Hot Topics in Machine Learning Safety

Anomaly Detection

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

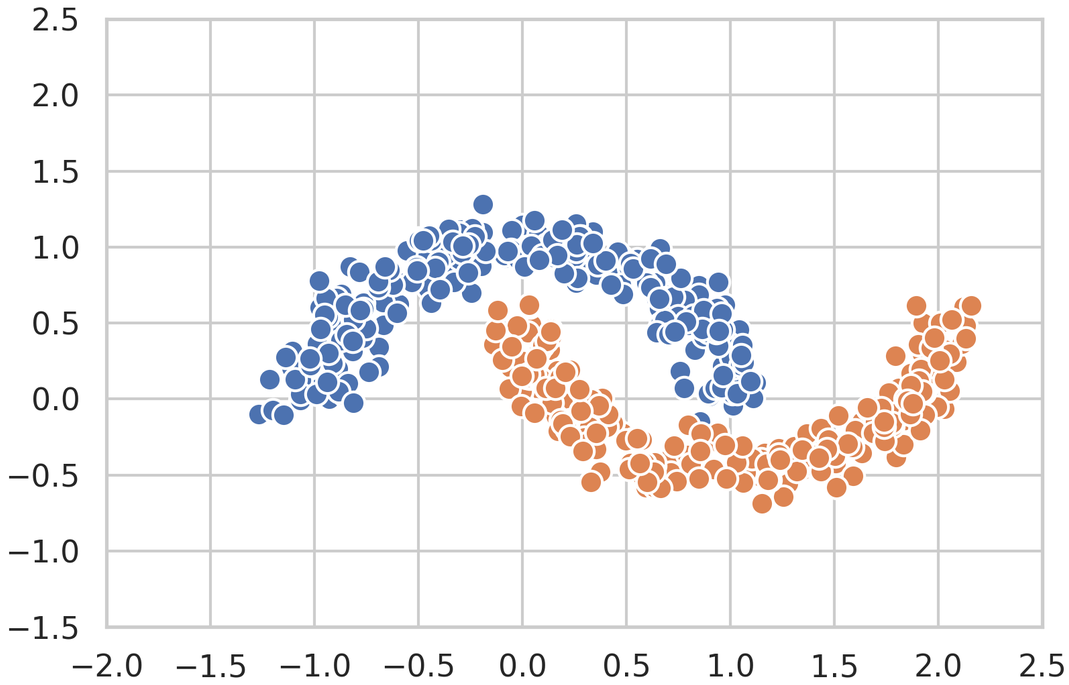
Seminar Schedule

- Introduction and Motivation
- ML Fundamentals
- Testing
- Explainability
- Uncertainty Estimation
- Anomaly Detection 👈
- Adversarial Machine Learning
- Alignment/Societal/Long-Term Risk

