Lecture 3: Energy-based models

Announcements

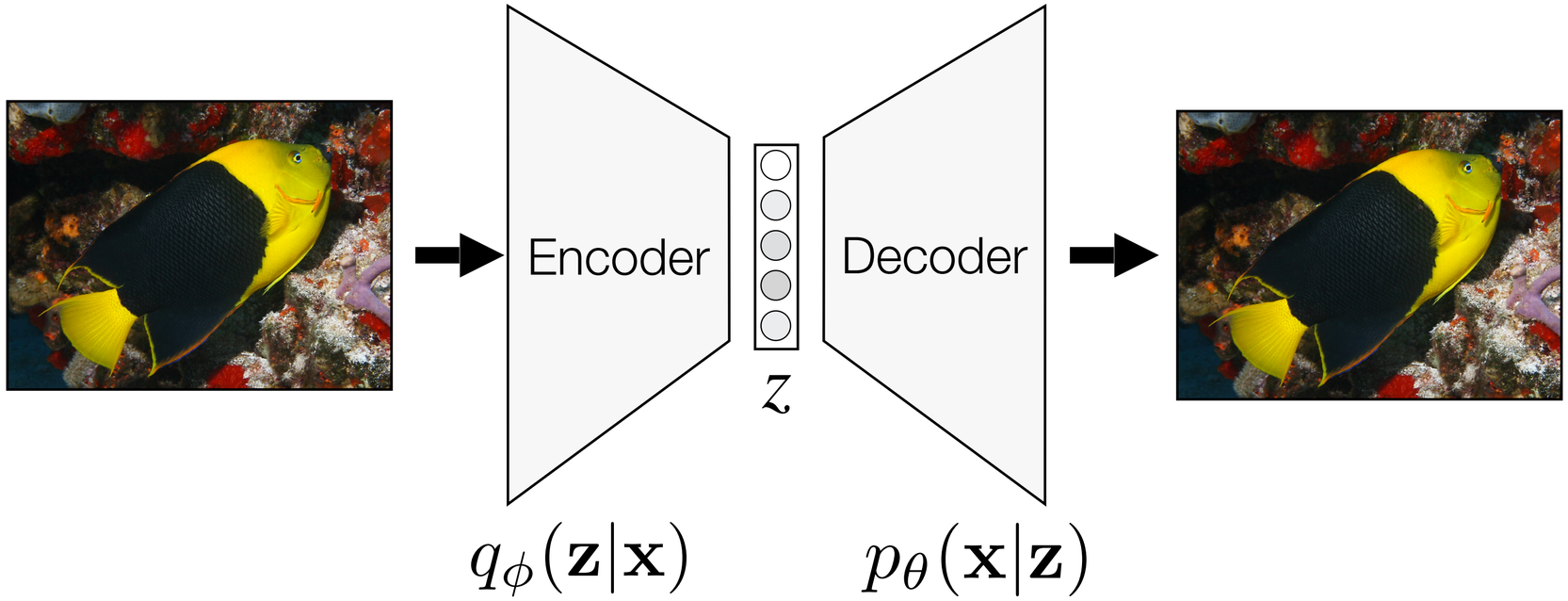
- Students setting up unofficial study groups on Piazza (thanks Zhengyuan Cui!)
- Still short on people for the first few weeks! Please sign up if you can.
- We'll start taking attendance next week.
- If you haven't submitted a paper review this week, please do so by 11:59pm tonight!
  - In future weeks, they'll be due before class.

