JEM++: Improved Techniques for Training JEM

Xiulong Yang, Shihao Ji
Department of Computer Science
Georgia State University
{xyang22,sji}@gsu.edu

Abstract

Joint Energy-based Model (JEM) [12] is a recently proposed hybrid model that retains strong discriminative power of modern CNN classifiers, while generating samples rivaling the quality of GAN-based approaches. In this paper, we propose a variety of new training procedures and architecture features to improve JEM's accuracy, training stability, and speed altogether. 1) We propose a proximal SGLD to generate samples in the proximity of samples from previous step, which improves the stability. 2) We further treat the approximate maximum likelihood learning of EBM as a multi-step differential game, and extend the YOPO framework [47] to cut out redundant calculations during backpropagation, which accelerates the training substantially. 3) Rather than initializing SGLD chain from random noise, we introduce a new informative initialization that samples from a distribution estimated from training data. 4) This informative initialization allows us to enable batch normalization in JEM, which further releases the power of modern CNN architectures for hybrid modeling.[1]

1. Introduction

Deep neural networks (DNNs) have made significant breakthroughs in various discriminative tasks and generative tasks, including image classification, object detection, and high-quality image and text generation [26, 18, 3, 5]. However, prior works on discriminative models and generative models are largely separated. Even though a few researches (e.g., [6, 8]) have shown that generative training is beneficial to discriminative models, most recent works on generative models focus primarily on qualitative sample quality [4, 37, 40], and the discriminative performances of state-of-the-art generative models are still far behind discriminative ones [2, 7, 9].

Among different discriminative and generative models, energy-based models (EBMs) [27] are an appealing class of probabilistic models, which can be viewed as hybrid models with both discriminative and generative powers [12]. Compared to the popular generative models, such as VAE [25] and GAN [11], which train explicit functions to generate samples, EBMs only need to train a single network with a set of shared features for discriminative tasks and generative tasks, and exploit implicit sampling for generation. Since an EBM is the only object that needs to be trained, it generally achieves a higher simplicity and stability than approaches that use multiple networks. Hence, there is a great interest recently in encompassing the generative capabilities into discriminative models without sacrificing their discriminative powers. Specifically, a series of recent works propose to train a CNN as an EBM for image classification and generation [45, 16, 9, 12]. Among them, JEM [12] is one of the most representative ones, which reinterprets the modern CNN classifier (e.g., Wide-ResNet [46]) as an EBM for image generation and achieves impressive performances in image classification and image generation simultaneously. JEM demonstrates the potential of EBMs in hybrid modeling and ignites a series of follow-up works [48, 13, 10, 14].

However, training EBMs is still a challenging task. As shown in Table 1, existing methods demonstrate a great deal of tradeoffs among different algorithmic features in the quest of improved training algorithms. Most of the works [32, 9, 12] adopt the SGLD sampling [42] to train EBMs, where K sweeps of forward and backward propagations are required in each sampling step. These training methods can be prolonged with a large K, preventing them from long training procedures required by large-scale datasets. In addition, SGLD can be precarious and easily diverged, which further hinders the prevalence of EBMs. To avoid the long sampling process of SGLD, recent works introduce auxiliary models [17, 44, 14] or use special architectures [13, 41] to amortize the SGLD sampling or improve its stability. Given the architectural simplicity of the SGLD-based methods, especially JEM [12], we ask the following question: Is it possible to develop new training methods of JEM to reduce the number of sampling steps required by SGLD while improving its training stability?

[1] Code: https://github.com/sndnyang/JEMPP

Table 1. Characteristics of different EBM training methods.

Training Method	Fast	Stable	High dim	No aux. model	Unrestricted arch	Approx. likelihood
SGLD-based [32, 9, 12]	✗	✗	✓	✓	✓	✓
Score Matching [41, 21]	✓	✗	✓	✓	✗	✗
Noise Contrastive [10, 15]	✓	✓	✗	✗	✓	✗
Regularized Generator [17, 14]	✓	✓	✓	✗	✓	✓
JEM++ (ours)	↑	↑	✓	✓	✓	✓

In this paper, we introduce a variety of training procedures and architecture features to improve JEM's accuracy, training stability, and speed altogether. After a thorough investigation on JEM, we find that JEM sometimes generates abnormal images containing pixels with extreme values beyond a reasonable range. This motivates us to constrain the SGLD sampling by projecting samples to an L_p-norm ball of previous samples. Secondly, JEM does not support modern architecture features such as batch norm [22][2]. We find that a huge statistic gap between the initial noisy samples of SGLD and real data incurs the training difficulty of JEM when batch norm is enabled. Hence, we introduce a new informative initialization that closes the gap between initial samples and real data. Moreover, we find that batch-norm-enabled JEM supports a larger learning rate, which further increases the convergence rate of JEM. Finally, we extend YOPO [47], a general framework for PGD [28] acceleration, to the maximum likelihood learning of EBM and speed up the training of JEM even further. Our main contributions are summarized as follows:

1. We propose a proximal SGLD to generate samples in the proximity of samples from previous step, which improves the stability of JEM.
2. We further treat the approximate maximum likelihood learning of EBM as a multi-step differential game, which can be accelerated by cutting out redundant calculations during backpropagation, while retaining the overall predictive performance.
3. We introduce a new informative initialization to initialize the SGLD chain, which stabilizes the training further and accelerates the convergence rate of SGLD sampling.
4. This new informative initialization also enables batch norm to train JEM successfully and release the power of modern CNN architectures. What's more, with the informative initialization and batch norm, JEM++ can be optimized with a large learning rate, while JEM fails to.
5. JEM++ matches or outperforms prior state-of-the-art hybrid models on discriminative and generative tasks, while enjoying improved stability and training speed over the original JEM.