Agenda

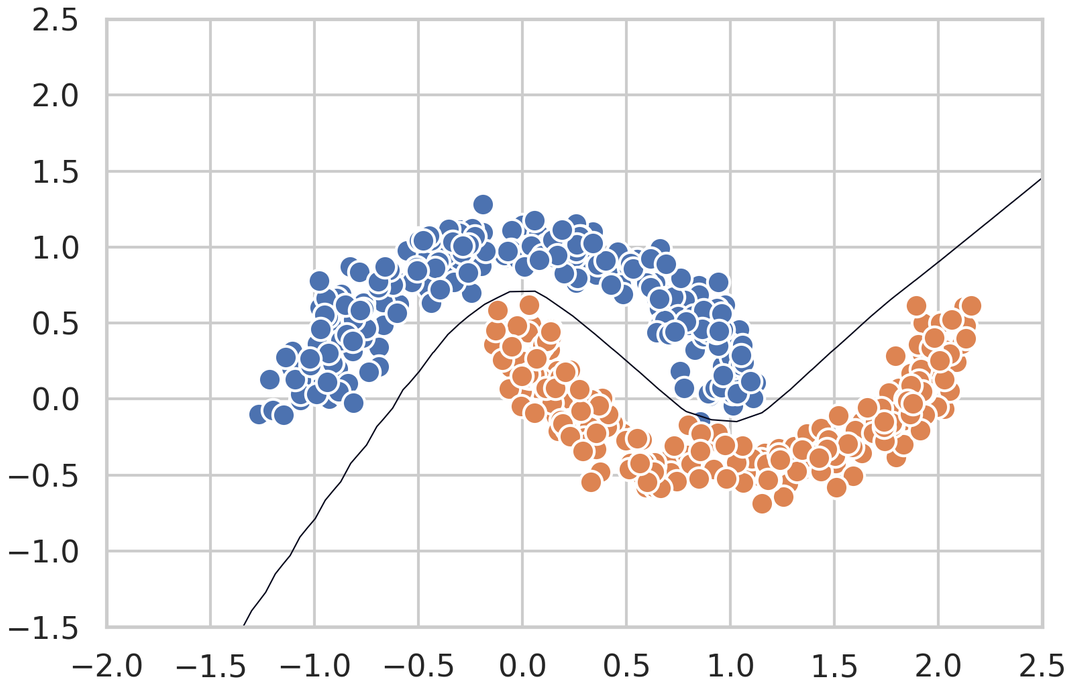
- What is anomaly detection?
- Why do we need anomaly detection?
- How do we do anomaly detection?
  - Baseline
  - Energy-based
  - State-of-the-Art
  - Reconstruction-based

