Hybrid Energy-based Models for Image Generation and Classification

Xiulong Yang
Georgia State University

Follow this and additional works at: https://scholarworks.gsu.edu/cs_diss

Recommended Citation
https://scholarworks.gsu.edu/cs_diss/190

This Dissertation is brought to you for free and open access by the Department of Computer Science at ScholarWorks @ Georgia State University. It has been accepted for inclusion in Computer Science Dissertations by an authorized administrator of ScholarWorks @ Georgia State University. For more information, please contact scholarworks@gsu.edu.
Hybrid Energy-based Models for Image Generation and Classification

by

Xiulong Yang

Under the Direction of Shihao Ji, Ph.D.

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2022
ABSTRACT

In recent years, deep neural networks (DNNs) have achieved state-of-the-art performance on a wide range of learning tasks. Among those tasks, two fundamental tasks are discriminative models and generative models. However, they are largely separated although prior works have shown that generative training is beneficial to classifiers to alleviate several notorious issues. Energy-based Model (EBM) especially Joint Energy-based Model (JEM) only needs to train a single network with shared features for discriminative and generative tasks. However, EBMs are expensive to train, and very unstable. It is crucial to understand the behaviour of EBM training and thus improve the stability, speed, accuracy, and generative quality altogether.

This dissertation mainly summarizes my research on EBMs for Hybrid Image Discriminative-Generative Models. We first proposed GMMC which models the joint density $p_\theta(x, y)$. As an alternative to the SoftMax classifier utilized in JEM, GMMC has a well-formulated latent feature distribution, which fits well with the generative process of image synthesis. Then we came up with a variety of new training techniques to improve JEM’s accuracy, training stability, and speed altogether, and we named it JEM++. Based on JEM++, we analyzed and improved it from three different aspects, 1) the manifold, 2) the data augmentation, 3) the energy landscape. Hence, we propose Manifold-Aware EBM/JEM and Sharpness-Aware JEM to further improve the speed, generation quality, stability, and classification significantly. Beyond MCMC-based EBM, we found we can combine two recent emergent approaches Vision Transformer (ViT) and Denoising Diffusion Probabilistic Model (DDPM) to learn a simple but powerful model for image classification and generation. The new direction can get rid of most disadvantages of EBM, such as the expensive MCMC sampling and instability. Finally, we discuss future research topics including the speed, generation quality, and the applications of hybrid models.
INDEX WORDS: Energy-based Model, Generative Model, Discriminative Model, Hybrid Model, Vision Transformer, Denoising Diffusion Model.
Hybrid Energy-based Models for Image Generation and Classification

by

Xiulong Yang

Committee Chair: Shihao Ji
Committee: Rajshekhar Sunderraman
Xiaojing Ye
Daniel Takabi

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
December 2022
DEDICATION

This dissertation is dedicated to my parents Shengmin Yang and Yinxian Kong for their endless support during my Ph.D. years. I cannot finish my Ph.D. without their encouragement and support.
ACKNOWLEDGMENTS

First, I would like to express the gratitude to my advisor Dr. Jonathon Shihao Ji for the support of the Ph.D. study, for his knowledge and patience. His guidance helped a lot in the time of research and writing during the Ph.D. study.

I would like to thank my thesis committee: Dr. Rajshekhar Sunderraman, Dr. Xiaojing Ye and Dr. Daniel Takabi, for their insightful comments, but also for the hard question which encouraged me to deepen and widen my research.

My sincere thanks also goes to professors and colleagues in my group and department. I would like to show my great gratitude to Dr. Yingshu Li, Ms. Tammie Dudley, Ms. Tyra Robinson, and Mr. Paul Bryan, for their patient help, which makes my study at our department much easier and more convenient.

Thanks for all my group colleagues Xiang Li, Yang Ye, Qin Su, Hui Ye, and Leo Yang Li, and all roommates Ke Yan, Peng Wang, Yixian Chen, and Dr. Kiril Kuzmin. Without their precious support it would not be possible to conduct this research.

Special thank to the special friend Zhenzhen.

I would like to thank everybody who made the dissertation possible, as well as express my apologies that I could not mention personally one by one.
TABLE OF CONTENTS

ACKNOWLEDGMENTS ................................................................. v
LIST OF TABLES ................................................................. xii
LIST OF FIGURES ............................................................... xv

1 INTRODUCTION ................................................................. 1
  1.1 Motivation ................................................................. 1
  1.2 Dissertation Organization ................................................. 4
  1.3 List of Publications ..................................................... 5

2 BACKGROUND ................................................................. 7
  2.1 Energy-based Models ..................................................... 7
  2.2 Regularization and Generalization ...................................... 8
  2.3 Score Matching and Diffusion Denoising Model ....................... 9

3 GENERATIVE MAX-MAHALANOBIS CLASSIFIERS FOR IMAGE CLASSIFICATION, GENERATION AND MORE ........................................... 11
  3.1 Abstract ............................................................... 11
  3.2 Introduction ........................................................... 12
  3.3 Background and Related Work ........................................... 15
    3.3.1 Energy-based Models ............................................. 15
    3.3.2 Alternatives to the Softmax Classifier ......................... 16
  3.4 Methodology ............................................................ 17
    3.4.1 Approach 1: Discriminative Training ........................... 18
    3.4.2 Approach 2: Generative Training ................................ 20
    3.4.3 Approach 3: Joint Training ...................................... 23
    3.4.4 GMMC for Inference ............................................. 24
3.5 Experiments ...................................................... 24
   3.5.1 Hybrid Modeling ........................................... 24
   3.5.2 Calibration ................................................ 26
   3.5.3 Out-Of-Distribution Detection ............................ 27
   3.5.4 Robustness ................................................ 29
   3.5.5 Training Stability ......................................... 32
   3.5.6 Joint Training .............................................. 32

3.6 Conclusion and Future work ..................................... 33

4 JEM++: IMPROVED TECHNIQUES FOR TRAINING JEM ............. 34
   4.1 Abstract ..................................................... 34
   4.2 Introduction ................................................ 34
   4.3 Energy-Based Models ......................................... 38
   4.4 JEM++: The Improved Training of JEM ......................... 40
      4.4.1 Training EBM as a Minimax Optimization ............. 41
      4.4.2 Proximal SGLD .......................................... 43
      4.4.3 Training EBM as a Differential Game .................. 44
      4.4.4 Informative Initialization ............................... 47
      4.4.5 Batch Normalization and Learning Rate ................ 49
   4.5 Experiments .................................................. 51
      4.5.1 Hybrid Modeling ......................................... 51
      4.5.2 Training Stability and Speed ............................ 53
      4.5.3 Ablation Study ........................................... 57
      4.5.4 Classification Accuracy vs. Image Quality ............ 58
   4.6 Applications ................................................ 59
      4.6.1 Robustness .............................................. 59
      4.6.2 Calibration .............................................. 61
      4.6.3 Out-Of-Distribution Detection .......................... 63
   4.7 Conclusion .................................................. 64
### MA-EBM: TOWARDS UNDERSTANDING THE MANIFOLD OF ENERGY-BASED MODELS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Abstract</td>
<td>66</td>
</tr>
<tr>
<td>5.2</td>
<td>Introduction</td>
<td>66</td>
</tr>
<tr>
<td>5.3</td>
<td>Background</td>
<td>70</td>
</tr>
<tr>
<td>5.4</td>
<td>Method</td>
<td>73</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Informative Initialization and MA-EBM</td>
<td>73</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Energy Function Regularization</td>
<td>75</td>
</tr>
<tr>
<td>5.4.3</td>
<td>MA-JEM</td>
<td>76</td>
</tr>
<tr>
<td>5.5</td>
<td>Experiments</td>
<td>77</td>
</tr>
<tr>
<td>5.5.1</td>
<td>MA-EBM</td>
<td>79</td>
</tr>
<tr>
<td>5.5.2</td>
<td>MA-JEM</td>
<td>81</td>
</tr>
<tr>
<td>5.5.3</td>
<td>Analysis</td>
<td>82</td>
</tr>
<tr>
<td>5.5.4</td>
<td>Training Speed</td>
<td>84</td>
</tr>
<tr>
<td>5.6</td>
<td>Conclusion</td>
<td>86</td>
</tr>
</tbody>
</table>