[2]Although the authors stated they have been able to train JEM with batch norm successfully, no details are disclosed in their paper or code.

2. Energy-Based Models

Energy-based models (EBMs) [27] define an energy function that assigns low energy values to samples drawn from data distribution and high values otherwise, such that any probability density $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ can be expressed via a Boltzmann distribution as

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{\exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x})\right)}{Z(\boldsymbol{\theta})}, \tag{1}$$

where $E_{\boldsymbol{\theta}}(\boldsymbol{x})$ is an energy function that maps each input $\boldsymbol{x} \in \mathcal{X}$ to a scalar, and $Z(\boldsymbol{\theta})$ is the normalizing constant (also known as the partition function) such that $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is a valid density function.

The key challenge of training EBMs lies in estimating the partition function $Z(\boldsymbol{\theta})$, which is notoriously intractable. The standard maximum likelihood estimation of parameters $\boldsymbol{\theta}$ is not straightforward either, and a number of sampling-based approaches have been proposed to approximate it effectively. Specifically, the derivative of the log-likelihood of a single sample $\boldsymbol{x} \in \mathcal{X}$ w.r.t. $\boldsymbol{\theta}$ can be expressed as

$$\frac{\partial \log p_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}} = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{x}')} \frac{\partial E_{\boldsymbol{\theta}}(\boldsymbol{x}')}{\partial \boldsymbol{\theta}} - \frac{\partial E_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}}, \tag{2}$$

where the expectation is over the density function $p_{\boldsymbol{\theta}}(\boldsymbol{x}')$, sampling from which is challenging due to the intractable $Z(\boldsymbol{\theta})$. Therefore, MCMC and Gibbs sampling [20] have been proposed previously to estimate the expectation efficiently. To speed up the mixing for effective sampling, recently Stochastic Gradient Langevin Dynamics (SGLD) [42] has been employed to train EBMs by using the gradient information [32, 9, 12]. Specifically, to sample from $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, SGLD follows

$$\begin{aligned} &\boldsymbol{x}^0 \sim p_0(\boldsymbol{x}), \\ &\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \frac{\alpha}{2}\frac{\partial E_{\boldsymbol{\theta}}(\boldsymbol{x}^t)}{\partial \boldsymbol{x}^t} + \alpha\epsilon^t, \quad \epsilon^t \sim \mathcal{N}(0, 1), \end{aligned} \tag{3}$$

where $p_0(\boldsymbol{x})$ is typically a uniform distribution over $[-1, 1]$, whose samples are refined via a noisy gradient decent with step-size α over a SGLD chain.

Prior works [9, 12, 31] have investigated the effect of hyper-parameters in SGLD sampling in terms of stability

and speed, and showed that the SGLD-based approaches suffer from poor stability and computational challenges from sequential sampling at every iteration. Specifically, Nijkamp et al. [31] find that the noise term in SGLD is not important, and including a noise of low variance appears to improve synthesis quality. What's more, for unnormalized densities, it's desirable to generate samples from SGLD chain after it converges. This requires the step-size α to decay with a polynomial schedule and an infinite number of sampling steps, which is not realistic in practical applications. Instead, JEM [12] uses a constant step-size α during sampling and approximates the samples with a sampler that runs only for a finite number of steps. To improve the sampling stability, the model would require to quadruple the number of SGLD steps, which greatly increases the run-time.

3. JEM++: The Improved Training of JEM

We first give a brief introduction of JEM [12] and then discuss a variety of new training procedures to improve its accuracy, stability and speed.

Joint Energy-based Models (JEM) [12] reinterprets modern CNN classifiers as EBMs. Considering a CNN classifier of parameters $\boldsymbol{\theta}$, given an input $\boldsymbol{x}$ the classifier first maps the input to a vector of C real-valued numbers (or logits): $f_{\boldsymbol{\theta}}(\boldsymbol{x})[y], \forall y \in [1, \cdots, C]$, where C is the number of classes; the logits are then normalized via the softmax function to yield a probability vector: $p_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = e^{f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]} / \sum_{y'} e^{f_{\boldsymbol{\theta}}(\boldsymbol{x})[y']}$. Interestingly, the same vector of logits $f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]$ can also be used to define an EBM for the joint density: $p_{\boldsymbol{\theta}}(\boldsymbol{x}, y) = e^{f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]}/Z(\boldsymbol{\theta})$, where $Z(\boldsymbol{\theta})$ is an unknown normalizing constant (regardless of $\boldsymbol{x}$ or y). Then a marginal density of $\boldsymbol{x}$ can be achieved by marginalizing the joint density as: $p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_y p_{\boldsymbol{\theta}}(\boldsymbol{x}, y) = \sum_y e^{f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]}/Z(\boldsymbol{\theta})$. Comparing this density with Eq. 1, it is readily to show that the corresponding energy function of $\boldsymbol{x}$ is defined as

$$E_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\log \sum_y e^{f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]} = -\text{LSE}(f_{\boldsymbol{\theta}}(\boldsymbol{x})), \tag{4}$$

where $\text{LSE}(\cdot)$ denotes the Log-Sum-Exp function.

To optimize the model parameter $\boldsymbol{\theta}$, JEM proposes to maximize the joint density function $p_{\boldsymbol{\theta}}(\boldsymbol{x}, y)$, which can be factorized as:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, y) = \log p_{\boldsymbol{\theta}}(y|\boldsymbol{x}) + \log p_{\boldsymbol{\theta}}(\boldsymbol{x}), \tag{5}$$

where the first term is the conventional cross-entropy objective for classification, and the second term can be optimized by the maximum likelihood learning of EBM as shown in Eq. 2 with the SGLD sampling defined in (3). In the paper, we follow the same objective function of JEM and focus on how to improve the stability of SGLD sampling as well as accelerate the maximum likelihood learning of EBM.