Lecture 3: Energy-based models

Background / How to Train Your Energy-based Models

Yang Song & Diederik Kingma

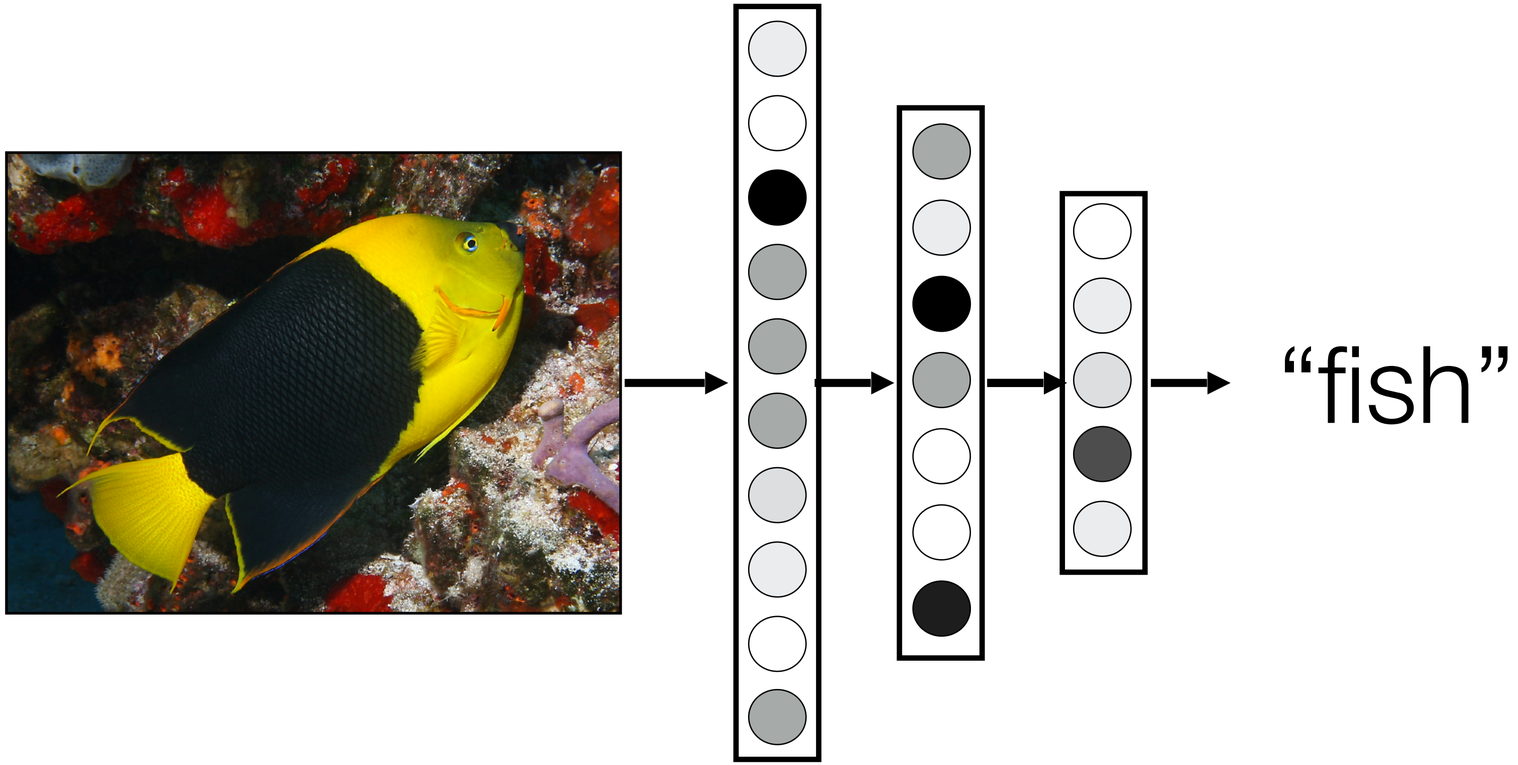
Latent variable model



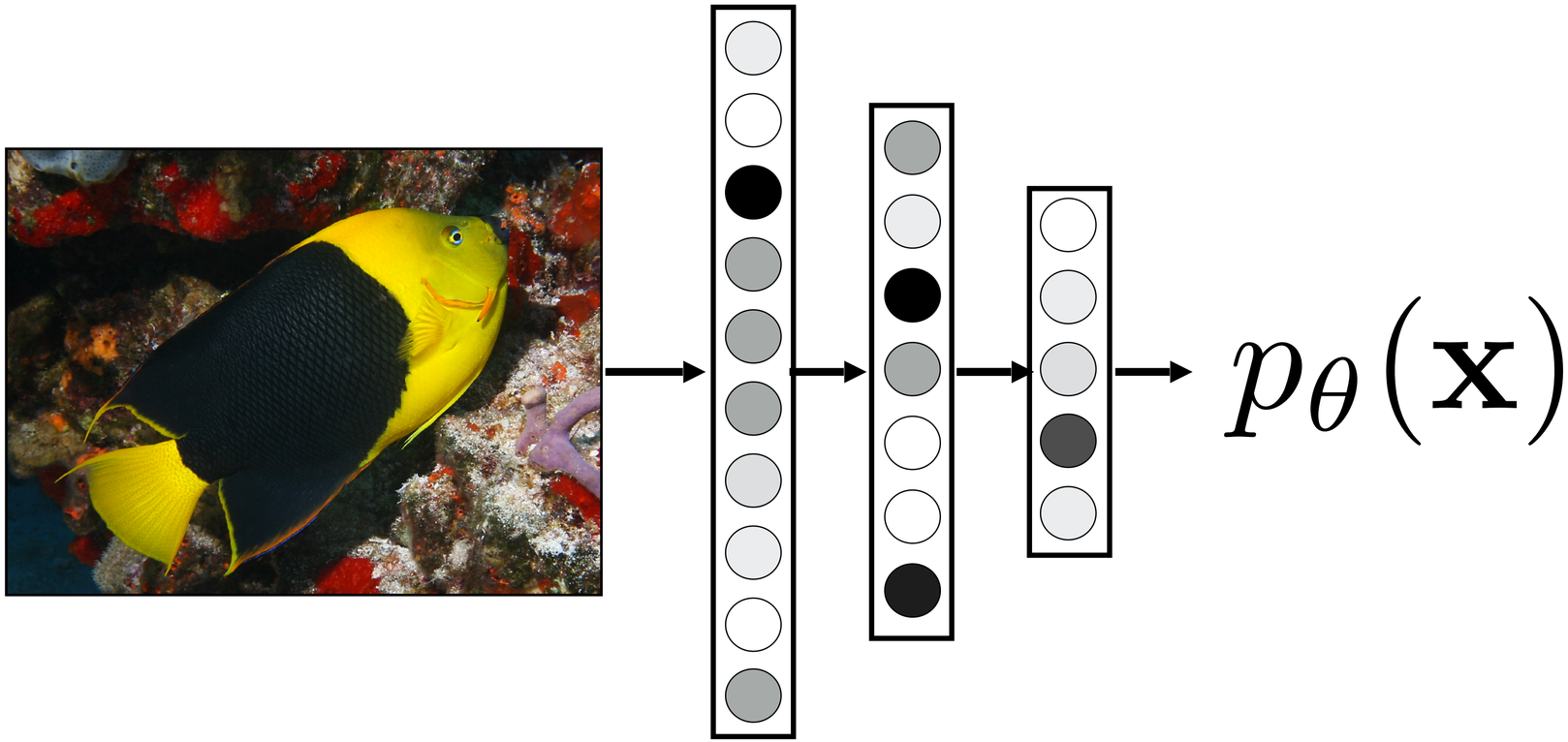
e.g. variational autoencoder

But isn’t this a bit of a roundabout way of solving the problem?

Simpler idea: train network to output probabilities?



Simpler idea: train network to output probabilities?



What makes this idea challenging? Needs to sum to 1!

Energy-based model

Energy function implemented by neural net, $E_\theta(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$

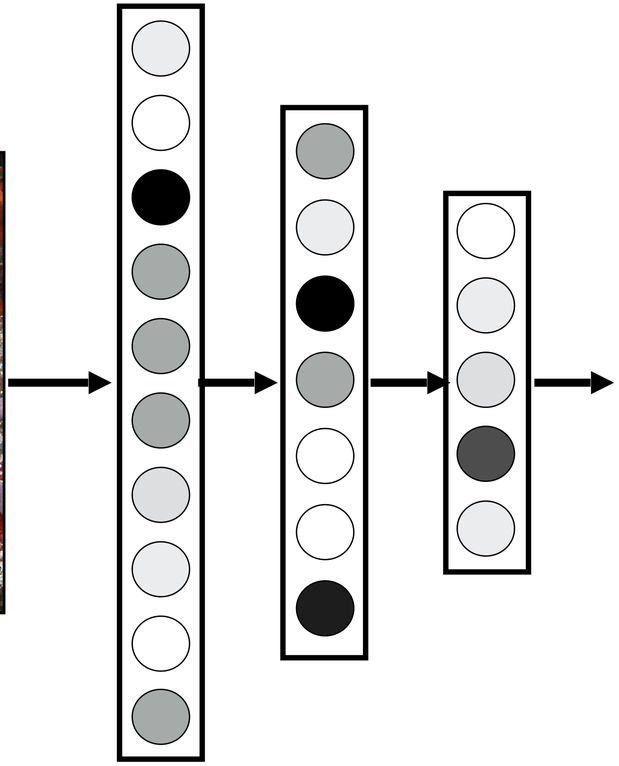
$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z(\theta)}$$