### SHARPNESS-AWARE JOINT ENERGY-BASED MODELS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Abstract</td>
<td>88</td>
</tr>
<tr>
<td>6.2</td>
<td>Introduction</td>
<td>88</td>
</tr>
<tr>
<td>6.3</td>
<td>Related Work</td>
<td>91</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Energy-Based Models</td>
<td>91</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Joint Energy-based Models</td>
<td>93</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Flat Minima and Generalization</td>
<td>94</td>
</tr>
<tr>
<td>6.4</td>
<td>SAJEM</td>
<td>95</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Sharpness-Aware Minimization</td>
<td>95</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Image Generation without Data Augmentation</td>
<td>96</td>
</tr>
<tr>
<td>6.5</td>
<td>Experiments</td>
<td>98</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Hybrid Modeling</td>
<td>99</td>
</tr>
<tr>
<td>6.5.2</td>
<td>Calibration</td>
<td>101</td>
</tr>
</tbody>
</table>
6.5.3 Out-Of-Distribution Detection ........................................ 103
6.5.4 Robustness ............................................................... 104
6.5.5 Mode Coverage .......................................................... 106
6.5.6 Ablation Study ............................................................ 108

6.6 Limitations ................................................................. 109
6.7 Conclusion ................................................................. 109

7 YOUR VIT IS SECRETLY A HYBRID DISCRIMINATIVE-GENERATIVE DIF-
FUSION MODEL ............................................................. 110

7.1 Abstract ........................................................................ 110
7.2 Introduction .................................................................... 110
7.3 Related Work .................................................................. 113
  7.3.1 Denoising Diffusion Probabilistic Models ...................... 113
7.4 Vision Transformers ....................................................... 116
  7.4.1 Hybrid models ............................................................ 118
  7.4.2 Energy-Based Models .................................................. 119
7.5 Method .......................................................................... 119
  7.5.1 A Single ViT is a Generative Model .............................. 119
  7.5.2 ViT as a Hybrid Model ............................................... 120
7.6 Experiments .................................................................... 121
  7.6.1 Hybrid Modeling .......................................................... 123
  7.6.2 Model Evaluation ......................................................... 126
  7.6.3 Ablation Study ............................................................ 130
7.7 Limitations ..................................................................... 133
7.8 Conclusion ...................................................................... 134

8 FUTURE WORK .................................................................. 135

8.1 Theoretical Analysis of JEM ........................................... 135
8.2 Improved ViT and Diffusion Model ................................. 136
E YOUR VIT IS SECRETLY A HYBRID DISCRIMINATIVE-GENERATIVE DIFFUSION MODEL ............................ 172

A Image Datasets ................................................................. 172

B Model Evaluation ............................................................. 173
   B.1 Qualitative Analysis of Samples ................................. 173
   B.2 Data Augmentation ..................................................... 175
   B.3 Out-of-Distribution Detection ..................................... 176
   B.4 Robustness ............................................................... 176
   B.5 Calibration .............................................................. 177
   B.6 Training Speed ......................................................... 177

C Additional Generated Samples ....................................... 178

REFERENCES ................................................................. 184
LIST OF TABLES

Table 3.1 Hybrid Modeling Results on CIFAR10. ............................ 25
Table 3.2 Test Accuracy (%) on SVHN and CIFAR100. .................... 25
Table 3.3 OOD Detection Results. Models are trained on CIFAR10. Values are AUROC. 29
Table 3.4 Histograms of $\log \theta p(x)$ for OOD detection. Green corresponds to in-distribution dataset, while red corresponds to OOD dataset. ................................. 29
Table 3.5 Classification accuracies when models are under $L_{\infty}$ PGD attack with different $\epsilon$’s. All models are trained on CIFAR10. ......................... 30
Table 3.6 Example adversarial noises generated from the GMMC (Gen) model under C&W attack on CIFAR10. ................................. 32
Table 3.7 $L_2$ norms of adversarial perturbations under C&W attack on CIFAR10. .... 32
Table 4.1 Characteristics of different EBM training methods. .............. 37
Table 4.2 Hybrid Modeling Results on CIFAR10. ............................ 52
Table 4.3 Test Accuracy (%) on SVHN and CIFAR100. .................... 52
Table 4.4 Run-time comparison of JEM and JEM++ on CIFAR10. ........... 56
Table 4.5 Ablation study of different components of JEM++. All the models are trained on CIFAR10 with $M = 10$ and $N = 5$. ................................. 58
Table 4.6 OOD detection results. Models are trained on CIFAR10. Values are AUROC. 64
Table 4.7 Histograms of $\log \theta p(x)$ for OOD detection. Green corresponds to in-distribution dataset, while red corresponds to OOD dataset. ................................. 65
Table 5.1 Inception and FID scores of MA-EBM on CIFAR10. .............. 79
Table 5.2 FID results of MA-EBM on CIFAR100, CelebA-HQ 128, and ImageNet 32x32. ....................................................... 80
Table 5.3 Hybrid Modeling Results on CIFAR10. ............................ 83
Table 5.4 Hybrid Modeling Results on CIFAR100. ............................ 84
Table 5.5  Run-time comparison on CIFAR10. We set 390 iterations as one epoch and the training epochs are 200 epochs. The batch size is 64 for all models except DiffuRecov and VAEBM. 87

Table 6.1  Results on CIFAR10. 99

Table 6.2  Hybrid Modeling Results on CIFAR100. 101

Table 6.3  Histograms of $\log_\theta p(\mathbf{x})$ for OOD detection. Green corresponds to in-distribution dataset, while red corresponds to OOD dataset. 102

Table 6.4  OOD detection results. Models are trained on CIFAR10. Values are AUROC. 103

Table 6.5  Ablation study of SAJEM. All the models are trained on CIFAR10 with $K = 10$. 108

Table 6.6  Ablation Study of $\rho$. All the models are trained on CIFAR10 with $K = 10$. 108

Table 7.1  Results on CIFAR10. 124

Table 7.2  Results on STL10, CelebA 128, Tiny-ImageNet, and ImageNet 32x32. 126

Table 7.3  OOD detection results. Models are trained on CIFAR10. Values are AUROC. 128

Table 7.4  NLL measured in bits/dim on CIFAR-10. 129

Table 7.5  Ablation study of epochs. 130

Table 7.6  Ablation Study on CIFAR10. The default configurations of HybViT are $\alpha = 100, \text{head}=12, \text{depth}=9, \text{dim}=384$. 131