3.1. Training EBM as a Minimax Optimization

In practice, when we employ the maximum likelihood estimate of model parameters $\boldsymbol{\theta}$ with Eq. 2, a minibatch of B samples $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_B\} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})$ and a minibatch of B real data samples $\{\boldsymbol{x}_1^r, \boldsymbol{x}_2^r, \cdots, \boldsymbol{x}_B^r\} \sim \mathcal{X}$ are used. To avoid notational clutter, we assume $B = 1$ in the rest of the paper, but the results are readily extended to $B > 1$.

Similar to Nijkamp et al. [31] who have found the insignificance of noise term in the SGLD sampling (3), our empirical study also confirms this observation. Thus, we ignore the noise term in Eq. 3 and treat it as an artifact that generates some stochasticity in the sampling process to facilitate the optimization. Under this assumption, the SGLD sampling (3) can be reinterpreted approximately as an SGD iteration, with a learning rate of $\alpha/2$, initialized from a random sample of $p_0(\boldsymbol{x})$. Assume the convergence can be achieved, the objective of the SGLD sampling (3) is to solve the following optimization problem approximately [3]

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\text{argmin}}\, E_{\boldsymbol{\theta}}(\boldsymbol{x}). \tag{6}$$

Therefore, the maximum likelihood learning of EBM with Eq. 2 is to approximately solve the following minimax game

$$\max_{\boldsymbol{\theta}} \left[\min_{\boldsymbol{x}} E_{\boldsymbol{\theta}}(\boldsymbol{x}) - E_{\boldsymbol{\theta}}(\boldsymbol{x}^r) \right]. \tag{7}$$

To have a robust convergence behavior, we can solve the inner minimization problem of (7) by using the Proximal Point Method [33]. We can further treat the minimax optimization problem (7) as a multi-step differential game and extend YOPO [47], a general framework of accelerating PGD, to speed up the training of EBM. Next, we describe these new training procedures in details.

3.2. Proximal SGLD

Prior works on EBMs reveal the tradeoff between training stability and computational time of SGLD-based approaches [32, 9, 12]. However, the cause of instability of SGLD-based EBMs is still under investigation. Empirically, we observe that upon the divergence of EBM, SGLD generates abnormal samples with extreme values that have a severe negative impact on model parameter update. Hence, we introduce our first improvement to stabilize the inner minimization problem with a proximal SGLD.

Proximal point methods are widely used in optimization [35, 33]. To solve the inner minimization problem of (7), the algorithm generates a sequence $\{\boldsymbol{x}^t\}_{t=1,2,\cdots}$ by the following proximal point iteration:

$$\boldsymbol{x}^{t+1} = \underset{\boldsymbol{x}}{\text{argmin}}\, E_{\boldsymbol{\theta}}(\boldsymbol{x}) \; s.t. \; ||\boldsymbol{x} - \boldsymbol{x}^t||_p < \varepsilon, \tag{8}$$

[3] The entire pipeline is still a stochastic sampler since samples are generated by running finite-length stochastic gradient decent with random initialization.

which solves a constrained minimization problem at each iteration t, i.e., the current solution should be in the proximity of previous one, measured by an L_p norm. Compared with the standard SGD iteration, the proximal point iteration has a robust convergence behavior. Moreover, even if the proximal operator defined in Eq. 8 is not exactly minimized in each iteration, it still has a stronger convergence guarantee than standard SGD, giving rise to the inexact proximal point method [35]. Thus, if we solve each minimization problem (8) inexactly with one step of SGD, we obtain an inexact proximal point iteration

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \frac{\alpha}{2} L_p(\nabla_{\boldsymbol{x}} E_{\boldsymbol{\theta}}(\boldsymbol{x}^t), \varepsilon), \tag{9}$$

where $L_p(\cdot, \varepsilon)$ projects the gradient to an L_p-norm ball of a radius ε. Empirically, we find the L_∞-norm works well across different architectures and datasets. Hence, we only consider the L_∞-norm in the rest of the paper. With an L_∞-norm, Eq. 9 can be rewritten as

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \frac{\alpha}{2} \text{clamp}(\nabla_{\boldsymbol{x}} E_{\boldsymbol{\theta}}(\boldsymbol{x}^t), \varepsilon) + \alpha \epsilon^t, \tag{10}$$

where the clamp$(\cdot, \varepsilon)$ operator clamps the gradient in the range of $[-\varepsilon, \varepsilon]$. Note that to incorporate stochasticity into the inexact proximal point iteration, we add the noise term back to Eq. 10, which resembles the original SGLD sampling (3) but with a gradient clamping operator used to enforce the proximity constraint.

3.3. Training EBM as a Differential Game

As discussed in Section 3.1, the maximum likelihood learning of EBM (7) solves a minimax game approximately. This objective has a close relationship to adversarial training with the PGD attack [28]. Hence, we can extend methods for accelerating adversarial training to EBM and reduce the computational complexity of multi-step SGLD.

Inspired by Pontryagin's Maximum Principle [34], a general framework in optimal control, Zhang et al. [47] propose an optimization method called YOPO (**Y**ou-**P**ropogate **O**nly **O**nce) to accelerate multi-step adversarial training such as PGD. The key factor in YOPO is that the adversarial perturbation is only coupled with the first layer's weights in a neural network. Then YOPO can decouple the adversary update from training of network parameters, and reduce the total number of full forward and backward propagations to only one in each group of adversary updates.