Anomaly Detection

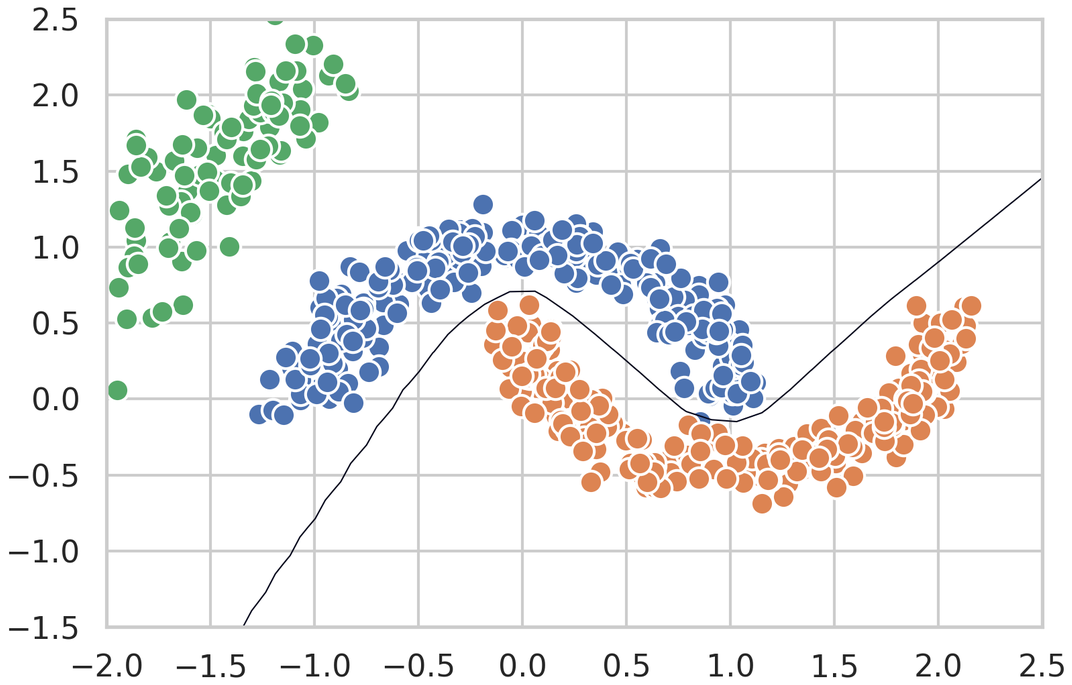


OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

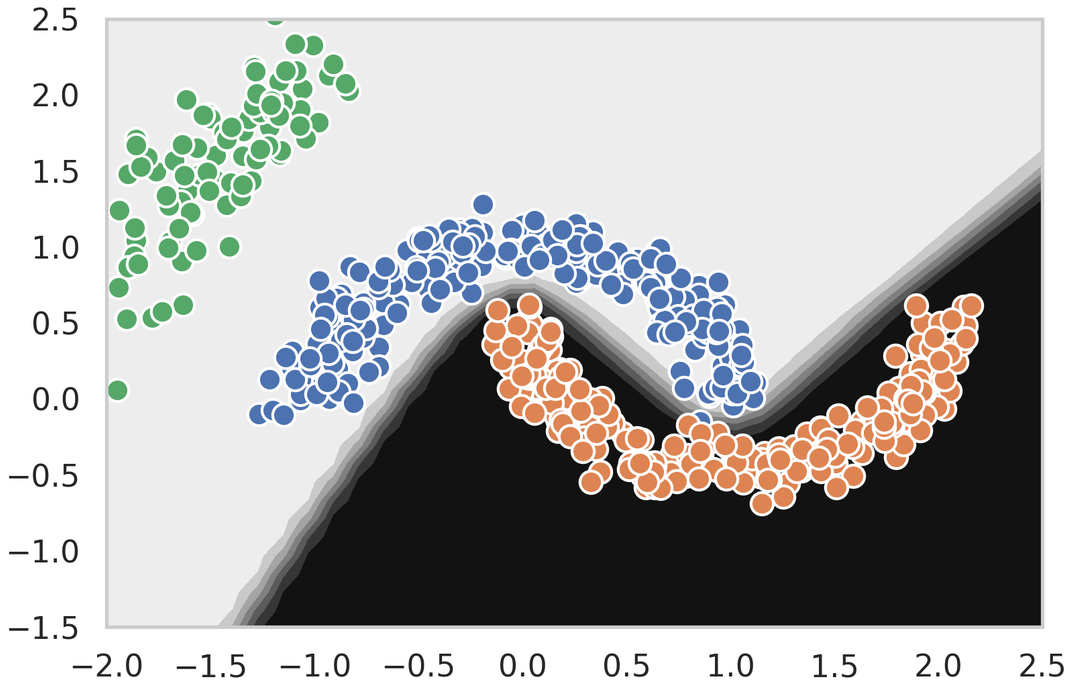
Anomaly Detection



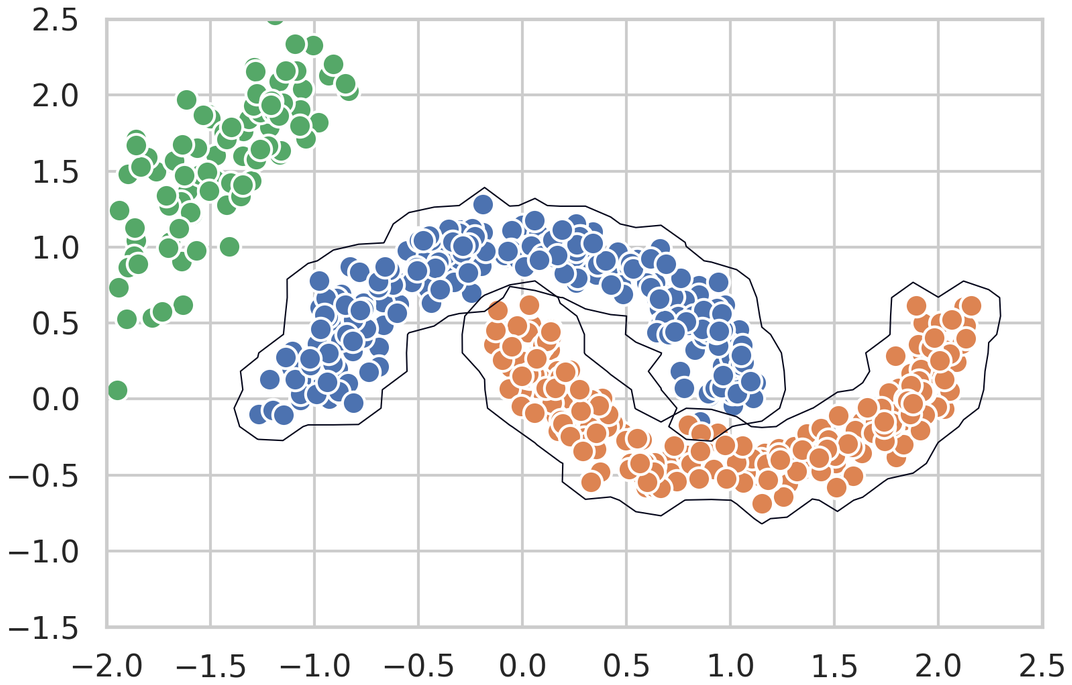
Anomaly Detection