Table 7.7  Ablation Study on STL10. All models are trained for 500 epochs. NoP means Number of Parameters. ps means Patch Size. 132

Table 7.8  Run-time comparison on CIFAR10. We set 1170 iterations as one epoch for all ViTs, and 390 for WRN-based models. All ViTs are trained for 500 epochs and WRN-based models are trained for 200 epochs. 133

Table A.1  Hyperparameters for generative training of GMMC 141

Table A.2  Calibration results on SVHN and CIFAR100. 143

Table A.3  The energy scores of real data and sampled data after switching from discriminative training to JEM on CIFAR10. 146
Table B.1  Hyperparameters of JEM++ for CIFAR10 . . . . . . . . . . . . . . . . . . . . . . . . . 152

Table C.1  Hyper-parameters of MA-EBM/JEM . . . . . . . . . . . . . . . . . . . . . . . . . . . . 160

Table D.1  Hyper-parameters of SAJEM for CIFAR10 and CIFAR100 . . . . . . . . . . . . . 164

Table D.2  Histograms of $\log_\theta p(x)$ for OOD detection. Green corresponds to in-distribution
dataset, while red corresponds to OOD dataset. . . . . . . . . . . . . . . . . . . . . . . . . . 167

Table E.1  Ablation Study of Data Augmentation on CIFAR10. . . . . . . . . . . . . . . . . . . 175

Table E.2  OOD detection results. Models are trained on CIFAR10. Values are AUROC. 176

Table E.3  Classification accuracies when models are under $L_\infty$ and $L_2$ PGD attacks
with different $\epsilon$’s. All models are trained on CIFAR10. . . . . . . . . . . . . . . . . . . . . 177

Table E.4  Run-time comparison on ImageNet 32x32. All experiments are performed
on a single GPU for 100 epochs. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 177
LIST OF FIGURES

Figure 3.1 t-SNE visualization of the latent feature spaces learned by different models trained on CIFAR10. .................................................. 12

Figure 3.2 Overview of GMMC for training and test, where the model can be trained discriminatively, generatively or jointly. \{\mu_1^*, \mu_2^*, \ldots, \mu_C^*\} are pre-designed according to MMD (Pang et al. 2018). Only \theta = \{\phi\} is learned from data. ............ 19

Figure 3.3 Generated CIFAR10 Samples. .......................................................... 26

Figure 3.4 Calibration Results on CIFAR10. The smaller ECE is, the better. ............ 27

Figure 4.1 Comparison between SGLD-\$K\$ and PYLD-M-N. ............................. 46

Figure 4.2 The categorical centers of CIFAR10. ................................................. 49

Figure 4.3 JEM++ generated CIFAR10 samples. ................................................. 54

Figure 4.4 JEM++ generated class-conditional samples of SVHN and CIFAR100. Each row corresponds to one class. ......................................................... 55

Figure 4.5 The learning curves of JEM++ trained on CIFAR10 with (1) Number of steps, and (2) the proximity constraint. ................................. 56

Figure 4.6 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. ................................. 59

Figure 4.7 Adversarial robustness under the PGD attacks. ................................. 60

Figure 4.8 Calibration results on CIFAR10. The smaller ECE is, the better. ............ 62

Figure 4.9 Calibration results on CIFAR100. The smaller ECE is, the better. ............ 62

Figure 5.1 Generated samples of CelebA-HQ 128x128 from our MA-EBM. .......... 67

Figure 5.2 The architecture of MA-JEM. A energy head \textbf{f}_e is augmented for energy magnitude regularization and two mini-batches are used for the training of classifier and the maximum likelihood estimate of EBM, respectively. ................. 75

Figure 5.3 MA-EBM generated samples of CIFAR10 and CIFAR100. ................. 81
Figure 5.4  MA-JEM generated samples of CIFAR10 and CIFAR100.  
Figure 5.5  Comparison between LSE-JEM and MA-JEM on CIFAR100. LSE-JEM diverges before 50 epochs while MA-JEM is more stable. LSE-JEM is JEM with our simplified initialization.

Figure 5.6  t-SNE visualization of the latent feature spaces learned by different models trained on CIFAR10. a) shows the manifolds from MA-JEM are denser than MA-EBM and better formulated than LSE-JEM. b) shows The manifolds with regularization are denser and separate $x^+$ and $x^0$.

Figure 6.1  Visualizing the energy landscapes (Li et al. 2018) of different models trained on CIFAR10. Please note the dramatic differences of y-axis scales, indicating that SAJEM identifies the smoothest local optimal among all methods considered.

Figure 6.2  Generated CIFAR10 samples from SAJEM.

Figure 6.3  Generated CIFAR100 samples from SAJEM.

Figure 6.4  Calibration results on CIFAR10. The smaller ECE is, the better.

Figure 6.5  Adversarial robustness under the PGD attacks.

Figure 6.6  SAJEM generated CIFAR10 samples.

Figure 7.1  A graphical model of diffusion process.

Figure 7.2  The backbone architecture for GenViT and HybViT. For generative modeling, $x_t$ with a time embedding of $t$ is fed into the model. For the classification task in HybViT, we compute logits from CLS with the input $x_0$.

Figure 7.3  The pipeline of HybViT.

Figure 7.4  GenViT Generated samples of CIFAR10 and CelebA 128.

Figure 7.5  HybViT Generated samples of CIFAR10 and STL10.

Figure 7.6  Calibration results on CIFAR10. The smaller ECE is, the better. However, ViTs are better calibrated.

Figure 7.7  Adversarial robustness under the PGD attacks.

Figure 8.1  The visualization of SGLD sampling process with JEM trained on CIFAR10.
Figure A.1  GMMC calibration results on SVHN and CIFAR100.  

Figure A.2  Validation accuracy curves of GMMC trained with discriminative, generative or joint training.  

Figure A.3  Generated CIFAR10 samples (unconditional) by GMMC with joint training.  
Compared to the samples from GMMC with generative training, these samples are more blurry.  

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

Figure A.5  GMMC generated class-conditional samples of Plane.  

Figure A.6  GMMC generated class-conditional samples of Car.  

Figure A.7  GMMC generated class-conditional samples of Bird.  

Figure A.8  GMMC generated class-conditional samples of Cat.  

Figure A.9  GMMC generated class-conditional samples of Deer.  

Figure A.10  GMMC generated class-conditional samples of Dog.  

Figure A.11  GMMC generated class-conditional samples of Frog.  

Figure A.12  GMMC generated class-conditional samples of Horse.  

Figure A.13  GMMC generated class-conditional samples of Ship.  

Figure A.14  GMMC generated class-conditional samples of Truck.  

Figure B.1  The categorical centers of SVHN and CIFAR100.  

Figure B.2  The categorical centers of and corresponding samples of CIFAR10.  

Figure B.3  JEM++ generated class-conditional samples of SVHN and CIFAR100. Each row corresponds to one class.  

Figure B.4  JEM++ generated class-conditional samples of Plane.  

Figure B.5  JEM++ generated class-conditional samples of Car.  

Figure B.6  JEM++ generated class-conditional samples of Bird.  

Figure B.7  JEM++ generated class-conditional samples of Cat.  

Figure B.8  JEM++ generated class-conditional samples of Deer.
Figure E.1  The evolution of HybViT’s classification accuracy, FID as a function of training epochs on CIFAR10 and ImageNet 32x32.  