Similarly, we can extend YOPO to the maximum likelihood learning of EBM because the objective (7) can also be treated as a multi-step differential game and the sampled image $\boldsymbol{x}$ from proximal SGLD (10) is only coupled with the first layer's weights. By inserting the energy function (4) into (7), we can rewrite the minimax objective as:

$$\max_{\boldsymbol{\theta}} \left[\min_{\boldsymbol{x}} -\text{LSE}\left(g_{\tilde{\boldsymbol{\theta}}}\left(f_0\left(\boldsymbol{x}, \boldsymbol{\theta}_0\right)\right)\right) - E_{\boldsymbol{\theta}}(x^r) \right] \tag{11}$$

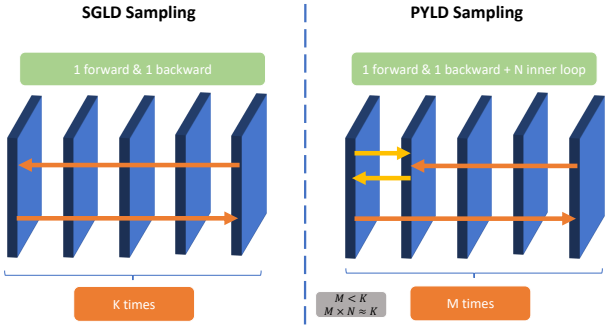


Figure 1. Comparison between SGLD-K sampling and PYLD-M-N sampling.

where f_0 denotes the first layer of a CNN-based EBM, $g_{\tilde{\boldsymbol{\theta}}} = f_{T-1}^{\boldsymbol{\theta}_{T-1}} \circ f_{T-2}^{\boldsymbol{\theta}_{T-2}} \circ \cdots f_1^{\boldsymbol{\theta}_1}$ denotes the network without the first layer, such that $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = g_{\tilde{\boldsymbol{\theta}}}\left(f_0\left(\boldsymbol{x}, \boldsymbol{\theta}_0\right)\right)$. Given a sample $\boldsymbol{x}$, the gradient of energy function (4) can be calculated by chain rule as:

$$\begin{aligned} \frac{\partial E_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{x}} = & -\nabla_{g_{\tilde{\boldsymbol{\theta}}}} \text{LSE}\left(g_{\tilde{\boldsymbol{\theta}}}\left(f_0\left(\boldsymbol{x}, \boldsymbol{\theta}_0\right)\right)\right) \\ & \cdot \nabla_{f_0} g_{\tilde{\boldsymbol{\theta}}}(f_0(\boldsymbol{x}, \boldsymbol{\theta}_0)) \cdot \nabla_{\boldsymbol{x}} f_0(\boldsymbol{x}, \boldsymbol{\theta}_0). \end{aligned} \tag{12}$$

Proximal SGLD (10) conducts K sweeps of full forward and backward propagations for each update of $\boldsymbol{\theta}$. To stabilize the training of EBM, it requires a large K, which greatly increases the run-time. To reduce the total number of thorough forward and backward propagations, we follow YOPO and introduce a slack variable:

$$p = -\nabla_{g_{\tilde{\boldsymbol{\theta}}}} \text{LSE}\left(g_{\tilde{\boldsymbol{\theta}}}\left(f_0(\boldsymbol{x}, \boldsymbol{\theta}_0)\right)\right) \cdot \nabla_{f_0} g_{\tilde{\boldsymbol{\theta}}}\left(f_0(\boldsymbol{x}, \boldsymbol{\theta}_0)\right), \tag{13}$$

and freeze it as a constant in the inner loop of the sample update. We call our accelerated Proximial SGLD algorithm PYLD-M-N (**P**roximal-**Y**OPO-SG**LD**) with M outer loops and N inner loops. Figure 1 demonstrates a conceptual comparison between SGLD-K and PYLD-M-N. SGLD-K accesses the data K times requiring K full forward and backward propagations. On the contrary, PYLD-M-N accesses the data $M \times N$ times, while only requiring M full forward and backward propagations and a inner loop of $M \times N$ cheap sample updates. Similar to YOPO [47], when $M \times N \approx K$, PYLD can achieve a similar sample quality as SGLD. But PYLD-M-N has the flexibility of increasing N and reducing M to achieve approximately the same level of movement with much less computation cost. We will demonstrate this when we present results.

The pseudo-code of our PYLD is described in Algorithm 1. For more details of YOPO, we refer the readers to [47].

3.4. Informative Initialization

The initial sampling distribution $p_0(\boldsymbol{x})$ also plays an important role in the training of EBM. Nijkamp et al. [31] sum-

Algorithm 1 PYLD-M-N sampling: Given network $g_{\tilde{\boldsymbol{\theta}}}$ and f_0 with $\boldsymbol{\theta}_0$, step-size α, number of steps M and N

$\boldsymbol{x}^0 \sim p_0(\boldsymbol{x})$
for $t \in [0, 1, \cdots, M-1]$ **do**
  % calculate the slack variable
  $p = -\nabla_{g_{\tilde{\boldsymbol{\theta}}}} \mathrm{LSE}\big(g_{\tilde{\boldsymbol{\theta}}}(f_0(\boldsymbol{x}^t, \boldsymbol{\theta}_0))\big) \cdot \nabla_{f_0} g_{\tilde{\boldsymbol{\theta}}}(f_0(\boldsymbol{x}^t, \boldsymbol{\theta}_0))$
  $\boldsymbol{x}^{t,0} = \boldsymbol{x}^t$
  for $s \in [0, 1, \cdots, N-1]$ **do**
    $\gamma = \mathrm{clamp}(p \cdot \nabla_{\boldsymbol{x}^{t,s}} f_0(\boldsymbol{x}^{t,s}, \boldsymbol{\theta}_0), \varepsilon)$
    $\boldsymbol{x}^{t,s+1} = \boldsymbol{x}^{t,s} - \alpha/2 \cdot \gamma$
  end for
  $\boldsymbol{x}^{t+1} = \boldsymbol{x}^{t,N} + \alpha \epsilon^t$
end for
return $\boldsymbol{x}^M$

