=TP/(TP+FP) and recall=TP/(TP+FN)
Experimental Results

- Measure the detection performance of threshold-based detectors
- Confidence loss with some explicit out-of-distribution dataset

<table>
<thead>
<tr>
<th>In-dist</th>
<th>Out-of-dist</th>
<th>Classification accuracy</th>
<th>TNR at TPR 95%</th>
<th>AUROC</th>
<th>Detection accuracy</th>
<th>AUPR in</th>
<th>AUPR out</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVHN</td>
<td>CIFAR-10 (seen)</td>
<td>93.82 / 94.23</td>
<td>47.4 / 99.9</td>
<td>62.6 / 99.9</td>
<td>78.6 / 99.9</td>
<td>71.6 / 99.9</td>
<td>91.2 / 99.4</td>
</tr>
<tr>
<td></td>
<td>TinyImageNet (unseen)</td>
<td></td>
<td>49.0 / 100.0</td>
<td>64.6 / 100.0</td>
<td>79.6 / 100.0</td>
<td>72.7 / 100.0</td>
<td>91.6 / 99.4</td>
</tr>
<tr>
<td></td>
<td>LSUN (unseen)</td>
<td></td>
<td>46.3 / 100.0</td>
<td>61.8 / 100.0</td>
<td>78.2 / 100.0</td>
<td>71.1 / 100.0</td>
<td>90.8 / 99.4</td>
</tr>
<tr>
<td></td>
<td>Gaussian (unseen)</td>
<td></td>
<td>56.1 / 100.0</td>
<td>72.0 / 100.0</td>
<td>83.4 / 100.0</td>
<td>77.2 / 100.0</td>
<td>92.8 / 99.4</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>SVHN (seen)</td>
<td>80.14 / 80.56</td>
<td>13.7 / 99.8</td>
<td>46.6 / 99.9</td>
<td>66.6 / 99.8</td>
<td>61.4 / 99.9</td>
<td>73.5 / 99.8</td>
</tr>
<tr>
<td></td>
<td>TinyImageNet (unseen)</td>
<td></td>
<td>13.6 / 9.9</td>
<td>39.6 / 31.8</td>
<td>62.6 / 58.6</td>
<td>58.3 / 55.3</td>
<td>71.0 / 66.1</td>
</tr>
<tr>
<td></td>
<td>LSUN (unseen)</td>
<td></td>
<td>14.0 / 10.5</td>
<td>40.7 / 34.8</td>
<td>63.2 / 60.2</td>
<td>58.7 / 56.4</td>
<td>71.5 / 68.0</td>
</tr>
<tr>
<td></td>
<td>Gaussian (unseen)</td>
<td></td>
<td>2.8 / 3.3</td>
<td>10.2 / 14.1</td>
<td>50.0 / 50.0</td>
<td>48.1 / 49.4</td>
<td>39.9 / 47.0</td>
</tr>
</tbody>
</table>

Table 1: Performance of the baseline detector (Hendrycks & Gimpel, 2016) using VGGNet. All values are percentages and boldface values indicate relative the better results. For each in-distribution, we minimize the KL divergence term in (1) using training samples from an out-of-distribution dataset denoted by “seen”, where other “unseen” out-of-distributions were only used for testing.

- Classifier trained by our method drastically improves the detection performance across all out-of-distributions

Realistic images such as TinyImageNet (aqua line) and LSUN(green line) are more useful than synthetic datasets (orange line) for improving the detection performance.
Experimental Results

• Joint confidence loss

• Confidence loss with the original GAN (orange bar) is often useful for improving the detection performance

• Joint confidence loss (blue bar) still outperforms all baseline it in all cases
Experimental Results

• Comparison with ODIN [Liang’ 18]

\[
S_i(x; T) = \frac{\exp \left( \frac{f_i(x)}{T} \right)}{\sum_{j=1}^{N} \exp \left( \frac{f_j(x)}{T} \right)},
\]

\[
\tilde{x} = x - \varepsilon \text{sign}(-\nabla_x \log S_{ij}(x; T)),
\]

\[
g(x; \delta, T, \varepsilon) = \begin{cases} 
1 & \text{if } \max_i p(\tilde{x}; T) \leq \delta, \\
0 & \text{if } \max_i p(\tilde{x}; T) > \delta.
\end{cases}
\]
Experimental Results

- Comparison with ODIN [Liang’ 18]

![Graphs showing experimental results](Figure 7: Performances of the baseline detector (Hendrycks & Gimpel, 2016) and ODIN detector (Liang et al., 2017) under various training losses.)
Experimental Results

- Interpretability of trained classifier

![Image](image_url)

(a) In-distribution: SVHN
(b) In-distribution: CIFAR-10

Figure 5: Guided gradient (sensitivity) maps of the top-1 predicted class with respect to the input image under various training losses.