Normalization constant a.k.a. **partition function**

where $$Z(\theta) = \int_{\mathbf{x}} \exp(-E_\theta(\mathbf{x}))d\mathbf{x}$$

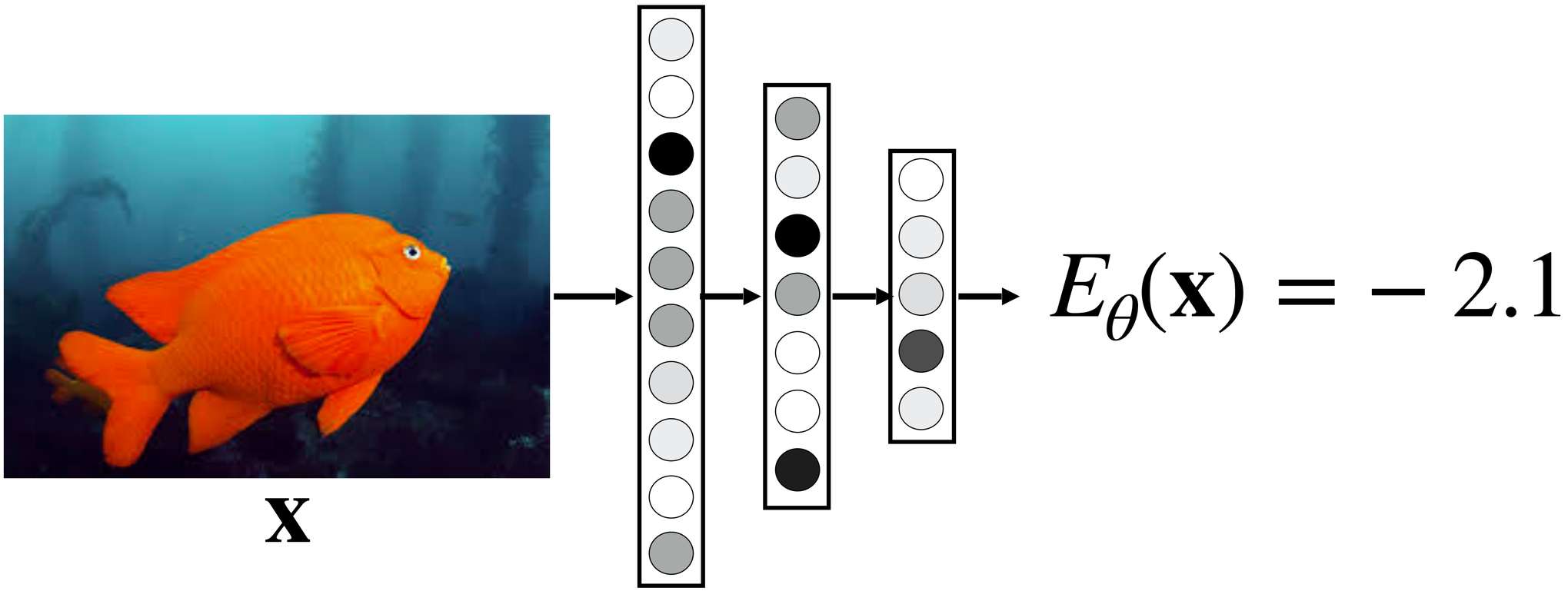
Energy function

$\mathbf{x}$

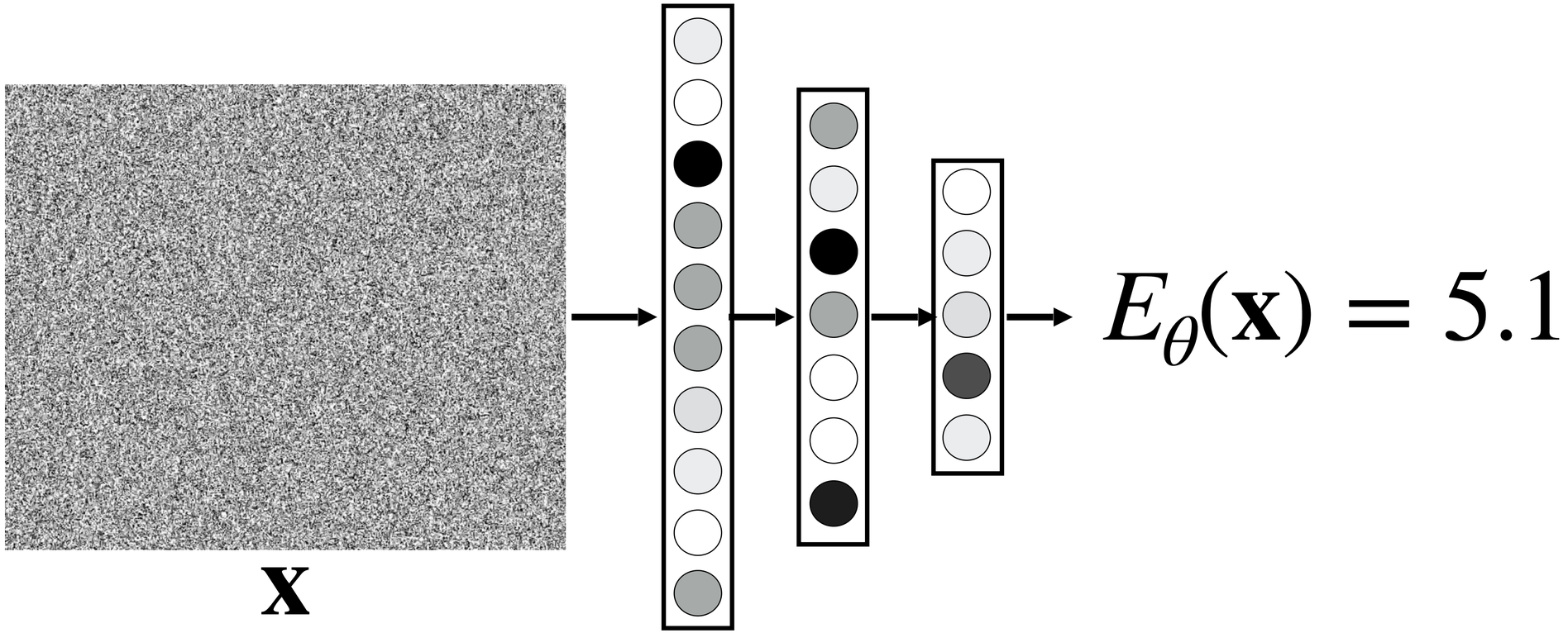


$$E_\theta(\mathbf{x}) = 0.1$$

Energy function



Energy function



When computing $Z(\theta)$, we need to integrate over the full input space, which makes it challenging.

Energy-based models

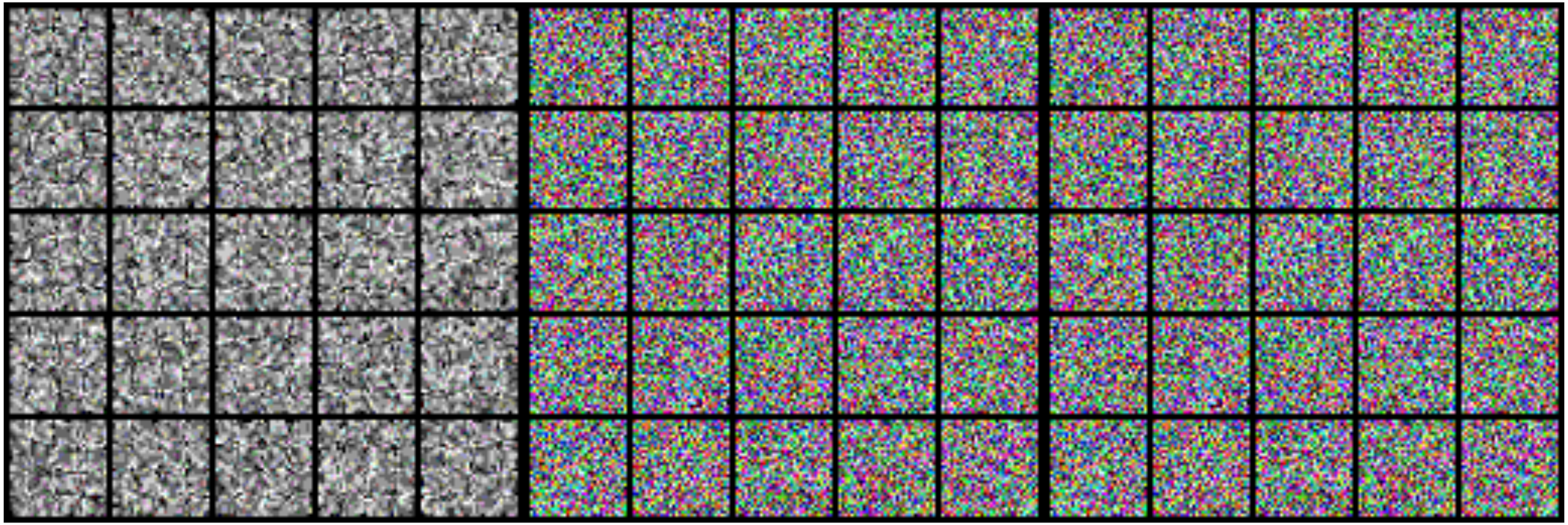
+ Very flexible. Define $E_\theta(\,.\,)$ architecture however you'd like!
+ Easy to estimate (unnormalized) probabilities.
+ Simple idea. No encoder, latent code (unlike VAE)

– No latent code (unlike VAE)
– Hard to generate samples.
– Hard to train. Generally requires MCMC.

Sampling from an EBM

- We want to draw a **random sample** from $p_\theta(\mathbf{x})$.
- Let's use the most general-purpose tool we have for this: Markov Chain Monte Carlo (MCMC).
- In generally, this is very hard to do! But we can exploit the fact that we can take compute $\nabla_{\mathbf{x}} \log(p_\theta(\mathbf{x}))$ easily.

Sampling example



[Song & Ermon, “Generative Modeling by Estimating Gradients of the Data Distribution”, 2019]