marize two main types of SGLD initializations for $\boldsymbol{x}^0$: non-informative initialization and informative initialization. The former initializes the sample $\boldsymbol{x}^0$ from a noise distribution independent to the training data, such as a uniform or Gaussian distribution, while the latter samples from an approximate distribution close to the data distribution. One typical informative initialization is to use samples from training data directly, as proposed in Contrastive Divergence (CD) [20]. Based on this, Tieleman [39] proposes Persistent Contrastive Divergence (PCD) and uses samples from previous learning iteration as the initial samples for the current iteration. In contrast to common wisdom, Nijkamp et al. [32] propose a short-run MCMC sampler which always starts from the random noise distribution such as a uniform distribution. Moreover, to train EBMs, Xie et al. [45] propose another persistent initialization, which combines non-informative and informative initialization and samples short SGLD chains from data samples of previous iterations and occasionally (with a small probability ρ) reinitializes the chains from random noise. This is also the sampling approach adopted by IGEBM [9] and JEM [12], which maintain a replay buffer of samples from previous iterations and replace a small percentage of samples in the buffer with random noise to train EBMs.

In this paper, we explore informative initialization to initialize the SGLD chain, and use the PCD with a replay buffer. The main difference is that we substitute the random noise samples with samples from a Gaussian mixture distribution estimated from the training dataset. That is, we define the initial sampling distribution as

$$p_0(\boldsymbol{x}) = \sum\nolimits_y \pi_y N(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y) \tag{14}$$
$$\text{with} \quad \pi_y = |\mathcal{D}_y| / \sum\nolimits_{y'} |\mathcal{D}_{y'}|, \quad \boldsymbol{\mu}_y = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_y}[\boldsymbol{x}],$$
$$\boldsymbol{\Sigma}_y = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_y}\left[(\boldsymbol{x} - \boldsymbol{\mu}_y)(\boldsymbol{x} - \boldsymbol{\mu}_y)^\top\right],$$

where $\mathcal{D}_y$ denotes the set of training samples with label y.

Figure 2. The categorical centers of CIFAR10.

As an example, Figure 2 visualizes the $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_{10}\}$ (categorical centers) estimated from the CIFAR10 training dataset. Similar visualizations on CIFAR100 and SVHN as well as example samples from the informative initialization can be found in the supplementary material.

The informative initialization brings sufficient information into $\boldsymbol{x}^0$ to guide the SGLD chain to converge faster than from a random noise since the initial sample $\boldsymbol{x}^0$ is now much closer to the real data manifold. Empirically, we also observe the improved training stability. What's more, the informative initialization allows us to enable batch norm [22], a modern architecture feature of DNNs, that is excluded by IGEBM and JEM due to the training difficulty introduced by batch norm.

3.5. Batch Normalization and Learning Rate

Batch norm [22] is an essential component in many state-of-the-art CNN architectures. Batch norm normalizes input features by the mean and variance computed within each mini-batch, which mitigates the vanishing gradient issue of training very deep networks and dramatically improves the convergence rate of gradient-based methods. Moreover, batch norm allows a much larger learning rate and mitigates the need of tedious finetuning.

However, state-of-the-art EBMs, such as IGEBM [9] and JEM [12], do not support batch norm. If batch norm is enabled in JEM, the model can neither achieve a high classification accuracy nor generate realistic images. This is because one intrinsic assumption of batch norm is that the input features should come from a single or similar distributions. This normalization behavior could be problematic if the mini-batch contains data from different distributions, therefore resulting in inaccurate statistics estimation. Unfortunately, this might be the case for the original IGEBM and JEM. Apparently, if the initial samples $\boldsymbol{x}^0$ are sampled from a uniform or Gaussian distribution as in IGEBM and JEM, $\boldsymbol{x}^0$ and real data samples have different underlying distributions, violating the assumption of batch norm.

Similar phenomenon has also been observed by Xie et al. [43] who demonstrates the different statistics between clean data and adversarial examples. They show that both clean data accuracy and adversarial robustness can be improved by using two branches of batch norm: one main branch for clean data and one auxiliary branch for adversarial examples. Instead of using two batch norms, we mitigate the training difficulty of batch norm from a different perspective. Since we have the choice of designing sampling distribution $p_0(\boldsymbol{x})$, we can use the informative initialization discussed above to enable batch norm in the EBM

Algorithm 2 Training JEM++: Given network $f_{\boldsymbol{\theta}}$, step-size α, replay buffer $\mathbb{B}$, number of steps M and N, reinitialization frequency ρ, and number of classes C

while not converged **do**
  Sample $(\boldsymbol{x}^r, y^r) \sim \mathcal{D}$
  Sample $\boldsymbol{x}^0 \sim \mathbb{B}$ with probability $1-\rho$, else $\boldsymbol{x}^0 \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y), y \sim p(y) = \boldsymbol{\pi}$
  Apply PYLD in Algo. 1 to sample $\boldsymbol{x}^M$ from $\boldsymbol{x}^0$
  Calculate gradient with Eq. 2 from $\boldsymbol{x}^r$ and $\boldsymbol{x}^M$, and gradient of CE loss from $(\boldsymbol{x}^r, y^r)$, and update model parameter $\boldsymbol{\theta}$
  Add / replace sample $\boldsymbol{x}^M$ back to $\mathbb{B}$
end while

training. Since the Gaussian mixture distribution (14) is actually estimated from real training examples, we can close the statistic gap between initial samples of SGLD and real data and enable batch norm in JEM++ successfully. What's more, with the informative initialization and batch norm, JEM++ can also use a much larger learning rate to improve convergence rate even further.