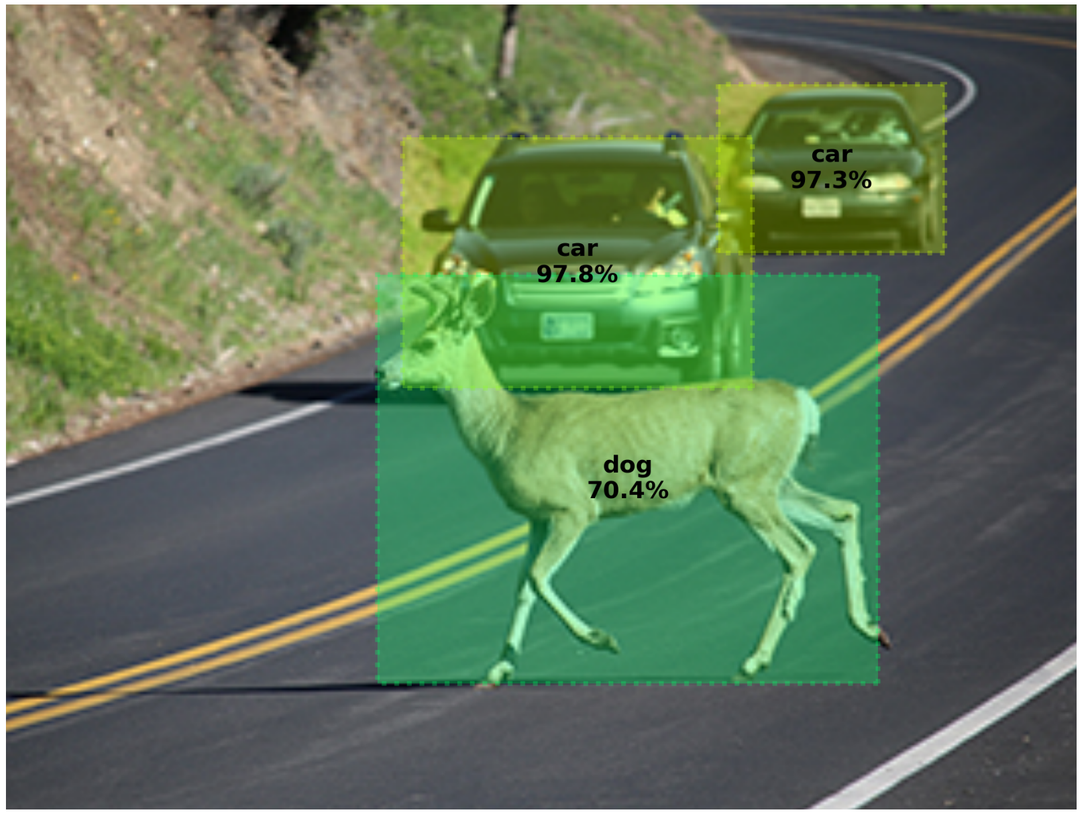
Anomaly Detection



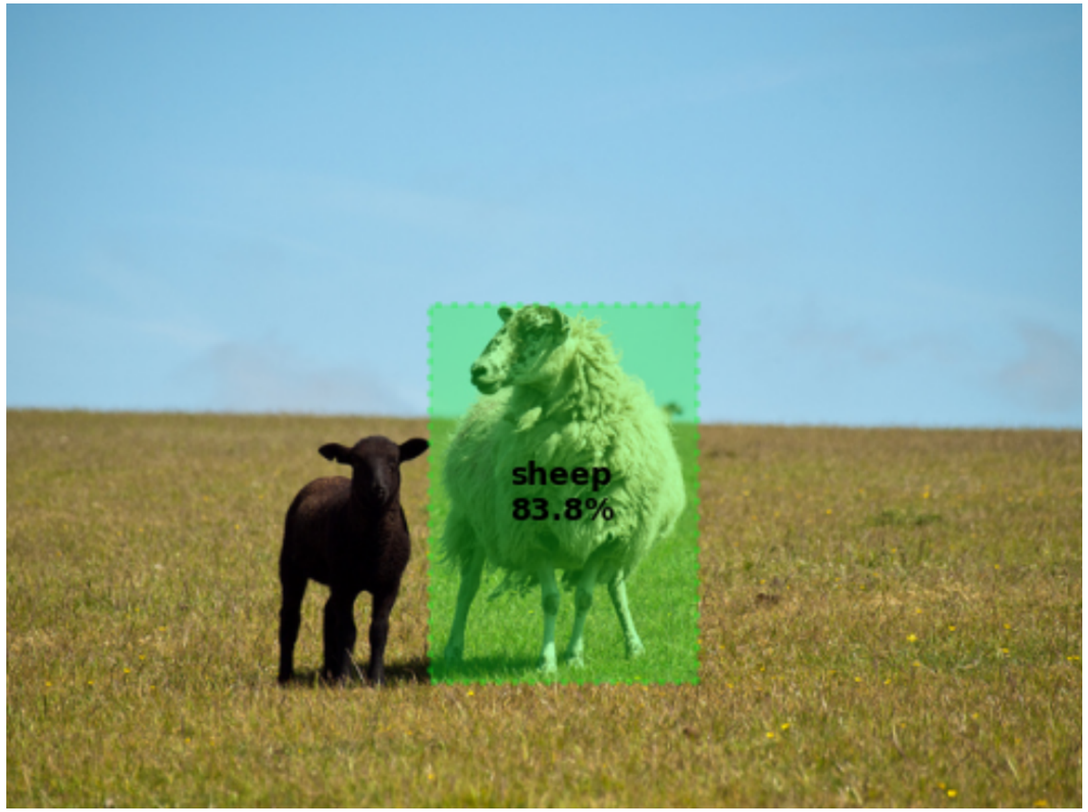
Anomaly Detection



Anomaly Detection



Anomaly Detection



General Theoretical Framework

- Training distribution p
- Training dataset $\mathcal{D} = \{\mathbf{x} \sim p\}$ drawn from p
- Anomalies $\mathcal{A} = \{\mathbf{x} : p(\mathbf{x}) < \alpha\}$ where α is some threshold

Detection

- Detector $D : \mathcal{X} \to \mathbb{R}$
- For decision: apply threshold τ to $D(\mathbf{x})$

