Improving GAN Inference with EBM model. Project Report

Vasiliy Viskov [1] **Bogdan Alexandrov** [1] **Nikita Sukhorukov** [1] **Nikolay Ivanov** [1] **Mark Nuzhnov** [1]

Abstract

In this work, we have provided motivation and a mathematical description of the combination of generative adversarial networks with energy-based models. With combining these two models we want to stabilize the GAN and make it possible to generate a disjoint distribution. Also the big advantage of this type of model is that after learning our model we get not only sampler (generator) from learned distribution, but its unnormalized Probability Density Function. We showed our results and compare it with GAN.

Github repo: https://github.com/v-vskv-v/GAN_EBM
Presentation file: Presentation_GAN_EMB.pdf

1. Introduction

1.1. GANs

Generative Adversarial Networks (Ian J. Goodfellow & Xu, 2014) are type of neural network that can generate complex images, sounds, and other data by learning from existing datasets. It is a generative model that is trained on real data and learns to generate similar synthetic data that is difficult to distinguish from the original data.

GANs consist of two neural networks working together, the generator and the discriminator. The generator generates synthetic data, while the discriminator identifies whether the generated data is real or fake. These two networks work in tandem, with the objective of the generator being to produce synthetic data that is nearly indistinguishable from the real data while the discriminator's objective is to distinguish between the real and fake data with increasing accuracy.

One of the main advantages of GANs is their ability to generate high-quality, realistic data. This makes them useful for applications such as image and video generation, where realistic results are crucial, and they still give SOTA results in some fields. Moreover, GANs can generate samples that have never been seen before, making them useful in situations where the data is scarce or difficult to collect.

Despite their advantages, GANs also have several disadvantages (Saxena & Cao, 2020). One of the major issues with GANs is instability, which arises from the min-max adversarial objective that the generator and discriminator networks have to optimize. Additionally, GANs may suffer from mode collapse, where the generator produces the same output for different input data, leading to a lack of diversity in generated samples. Also, these networks can suffer from gradient vanishing, when discriminator learned real data too well and doesn't provide enough information for the generator to make progress. Our work shows some attempts to overcome this problem via Energy Based Models.

1.2. EBMs

Energy-Based Models (LeCun Y. & Ranzato M., 2006) are a type of machine learning model that takes a different approach than traditional probabilistic models. Instead of utilizing probabilities to estimate the likelihood of a sample belonging to a certain class or category, EBMs use an energy function to evaluate the compatibility of a given input and the model's weights. These networks work by training the model to assign higher energy values for unlikely or anomalous events, while assigning lower energy values to more probable occurrences.

Because EBMs use an energy function to evaluate the likelihood of a given data sample, they can incorporate complex, high-level features that may not be captured by other traditional probabilistic models. Additionally, EBMs do not suffer from the overfitting problem that affects many traditional models, which means they can perform better in the presence of smaller amounts of training data.

However, for implementing EBMs is the high computational requirements needed to train them. EBMs use Markov Chain Monte Carlo methods for training and inference, which is time-consuming operation. Instead of using EBM for sampling, we can train a generator, which will decrease

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Vasiliy Viskov <Vasiliy.Viskov@skoltech.ru>, Bogdan Alexandrov <Bogdan.Alexanrov@skoltech.ru>, Nikita Sukhorukov <Nikita.Sukhorukov@skoltech.ru>, Nikolay Ivanov <Nikolay.Ivanov3@skoltech.ru>, Mark Nuzhnov <Mark.Nuzhnov@skoltech.ru>.

Final Projects of the Machine Learning 2023 Course, Skoltech, Moscow, Russian Federation, 2023.

time of inference.

1.3. Motivation

The motivation behind combining Generative Adversarial Networks (GANs) and Energy-Based Models (EBMs) is to address the limitations of each framework and leverage their strengths to improve generative modeling tasks.

GANs are known for their ability to generate high-quality images, but they can suffer from instability during training and mode collapse, where the generator produces limited variations of the same samples. On the other hand, EBMs are more stable during training and can handle more complex data distributions, but they have been less successful in image generation tasks.

By combining GANs and EBMs, we aim to overcome these limitations and create a framework that is more stable, efficient, and effective in generating high-quality samples. The EBM provides a way to model the energy of the data distribution and guide the generator towards producing more diverse and realistic samples. Additionally, GAN-EBMs offer a natural way to incorporate prior knowledge into the generative model by adjusting the energy function accordingly.

