Your Classifier is Secretly an Energy Based Model and You Should Treat it Like One[1]

January 30, 2020

[1] Will Grathwohl et al., ICLR 2020

What is an Energy Based Model (EBM)?

What is an Energy Based Model (EBM)?

Any pdf $p_\theta(x)$ can be written down as:

$$p_\theta(\mathbf{x}) = \frac{\exp\left(-E_\theta(\mathbf{x})\right)}{Z_\theta} \tag{1}$$

What is an Energy Based Model (EBM)?

Any pdf $p_\theta(x)$ can be written down as:

$$p_\theta(\mathbf{x}) = \frac{\exp\left(-E_\theta(\mathbf{x})\right)}{Z_\theta} \tag{1}$$

where Z_θ is a normalizing constant:

$$Z_\theta = \int_{\boldsymbol{x}} \exp\left(-E_\theta(\boldsymbol{x})\right) d\boldsymbol{x}$$

What is an Energy Based Model (EBM)?

Any pdf $p_\theta(x)$ can be written down as:

$$p_\theta(\mathbf{x}) = \frac{\exp\left(-E_\theta(\mathbf{x})\right)}{Z_\theta} \tag{1}$$

where Z_θ is a normalizing constant:

$$Z_\theta = \int_{\boldsymbol{x}} \exp\left(-E_\theta(\boldsymbol{x})\right) d\boldsymbol{x}$$

- This parametrization allows us to forget about $p_\theta(x)$ and work with $E_\theta(x)$

What is an Energy Based Model (EBM)?

Any pdf $p_\theta(x)$ can be written down as:

$$p_\theta(\mathbf{x}) = \frac{\exp\left(-E_\theta(\mathbf{x})\right)}{Z_\theta} \tag{1}$$

where Z_θ is a normalizing constant:

$$Z_\theta = \int_{\boldsymbol{x}} \exp\left(-E_\theta(\boldsymbol{x})\right) d\boldsymbol{x}$$

- This parametrization allows us to forget about $p_\theta(x)$ and work with $E_\theta(x)$
- Function $E_\theta(x)$ is an *energy function* (hence the name — EBM)

What is an Energy Based Model (EBM)?

Any pdf $p_\theta(x)$ can be written down as:

$$p_\theta(\mathbf{x}) = \frac{\exp\left(-E_\theta(\mathbf{x})\right)}{Z_\theta} \tag{1}$$

where Z_θ is a normalizing constant:

$$Z_\theta = \int_{\boldsymbol{x}} \exp\left(-E_\theta(\boldsymbol{x})\right) d\boldsymbol{x}$$

- This parametrization allows us to forget about $p_\theta(x)$ and work with $E_\theta(x)$
- Function $E_\theta(x)$ is an *energy function* (hence the name — EBM)
- Energy functions are *easier* to work with: you do not have any restrictions on $E_\theta(x)$ ($p_\theta(x)$ is nonnegative and integrates to 1).

What is an Energy Based Model (EBM)?

Any pdf $p_\theta(x)$ can be written down as:

$$p_\theta(\mathbf{x}) = \frac{\exp\left(-E_\theta(\mathbf{x})\right)}{Z_\theta} \tag{1}$$

where Z_θ is a normalizing constant:

$$Z_\theta = \int_{\boldsymbol{x}} \exp\left(-E_\theta(\boldsymbol{x})\right) d\boldsymbol{x}$$

- This parametrization allows us to forget about $p_\theta(x)$ and work with $E_\theta(x)$
- Function $E_\theta(x)$ is an *energy function* (hence the name — EBM)
- Energy functions are *easier* to work with: you do not have any restrictions on $E_\theta(x)$ ($p_\theta(x)$ is nonnegative and integrates to 1).
- Energy functions are *harder* to work with: Z_θ is hard or impossible to compute, so we can't optimize EBM for θ

How to train EBMs

How to train EBMs

- We perform standard MLE: maximize $\log p_\theta(x_i)$ at our points x_i.

How to train EBMs

- We perform standard MLE: maximize $\log p_\theta(x_i)$ at our points x_i.
- By optimizing $\log p_\theta(x)$ we also optimize $E_\theta(x)$.