In summary, Algorithm 2 provides the pseudo-code for JEM++ training, which follows a similar design of JEM [12] and IGEBM [9] with a replay buffer. For brevity, only one real sample $(\boldsymbol{x}^r, y^r) \sim \mathcal{D}$ and one generated sample $\boldsymbol{x}^M \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})$ are used to optimize the parameter $\boldsymbol{\theta}$. It is straightforward to generalize the pseudo-code above to a mini-batch setting, which we use in the experiments.

4. Experiments

We evaluate the performance of JEM++ on multiple discriminative and generative tasks, including image classification, image generation, adversarial robustness, calibration of uncertainty, and out-of-distribution (OOD) detection. Since our main goal is to improve JEM's accuracy, training stability and speed, we present these results in the main text and relegate its downstream applications, such as adversarial robustness, calibration and OOD detection, to the supplementary material. For a fair comparison with JEM [12], our experiments closely follow the settings provided in the source code of JEM[4]. All our experiments are performed with PyTorch on Nvidia RTX GPUs.

4.1. Hybrid Modeling

We train JEM++ on three benchmark datasets: CIFAR10, CIFAR100 [26] and SVHN [30], and compare it to the state-of-the-art hybrid models, as well as standalone generative and discriminative models. Following the settings of JEM [12], all our experiments are based on the Wide-ResNet architecture [46], with the details of hyper-parameter settings of JEM++ provided in the supplementary material. It's worth mentioning that applying the SGD optimizer with $lr = 0.1$ to JEM++ achieves better accuracy than the default setting of JEM using Adam with $lr = 0.0001$[5]. To evaluate the quality of generated images, we adopt the Inception Score (IS) [36] and Fréchet Inception Distance (FID) [19].

Table 2. Hybrid Modeling Results on CIFAR10. We report JEM++'s performance with different Ms when $N = 5$ is fixed. We also report the per epoch speedup between JEM and JEM++.

Class	Model	Acc % ↑	IS* ↑	FID* ↓
Single Hybrid Model	Residual Flow [7]	70.3	3.60	46.4
	Glow [24]	67.6	3.92	48.9
	IGEBM [9]	49.1	8.30	37.9
	JEM (K=20) [12] 1×	92.9	**8.76**	38.4
	JEM++ (M=5) 2.4×	91.1	7.81	37.9
	JEM++ (M=10) 1.5×	93.5	8.29	37.1
	JEM++ (M=20) .92×	**94.1**	8.11	38.0
Reg Gen.	VERA† (α=100) 2.8×	93.2	8.11	30.5
	VERA [14] (α=1) 2.8×	76.1	8.00	27.5
Disc.	WRN w/ BN	95.8	N/A	N/A
Gen.	SNGAN [29]	N/A	8.59	25.5
	NCSN [38]	N/A	8.91	25.3

†VERA uses an auxiliary generator to amortize the SGLD sampling and reports a 2.8× speedup without much details on how the evaluation is performed.
*A fair evaluation of IS and FID is challenging as different methods use different ways to measure the image quality. JEM uses an ensemble of models to evaluate its IS and FID, while JEM++ only uses a single model for evaluation. No more details are provided in JEM. Thus, it is difficult to have a fair comparison.

Table 3. Test Accuracy (%) on SVHN and CIFAR100.

Model	SVHN	CIFAR100
Softmax (w/ BN)	97.0	78.9
VERA [14]	96.8	72.2
JEM (K=20)	96.7	72.2
JEM++ (M=5)	96.7	72.0
JEM++ (M=10)	**96.9**	**74.5**

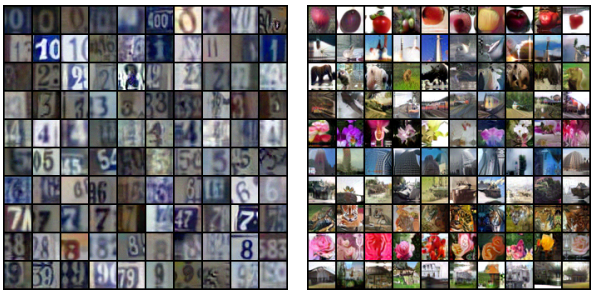
The results on CIFAR10, CIFAR100 and SVHN are reported in Table 2 and 3, respectively. It can be observed that JEM++ ($M = 10$) outperforms JEM and other single-network hybrid models in terms of accuracy (93.5%), FID score (37.1) and per epoch speedup (1.5×), while being slightly worse in IS score. Since no IS and FID scores are commonly reported on SVHN and CIFAR100, we present the classification accuracy and generated samples on these two benchmarks. Our JEM++ (M=10) model achieves an accuracy of 96.9% and 74.5% on SVHN and CIFAR100, respectively, outperforming JEM by notable margins. Example images generated by JEM++ for CIFAR10, SVHN

[4] https://github.com/wgrathwohl/JEM

[5] JEM cannot use a learning rate larger than 0.0001. Otherwise, it is extremely unstable and diverges easily at early epochs.

(a) Unconditional Samples (b) Class-conditional Samples

Figure 3. JEM++ generated CIFAR10 samples.



(a) SVHN (b) CIFAR100

Figure 4. JEM++ generated class-conditional samples of SVHN and CIFAR100. Each row corresponds to one class.

and CIFAR100 are shown in Figure 3 and 4, respectively. Additional JEM++ generated images can be found in the supplementary material.

We also investigated JEM++'s performances on several downstream applications, including adversarial robustness, calibration of uncertainty, and OOD detection, where JEM++ achieves improved performances over the original JEM in most of the cases. Due to page limit, the details are relegated to the supplementary material.