1.4. Our contribution

The main contributions of this report are as follows:

1. Creating model, which combines EBM and GAN architectures
2. Providing results of several experiments on datasets Make Moons, Swiss Roll, MNIST and CIFAR10
3. Explaining hyperparameters influence on model performance

2. Preliminaries

Generative Adversarial Networks The whole process of training and mathematical basis were taken from this work (Ian J. Goodfellow & Xu, 2014). When both models are multilayer perceptrons, the adversarial modeling framework can be applied in a straightforward manner. The goal is to learn the distribution p_g of data x generated by the generator, which is represented by a differentiable function $G(z;\theta_g)$ mapping input noise variables $pz(z)$ to data space. θ_g represents the parameters of the multilayer perceptron. We also introduce a second multilayer perceptron $D(x;\theta_d)$, which outputs a single scalar. The function $D(x)$ estimates the probability that the input x came from the real data distribution rather than the generator p_g. The models are trained in a way that D maximizes the probability of assigning the correct label to both training examples and samples from G, while G is trained to minimize $log(1 - D(G(z)))$:

Total optimization function for Discriminator and Generator $V(G,D)$:

$$\min_G \max_D V(D,G) = \\ = \mathbb{E}_{p_{data}(x)}[\log D(x)] + \mathbb{E}_{p_z(z)}[\log(1 - D(G(z)))] \tag{1}$$

Energy-based models (EBMs) (*LeCun Y.&Ranzato M.*, 2006) In this work authors discover main concepts and basics of Energy-based models. One way to represent probability distributions is by using a scalar function f_θ, which outputs values that can be converted into probabilities using a Gibbs distribution:

$$q_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta} \tag{2}$$

The partition function, also known as the normalizing constant $Z(\theta)$, is defined as an integral over the unnormalized probability of all states: $Z(\theta) = \int exp(f_\theta(x))dx$. The energy function, which is defined as $E_\theta(x) = -f_\theta(x)$, attributes low energy outputs on the support of the target data distribution and high energy outputs in other regions.

However, for many complex models, computing the partition function $Z(\theta)$ is intractable, making the maximum likelihood estimation (MLE) of the model parameters θ not directly applicable. To address this, standard maximum likelihood learning of Energy-Based Models (EBMs) relies on the gradient of the log likelihood function. Specifically, denoting the distribution of the observed data as p_{data}, the gradient of the log likelihood takes the following form:

$$\nabla_\theta \mathcal{L}\text{MLE}(\theta;p) = -\mathbb{E}_{p(\mathbf{x})}\left[\nabla_\theta \log q_\theta(\mathbf{x})\right] = \\ = \mathbb{E}_{p(\mathbf{x})}\left[\nabla \theta E_\theta(\mathbf{x})\right] - \mathbb{E}_{q_\theta(\mathbf{x})}\left[\nabla_\theta E_\theta(\mathbf{x})\right] \tag{3}$$

To evaluate $E_{x\sim p_\theta(x)}\nabla_\theta f_\theta(x)$, it is necessary to sample from the model distribution. One common way to achieve this is through Markov chain Monte Carlo (MCMC) methods. In our work, we have drawn inspiration from the ideas presented in (Xie & Lu, 2016), which address the challenge of scaling up training of Energy-Based Models (EBMs) to high-dimensional data. Specifically, we make use of the Stochastic Gradient Langevin Dynamics (SGLD) method (Welling, 2011) to sample from the model distribution.

$$x_0 \sim p_0, x_{i+1} = x_i + \frac{\lambda}{2}\nabla_x f_\theta(x_i) + \epsilon, \epsilon \sim \mathcal{N}(0,\lambda) \tag{4}$$

where p_0 is some random noise distribution.

To ensure that the distribution of sampled data matches p_θ, an effective Stochastic Gradient Langevin Dynamics (SGLD) sampler typically requires a large number of update steps. However, due to the significant computational cost associated with this sampling process, in our work, we employ short-run non-convergent Markov chain Monte Carlo (MCMC) methods to improve the efficiency of the sampling process (Nijkamp).