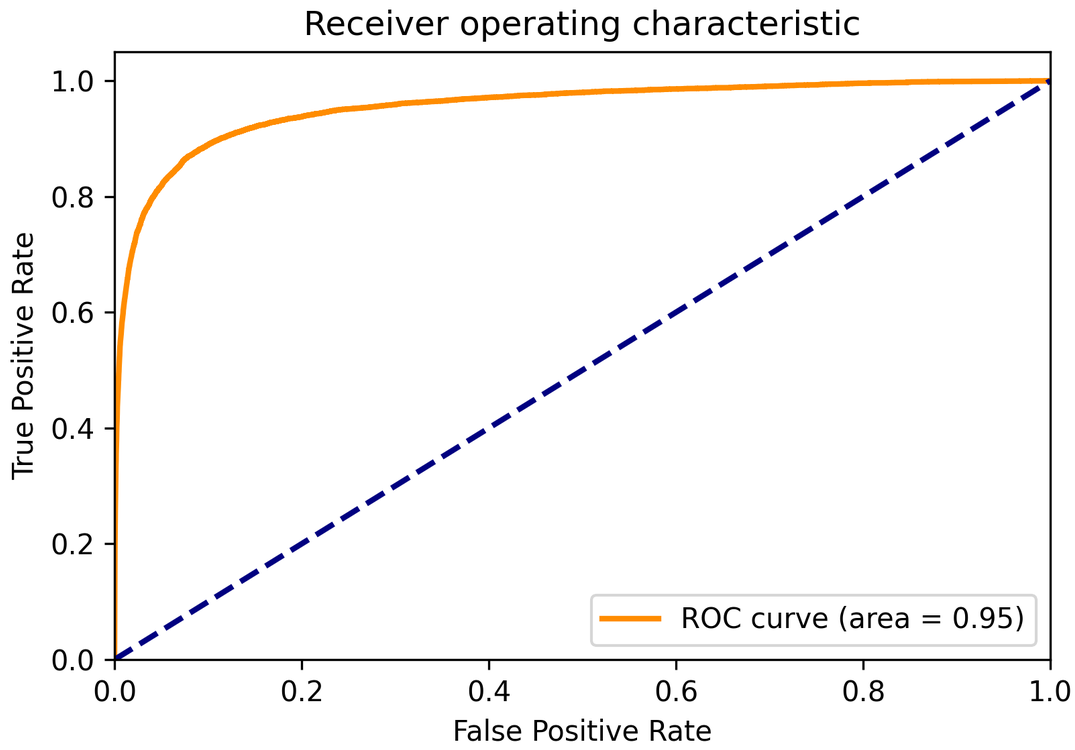
Max-SoftMax Probability (2016)

$$D(\mathbf{x}) = \max_y p(y \mid \mathbf{x})$$

- Usually works to some degree
- Not-Optimal

Area Under Receiver Operating Characteristic

- Plot FPR vs. TPR for all possible τ, measure area
- Probability that randomly seclected outlier receives higher outlier-score then random inlier



Energy-Based (2020)

The essence of the energy-based model (EBM) [20] is to build a function $E(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$ that maps each point $\mathbf{x}$ of an input space to a single, non-probabilistic scalar called the *energy*. A collection of energy values could be turned into a probability density $p(\mathbf{x})$ through the Gibbs distribution:

$$p(y \mid \mathbf{x}) = \frac{e^{-E(\mathbf{x},y)/T}}{\int_{y'} e^{-E(\mathbf{x},y')/T}} = \frac{e^{-E(\mathbf{x},y)/T}}{e^{-E(\mathbf{x})/T}}, \tag{1}$$

where the denominator $\int_{y'} e^{-E(\mathbf{x},y')/T}$ is called the partition function, which marginalizes over y, and T is the temperature parameter. The *Helmholtz free energy* $E(\mathbf{x})$ of a given data point $\mathbf{x} \in \mathbb{R}^D$ can be expressed as the negative of the log partition function:

$$E(\mathbf{x}) = -T \cdot \log \int_{y'} e^{-E(\mathbf{x},y')/T} \tag{2}$$

Energy-Based (2020)