Langevin MCMC

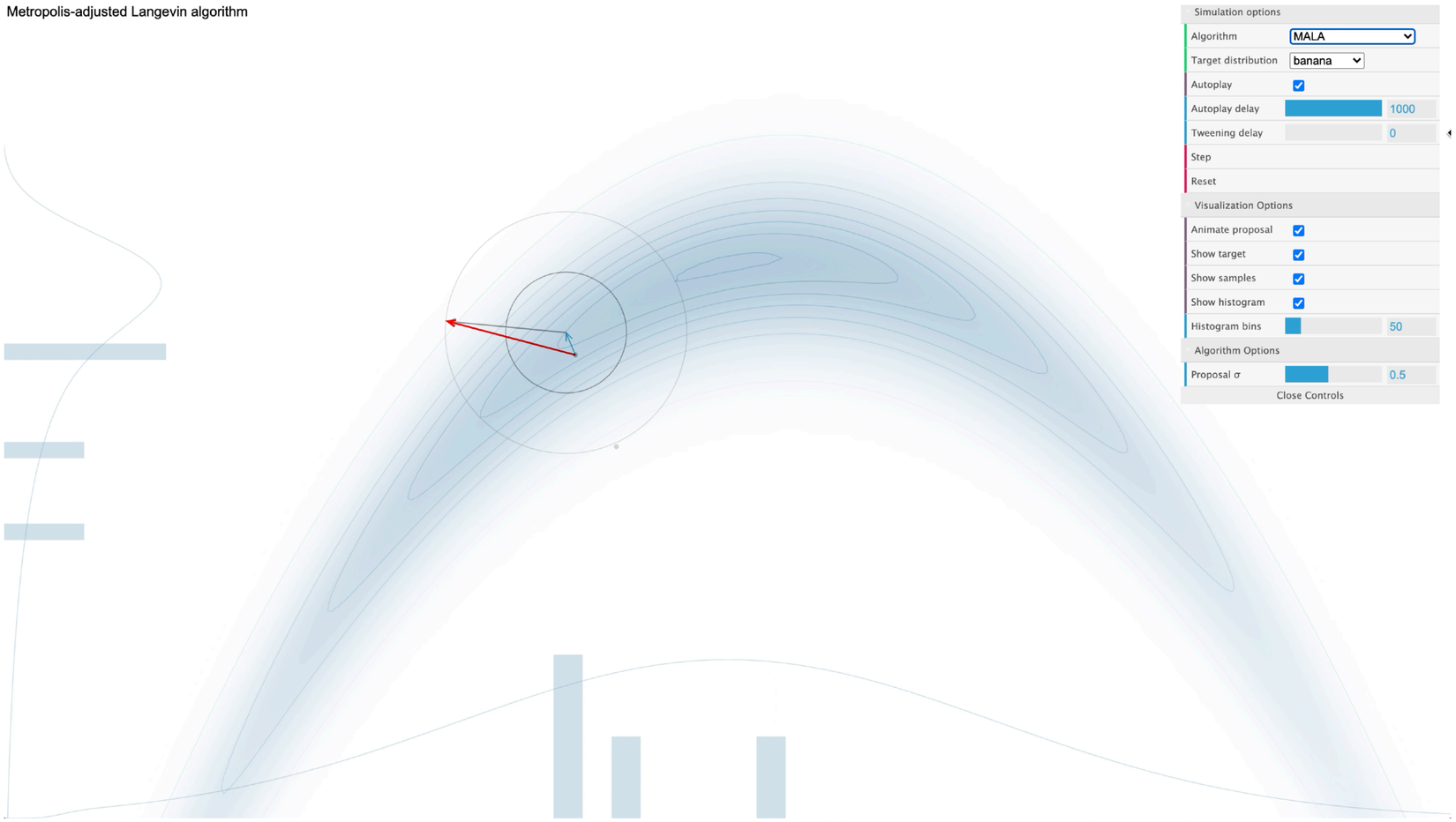
- MCMC method that exploits gradients
- Start with a random noise image $\mathbf{x}_0 \sim U(0,1)$
- Update the random noise by taking a gradient step and adding noise.

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \frac{\alpha}{2}\nabla_{\mathbf{x}}\log(p_\theta(\mathbf{x}_t)) + \epsilon, \qquad \epsilon \sim N(0,\ \alpha)$$

where α controls the step size.

- Recall: $\nabla_{\mathbf{x}}\log(p_\theta(\mathbf{x}_t)) = \nabla_{\mathbf{x}}\frac{1}{Z(\theta)}\exp(-E_\theta(\mathbf{x})) = -\nabla_{\mathbf{x}}E_\theta(\mathbf{x})$

Langevin MCMC



Maximum likelihood

Want to maximize:

$$\mathbb{E}_{\mathbf{x}\sim p_{data}}\left[\log p_\theta(\mathbf{x})\right] \approx \frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{\exp(-E_\theta(\mathbf{x}_i))}{Z(\theta)}\right))$$

Consider doing gradient ascent:

$$\nabla_\theta \log(p_\theta(\mathbf{x})) = -\nabla_\theta E_\theta(\mathbf{x}) - \nabla_\theta \log Z(\theta)$$

Maximum likelihood

$$\nabla_\theta \log(p_\theta(\mathbf{x})) = -\nabla_\theta E_\theta(\mathbf{x}) - \nabla_\theta \log Z(\theta)$$

Problem: The normalizing constant depends on θ, and it's intractable to compute it (or its derivative) directly!

$$\log Z(\theta) = \log \int_{\mathbf{x}} \exp(-E_\theta(\mathbf{x}))d\mathbf{x} \qquad \nabla_\theta \log Z(\theta) = ?$$

Maximum likelihood via sampling

By algebraic manipulation, we can show that:

$$\nabla_\theta \log Z(\theta) = \mathbb{E}_{\mathbf{x} \sim p_\theta} \left[-\nabla_\theta E_\theta(\mathbf{x}) \right]$$

This is convenient, since we already know how to draw samples!

$\nabla_\theta \log Z(\theta) \approx \nabla_\theta E_\theta(\tilde{\mathbf{x}})$ where $\tilde{\mathbf{x}}$ is an MCMC sample

Sometimes called stochastic gradient Langevin dynamics (SGLD)

Training tricks

Putting these equations together:

- $\nabla_\theta \log(p_\theta(\mathbf{x})) = -\nabla_\theta E_\theta(\mathbf{x}) - \nabla_\theta \log Z(\theta)$
- $\nabla_\theta \log Z(\theta) \approx \nabla_\theta E_\theta(\tilde{\mathbf{x}})$ where $\tilde{\mathbf{x}}$ is an MCMC sample

Learning rule:

$$\nabla_\theta \log(p_\theta(\mathbf{x})) \approx -\nabla_\theta E_\theta(\mathbf{x}) - \nabla_\theta E_\theta(\tilde{\mathbf{x}})$$

Contrastive divergence [Hinton 2002]: instead of sampling $\tilde{\mathbf{x}}$ from scratch, which is slow, initialize the MCMC chain using our training example, $\mathbf{x}$. Also only run the chain for a few iterations.

Questions?

Score matching

If you only want samples, don't even bother modeling $E_\theta(\mathbf{x})$, train net to predict $\nabla \log(p_\theta(\mathbf{x}))$, known as the "score" directly. This is all you need!

$$\text{minimize} \quad \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\frac{1}{2}\left\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x})\right\|^2\right]$$

Samples from [Song & Ermon 2020]

Recall

+ **Very flexible. Define $E_\theta(\,.\,)$ architecture however you'd like!**
+ Easy to estimate (unnormalized) probabilities.
+ Simple idea. No encoder, latent code (unlike VAE)
– No latent code (unlike VAE)
– Hard to generate samples.
– Hard to train. Generally requires MCMC.

Your Classifier is Secretly an Energy Based Model and You Should Treat it Like One

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, Kevin Swersky

ICLR 2020

In principle, unsupervised learning should let you:

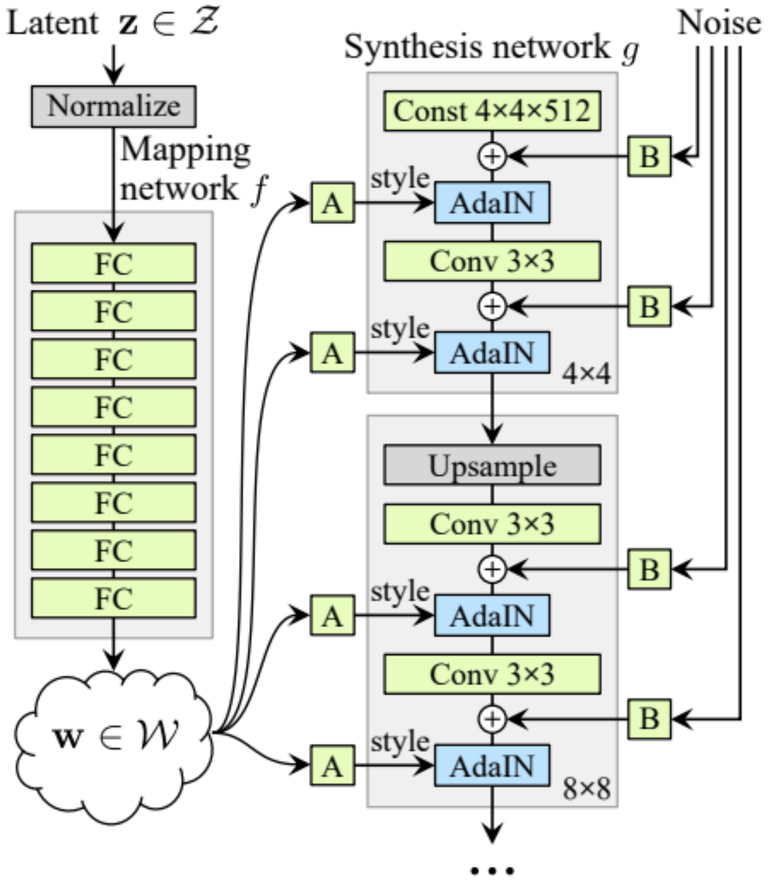
- Draw samples
- Get good features
- Get good log likelihood estimates