3. Related work

We have reviewed and studied articles related to the training and inference of GANS and EBM. Here's their main concepts and ideas.
In this (Jianwen Xie & Wu, 2022) paper authors introduce novel architecture of GAN, where both Generator and Discriminator perform as EBM. They propose cooperative learning algorithm which work as follows:

- Generator produce synthetic data from noise, let's call it X
- Discriminator takes Generator output X and performs finite steps of MCMC on it. Let's call this output Y.
- Discriminator takes real data, calculate loss and update weights
- Generator takes Discriminator output Y, takes initial latent representation X, which was used for it's generation, also perform MCMC on it and then calculate loss with Y as real data.

Authors show, that this algorithm is stable and not prone to mode collapse issue. And in comparison to other models, like W-GAN, DCGAN, it's shows less restoration error and less Frechet Inception Distance. However, no code was provided, so we can't reproduce their experiment. This work shows high results, but don't solve problem of time-consuming inference of EBM.

Another approach for more stable GAN training was proposed in this (Takeru Miyato & Yoshida, 2018) work, where authors propose Spectral Normalization technique, when layers of neural network divided by their spectral norm. They study, how their invention performs on real datasets and show, that their normalization more robust to wide spectre of hyperparameters and provide better scores compare to other normalizations. Disadvantage of this approach is that it time-consuming.

As mentioned in another paper (Saxena & Cao, 2020), several ways for improving training of GANs exist: building stronger architectures for Generator and Discriminator, optimizing another losses, applying different regularization techniques. For example, in this paper (Alec Radford & Chintala, 2016) different CNN architectures were examined on images. Authors give some intuitions, how to choose appropriate architectures, but they don't solve problem of collapsing and instability in training GANs.

Also, there were several researches upon how to improve EBM training, like in (Xuwang Yin & Rohde, 2022), (Erik Nijkamp & Wu, 2019), but their inference still slow due to MCMC. In another paper (Will Grathwohl & Swersky, 2021) authors try to amortize time-consuming MCMC and made their model twice faster than alternative EBMs, but they don't solve problem in essense and performance of approach too low with comapre to GANs.

As we investigated, there are exists a lot of works showing how improve GANs and EBMs performance, but none of them propose how to get both sampler from learned distribution and its PDF or how to remove computational difficult MCMC method. Our work attempts to solve this problem.

4. Algorithms and Models

4.1. MCMC algorithm

To sample from an energy-based model, one approach is to use Markov Chain Monte Carlo with Langevin Dynamics. The algorithm starts from a random point and gradually moves towards the direction of higher probability by using the gradients of E_θ. However, this alone is insufficient to fully capture the probability distribution (Du & Mordatch, 2020), so we introduce noise ω at each gradient step to the current sample. While it's theoretically possible to create an exact sample from our modeled distribution by performing an infinite number of gradient steps, it's impractical to do so. Hence, we usually limit the chain to K steps (with K being a hyperparameter that requires fine-tuning). The overall sampling procedure can be summarized as follows:

Algorithm 1 MCMC algorithm for energy-based models

```
input: $\widetilde{x}^0$
for sample step $k = 1$ to $K$ do
    ▷ Generate sample via Langevin dynamics
    $\widetilde{x}^k \leftarrow \widetilde{x}^{k-1} - \eta\nabla_x E_\theta(\widetilde{x}^{k-1}) + \omega, \triangleright\, \omega \sim \mathcal{N}(0, \sigma)$
end for
```

4.2. Sampling algorithm

For stabilizing of EBM training (Du & Mordatch, 2020) we may store outputs from previous MCMC runs. Once the sampling buffer is prepared, the sampling algorithm for an energy-based model on image modeling can be completed. The following is a summary of the complete algorithm:

Algorithm 2 Sampling from an energy-based model

```
Initialize empty buffer $\mathcal{B} \leftarrow \emptyset$
▷ Sample initial fake data:
$z \sim \mathcal{N}(0,1)$
$x_i^0 \sim \mathcal{B}$, with 95 percent probability, else $G(z)$
$\widetilde{x}^K = MCMC(x^0)$
$x^- \leftarrow \Omega(\widetilde{x}^K)$                ▷$\Omega$: Stop gradient operator
▷Add samples to buffer: $\mathcal{B} \leftarrow \mathcal{B} \cup x^-$
return $x^-$
```

4.3. Training algorithm

The overall training algorithm of our architecture is as follows: Generator produce samples, then EBM takes this output to produce new samples. After it, Discriminator takes Generator output as fake data and EBM output as real data. For updating weights, EBM takes real data and compute loss, while Discriminator and Generator use EBM's output for learning. Here you can see algorithm step by step:

Algorithm 3 Training pipeline

```
while not converged do
    ▷ sample noise for Generator
    $z \sim \mathcal{N}(0,1)$
    $x_g = G(z)$
    $x_{EBM} = EBM\_Sample(x_g)$
    ▷ provide Discriminator EBM output as data
    $d\_real = D(x_{EBM})$
    $d\_fake = D(x_g)$
    ▷ update D weights
    $D_{loss} = BCE(true\_labels, d\_real) + BCE(fake\_labels, d\_fake)$
    update $\theta_D$
    ▷ update G weights
    $\hat{z} \sim \mathcal{N}(0,1)$
    $G_{loss} = log(D(G(\hat{z})))$
    update $\theta_G$
    ▷ Sample data from dataset:
    $x^+ \sim \rho_{\mathcal{D}}$
    ▷ Constrastive divergence:
    $\mathcal{L}_{CD} = \frac{1}{N}\sum_i (E_\theta(x_i^+) - E_\theta(x_{EBM_i}))$
    ▷ Regularization loss:
    $\mathcal{L}_{RG} = \frac{1}{N}\sum_i (E_\theta(x_i^+)^2 + E_\theta(x_{EBM_i})^2)$
    ▷ update EBM weights
    Perform SGD/ Adam on $\nabla_\theta(\mathcal{L}_{CD} + \alpha\mathcal{L}_{RG})$
end while
```

For computing Discriminator loss we used Binary Cross Entropy loss. To learn EBM, loss was taken from (Du & Mordatch, 2020), with regularization parameter α. In our experimenters we set this parameter to 0.1, this value we found the most appropriate during manual search.

4.4. Models

Generator and Discriminator acrhitectures were Feed Forward Networks with LeakyReLU with $\alpha = 0.2$ and adding batch normalization, optimizator - Adam. Main hyperparameters of the EBM were amount of steps (*steps_cnt*) and η - step size in MCMC algorithm, which optimal values we also found manually. Generator input distribution was multivariate normal distribution of size 20 with zero mean vector and unit covariance matrix.

As EBM we used different architectures for 2D-datasets, MNIST and CIFAR. Here's explanation of these architectures:

- For Make Moons and Swiss Roll datasets EBM is FFN with 6 layers, SiLU as activation function and BatchNorm layers after last two hidden layers.
- For MNIST and CIFAR10 datasets EBM is CNN with 4 convolutional layers and 2 linear, SiLU as activation function.

We used simple and light weighted NNs for fast experiments.

All details and architecture implementations are provided in our github.[1]

5. Experiments and Results

In our experiments we considered 2D-datasets (Make Moons[2] and Swiss roll[3]) and image datasets (MNIST[4] and CIFAR10[5]). Data was scaled to be in the segment $[-1, 1]$ for every dimension. We compared vanilla GAN (Ian J. Goodfellow & Xu, 2014) model with proposed one.

5.1. Make Moons

We used 100 epochs for training both GAN and GAN + EBM.

[1] https://github.com/v-vskv-v/GAN_EBM

[2] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html

[3] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_swiss_roll.html

[4] https://pytorch.org/vision/main/generated/torchvision.datasets.MNIST.html

[5] https://pytorch.org/vision/main/generated/torchvision.datasets.CIFAR10

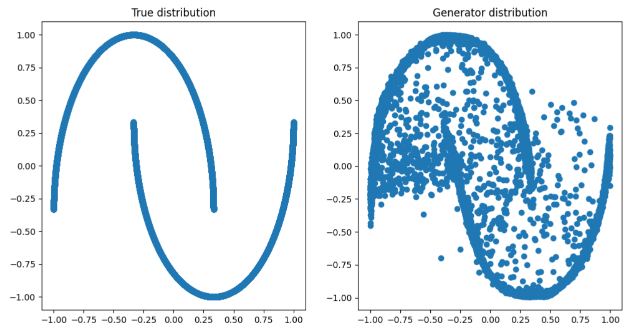

Figure 1. True distribution and generator distribution after GAN training

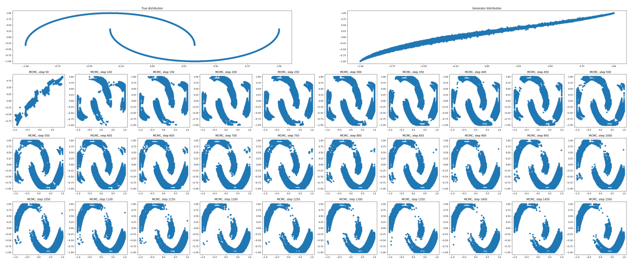

Figure 2. True distribution and generator distribution after GAN + EBM training (*steps_cnt*: 60, *step_size*: 3)

For evaluation *steps_cnt* and *step_size* were fixed with 1500 and 2.