How to train EBMs

- We perform standard MLE: maximize $\log p_\theta(x_i)$ at our points x_i.
- By optimizing $\log p_\theta(x)$ we also optimize $E_\theta(x)$.
- We can't compute $\log p_\theta(x)$, but we can (approximately) compute its log-derivative via:

$$\frac{\partial \log p_\theta(\mathbf{x})}{\partial \theta} = \mathbb{E}_{p_\theta(\mathbf{x}')}\left[\frac{\partial E_\theta(\mathbf{x}')}{\partial \theta}\right] - \frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \tag{2}$$

How to train EBMs

- We perform standard MLE: maximize $\log p_\theta(x_i)$ at our points x_i.
- By optimizing $\log p_\theta(x)$ we also optimize $E_\theta(x)$.
- We can't compute $\log p_\theta(x)$, but we can (approximately) compute its log-derivative via:

$$\frac{\partial \log p_\theta(\mathbf{x})}{\partial \theta} = \mathbb{E}_{p_\theta(\mathbf{x}')}\left[\frac{\partial E_\theta\left(\mathbf{x}'\right)}{\partial \theta}\right] - \frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \tag{2}$$

- Proof: follows directly from the definition.

How to train EBMs

- We perform standard MLE: maximize $\log p_\theta(x_i)$ at our points x_i.
- By optimizing $\log p_\theta(x)$ we also optimize $E_\theta(x)$.
- We can't compute $\log p_\theta(x)$, but we can (approximately) compute its log-derivative via:

$$\frac{\partial \log p_\theta(\mathbf{x})}{\partial \theta} = \mathbb{E}_{p_\theta(\mathbf{x}')}\left[\frac{\partial E_\theta(\mathbf{x}')}{\partial \theta}\right] - \frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \tag{2}$$

- Proof: follows directly from the definition.
- But a new problem arises: how to compute expectation?

How to train EBMs

- We perform standard MLE: maximize $\log p_\theta(x_i)$ at our points x_i.
- By optimizing $\log p_\theta(x)$ we also optimize $E_\theta(x)$.
- We can't compute $\log p_\theta(x)$, but we can (approximately) compute its log-derivative via:

$$\frac{\partial \log p_\theta(\mathbf{x})}{\partial \theta} = \mathbb{E}_{p_\theta(\mathbf{x}')}\left[\frac{\partial E_\theta\left(\mathbf{x}'\right)}{\partial \theta}\right] - \frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \tag{2}$$

- Proof: follows directly from the definition.
- But a new problem arises: how to compute expectation?
- Answer: we can't do that directly, so let's use SGLD sampling.

Stochastic Gradient Langevin Dynamics (SGLD)

Motivation:

Stochastic Gradient Langevin Dynamics (SGLD)

Motivation:

- Imagine we have a “magic” function $p_\theta(x)$ that takes an input x and tells us its density.

Stochastic Gradient Langevin Dynamics (SGLD)

Motivation:

- Imagine we have a "magic" function $p_\theta(x)$ that takes an input x and tells us its density.
- Unfortunately, we cannot sample new x-s from it directly, i.e. $p_\theta(x)$ does not support such operations.

Stochastic Gradient Langevin Dynamics (SGLD)

Motivation:

- Imagine we have a "magic" function $p_\theta(x)$ that takes an input x and tells us its density.
- Unfortunately, we cannot sample new x-s from it directly, i.e. $p_\theta(x)$ does not support such operations.
- But we really want to sample from it! What should we do?

Stochastic Gradient Langevin Dynamics (SGLD)

Motivation:

- Imagine we have a "magic" function $p_\theta(x)$ that takes an input x and tells us its density.
- Unfortunately, we cannot sample new x-s from it directly, i.e. $p_\theta(x)$ does not support such operations.
- But we really want to sample from it! What should we do?
- SGLD is a method to sample from such functions.

Stochastic Gradient Langevin Dynamics (SGLD)

Motivation:

- Imagine we have a "magic" function $p_\theta(x)$ that takes an input x and tells us its density.
- Unfortunately, we cannot sample new x-s from it directly, i.e. $p_\theta(x)$ does not support such operations.
- But we really want to sample from it! What should we do?
- SGLD is a method to sample from such functions.
- What is cool about SGLD is that it allows to use *unnormalized* density $\tilde{p}(x)$ (a very important property in our case!).