… but also a lot more stuff, too:

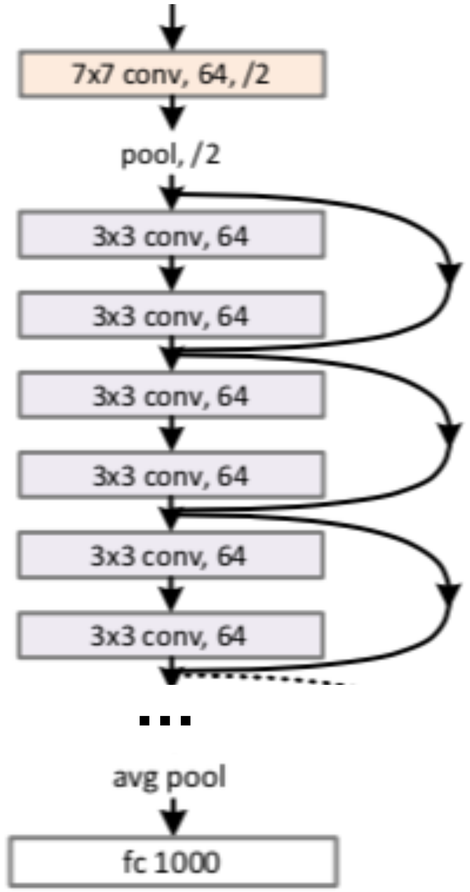
- Semi-supervised learning
- Fill in missing data
- Calibration of uncertainty
- Better adversarial robustness

Architectures have diverged from discriminative models

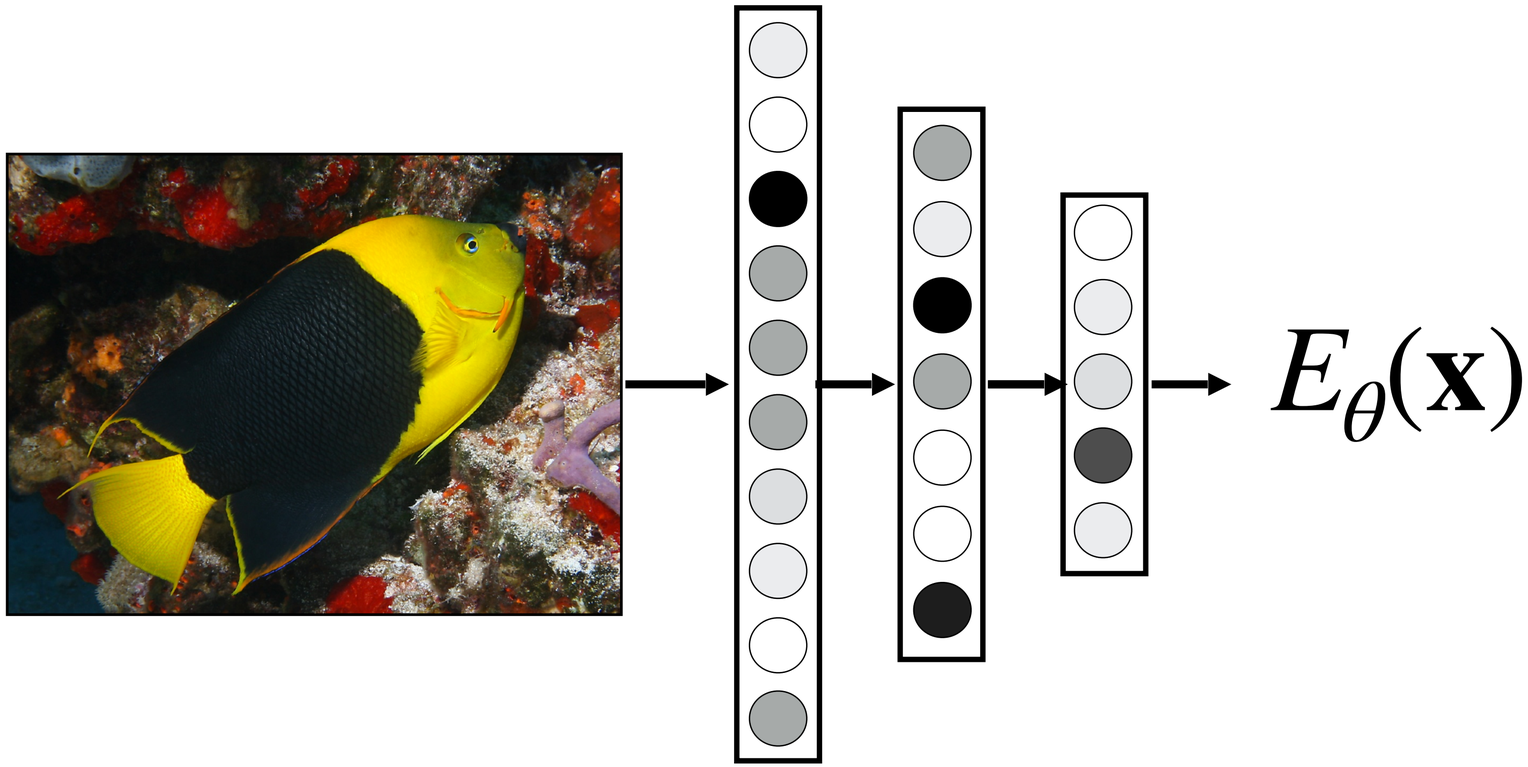
StyleGAN
[Karras, Laine, Aila 2019]



ResNet
[He et al., 2016]

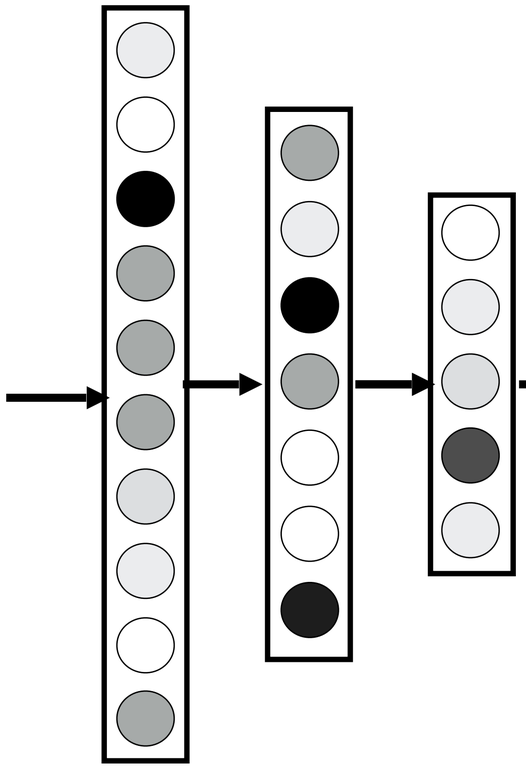


But energy-based models are flexible!