The best visual results were obtained with hyperparameter pair (60, 3). Check rest results in Github repo.

5.2. Swiss roll

We used 150 epochs for training both GAN and 100 - for GAN + EBM.

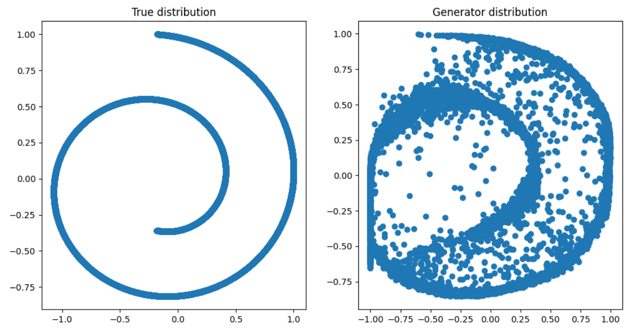

Figure 3. True distribution and generator distribution after GAN training

For evaluation *steps_cnt* and *step_size* were fixed with 1500 and 3.

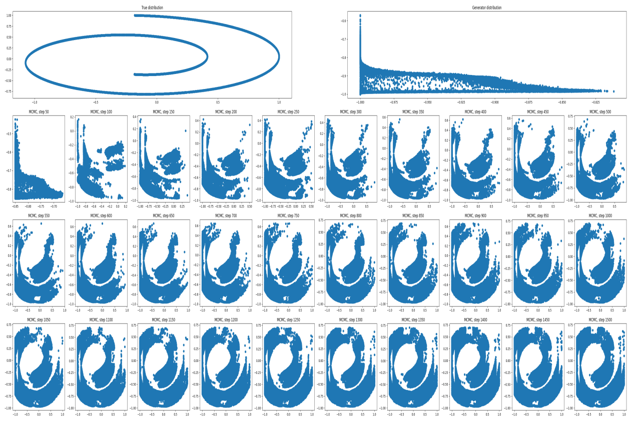

Figure 4. True distribution and generator distribution after GAN + EBM training (*steps_cnt*: 60, *step_size*: 3)

5.3. MNIST

Generator input was MNIST dataset distribution. Here are images by vanilla GAN generator on a Figure 5 and 500 steps of MCMC applied to generator output with step size 2 on a Figure 6.

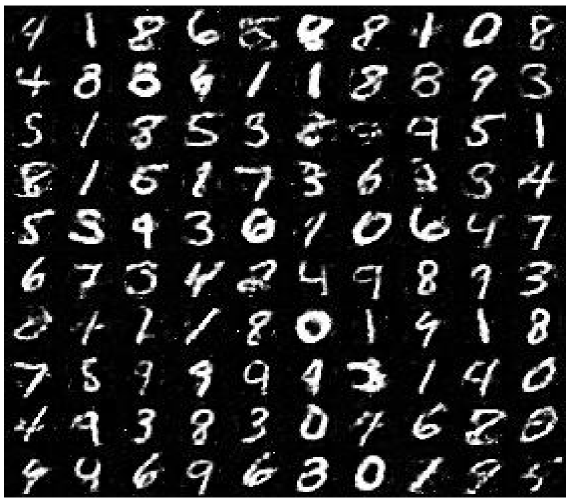

Figure 5. Image generated by vanilla GAN generator

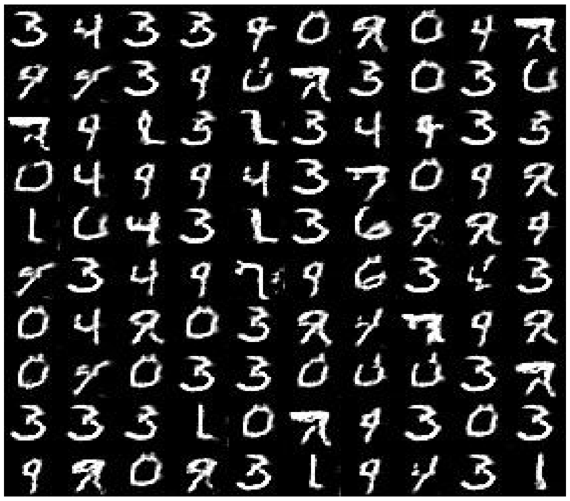

Figure 6. Image generated by proposed generative model (*steps_cnt*: 500, *step_size*: 2)

5.4. CIFAR10

Here are the comparison between generated images by vanilla GAN generator and 1500 steps of MCMC applied to generator output with step size 2.