Stochastic Gradient Langevin Dynamics (SGLD)

SGLD in a (very) few words:

Stochastic Gradient Langevin Dynamics (SGLD)

SGLD in a (very) few words:

1. Start from any random point x_0.

Stochastic Gradient Langevin Dynamics (SGLD)

SGLD in a (very) few words:

1. Start from any random point x_0.
2. Compute the gradient of $\nabla \log p_\theta(x_0)$.

Stochastic Gradient Langevin Dynamics (SGLD)

SGLD in a (very) few words:

1. Start from any random point x_0.
2. Compute the gradient of $\nabla \log p_\theta(x_0)$.
3. Perform a small step in that direction.

Stochastic Gradient Langevin Dynamics (SGLD)

SGLD in a (very) few words:

1. Start from any random point x_0.
2. Compute the gradient of $\nabla \log p_\theta(x_0)$.
3. Perform a small step in that direction.
4. Add a little bit of noise.

Stochastic Gradient Langevin Dynamics (SGLD)

SGLD in a (very) few words:

1. Start from any random point x_0.
2. Compute the gradient of $\nabla \log p_\theta(x_0)$.
3. Perform a small step in that direction.
4. Add a little bit of noise.
5. Obtain x_{t+1}. Accept it if:

$$\frac{p\left(x_t|x_{t+1}\right)p\left(x_t\right)}{p\left(x_{t+1}|x_t\right)p\left(x_{t+1}\right)} < u, u \sim \mathcal{U}[0,1] \tag{3}$$

Stochastic Gradient Langevin Dynamics (SGLD)

SGLD in a (very) few words:

1. Start from any random point x_0.
2. Compute the gradient of $\nabla \log p_\theta(x_0)$.
3. Perform a small step in that direction.
4. Add a little bit of noise.
5. Obtain x_{t+1}. Accept it if:

$$\frac{p\left(x_t|x_{t+1}\right)p\left(x_t\right)}{p\left(x_{t+1}|x_t\right)p\left(x_{t+1}\right)} < u, u \sim \mathcal{U}[0,1] \tag{3}$$

In other words, run the following recurrence a lot of times:

$$\mathbf{x}_0 \sim p_0(\mathbf{x}), \quad \mathbf{x}_{i+1} = \mathbf{x}_i - \frac{\alpha}{2}\frac{\partial E_\theta\left(\mathbf{x}_i\right)}{\partial \mathbf{x}_i} + \epsilon, \quad \epsilon \sim \mathcal{N}(0,\alpha) \tag{4}$$

Stochastic Gradient Langevin Dynamics (SGLD)

SGLD in a (very) few words:

1. Start from any random point x_0.
2. Compute the gradient of $\nabla \log p_\theta(x_0)$.
3. Perform a small step in that direction.
4. Add a little bit of noise.
5. Obtain x_{t+1}. Accept it if:

$$\frac{p\left(x_t \mid x_{t+1}\right) p\left(x_t\right)}{p\left(x_{t+1} \mid x_t\right) p\left(x_{t+1}\right)} < u, u \sim \mathcal{U}[0,1] \tag{3}$$

In other words, run the following recurrence a lot of times:

$$\mathbf{x}_0 \sim p_0(\mathbf{x}), \quad \mathbf{x}_{i+1} = \mathbf{x}_i - \frac{\alpha}{2} \frac{\partial E_\theta\left(\mathbf{x}_i\right)}{\partial \mathbf{x}_i} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \alpha) \tag{4}$$

Justification: if your model and your choice of α are “good” (in some sense), then after infinite amount of steps your samples will be “good” (in some sense).

Ok, how to treat a classifier as an EBM?

Ok, how to treat a classifier as an EBM?

- Remember that *any* function can be used as $E_\theta(x)$

Ok, how to treat a classifier as an EBM?

- Remember that *any* function can be used as $E_\theta(x)$
- Let $f_\theta(x)$ be our neural network.

Ok, how to treat a classifier as an EBM?

- Remember that *any* function can be used as $E_\theta(x)$
- Let $f_\theta(x)$ be our neural network.
- Let $f_\theta(x)[y]$ be the y-th logit (i.e. logit value for class y).

Ok, how to treat a classifier as an EBM?

- Remember that *any* function can be used as $E_\theta(x)$
- Let $f_\theta(x)$ be our neural network.
- Let $f_\theta(x)[y]$ be the y-th logit (i.e. logit value for class y).
- Let our examples be $s = (x, y)$

Ok, how to treat a classifier as an EBM?