- We can define $E_\theta(\mathbf{x})$ any way we like!
- We have (decent) methods for learning them if we can take gradients

Repurposing a classifier as an EBM



“fish”

$$p_\theta(y \mid \mathbf{x}) = \frac{\exp\left(f_\theta(\mathbf{x})[y]\right)}{\sum_{y'} \exp\left(f_\theta(\mathbf{x})[y']\right)}$$

Repurposing a classifier as an EBM

Consider the joint distribution:

$$p_\theta(\mathbf{x}, y) = \frac{\exp\left(f_\theta(\mathbf{x})[y]\right)}{Z(\theta)}$$

Marginalizing out y:

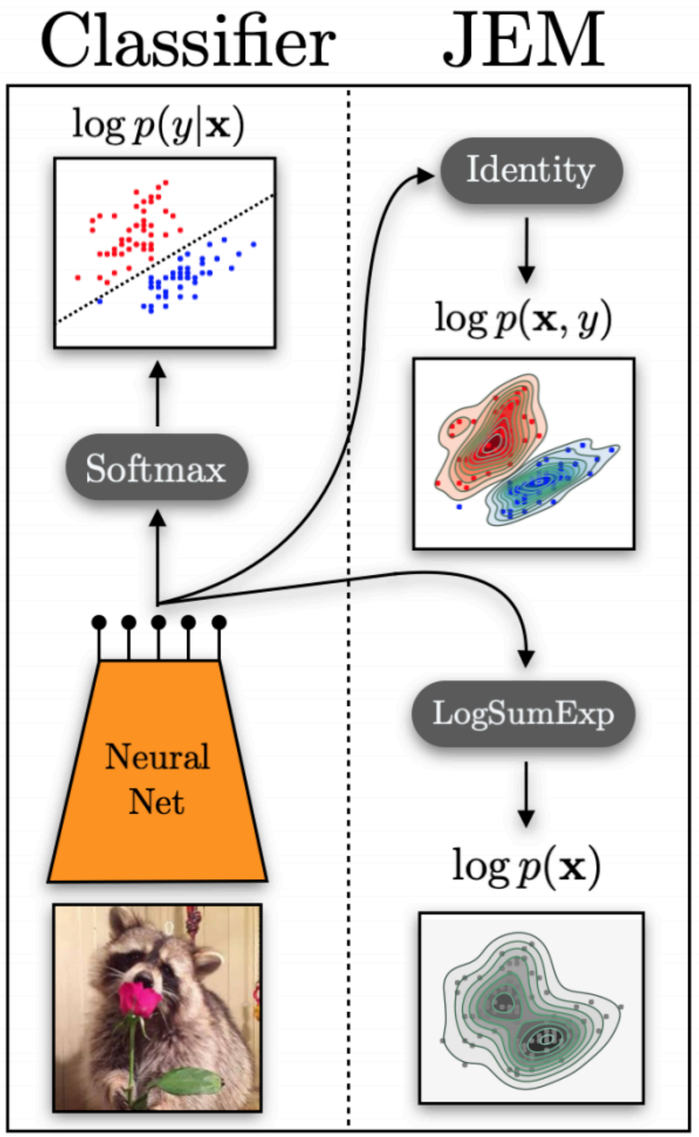
$$p_\theta(\mathbf{x}) = \sum_y p_\theta(\mathbf{x}, y) = \frac{\sum_y \exp\left(f_\theta(\mathbf{x})[y]\right)}{Z(\theta)}$$

This gives us an EBM:

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

Unlike traditional classifier, logits are not shift-invariant. We use this degree of freedom to define $E_\theta(\mathrm{x})$!

Joint Energy-based Model (JEM)



Training

Joint distribution:

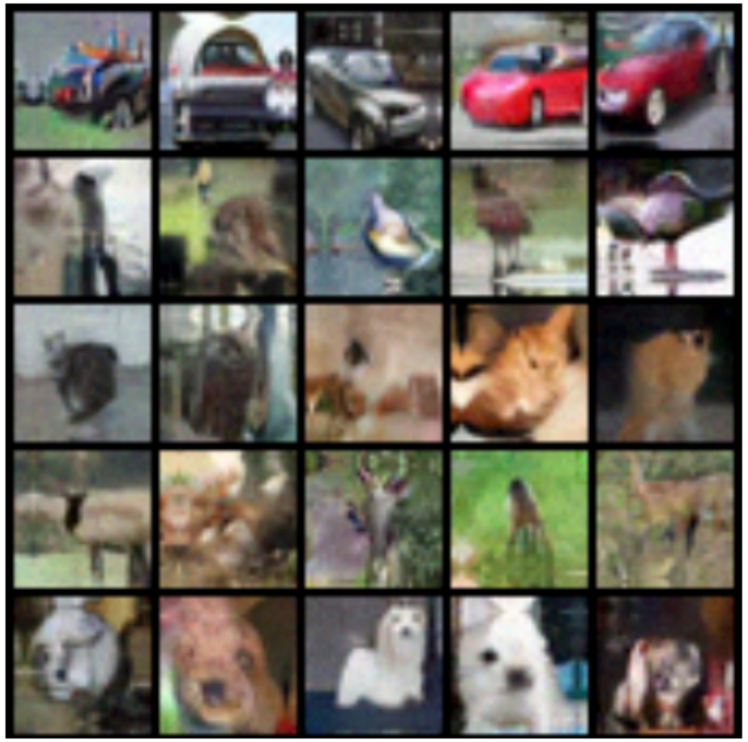
$$\log p_\theta(\mathbf{x}, y) = \log p_\theta(\mathbf{x}) + \log p_\theta(y|\mathbf{x})$$

- Optimize $\log p_\theta(y|\mathbf{x})$ using standard gradient descent
- Optimize $\log p_\theta(\mathbf{x})$ using SGLD (like we just saw).

Observations:

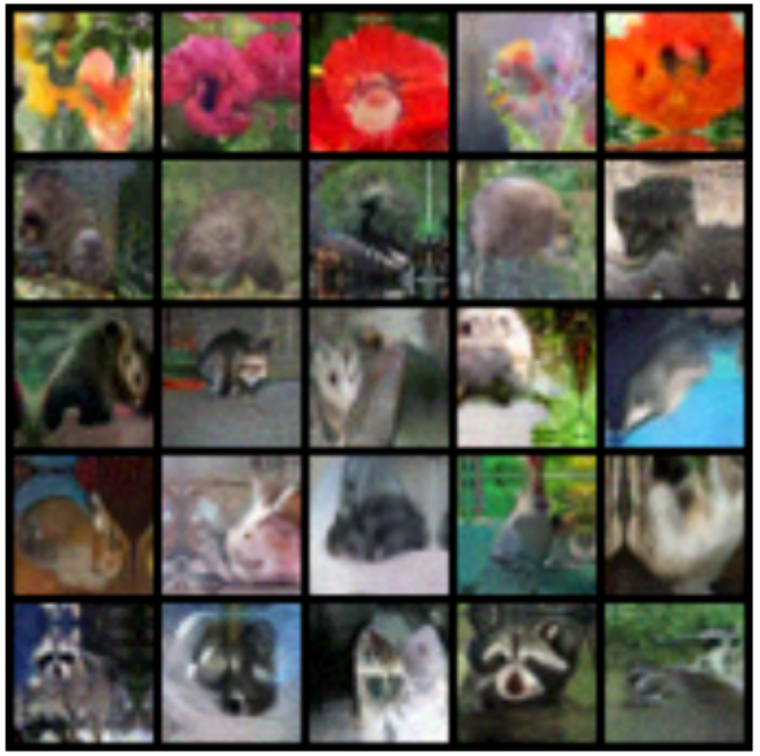
- Other factorizations, like $\log(p(\mathrm{x}\,|\,y)) + \log(p(y))$ don't work as well
- Very hard to train!
- Need 20 MCMC steps, even after adding lots of tricks.
- Still often diverges