Figure E.2  The comparison between samples with FID=40 and FID=20. The difference is visually imperceptible for human.  

Figure E.3  HybViT generated class-conditional (best and worst) samples. Each row corresponds to 1 class.  

Figure E.4  Histograms (oriented horizontally for easier visual alignment) of $x$ arranged by class for CIFAR10.  

Figure E.5  Calibration results on CIFAR10. The smaller ECE is, the better.  

Figure E.6  Generated Images  

Figure E.7  Generated Images  

Figure E.8  HybViT generated class-conditional samples of Plane  

Figure E.9  HybViT generated class-conditional samples of Car  

Figure E.10  HybViT generated class-conditional samples of Bird  

Figure E.11  HybViT generated class-conditional samples of Cat  

Figure E.12  HybViT generated class-conditional samples of Deer  

Figure E.13  HybViT generated class-conditional samples of Dog  

Figure E.14  HybViT generated class-conditional samples of Frog  

Figure E.15  HybViT generated class-conditional samples of Horse  

Figure E.16  HybViT generated class-conditional samples of Ship  

Figure E.17  HybViT generated class-conditional samples of Truck
CHAPTER 1
INTRODUCTION

1.1 Motivation

Deep neural networks (DNNs) have made significant achievements in various discriminative tasks and generative tasks, including image classification and high-quality generation (Krizhevsky and Hinton 2009; He et al. 2016; Brock et al. 2019a; Brown et al. 2020; Ho et al. 2020). However, discriminative models and generative models are generally trained and researched separately as two areas, while human can do both in one. Classification tasks are one of the most fundamental problems in machine learning and discriminative models are notorious to be exposed to several critical issues, such as adversarial examples (Goodfellow et al. 2015a), calibration of uncertainty (Guo et al. 2017a) and out-of-distribution detection (Hendrycks and Gimpel 2016). Existing works (e.g., (Chapelle et al. 2009; Dempster et al. 1977)) have shown that generative training is beneficial to discriminative models to alleviate those crucial flaws. Yet, most recent works on generative models focus primarily on qualitative sample quality (Brock et al. 2019b; Santurkar et al. 2019; Vahdat and Kautz 2020).

Recently, there is a flurry of interest in closing the performance gap between generative models and discriminative models (Du and Mordatch 2019; Grathwohl et al. 2020a; Gao et al. 2020; Ardizzone et al. 2020). One representative branch is to utilize the EBM learning. (Du and Mordatch 2019) summaries the advantages of EBM as 1) Simplicity and Stability, 2) Sharing of Statistical Strength, 3) Adaptive Computation Time, 4) Flexibility Of Generation, 5) Compositionality. Furthermore, JEM (Grathwohl et al. 2020a) reinterprets CNN classifiers as the energy-based models.
(EBMs) for image generation. Since the CNN classifier is the only trained model, which has a high compositionality, it is possible that a single trained CNN model may encompass the generative capabilities into the discriminative model without sacrificing its discriminative power. Their work realizes the potential of EBMs in hybrid modeling and achieve improved performances on discriminative and generative tasks, and ignites a series of follow-up works (Zhao et al. 2020; Grathwohl et al. 2020b; Gao et al. 2020).

Despite a number of desirable properties, two challenges remain for training EBMs on high-dimensional datasets. First, learning EBMs by maximum likelihood requires Markov Chain Monte Carlo (MCMC) to generate samples from the model, which can be extremely expensive. Second, if we use less MCMC steps, the training of EBMs can be very unstable and we still can’t figure out the reason and premonition of the instability. A consequential challenges is the accuracy gap between the JEM and standard softmax classifier.

Therefore, the goal of our work is to improve the stability, speed, accuracy, and generation quality of hybrid discriminative-generative models, especially the tradeoff between stability and speed. We improve JEM from several interesting aspects.

We investigates an LDA classifier for image classification and generation. In particular, the Max-Mahalanobis Classifier (MMC) (Pang et al. 2020b), a special case of LDA, fits our goal very well. We show that our Generative MMC (GMMC) can be trained discriminatively, generatively or jointly for image classification and generation. Extensive experiments on multiple datasets show that GMMC achieves state-of-the-art discriminative and generative performances, while outperforming JEM in calibration, adversarial robustness and out-of-distribution detection.
by a significant margin. the softmax classifier that JEM exploits is inherently discriminative and its latent feature space is not well formulated as probabilistic distributions, which may hinder its potential for image generation and incur training instability.

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 (Zhang et al. 2019) 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.

We further study and deal with some weaknesses in JEM++. 1) We propose a simplified informative initialization to initialize the SGLD chain, which stabilizes and accelerates the training of unconditional EBM and JEM. 2) We find adding an $L_2$-regularized energy head on top of a CNN feature extractor to represent an energy function stabilizes the training of JEM significantly. 3) We improve image generation quality of MA-JEM by using two mini-batches: one with data augmentation for classification, and the other one without data augmentation for maximum likelihood estimation of EBMs. We build a visualization to show the manifolds learned by different models.

Moreover, we incorporate the sharpness-aware minimization technique into JEM++ to reduce
the classification gap to a standard classifier. We investigate the energy landscapes of different models and find that JEM leads to the sharpest one, which potentially undermines the generalizability of trained models.

Finally, we propose a new type of generative model GenViT and hybrid model HybViT. By integrating ViT (Dosovitskiy et al. 2021) into Denoising Diffusion Model (Ho et al. 2020), they successfully remove the MCMC sampling during training and are absolutely stable. Our models show superior performances over prior state-of-the-arts in both generative and discriminative tasks. We believe it will be a promising direction for future work.

1.2 Dissertation Organization

The rest of the dissertation is organized as follows: Chapter 2 summarizes the background and related literature. From Chapter 3 to Chapter 7, we introduce 5 proposed algorithms on hybrid energy-based models for image classification and generation. Chapter 3 introduces GMMC, an Max-Mahalanobis Classifier (MMC) for image classification and generation. Chapter 4 proposes JEM++, a variety of new training procedures and architecture features to improve JEM’s accuracy, training stability, and speed altogether. Chapter 5 proposes Manifold-Aware EBM and JEM to deal with some weaknesses in JEM++ and build a visualization to show the manifolds learned by different models. Chapter 6 further incorporate the sharpness-aware minimization technique into JEM++ to reduce the classification gap to a standard classifier. Beyond the scope, we propose GenViT and HybViT by building a connection between ViT and Diffusion Denoising Model in Chapter 7. Finally, we discuss about some challenges for our future work in Chapter 8 and
Chapter 9 concludes our work.