- Remember that *any* function can be used as $E_\theta(x)$
- Let $f_\theta(x)$ be our neural network.
- Let $f_\theta(x)[y]$ be the y-th logit (i.e. logit value for class y).
- Let our examples be $s = (x, y)$
- We define energy $E_\theta(s) = E_\theta(x, y) = -f_\theta(x)[y]$.

Ok, how to treat a classifier as an EBM?

- Remember that *any* function can be used as $E_\theta(x)$
- Let $f_\theta(x)$ be our neural network.
- Let $f_\theta(x)[y]$ be the y-th logit (i.e. logit value for class y).
- Let our examples be $s = (x, y)$
- We define energy $E_\theta(s) = E_\theta(x, y) = -f_\theta(x)[y]$.
- Using $E_\theta(x, y)$ we can define $p_\theta(x)$:

$$p_\theta(\mathbf{x}) = \sum_y p_\theta(\mathbf{x}, y) = \frac{1}{Z_\theta} \sum_y \exp\left(f_\theta(\mathbf{x})[y]\right) \tag{5}$$

Ok, how to treat a classifier as an EBM?

- Remember that *any* function can be used as $E_\theta(x)$
- Let $f_\theta(x)$ be our neural network.
- Let $f_\theta(x)[y]$ be the y-th logit (i.e. logit value for class y).
- Let our examples be $s = (x, y)$
- We define energy $E_\theta(s) = E_\theta(x, y) = -f_\theta(x)[y]$.
- Using $E_\theta(x, y)$ we can define $p_\theta(x)$:

$$p_\theta(\mathbf{x}) = \sum_y p_\theta(\mathbf{x}, y) = \frac{1}{Z_\theta} \sum_y \exp\left(f_\theta(\mathbf{x})[y]\right) \quad (5)$$

- Using $p_\theta(x)$ we can define $E_\theta(x)$:

$$E_\theta(\mathbf{x}) = -\log \sum_y \exp\left(f_\theta(\mathbf{x})[y]\right) \quad (6)$$

Ok, how to treat a classifier as an EBM?

- Remember that *any* function can be used as $E_\theta(x)$
- Let $f_\theta(x)$ be our neural network.
- Let $f_\theta(x)[y]$ be the y-th logit (i.e. logit value for class y).
- Let our examples be $s = (x, y)$
- We define energy $E_\theta(s) = E_\theta(x, y) = -f_\theta(x)[y]$.
- Using $E_\theta(x, y)$ we can define $p_\theta(x)$:

$$p_\theta(\mathbf{x}) = \sum_y p_\theta(\mathbf{x}, y) = \frac{1}{Z_\theta} \sum_y \exp\left(f_\theta(\mathbf{x})[y]\right) \tag{5}$$

- Using $p_\theta(x)$ we can define $E_\theta(x)$:

$$E_\theta(\mathbf{x}) = -\log \sum_y \exp\left(f_\theta(\mathbf{x})[y]\right) \tag{6}$$

- Higher the logits — lower the energy of x.

Ok, how to treat a classifier as an EBM?

- Remember that *any* function can be used as $E_\theta(x)$
- Let $f_\theta(x)$ be our neural network.
- Let $f_\theta(x)[y]$ be the y-th logit (i.e. logit value for class y).
- Let our examples be $s = (x, y)$
- We define energy $E_\theta(s) = E_\theta(x, y) = -f_\theta(x)[y]$.
- Using $E_\theta(x, y)$ we can define $p_\theta(x)$:

$$p_\theta(\mathbf{x}) = \sum_y p_\theta(\mathbf{x}, y) = \frac{1}{Z_\theta} \sum_y \exp\left(f_\theta(\mathbf{x})[y]\right) \tag{5}$$

- Using $p_\theta(x)$ we can define $E_\theta(x)$:

$$E_\theta(\mathbf{x}) = -\log \sum_y \exp\left(f_\theta(\mathbf{x})[y]\right) \tag{6}$$

- Higher the logits — lower the energy of x.
- Our final loss looks like:

$$\mathcal{L}(x) = \log p(x, y) = \underbrace{\log p(y|x)}_{\text{classification loss}} + \underbrace{\log p(x)}_{\text{generative loss}} \tag{7}$$

Is JEM using adversarial training?!

Is JEM using adversarial training?!

- Let's take a closer look to the gradient of the part of the loss:

$$\frac{\partial \log p_\theta(\mathbf{x})}{\partial \theta} = \mathbb{E}_{p_\theta(\mathbf{x}')}\left[\frac{\partial E_\theta(\mathbf{x}')}{\partial \theta}\right] - \frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \tag{8}$$

Is JEM using adversarial training?!