4.2. Training Stability and Speed

The main limitation of the SGLD-based training is the tradeoff between training time and stability. The more SGLD sampling steps are used, the more stable and better performance EBMs can achieve. In this section, we evaluate JEM and JEM++ in terms of training stability and speed.

We first compare the training stability of JEM and JEM++. From our empirical study, the official JEM (K-step SGLD with $K = 20$) suffers from training instability, i.e., it regularly diverges before 60 epochs. Prior works [9, 12], including JEM, fail to find a reasonably small K to completely stabilize the training of EBMs, and thus rely on checkpoints to resume the training when divergences occur. Figure 5 shows the learning curves of JEM++ trained on CIFAR10 with different configurations. As can be seen, JEM++ is much more stable and does not diverge when $M = 20$. What's more, JEM++ with $M = 10$ can achieve high stability; even JEM++ with $M = 5$ is more stable than JEM with $K = 20$. As discussed in Section 3, the informative initialization improves JEM's stability because the initial samples $\boldsymbol{x}^0$ of SGLD are now close to the real data manifold. Hence, the sampling process requires fewer steps to reach the low energy region of the energy function, which we conjure should be much smoother than other regions. In addition, the proximity constraint also improves the stability of JEM++ as demonstrated in Figure 5.

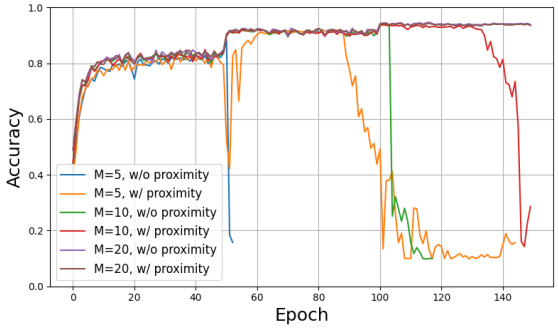


Figure 5. The learning curves of JEM++ trained on CIFAR10 with different configurations: (1) Number of steps M with $N = 5$, and (2) the proximity constraint. The official JEM uses $K = 20$, but it regularly diverges before 60 epochs.

We further compare the training speed between JEM and JEM++ in terms of run-time per epoch. The results are reported in Table 4, where we compare JEM and JEM++ trained on CIFAR10 with different configurations of M and N. It can be observed that M specifies the total number of forward and backward propagations of PYLD, consuming most of the run-time, while N has a minor impact on the run-time as it specifies the number of inner loops for sample update, which is relatively inexpensive. Therefore, we can increase N and reduce M to achieve approximately the same level of sample quality with much less computation cost. Considering the training stability (Figure 5) and training speed (Table 4), $M = 10$ and $N = 5$ achieves a good balance between the two criteria and therefore is our default configuration of JEM++.

Table 4. Run-time comparison of JEM and JEM++ on CIFAR10.

Model	Minutes per epoch	Speedup
JEM	30.1	1×
JEM++, $M = 5$		
$N = 5$	12.5	2.41×
$N = 10$	12.6	2.39×
$N = 20$	13.0	2.31×
JEM++, $M = 10$		
$N = 5$	20.1	1.49×
$N = 10$	20.3	1.48×
$N = 20$	20.4	1.47×
JEM++, $M = 20$		
$N = 5$	32.5	.93×
$N = 10$	32.7	.92×
$N = 20$	32.9	.91×

4.3. Ablation Study

JEM++ introduces a variety of new training procedures and architecture features to improve JEM's accuracy, training stability and speed. In this section, we study the effect of different components of JEM++ on the performance of image classification and image generation. Specifically, we conduct the ablation study on CIFAR10 with an exhaustive comparison of different components. We measure the effects of 1) w/o proximity constraint, 2) with Adam optimizer, 3) random initialization with batch norm enabled, and 4) two different types of initialization w/o batch norm.

The results are reported in Table 5. It can be observed that each component contributes to JEM++'s performance positively. The proximity constraint in Proximal SGLD improves both stability and accuracy. Our experiments show that when a smaller M enlarges the instability, the proximity constraint not only helps to stabilize the training, but also improves the accuracy of the trained models. The informative initialization also takes a significant role in JEM++, which enables both batch norm and the use of SGD with larger learning rates. When batch norm is enabled in JEM, we find that it can neither achieve a high classification accuracy nor generate realistic images. On the other hand, JEM without batch norm can achieve decent classification accuracy and generate quality images, but it's precarious and easily diverged at early epochs. The informative initialization itself w/o batch norm is still beneficial to stabilize the training, as manifested by the improved classification accuracy and image quality. It's worth mentioning that when batch norm is disabled, only Adam [23] with a small learning rate no greater than 0.0001 yields a stable training. However, when batch norm is enabled, the SGD optimizer with a much larger learning rate can be applied to train JEM++ successfully, outperforming the default Adam optimizer (with a very small learning rate) used in JEM.

Table 5. Ablation study of different components of JEM++. All the models are trained on CIFAR10 with $M = 10$ and $N = 5$.

Ablation	Acc % ↑	IS ↑	FID ↓
JEM++	**93.5**	**8.29**	37.1
w/o Proximity	92.9	7.92	36.0
w/ Adam	92.5	7.65	42.7
random init (w/ BN)[1]	-	-	-
random init (w/o BN)[2]	88.6	7.64	**35.1**
informative init (w/o BN)[3]	91.1	7.92	39.8

[1] It fails to achieve a high accuracy and generate realistic images.
[2] It diverges early at epoch 28.
[3] Without batch norm, only ADAM with $lr = 0.0001$ can be used.