1.3 List of Publications


CHAPTER 2  
BACKGROUND

2.1 Energy-based Models

Energy-based models (EBMs) (LeCun et al. 2006) define an energy function that assigns low energy values to samples from data distribution and high values otherwise, and then any probability density $p_\theta(x)$ can be expressed via a Boltzmann distribution as

$$p_\theta(x) = \frac{\exp \left( -E_\theta(x) \right)}{Z(\theta)}, \tag{2.1}$$

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

The key challenge is estimating the intractable partition function $Z(\theta)$. The standard maximum likelihood estimation of parameters $\theta$ is not straightforward, and a number of sampling-based approaches have been proposed to approximate it effectively. Specifically, the gradient of the log-likelihood of a single sample $x$ w.r.t. $\theta$ can be expressed as

$$\frac{\partial \log p_\theta(x)}{\partial \theta} = \mathbb{E}_{p_\theta(x')} \frac{\partial E_\theta(x')}{\partial \theta} - \frac{\partial E_\theta(x)}{\partial \theta}, \tag{2.2}$$

where the expectation is over model distribution $p_\theta(x')$, sampling from which is challenging due to the intractable $Z(\theta)$. Therefore, MCMC and Gibbs sampling (Hinton 2002a) have been proposed previously to estimate the expectation efficiently. To speed up the mixing for effective sampling,
recently Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh 2011) has been used to train EBMs by exploiting the gradient information (Nijkamp et al. 2019b; Du and Mordatch 2019; Grathwohl et al. 2020a). Specifically, to sample from $p_\theta(x)$, SGLD follows

$$x_0 \sim p_0(x), \quad x_{i+1} = x_i - \frac{\alpha}{2} \frac{\partial E_\theta(x_i)}{\partial x_i} + \alpha \epsilon, \quad \epsilon \sim N(0, 1),$$

where $p_0(x)$ is typically a uniform distribution over $[-1, 1]$, whose samples are refined via noisy gradient decent with step-size $\alpha$, which should be decayed following a polynomial schedule.

Besides JEM (Grathwohl et al. 2020a) that we discussed in the introduction, (Xie et al. 2016) is an earlier work that derives a generative CNN model from the classical CNN architecture by treating it as an EBM, where the authors factorize the loss function $\log p(x|y)$ as an EBM. Following-up works, such as (Nijkamp et al. 2019a; Du and Mordatch 2019), scale the training of EBMs to high-dimensional data using SGLD.

### 2.2 Regularization and Generalization

To improve the generalization of trained DNNs on unseen data examples, a variety of regularization techniques have been proposed in the past few years, such as dropout (Srivastava et al. 2014), weight decay (Hinton and van Camp 1993), batch norm (Ioffe and Szegedy 2015), spectral norm (Miyato et al. 2018), weight standardization (Salimans and Kingma 2016; Qiao et al. 2020; Huang et al. 2017), and gradient clipping (Zhang et al. 2020).

IGEBM and JEM (Du and Mordatch 2019; Grathwohl et al. 2020a) make a huge effort on trying to improve the stability of EBMs training with different regularization techniques and heuristic
approaches such as gradient clipping, energy clipping, and ignoring examples with atypical energy values. They provide exhaustive investigation results and find none of them can help to improve the stability with reduced number of sampling steps and the key factor to improve the stability is the sampling steps.

A great number of previous works have investigated the generalizability of learned models (Li et al. 2018; Keskar et al. 2017; Wei et al. 2020; Chen et al. 2022; Foret et al. 2021; Kwon et al. 2021). It is widely accepted that flat minima tends to give better generalization performance. We also adopt this idea to investigate JEM and propose to explore the sharpness-aware minimization (SAM) (Foret et al. 2021) for JEM.

2.3 Score Matching and Diffusion Denoising Model

These is a connection between score matching and EBM. Both learn unnormalized statistical models. In particular, Score matching-based methods (Hyvärinen 2005a; Swersky et al. 2011; Song and Ermon 2019; Song et al. 2020) learn unnormalized statistical models by matching the gradient of the log probability density of the model distribution to that of the data distribution. An example method of score matching is NCSN (Song and Ermon 2019).

Another branch is the Diffusion Models. DDPMs (Ho et al. 2020) optimize a variational lower bound to the log-likelihood, whereas NCSNs optimize the score matching objective (Hyvärinen 2005b) over a non-parametric Parzen density estimator of the data (Vincent 2011; Raphan and Simoncelli 2011). Despite their different motivations, DDPMs and NCSNs are closely related. Both use a denoising autoencoder objective for many noise levels, and both use a procedure similar
to Langevin dynamics to produce samples.

In our work, we utilize the pipeline of DDPM to train the GenViT. It’s not hard to switch to a score matching-based method.
CHAPTER 3
GENERATIVE MAX-MAHALANOBIS CLASSIFIERS FOR IMAGE CLASSIFICATION, GENERATION AND MORE

3.1 Abstract

Joint Energy-based Model (JEM) of (Grathwohl et al. 2020a) shows that a standard softmax classifier can be reinterpreted as an energy-based model (EBM) for the joint distribution $p(x, y)$; the resulting model can be optimized to improve calibration, robustness and out-of-distribution detection, while generating samples rivaling the quality of recent GAN-based approaches. However, the softmax classifier that JEM exploits is inherently discriminative and its latent feature space is not well formulated as probabilistic distributions, which may hinder its potential for image generation and incur training instability. We hypothesize that generative classifiers, such as Linear Discriminant Analysis (LDA), might be more suitable for image generation since generative classifiers model the data generation process explicitly. This paper therefore investigates an LDA classifier for image classification and generation. In particular, the Max-Mahalanobis Classifier (MMC) (Pang et al. 2020b), a special case of LDA, fits our goal very well. We show that our Generative MMC (GMMC) can be trained discriminatively, generatively or jointly for image classification and generation. Extensive experiments on multiple datasets show that GMMC achieves state-of-the-art discriminative and generative performances, while outperforming JEM in calibration, adversarial robustness and out-of-distribution detection by a significant margin. Our source code is available at https://github.com/sndnyang/GMMC.
Figure 3.1 t-SNE visualization of the latent feature spaces learned by different models trained on CIFAR10.

3.2 Introduction

Over the past few years, deep neural networks (DNNs) have achieved state-of-the-art performance on a wide range of learning tasks, such as image classification, object detection, segmentation and image captioning (Krizhevsky and Hinton 2009; He et al. 2016). All of these breakthroughs, however, are achieved in the framework of discriminative models, which are known to be exposed to several critical issues, such as adversarial examples (Goodfellow et al. 2015a), calibration of uncertainty (Guo et al. 2017a) and out-of-distribution detection (Hendrycks and Gimpel 2016).

Prior works have shown that generative training is beneficial to these models and can alleviate some of these issues at certain levels (Chapelle et al. 2009; Dempster et al. 1977). Yet, most recent research on generative models focus primarily on qualitative sample quality (Brock et al. 2019b; Santurkar et al. 2019; Vahdat and Kautz 2020), and the discriminative performances of state-of-the-art generative models are still far behind discriminative ones (Behrmann et al. 2018; Chen et al. 2019; Du and Mordatch 2019).

Recently, there is a flurry of interest in closing the performance gap between generative models and discriminative models (Du and Mordatch 2019; Grathwohl et al. 2020a; Gao et al. 2020;
Ardizzone et al. 2020). Among them, IGEBM (Du and Mordatch 2019) and JEM (Grathwohl et al. 2020a) are the two most representative ones, which reinterpret CNN classifiers as the energy-based models (EBMs) for image generation. Since the CNN classifier is the only trained model, which has a high compositionality, it is possible that a single trained CNN model may encompass the generative capabilities into the discriminative model without sacrificing its discriminative power. Their works realize the potential of EBMs in hybrid modeling and achieve improved performances on discriminative and generative tasks. Specifically, JEM (Grathwohl et al. 2020a) reinterprets the standard softmax classifier as an EBM and achieves impressive performances in image classification and generation simultaneously, and ignites a series of follow-up works (Zhao et al. 2020; Grathwohl et al. 2020b; Gao et al. 2020).