- Classifier trained by cross entropy loss shows sharp gradient maps for both samples from in- and out-of-distributions
- Classifiers trained by the confidence losses do only on samples from in-distribution.
Outline

• Introduction
  • Predictive uncertainty of deep neural networks
  • Summary of contributions

• How to train confident neural networks
  • Training Confidence-Calibrated Classifiers for Detecting Out-of-Distribution Samples [Lee’ 18a]

• Applications
  • Hierarchical novelty detection [Lee’ 18b]

• Conclusion
  • Future work

Hierarchical Novelty Detection

- Novelty detection

Figure 1. An illustration of our hierarchical novelty detection task
Hierarchical Novelty Detection

• Objective

Figure 1. An illustration of our hierarchical novelty detection task
Hierarchical Novelty Detection

• Objective
  • 1. Find the closest known (super-)category in taxonomy
  • 2. Find fine-grained classification for novel categories (i.e., out-of-distribution samples)

Figure 1. An illustration of our hierarchical novelty detection task
Two Main Approaches

- Top-down method (TD)
  - \( p(\text{child}) = \sum_{\text{super}} p(\text{child} \mid \text{super}) \, p(\text{super}) \)

- Inference

\[
\hat{y} = \begin{cases} 
\arg \max_{y'} Pr(y' \mid x, s; \theta_s) & \text{if confident,} \\
\mathcal{N}(s) & \text{otherwise,}
\end{cases}
\]

- Definition of confidence: \( D_{KL}(U(y \mid s) \parallel Pr(y \mid x, s; \theta_s)) \geq \lambda_s \).
Two Main Approaches

• Top-down method (TD)
  • \( p(\text{child}) = \sum_{\text{super}} p(\text{child} \mid \text{super}) \ p(\text{super}) \)

• Inference
  \[
  \hat{y} = \begin{cases} 
  \arg \max_{y'} Pr(y' \mid x, s; \theta_s) & \text{if confident,} \\
  \mathcal{N}(s) & \text{otherwise,}
  \end{cases}
  \]
  Novel class

• Definition of confidence: \( D_{KL}(U(y \mid s) \parallel Pr(y \mid x, s; \theta_s)) \geq \lambda_s \).

• Objective

\[
\min_{\theta_s} \mathbb{E}_{Pr(x,y \mid s)} \left[ \sum_{s} \left( -\log Pr(y \mid x, s; \theta_s) + \right) \right] \\
\quad + \mathbb{E}_{Pr(x,y \mid \mathcal{O}(s))} \left[ D_{KL}(U(y \mid s) \parallel Pr(y \mid x, s; \theta_s)) \right],
\]

\( Pr(x, y \mid \mathcal{O}(s)) \) denotes the data distribution of all exclusive classes from \( s \).
• ImageNet dataset
• 22K classes
• Taxonomy
  • 396 super classes of 1K known leaf classes
  • Rest of 21K classes can be used as novel class
• Example

Experimental Results on ImageNet Dataset

- **ImageNet dataset**
  - 22K classes
  - Taxonomy
    - 396 super classes of 1K known leaf classes
    - Rest of 21K classes can be used as novel class
  - Example

- **Hierarchical novelty detection performance**
  - Baseline: DARTS [Deng’ 12]
    - One can note that our methods have higher novel class accuracy than DARTS to have a same known class accuracy in most regions

Conclusion

• We propose a new method for training **confident** deep neural networks
  • It produce the uniform distribution when the input is not from target distribution

• We show that it can be applied to many machine learning problems:
  • Detecting out-of-distribution problem [Lee’ 18a]
  • Ensemble learning using deep neural networks [Lee’ 17]
  • Hierarchical novelty detection [Lee’ 18b]

• We believe that our new approach brings a refreshing angle for developing confident deep networks in many related applications:
  • Network calibration
  • Adversarial example detection
  • Bayesian probabilistic models
  • Semi-supervised learning