Class-conditional samples



CIFAR10

SVHN



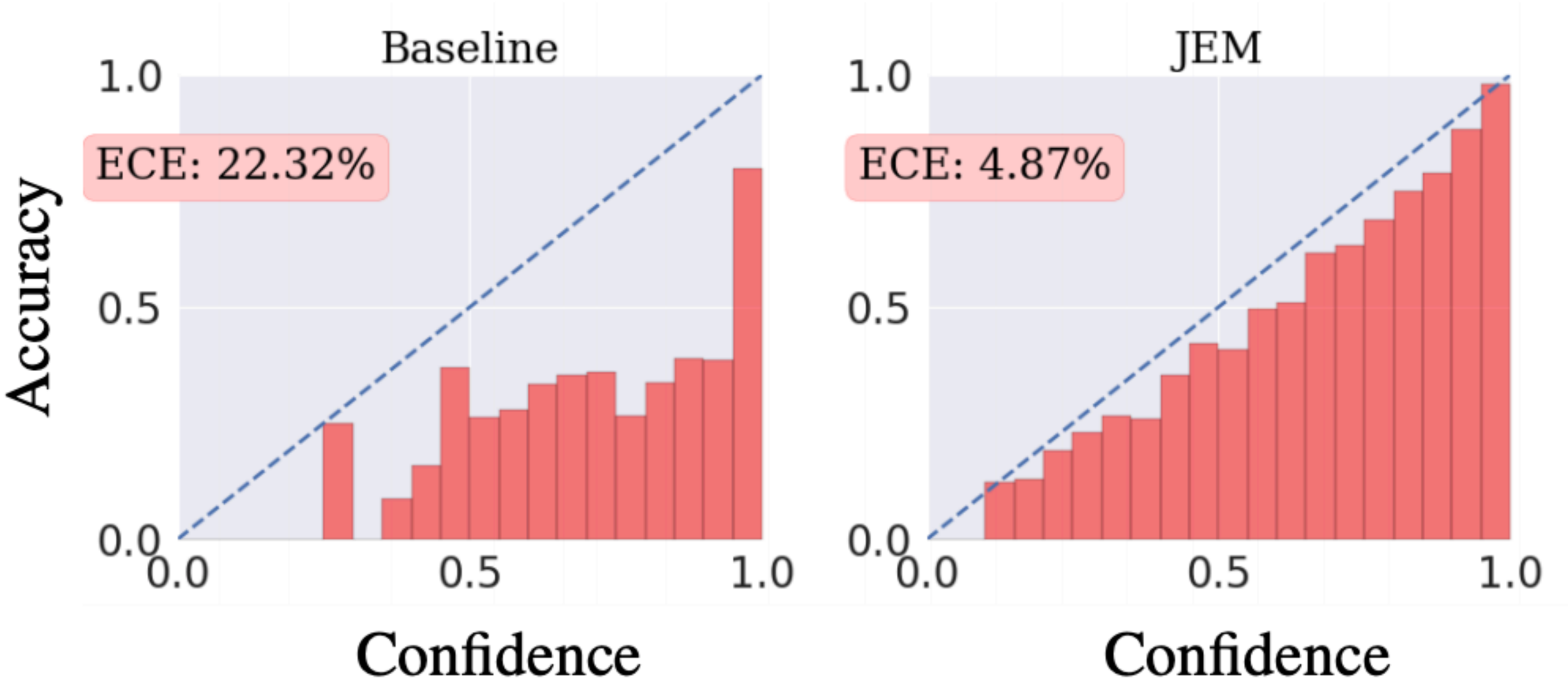
CIFAR100

Classification accuracy

Class	Model	Accuracy% ↑	FID↓
Hybrid	Residual Flow	70.3	46.4
	Glow	67.6	48.9
	IGEBM	49.1	**37.9**
	JEM $p(\mathbf{x}\|y)$ factored	30.1	61.8
	JEM (Ours)	**92.9**	38.4
Disc.	Wide-Resnet	95.8	N/A
Gen.	SNGAN	N/A	25.5
	NCSN	N/A	25.32

Slight performance drop. 93.6% with apples-to-apples comparison.

Calibration



Out-of-distribution detection

Low probability assigned by $s(\mathbf{x}) = p_\theta(\mathbf{x})$? Image is out of distribution!

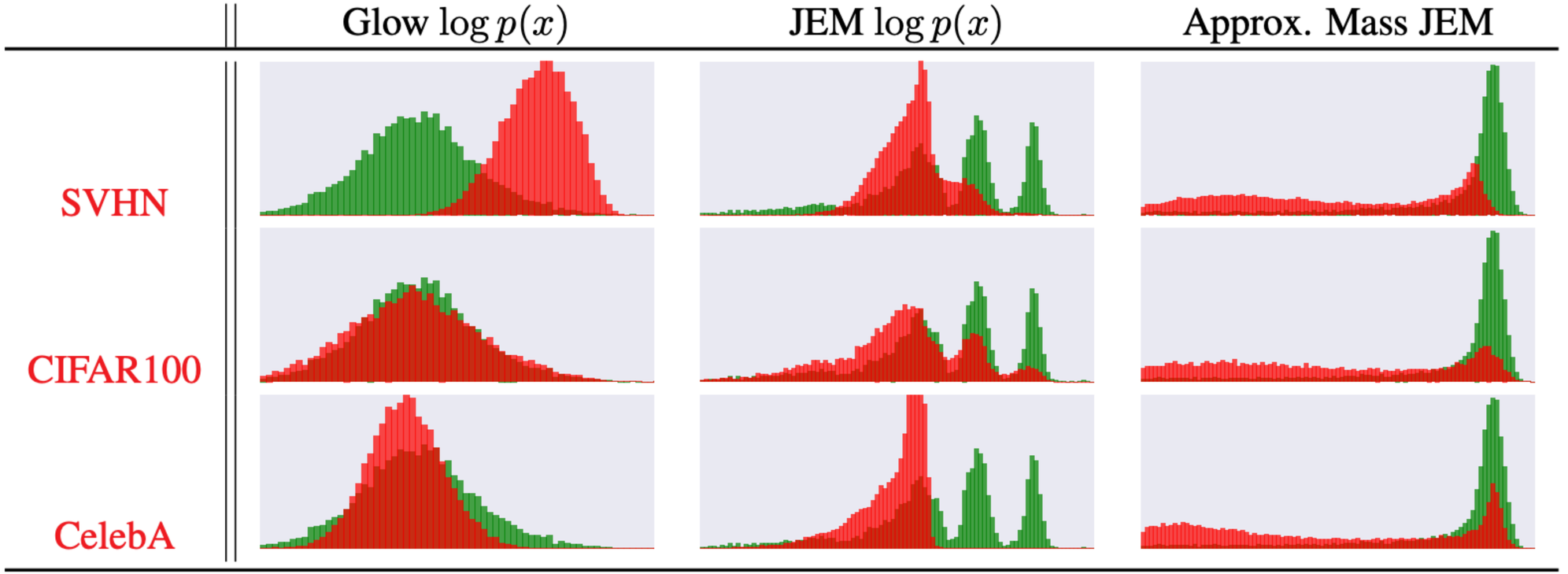


Table 2: Histograms for OOD detection. All models trained on CIFAR10. Green corresponds to the score on (in-distribution) CIFAR10, and red corresponds to the score on the OOD dataset.

Out-of-distribution detection

Can also use class predictions, though pretty similar to discriminative model

$$s(\mathrm{x}) = \min p(y \mid \mathrm{x})$$

$s_\theta(\mathbf{x})$	Model	CIFAR10 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**

Robustness

Hypothesis: use generative model to become more robust to adversaries

Adversarial example: Shift an example by a tiny amount to change the classification result:

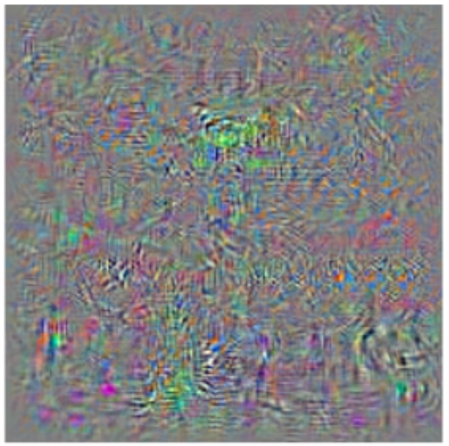
$\tilde{\mathrm{x}} = \mathrm{x} + \delta$**,** where δ is a tiny vector

$$\operatorname{argmax}\ p(y \mid \tilde{\mathrm{x}}) \neq \operatorname{argmax}\ p(y \mid \mathrm{x})$$