However, the softmax classifier that JEM exploits is inherently discriminative, which may hinder its potential in image generation. To investigate this, we visualize the latent feature spaces learnt by a standard softmax classifier and by JEM through t-SNE (Maaten and Hinton 2008) in Figs. 3.1a and 3.1b, respectively. Apparently, the feature space of the softmax classifier has been improved significantly by JEM as manifested by higher inter-class separability and intra-class compactness. However, JEM’s latent space is not well formulated as probabilistic distributions, which may limit its generative performance and incur training instability as observed in (Grathwohl et al. 2020a). We hypothesize that generative classifiers (e.g., LDA) might be more suitable for image classification and generation. This is because generative classifiers model the data generation process explicitly with probabilistic distributions, such as mixture of Gaussians, which aligns well with the generative process of image synthesis. Therefore, in this paper we investigate an LDA
classifier for image classification and generation. In particular, the Max-Mahalanobis Classifier (MMC) (Pang et al. 2020b), a special case of LDA, fits our goal very well since MMC formulates the latent feature space explicitly as the Max-Mahalanobis distribution (Pang et al. 2018). Distinct to (Pang et al. 2020b), we show that MMC can be trained discriminatively, generatively or jointly as an EBM. We term our algorithm Generative MMC (GMMC) given that it is a hybrid model for image classification and generation, while the original MMC (Pang et al. 2020b) is only for classification.

As a comparison, Figs. 3.1c and 3.1d illustrate the latent feature spaces of GMMC optimized with discriminative training and generative training (to be discussed in Sec. 3.4), respectively. It can be observed that the latent feature spaces of GMMC are improved even further over that of JEM’s with higher inter-class separability and intra-class compactness. Furthermore, the explicit generative modeling of GMMC leads to many auxiliary benefits, such as adversarial robustness, calibration of uncertainty and out-of-distribution detection, which will be demonstrated in our experiments. Our main contributions can be summarized as follows:

1. We introduce GMMC, a hybrid model for image classification and generation. As an alternative to the softmax classifier utilized in JEM, GMMC has a well-formulated latent feature distribution, which fits well with the generative process of image synthesis.

2. We show that GMMC can be trained discriminatively, generatively or jointly with reduced complexity and improved stability as compared to JEM.

3. Our model matches or outperforms prior state-of-the-art hybrid models on multiple discriminative and generative tasks, including image classification, image synthesis, calibration of
uncertainty, out-of-distribution detection and adversarial robustness.

3.3 Background and Related Work

3.3.1 Energy-based Models

Energy-based models (EBMs) (LeCun et al. 2006) define an energy function that assigns low energy values to samples drawn from data distribution and high values otherwise, such that any probability density \( p_\theta(x) \) can be expressed via a Boltzmann distribution as

\[
p_\theta(x) = \exp\left(-E_\theta(x)\right) / Z(\theta),
\]

(3.1)

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

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

\[
\frac{\partial \log p_\theta(x)}{\partial \theta} = \mathbb{E}_{p_\theta(x')} \frac{\partial E_\theta(x')}{\partial \theta} - \frac{\partial E_\theta(x)}{\partial \theta},
\]

(3.2)

where the expectation is over model distribution \( p_\theta(x') \), sampling from which is challenging due to the intractable \( Z(\theta) \). Therefore, MCMC and Gibbs sampling (Hinton 2002a) have been proposed...
previously to estimate the expectation efficiently. To speed up the mixing for effective sampling, recently Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh 2011) has been used to train EBMs by exploiting the gradient information (Nijkamp et al. 2019b; Du and Mordatch 2019; Grathwohl et al. 2020a). Specifically, to sample from \( p_\theta(x) \), SGLD follows

\[
x_0 \sim p_0(x), \quad x_{i+1} = x_i - \alpha \frac{\partial E_\theta(x_i)}{\partial x_i} + \alpha \epsilon, \quad \epsilon \sim N(0,1), \quad (3.3)
\]

where \( p_0(x) \) is typically a uniform distribution over \([-1, 1]\), whose samples are refined via noisy gradient decent with step-size \( \alpha \), which should be decayed following a polynomial schedule.

Besides JEM (Grathwohl et al. 2020a) that we discussed in the introduction, (Xie et al. 2016) is an earlier work that derives a generative CNN model from the commonly used discriminative CNN by treating it as an EBM, where the authors factorize the loss function \( \log p(x|y) \) as an EBM. Following-up works, such as (Nijkamp et al. 2019a; Du and Mordatch 2019), scale the training of EBMs to high-dimensional data using SGLD. However, all of these previous methods define \( p(x|y) \) or \( p(x) \) as an EBM, while our GMMC defines an EBM on \( p(x, y) \) by following a mixture of Gaussian distribution, which simplifies the maximum likelihood estimation and achieves improved performances in many discriminative and generative tasks.

### 3.3.2 Alternatives to the Softmax Classifier

Softmax classifier has been widely used in state-of-the-art models for discriminative tasks due to its simplicity and efficiency. However, softmax classifier is known particularly vulnerable to adversarial attacks because the latent feature space induced by softmax classifier is typically not
well separated (as shown in Fig. 3.1(a)). Some recent works propose to use generative classifiers to better formulate the latent space distributions in order to improve its robustness to adversarial examples. For example, Wan et al. (Wan et al. 2018) propose to model the latent feature space as mixture of Gaussians and encourages stronger intra-class compactness and larger inter-class separability by introducing large margins between classes. Different from (Wan et al. 2018), Pang et al. (Pang et al. 2018) pre-design the centroids based on the Max-Mahalanobis distribution (MMD), other than learning them from data. The authors prove that if the latent feature space distributes as an MMD, the LDA classifier will have the best robustness to adversarial examples. Taking advantage of the benefits of MMD, Pang et al. (Pang et al. 2020b) further propose a max-Mahalanobis center regression loss, which induces much denser feature regions and improves the robustness of trained models. Compared with softmax classifier, all these works can generate better latent feature spaces to improve the robustness of models for the task of classification. Our GMMC is built on the basic framework of MMC, but we reinterpret MMC as an EBM for image classification and generation. Moreover, we show that the generative training of MMC can further improve calibration, adversarial robustness and out-of-distribution detection.

3.4 Methodology

We assume a Linear Discriminant Analysis (LDA) classifier is defined as: $\phi(x), \mu = \{\mu_y, y = 1, 2, \cdots, C\}$ and $\pi = \{\pi_y = \frac{1}{C}, y = 1, 2, \cdots, C\}$ for $C$-class classification, where $\phi(x) \in \mathbb{R}^d$ is the feature representation of $x$ extracted by a CNN, parameterized by $\phi^1$, and $\mu_y \in \mathbb{R}^d$ is the mean

\footnote{To avoid notational clutter in later derivations, we use $\phi$ to denote a CNN feature extractor and its parameter. But the meaning of $\phi$ is clear given the context.}
of a Gaussians distribution with a diagonal covariance matrix $\gamma^2 I$, i.e., $p_\theta(\phi(x)|y) = N(\mu_y, \gamma^2 I)$. Therefore, we can parameterize LDA by $\theta = \{\phi, \mu\}^2$. Instead of using this regular LDA classifier, in this paper the max-Mahalanobis classifier (MMC) (Pang et al. 2020b), a special case of LDA, is considered. Different from the LDA modeling above, in MMC $\mu = \{\mu_y, y = 1, 2, \ldots, C\}$ is pre-designed to induce compact feature representations for model robustness. We found that the MMC modeling fits our goal better than the regular LDA classifier due to its improved training stability and boosted adversarial robustness. Therefore, in the following we focus on the MMC modeling for image classification and generation. As such, the learnable parameters of MMC reduce to $\theta = \{\phi\}$, and the pseudo-code of calculating pre-designed $\mu$ can be found in Algorithm 2 of the appendix C. Fig. 3.2 provides an overview of the training and test of our GMMC algorithm, with the details discussed below.