For vanilla GAN model we dealed with zero discriminant loss problem - for every iteration we see this noisy image, as we can see on a Figure 7.

And on the Figure 8 we can such an improvement of images by generative model.

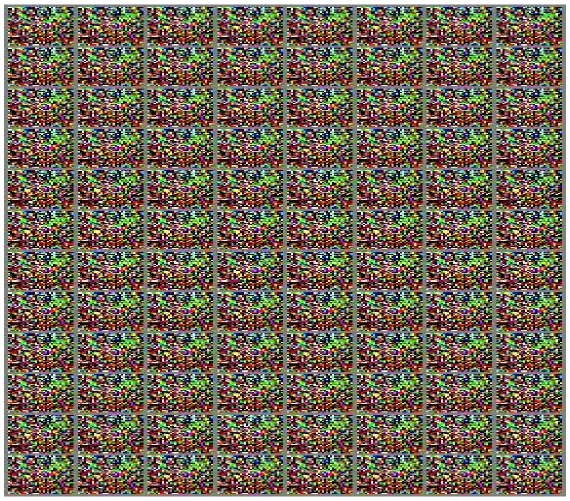

Figure 7. Image generated by Vanilla GAN

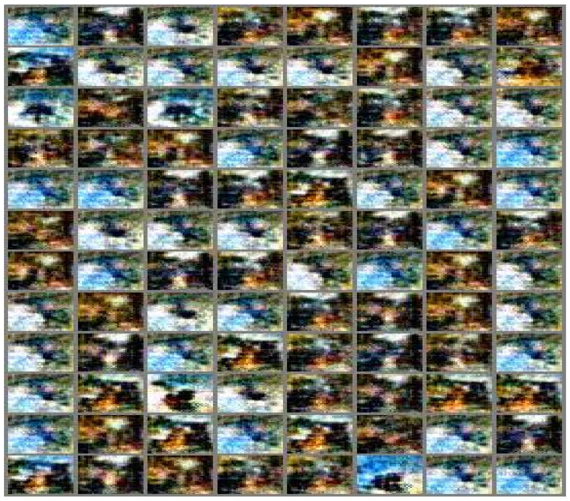

Figure 8. Image generated by proposed generative model (*steps_cnt*: 1500, *step_size*: 2)

5.5. Results explanation

We tried sample from trained generator and it worked bad. So we decided to use MCMC sampling-algorithm upon the generator output to produce better quality images. It was about 100 attempts to run all experiments and find optimal, appropriate parameters.

We got intuition for hyperparameter tuning for 2D-datasets: the higher step_size we use, the wider stripe we have on the plot. But for the small step_size we need more amount steps to get an appropriate distribution. For image datasets we didn't find any dependencies in that ground, here you need to tune it manually to get appropriate images or introduce evaluation metric and use grid search algorithm.

6. Conclusions

In this study, we explored the use of Generative Adversarial Networks (GANs) and Energy-Based Models (EBMs). We identified some limitations of these models, including the instability of GAN training and the difficulty of computing the partition function for EBMs. To overcome these challenges, we used a hybrid model that combines the strengths of both approaches and uses Monte Carlo Markov Chain (MCMC) techniques for sampling. Our experiments on various datasets showed that our model achieves better image quality and stability compared to GANs and EBMs alone.

Using different types of GANs, such as conditional GANs, InfoGANs, and others, is definitely a promising direction for future research. These variations of GANs can address specific problems, such as learning from limited or noisy data, controlling the output, and discovering disentangled representations.

In addition, exploring novel architectures and loss functions for GANs can also lead to significant improvements in their performance and stability. For example, recent research has shown the effectiveness of spectral normalization, self-attention mechanisms, and Wasserstein distance for training GANs.

Overall, our work contributes to advancing the state-of-the-art in generative modeling and provides a promising direction for future research in this area.

References

Alec Radford, L. M. and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2016.