Robustness

$\mathbf{x}$ $\mathbf{r}$ $\mathbf{x} + \mathbf{r}$

+



=

f f

y “School bus” “Ostrich”

$$\arg\max_{\mathbf{r}} p(y = \texttt{ostrich} | \mathbf{x} + \mathbf{r}) \quad \text{subject to} \quad \|\mathbf{r}\| < \epsilon$$

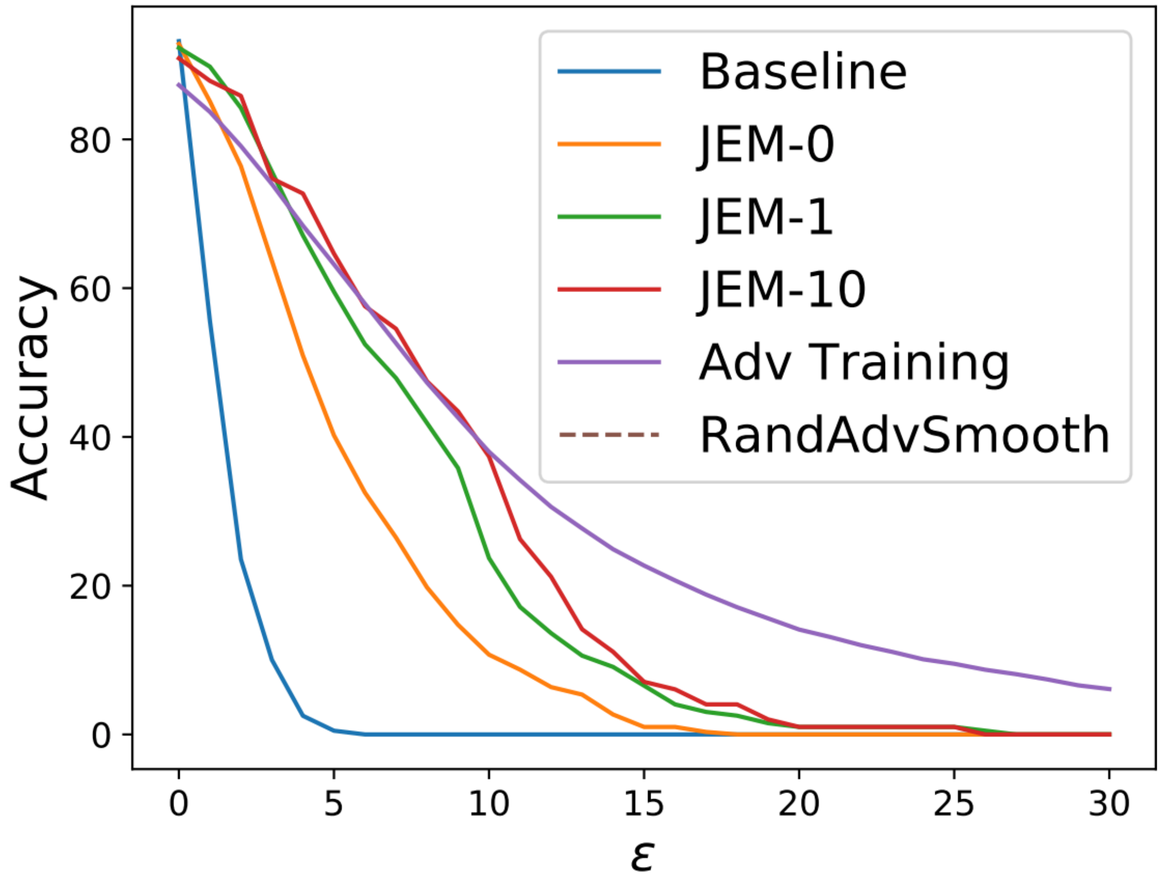
[“Intriguing properties of neural networks”, Szegedy et al. 2014]

Source: Isola, Torralba, Freeman

How can this give you robustness?

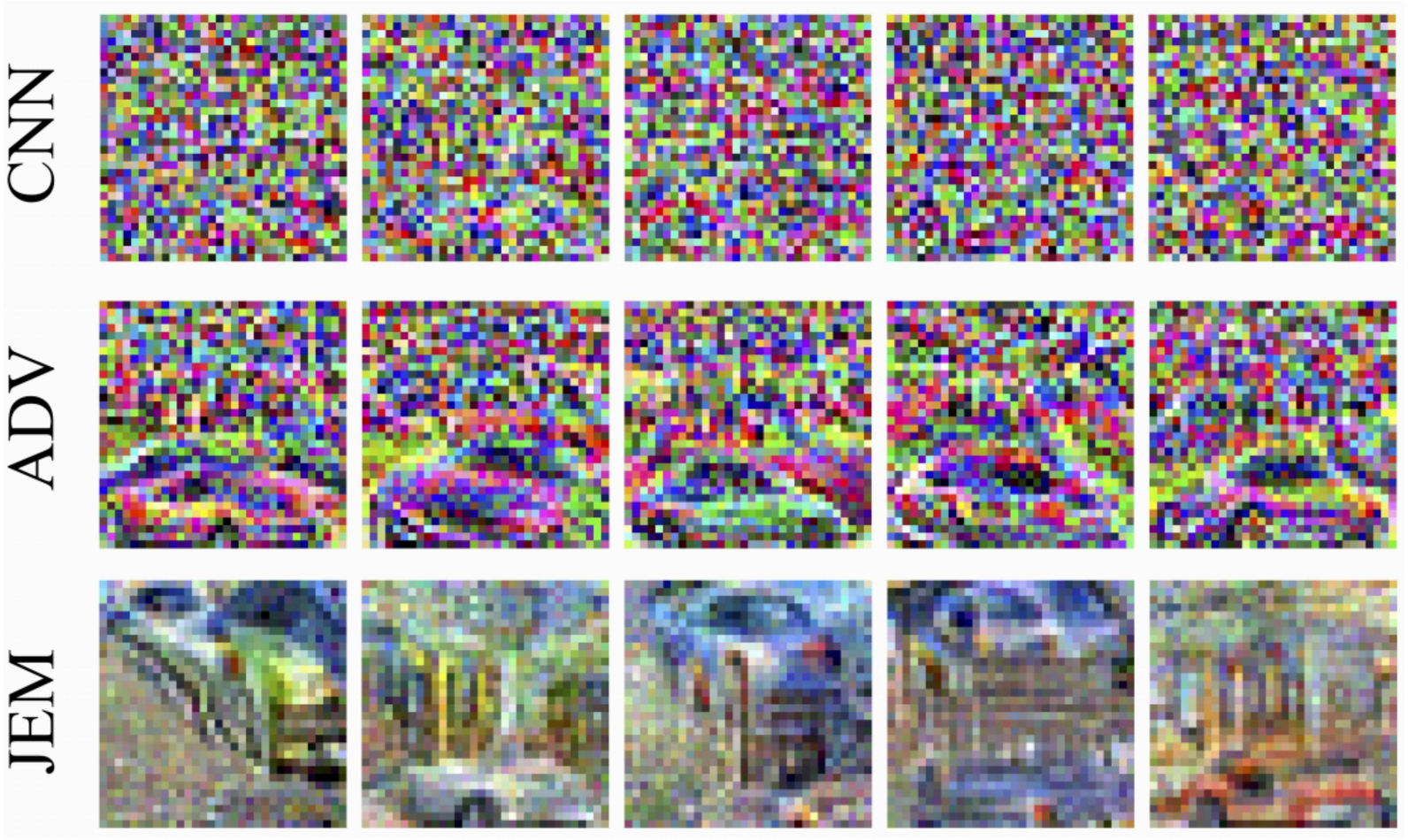
- Might be robust already due to seeing random noise during training
- Can snap an adversarial input example to a higher probability example first, then classify *that*, instead.
  - Run a few iterations of MCMC on the input, then classify.

How can this give you robustness?



(a) L_∞ Robustness

Distal adversaries



$$p(y = \text{"car"} \mid \mathrm{x})$$

What are the advantages to semisupervised training, e.g. $p(\mathbf{x}, y)$?

What are the strengths of VAEs vs. EBMs?

Next class: GANs
Presenters: Nathan Louis, Guanhua Wang, Chia-Ming Wang