Instead of maximizing $p_\theta(y|x)$ as in standard softmax classifier, following JEM (Grathwohl et al. 2020a) we maximize the joint distribution $p_\theta(x, y)$, which follows a mixture of Gaussians distribution in GMMC. To optimize $\log p_\theta(x, y)$, we can consider three different approaches.

### 3.4.1 Approach 1: Discriminative Training

According to the MMC modeling above, the joint distribution $p_\theta(x, y)$ can be expressed as

$$p_\theta(x, y) = p(y)p_\theta(x|y) \propto \frac{1}{C}(2\pi \gamma^2)^{-d/2} \exp\left(-\frac{1}{2\gamma^2}||\phi(x) - \mu_y||_2^2\right)$$

$$= \frac{\exp\left(-\frac{1}{2\gamma^2}||\phi(x) - \mu_y||_2^2\right)}{Z(\theta)} = \frac{\exp(-E_\theta(x, y))}{Z(\theta)} \tag{3.4}$$

\(^2\text{We can treat } \gamma \text{ as a tunable hyperparameter or we can estimate it by post-processing. In this work, we take the latter approach as discussed in Sec. 3.4.1.}\)
Figure 3.2 Overview of GMMC for training and test, where the model can be trained discriminatively, generatively or jointly. \{\mathbf{\mu}_1^*, \mathbf{\mu}_2^*, \cdots, \mathbf{\mu}_C^*\} are pre-designed according to MMD (Pang et al. 2018). Only \(\theta = \{\phi\}\) is learned from data.

where we define \(E_\theta(x, y) = \frac{1}{2\gamma^2}||\phi(x) - \mathbf{\mu}_y||_2^2\), and \(Z(\theta) = \int \exp(-E_\theta(x, y))dx\,dy\), which is an intractable partition function. To avoid the expense of evaluating the partition function, we follow Mnih and Teh (Mnih and Teh 2012) and approximate \(Z(\theta)\) as a constant (e.g., \(Z(\theta) = 1\)). This turns out to be an effective approximation for neural networks with lots of parameters as it encourages the model to have “self-normalized” outputs. With this approximation, the log of the joint distribution can be simplified as

\[
\log p_\theta(x, y) = -\frac{1}{2\gamma^2}||\phi(x) - \mathbf{\mu}_y||_2^2 - \log Z(\theta) \\
\approx -E_\theta(x, y) + \text{constant.} \tag{3.5}
\]
To optimize the parameters $\theta$, we can simply compute gradient of Eq. 3.5 w.r.t. $\theta$, and update the parameters by stochastic gradient descent (SGD) (Robbins and Monro 1951). Note that $\gamma$ is a constant in Eq. 3.5, its effect can be absorbed into the learning rate when optimizing Eq. 3.5 via SGD. After convergence, we can estimate $\gamma^2 = \frac{1}{d} \left( \frac{1}{N} \sum_{i=1}^{N} ||\phi(x_i) - \mu_i||_2^2 \right)$ from training set by using optimized $\phi$ and pre-designed $\mu$.

Note that Eq. 3.5 boils down to the same objective that MMC (Pang et al. 2020b) proposes, i.e. the center regression loss. While MMC reaches to this objective from the perspective of inducing compact feature representations for model robustness, we arrive at this objective by simply following the principle of maximum likelihood estimation of model parameter $\theta$ of joint density $p_\theta(x, y)$ with the “self-normalization” approximation (Mnih and Teh 2012).

### 3.4.2 Approach 2: Generative Training

Comparing Eq. 3.4 with the definition of EBM (3.1), we can also treat the joint density

$$p_\theta(x, y) = \frac{\exp\left(-\frac{1}{2\gamma^2}||\phi(x) - \mu_y||_2^2\right)}{Z(\theta)}$$

as an EBM with $E_\theta(x, y) = \frac{1}{2\gamma^2}||\phi(x) - \mu_y||_2^2$ defined as an energy function of $(x, y)$.

Following the maximum likelihood training of EBM (3.2), to optimize Eq. 3.6, we can compute its gradient w.r.t. $\theta$ as

$$\frac{\partial \log p_\theta(x, y)}{\partial \theta} = \beta E_{p_\theta(x', y')} \frac{\partial E_\theta(x', y')}{\partial \theta} - \frac{\partial E_\theta(x, y)}{\partial \theta},$$
where the expectation is over the joint density \( p_\theta(x', y') \), sampling from which is challenging due to the intractable \( Z(\theta) \), and \( \beta \) is a hyperparameter that balances the contributions from the two terms. Different value of \( \beta \) has a significant impact to the performance. From our experiments, we find that \( \beta = 0.5 \) works very well in all our cases. Therefore, we set \( \beta = 0.5 \) as the default value.

Notably, the two terms of our GMMC (Eq. 3.7) are both defined on the same energy function \( E_\theta(x, y) \), while the two terms of JEM (Grathwohl et al. 2020a) are defined on \( p(x) \) and \( p(y|x) \), respectively, which are computationally more expensive and might incur training instability as we will discuss in Sec. 3.4.3.

The tricky part is how to generate samples \((x', y') \sim p_\theta(x', y')\) to estimate the first term of Eq. 3.7. We can follow the mixture of Gaussians assumption of MMC. That is, \( p_\theta(x', y') = p(y')p_\theta(x'|y') \): (1) sample \( y' \sim p(y') = \frac{1}{C} \), and then (2) sample \( x' \sim p_\theta(x'|y') \propto \mathcal{N}(\mu_{y'}, \gamma^2 I) \). To sample \( x' \), again we can consider two choices.

**(1) Staged Sampling**

We can first sample \( z_{x'} \sim \mathcal{N}(\mu_{y'}, \gamma^2 I) \), and then find an \( x' \) to minimize \( E_\theta(x') = \frac{1}{2\gamma^2} ||\phi(x') - z_{x'}||_2^2 \). This can be achieved by

\[
x_0' \sim p_0(x'), \quad x'_{t+1} = x'_t - \alpha \frac{\partial E_\theta(x'_t)}{\partial x'_t}, \quad (3.8)
\]

where \( p_0(x) \) is typically a uniform distribution over \([-1, 1]\). Note that this is similar to SGLD (see Eq. 3.3) but without a noisy term. Thus, the training could be more stable. In addition, the function \( E_\theta(x') = \frac{1}{2\gamma^2} ||\phi(x') - z_{x'}||_2^2 \) is just an \( L_2 \) regression loss (not an LogSumExp function as used in
(2) Noise Injected Sampling