Du, Y. and Mordatch, I. Implicit generation and modeling with energy-based models. *arXiv preprint arXiv:1903.08689*, 2020.

Erik Nijkamp, Song-Chun Zhu, M. H. and Wu, Y. N. Learning non-convergent non-persistent short-run mcmc toward energy-based model. *arXiv preprint arXiv:1904.09770*, 2019.

Ian J. Goodfellow, Jean Pouget-Abadie, M. M. and Xu, B. Generative adversarial nets. *arXiv preprint arXiv:1406.2661v1*, 2014.

Jianwen Xie, Yang Lu, R. G. I. and Wu, Y. N. Cooperative learning of descriptor and generator networks. *https://ieeexplore.ieee.org/ielaam/34/8922815/8519332-aam.pdf*, 2022.

LeCun Y., Chopra S., H. R. and Ranzato M., H. F. A tutorial on energybased learning. In *Predicting structured data 1(0)*, 2006.

Nijkamp, E., H. M. Z. S. W. Y. Learning non-convergent non-persistent short-run mcmc toward energy-based model.

Saxena, D. and Cao, J. Generative adversarial networks (gans): Challenges, solutions, and future directions. *arXiv preprint arXiv:2005.00065v3*, 2020.

Takeru Miyato, Toshiki Kataoka, M. K. and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957v1*, 2018.

Welling, M., T. Y. *Bayesian learning via stochastic gradient langevin dynamics.*, volume 3. Citeseer, 2011.

Will Grathwohl, Milad Hashemi, D. D. J. K. M. N. and Swersky, K. No mcmc for me: Amortized sampling for fast and stable training of energy-based models. *arXiv preprint arXiv:2010.04230*, 2021.

Xie, J. and Lu, Y., Z. S. W. Y. *A theory of generative convnet. In: International Conference on Machine Learning*, volume 3. PMLR, 2016.

Xuwang Yin, S. L. and Rohde, G. K. Learning energy-based models with adversarial training. *arXiv preprint arXiv:2012.06568v4*, 2022.

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Vasiliy Viskov (20% of work)

- Coding the training algorithm
- Experimenting with model parameters on Make moons, Swiss roll and CIFAR10 datasets
- Preparing the GitHub Repo

Bogdan Alexandrov (20% of work)

- Coding the sampling algorithm
- Experimenting with model parameters on MNIST dataset

Nikita Sukhorukov (20% of work)

- Reviewing literate on the topics (4 papers)
- Preparing the Related Work, Conclusion, References, Appendixes of this report

Nikolay Ivanov (20% of work)

- Coding the GAN with EBM pipeline
- Preparing the Inroduction and Abstract of this report

Mark Nuzhnov (20% of work)

- Plot the result
- Train Vanilla GAN
- Preparing the Algorithms and Models, Experiments and Results of this report

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   - ☐ Yes.
   - ☑ No.
   - ☐ Not applicable.

   Students' comment: None

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   - ☑ Yes.
   - ☐ No.
   - ☐ Not applicable.

   Students' comment: None

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   - ☑ Yes.
   - ☐ No.
   - ☐ Not applicable.

   Students' comment: None

4. A complete description of the data collection process, including sample size, is included in the report.

   - ☑ Yes.
   - ☐ No.
   - ☐ Not applicable.

   Students' comment: None

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   - ☑ Yes.
   - ☐ No.
   - ☐ Not applicable.

   Students' comment: None

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   - ☐ Yes.
   - ☐ No.
   - ☑ Not applicable.

   Students' comment: None

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   - ☐ Yes.
   - ☐ No.
   - ☑ Not applicable.

   Students' comment: None

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   - ☑ Yes.
   - ☐ No.
   - ☐ Not applicable.

   Students' comment: None

9. The exact number of evaluation runs is included.

   - ☑ Yes.
   - ☐ No.
   - ☐ Not applicable.

   Students' comment: None

10. A description of how experiments have been conducted is included.

    - ☑ Yes.
    - ☐ No.
    - ☐ Not applicable.

    Students' comment: None

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    - ☐ Yes.
    - ☐ No.
    - ☑ Not applicable.

    Students' comment: None

12. Clearly defined error bars are included in the report.

    - ☐ Yes.
    - ☐ No.
    - ☑ Not applicable.

    Students' comment: None

13. A description of the computing infrastructure used is included in the report.

    - ☑ Yes.
    - ☐ No.
    - ☐ Not applicable.

    Students' comment: None