- Let's take a closer look to the gradient of the part of the loss:

$$\frac{\partial \log p_\theta(\mathbf{x})}{\partial \theta} = \mathbb{E}_{p_\theta(\mathbf{x}')}\left[\frac{\partial E_\theta\left(\mathbf{x}'\right)}{\partial \theta}\right] - \frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \tag{8}$$

- If we use a single sample to approximate expectation and compute the gradient across the dataset our gradient we'll look like:

$$\frac{\partial \log p_\theta(\mathbf{X})}{\partial \theta} \approx \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\partial E_\theta\left(\mathbf{x}^{\text{fake}}\right)}{\partial \theta} - \frac{\partial E_\theta(\mathbf{x}^{\text{real}})}{\partial \theta}\right) \tag{9}$$

Is JEM using adversarial training?!

- Let's take a closer look to the gradient of the part of the loss:

$$\frac{\partial \log p_\theta(\mathbf{x})}{\partial \theta} = \mathbb{E}_{p_\theta(\mathbf{x}')}\left[\frac{\partial E_\theta\left(\mathbf{x}'\right)}{\partial \theta}\right] - \frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \tag{8}$$

- If we use a single sample to approximate expectation and compute the gradient across the dataset our gradient we'll look like:

$$\frac{\partial \log p_\theta(\mathbf{X})}{\partial \theta} \approx \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\partial E_\theta\left(\mathbf{x}^{\text{fake}}\right)}{\partial \theta} - \frac{\partial E_\theta(\mathbf{x}^{\text{real}})}{\partial \theta}\right) \tag{9}$$

- And this gradient looks very similar to the gradient of the following loss:

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{p_\theta(x)}\left[E_\theta(x)\right] - \mathop{\mathbb{E}}_{p_{\text{data}}(x)}\left[E_\theta(x)\right] \tag{10}$$

Is JEM using adversarial training?!

- Let's take a closer look to the gradient of the part of the loss:

$$\frac{\partial \log p_\theta(\mathbf{x})}{\partial \theta} = \mathbb{E}_{p_\theta(\mathbf{x}')}\left[\frac{\partial E_\theta\left(\mathbf{x}'\right)}{\partial \theta}\right] - \frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \tag{8}$$

- If we use a single sample to approximate expectation and compute the gradient across the dataset our gradient we'll look like:

$$\frac{\partial \log p_\theta(\mathbf{X})}{\partial \theta} \approx \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\partial E_\theta\left(\mathbf{x}^{\text{fake}}\right)}{\partial \theta} - \frac{\partial E_\theta(\mathbf{x}^{\text{real}})}{\partial \theta}\right) \tag{9}$$

- And this gradient looks very similar to the gradient of the following loss:

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{p_\theta(x)}[E_\theta(x)] - \mathop{\mathbb{E}}_{p_{\text{data}}(x)}[E_\theta(x)] \tag{10}$$

- Which is a discriminator loss!

Is JEM using adversarial training?!

- Let's take a closer look to the gradient of the part of the loss:

$$\frac{\partial \log p_\theta(\mathbf{x})}{\partial \theta} = \mathbb{E}_{p_\theta(\mathbf{x}')}\left[\frac{\partial E_\theta\left(\mathbf{x}'\right)}{\partial \theta}\right] - \frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \tag{8}$$

- If we use a single sample to approximate expectation and compute the gradient across the dataset our gradient we'll look like:

$$\frac{\partial \log p_\theta(\mathbf{X})}{\partial \theta} \approx \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\partial E_\theta\left(\mathbf{x}^{\text{fake}}\right)}{\partial \theta} - \frac{\partial E_\theta(\mathbf{x}^{\text{real}})}{\partial \theta}\right) \tag{9}$$

- And this gradient looks very similar to the gradient of the following loss:

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{p_\theta(x)}\left[E_\theta(x)\right] - \mathop{\mathbb{E}}_{p_{\text{data}}(x)}\left[E_\theta(x)\right] \tag{10}$$

- Which is a discriminator loss!
- Reformulation of GANs via EBMs is not new.

Is JEM using adversarial training?!