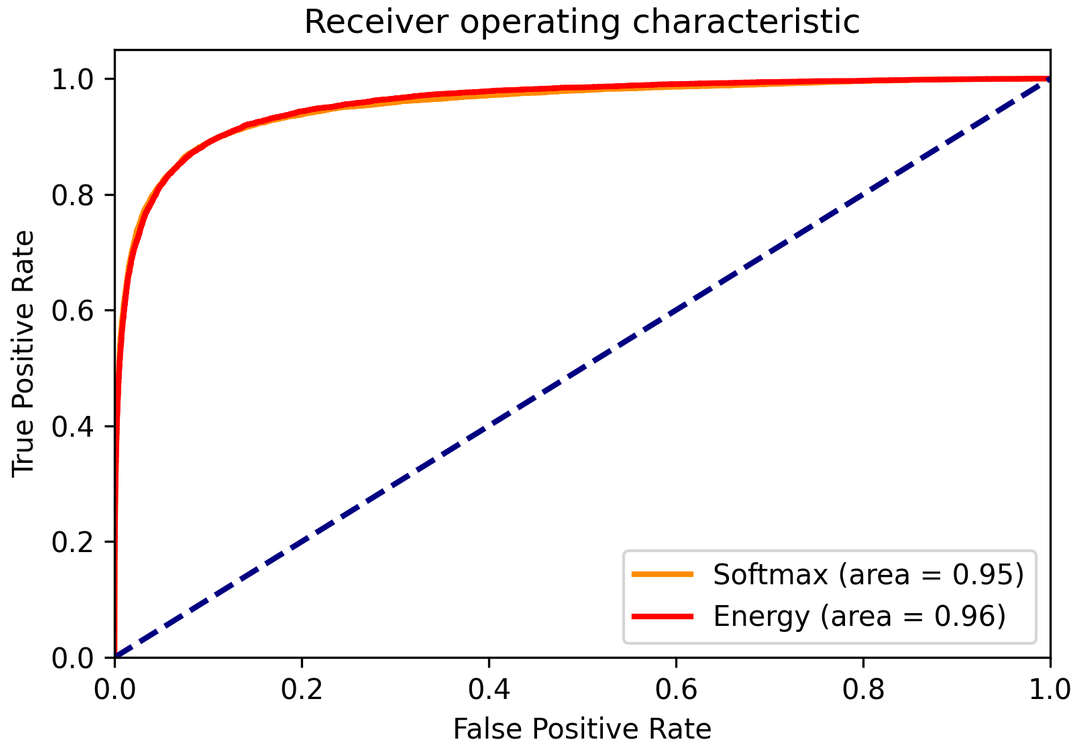
By connecting Eq. 1 and Eq. 3, we can define an energy for a given input $(\mathbf{x}, y)$ as $E(\mathbf{x}, y) = -f_y(\mathbf{x})$. More importantly, without changing the parameterization of the neural network $f(\mathbf{x})$, we can express the free energy function $E(\mathbf{x}; f)$ over $\mathbf{x} \in \mathbb{R}^D$ in terms of the denominator of the softmax activation:

$$E(\mathbf{x}; f) = -T \cdot \log \sum_{i}^{K} e^{f_i(\mathbf{x})/T}. \tag{4}$$

- Also: $T = 1$
- $D(\mathbf{x}) = -\log \sum_i^K e^{f_i(\mathbf{x})}$

Energy-Based (2020)

```
1 logits = model(imgs) # get activations before softmax aka "logits"
2 energies = - logits.exp().sum(dim=1).log()
3
4 # or
5 energies = - torch.logsumexp(logits)
```



Outlier Exposure (2018)

- Best method known today
- Train with example outliers: add another loss term to

$$\min \mathbb{E}_{\mathbf{x},y\sim p_{\text{in}}}[\mathcal{L}(\mathbf{x},y) + \lambda \mathbb{E}_{\mathbf{x}'\sim p_{\text{out}}}[\mathcal{L}(\mathbf{x},y,\mathbf{x}')]]$$

Take Home Assignments (Optional)

- Read "VOS: Learning What You Don't Know by Virtual Outlier Synthesis"
  - https://arxiv.org/abs/2202.01197

- Excercise notebooks will be available soon