4.4. Classification Accuracy vs. Image Quality

One interesting phenomenon we observed from our experiments is the tradeoff between classification accuracy and image quality. Figure 6 shows the evolution of classification accuracy, IS and FID scores as a function of the training epochs. At the early stage of training (before epoch 100), both classification accuracy and image quality can be improved jointly. After that, there is a clear competition between accuracy and image quality, where improving accuracy hurts image quality. This probably can be explained by our minimax objective (7), in which the classifier and the implicit generator compete with each other to achieve an equilibrium. Compared to the standard GANs [11], the difference is that we have only one network that serves both as classifier and generator. How to balance the discriminative and generative powers within one model is unclear. It would be interesting to investigate this further in the future.

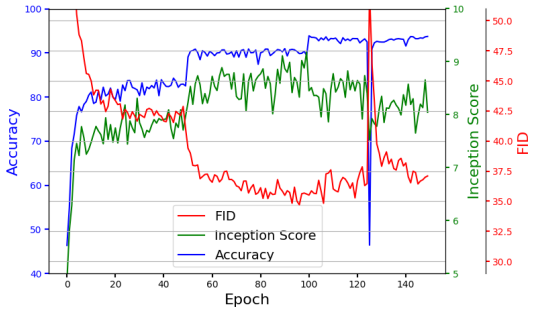


Figure 6. The evolution of JEM++'s classification accuracy, IS and FID scores as a function of training epochs on CIFAR10. The spike around epoch 125 is due to training instability and thanks to the proximity constraint, JEM++ stabilizes the training eventually.

5. Conclusion

In this paper, we propose JEM++ which improves JEM's accuracy, training stability and speed altogether with a number of new training procedures and architecture features. We demonstrate the effectiveness of these improvements on multiple benchmark datasets with state-of-the-art results in most of the tasks of image classification, image generation, adversarial robustness, uncertainty calibration and OOD detection. Most importantly, JEM++ enjoys stable and accelerated training over the original JEM.

As for future work, we plan to investigate the tradeoff between the classification accuracy and image quality as shown in Figure 6. We are interested in what the optimal tradeoff is and how we can achieve the optimum with architecture design and/or new training methodologies (e.g., [14, 1]). We also plan to explore JEM++ to large-scale benchmarks, such as ImageNet, and its application to other domains, such as NLP.

6. Acknowledgment

We would like to thank the anonymous reviewers for their comments and suggestions, which helped improve the quality of this paper. We would also gratefully acknowledge the support of VMware Inc. for its university research fund to this research.

References

[1] Lynton Ardizzone, Radek Mackowiak, Carsten Rother, and Ullrich Köthe. Training normalizing flows with the information bottleneck for competitive generative classification. In *Neural Information Processing Systems (NeurIPS)*, 2020. 8

[2] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning (ICML)*, 2018. 1

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations(ICLR)*, 2019. 1

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019. 1

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1

[6] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning. *IEEE Transactions on Neural Networks*, 2009. 1

[7] Ricky TQ Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *arXiv preprint arXiv:1906.02735*, 2019. 1, 6

[8] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1977. 1

[9] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1, 2, 3, 5, 6, 7

[10] Ruiqi Gao, Erik Nijkamp, Diederik P. Kingma, Zhen Xu, Andrew M. Dai, and Ying Nian Wu. Flow contrastive estimation of energy-based models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. 1, 8

[12] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations (ICLR)*, 2020. 1, 2, 3, 5, 6, 7

[13] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, and Richard Zemel. Learning the stein discrepancy for training and evaluating energy-based models without sampling. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020. 1

[14] Will Sussman Grathwohl, Jacob Jin Kelly, Milad Hashemi, Mohammad Norouzi, Kevin Swersky, and David Duvenaud. No mcmc for me: Amortized sampling for fast and stable training of energy-based models. In *International Conference on Learning Representations (ICLR)*, 2021. 1, 2, 6, 8

[15] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010. 2

[16] Tian Han, Erik Nijkamp, Xiaolin Fang, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Divergence triangle for joint training of generator model, energy-based model, and inference model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[17] Tian Han, Erik Nijkamp, Linqi Zhou, Bo Pang, Song-Chun Zhu, and Ying Nian Wu. Joint training of variational auto-encoder and latent energy-based model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[19] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 6

[20] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 2002. 2, 5

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. 2, 5

[23] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 8

[24] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, 2018. 6

[25] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014. 1

[26] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 1, 6

[27] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 2006. 1, 2

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018. 2, 4

[29] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018. 6

[30] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Ng andrew Y. Reading digits in natural images with unsupervised feature learning. 2011. 6

[31] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019. 2, 3, 4

[32] Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. On learning non-convergent short-run mcmc toward energy-based model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1, 2, 3, 5

[33] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014. 3

[34] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC, 1987. 4

[35] R. Rockafellar. Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1(2):97–116, 1976. 3, 4

[36] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems (NeurIPS)*, 2016. 6

[37] Shibani Santurkar, Andrew Ilyas, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Image synthesis with a single (robust) classifier. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1

[38] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *arXiv preprint arXiv:1907.05600*, 2019. 6

[39] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *International conference on Machine learning (ICML)*, 2008. 5

[40] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems*, 2020. 1

[41] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 2011. 1, 2

[42] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML)*, 2011. 1, 2

[43] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L. Yuille, and Quoc V. Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 5

[44] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of descriptor and generator networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:27–45, Jan. 2020. 1

[45] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, 2016. 1, 5

[46] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *The British Machine Vision Conference (BMVC)*, 2016. 1, 6

[47] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1, 2, 3, 4

[48] Stephen Zhao, Jorn-Henrik Jacobsen, and Will Grathwohl. Joint energy-based models for semi-supervised classification. In *ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning*, 2020. 1