- Let's take a closer look to the gradient of the part of the loss:

$$\frac{\partial \log p_\theta(\mathbf{x})}{\partial \theta} = \mathbb{E}_{p_\theta(\mathbf{x}')}\left[\frac{\partial E_\theta\left(\mathbf{x}'\right)}{\partial \theta}\right] - \frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \tag{8}$$

- If we use a single sample to approximate expectation and compute the gradient across the dataset our gradient we'll look like:

$$\frac{\partial \log p_\theta(\mathbf{X})}{\partial \theta} \approx \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\partial E_\theta\left(\mathbf{x}^{\text{fake}}\right)}{\partial \theta} - \frac{\partial E_\theta(\mathbf{x}^{\text{real}})}{\partial \theta}\right) \tag{9}$$

- And this gradient looks very similar to the gradient of the following loss:

$$\mathcal{L}(\theta) = \underset{p_\theta(x)}{\mathbb{E}}\left[E_\theta(x)\right] - \underset{p_{\text{data}}(x)}{\mathbb{E}}\left[E_\theta(x)\right] \tag{10}$$

- Which is a discriminator loss!
- Reformulation of GANs via EBMs is not new.
- The key novelty of the paper is that we can do that "inside" the classifier.

Results and discussion

Results and discussion

- Disclaimer: results are really good.

Results and discussion

- Disclaimer: results are really good.
- Authors built a model on SVHN/CIFAR10/CIFAR100 which:

Results and discussion

- Disclaimer: results are really good.
- Authors built a model on SVHN/CIFAR10/CIFAR100 which:
  - has good classification accuracy

Results and discussion

- Disclaimer: results are really good.
- Authors built a model on SVHN/CIFAR10/CIFAR100 which:
  - has good classification accuracy
  - has good IS/FID scores (i.e. has good samples)

Results and discussion

- Disclaimer: results are really good.
- Authors built a model on SVHN/CIFAR10/CIFAR100 which:
  - has good classification accuracy
  - has good IS/FID scores (i.e. has good samples)
  - has much better calibrated predictions (i.e. is not overconfident unlike traditional classifiers)

Results and discussion

- Disclaimer: results are really good.
- Authors built a model on SVHN/CIFAR10/CIFAR100 which:
  - has good classification accuracy
  - has good IS/FID scores (i.e. has good samples)
  - has much better calibrated predictions (i.e. is not overconfident unlike traditional classifiers)
  - has good out-of-distribution predictions (i.e. detecting when we feed OOD-samples into it)

Results and discussion

- Disclaimer: results are really good.
- Authors built a model on SVHN/CIFAR10/CIFAR100 which:
  - has good classification accuracy
  - has good IS/FID scores (i.e. has good samples)
  - has much better calibrated predictions (i.e. is not overconfident unlike traditional classifiers)
  - has good out-of-distribution predictions (i.e. detecting when we feed OOD-samples into it)
  - is robust to adversarial attacks

Results and discussion

- Disclaimer: results are really good.
- Authors built a model on SVHN/CIFAR10/CIFAR100 which:
  - has good classification accuracy
  - has good IS/FID scores (i.e. has good samples)
  - has much better calibrated predictions (i.e. is not overconfident unlike traditional classifiers)
  - has good out-of-distribution predictions (i.e. detecting when we feed OOD-samples into it)
  - is robust to adversarial attacks
  - ...and all of these at the same time!

Results and discussion

- Disclaimer: results are really good.
- Authors built a model on SVHN/CIFAR10/CIFAR100 which:
  - has good classification accuracy
  - has good IS/FID scores (i.e. has good samples)
  - has much better calibrated predictions (i.e. is not overconfident unlike traditional classifiers)
  - has good out-of-distribution predictions (i.e. detecting when we feed OOD-samples into it)
  - is robust to adversarial attacks
  - ...and all of these at the same time!
- A huge limitation: SGLD makes both the training unstable and the model hard to scale.

Results and discussion

- Disclaimer: results are really good.
- Authors built a model on SVHN/CIFAR10/CIFAR100 which:
  - has good classification accuracy
  - has good IS/FID scores (i.e. has good samples)
  - has much better calibrated predictions (i.e. is not overconfident unlike traditional classifiers)
  - has good out-of-distribution predictions (i.e. detecting when we feed OOD-samples into it)
  - is robust to adversarial attacks
  - ...and all of these at the same time!
- A huge limitation: SGLD makes both the training unstable and the model hard to scale.
- An approach was coined Joint Energy based Model (JEM) since we model (x, y) simultaneously.