We can first sample $z \sim N(0, I)$, then by the reparameterization trick we have $z' = \gamma z + \mu'$. Finally, we can find an $x'$ to minimize

$$E_{\theta}(x') = \frac{1}{2\gamma^2} ||\phi(x') - z'||_2^2 = \frac{1}{2\gamma^2} ||\phi(x') - \mu' - \gamma z||_2^2$$

$$= \frac{1}{2\gamma^2} ||\phi(x') - \mu'||_2^2 + \frac{1}{2} ||z||_2^2 - \frac{1}{\gamma^2} <\phi(x') - \mu', \gamma z>$$

$$= E_{\theta}(x', y') + \frac{1}{2} ||z||_2^2 - \frac{1}{\gamma} <\phi(x') - \mu', z>.$$  \hspace{1cm} (3.9)

This can be achieved by

$$x'_0 \sim p_0(x'),$$

$$x'_{t+1} = x'_t - \alpha \frac{\partial E_{\theta}(x'_t)}{\partial x'_t} = x'_t - \alpha \frac{\partial E_{\theta}(x'_t, y')}{\partial x'_t} + \alpha \frac{1}{\gamma} < \frac{\partial \phi(x'_t)}{\partial x'_t}, z >.$$  \hspace{1cm} (3.10)

where we sample a different $z$ at each iteration. As a result, Eq. 3.10 is an analogy of SGLD (see Eq. 3.3). The difference is instead of using an unit Gaussian noise $\epsilon \sim N(0, 1)$ as in SGLD, a gradient-modulated noise (the 3rd term) is applied.

Algorithm 1 provides the pseudo-code of the generative training of GMMC, which follows a similar design of IGEBM (Du and Mordatch 2019) and JEM (Grathwohl et al. 2020a) with a replay buffer $B$. For brevity, only one real sample $(x, y) \sim D$ and one generated sample $(x', y') \sim p_{\theta}(x', y')$ are used to optimize the parameter $\theta$. It is straightforward to generalize the pseudo-code...
Algorithm 1 Generative training of GMMC: Given model parameter $\theta = \{\phi\}$, step-size $\alpha$, replay buffer $B$, number of steps $\tau$, reinitialization frequency $\rho$

1: while not converged do
2: Sample $x$ and $y$ from dataset $\mathcal{D}$
3: Sample $(x'_0, y') \sim B$ with probability $1 - \rho$, else $x'_0 \sim \mathcal{U}(-1, 1)$, $y' \sim p(y') = \frac{1}{C}$
4: Sample $z_{x'} \sim \mathcal{N}(\mu_{y'}, \gamma^2 I)$ if staged sampling
5: for $t \in [1, 2, \ldots, \tau]$ do
6: Sample $z \sim \mathcal{N}(0, I)$, $z_{x'} = \mu_{y'} + \gamma z$ if noise injected sampling
7: $x'_t = x'_{t-1} - \alpha \frac{\partial E_\theta(x'_{t-1})}{\partial x'_{t-1}}$ (Eq. 3.8)
8: end for
9: Calculate gradient with Eq. 3.7 from $(x, y)$ and $(x'_{\tau}, y')$ for model update
10: Add / replace updated $(x'_{\tau}, y')$ back to $B$
11: end while

above to a mini-batch training, which is used in our experiments. Compared to JEM, GMMC needs no additional calculation of $p_\theta(y|x)$ and thus has reduced computational complexity.

3.4.3 Approach 3: Joint Training

Comparing Eq. 3.5 and Eq. 3.7, we note that the gradient of Eq. 3.5 is just the second term of Eq. 3.7. Hence, we can use (Approach 1) discriminative training to pretrain $\theta$, and then finetune $\theta$ by (Approach 2) generative training. The transition between the two can be achieved by scaling up $\beta$ from 0 to a predefined value (e.g., 0.5). Similar joint training strategy can be applied to train JEM as well. However, from our experiments we note that this joint training of JEM is extremely unstable. We hypothesize that this is likely because the two terms of JEM are defined on $p(x)$ and $p(y|x)$, respectively, while the two terms of our GMMC (Eq. 3.7) are defined on the same energy function $E_\theta(x, y)$. Hence, the learned model parameters from the two training stages are more compatible in GMMC than in JEM. We will demonstrate the training issues of JEM and GMMC
3.4.4 GMMC for Inference

After training with one of the three approaches discussed above, we get the optimized GMMC parameters $\theta = \{\phi\}$, the pre-designed $\mu$ and the estimated $\gamma^2 = \frac{1}{d} \left( \frac{1}{N} \sum_{i=1}^{N} ||\phi(x_i) - \mu_i||^2 \right)$ from training set. We can then calculate class probabilities for classification

$$p_{\theta}(y|x) = \frac{\exp\left(-\frac{1}{2\gamma^2}||\phi(x) - \mu_y||^2\right)}{\sum_{y'} \exp\left(-\frac{1}{2\gamma^2}||\phi(x) - \mu_{y'}||^2\right)}.$$ (3.11)

3.5 Experiments

We evaluate the performance of GMMC on multiple discriminative and generative tasks, including image classification, image generation, calibration of uncertainty, out-of-distribution detection and adversarial robustness. Since GMMC is inspired largely by JEM (Grathwohl et al. 2020a), for a fair comparison, our experiments closely follow the settings provided in the source code of JEM$^3$. All our experiments are performed with PyTorch on Nvidia RTX GPUs. Due to page limit, details of the experimental setup are relegated to Appendix B.

3.5.1 Hybrid Modeling

We train GMMC on three benchmark datasets: CIFAR10, CIFAR100 (Krizhevsky and Hinton 2009) and SVHN (Netzer et al. 2011), and compare it to the state-of-the-art hybrid models, as well as standalone generative and discriminative models. Following the settings of JEM, we use

$^3$https://github.com/wgrathwohl/JEM
the Wide-ResNet (Zagoruyko and Komodakis 2016) as the backbone CNN model for JEM and GMMC. To evaluate the quality of generated images, we adopt Inception Score (IS) (Salimans et al. 2016) and Fréchet Inception Distance (FID) (Heusel et al. 2017) as the evaluation metrics.

Table 3.1 Hybrid Modeling Results on CIFAR10.

<table>
<thead>
<tr>
<th>Class</th>
<th>Model</th>
<th>Acc % ↑</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid</td>
<td>Residual Flow</td>
<td>70.3</td>
<td>3.60</td>
<td>46.4</td>
</tr>
<tr>
<td></td>
<td>Glow</td>
<td>67.6</td>
<td>3.92</td>
<td>48.9</td>
</tr>
<tr>
<td></td>
<td>IGBM</td>
<td>49.1</td>
<td>8.30</td>
<td>37.9</td>
</tr>
<tr>
<td></td>
<td>JEM</td>
<td>92.9</td>
<td>8.76</td>
<td>38.4</td>
</tr>
<tr>
<td></td>
<td>GMMC (Ours)</td>
<td>94.08</td>
<td>7.24</td>
<td>37.0</td>
</tr>
<tr>
<td>Disc.</td>
<td>WRN w/ BN</td>
<td>95.8 N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>WRN w/o BN</td>
<td>93.6 N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>GMMC (Dis)</td>
<td>94.3 N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Gen.</td>
<td>SNGAN</td>
<td>N/A</td>
<td>8.59</td>
<td>25.5</td>
</tr>
<tr>
<td></td>
<td>NCSN</td>
<td>N/A</td>
<td>8.91</td>
<td>25.3</td>
</tr>
</tbody>
</table>

Table 3.2 Test Accuracy (%) on SVHN and CIFAR100.

<table>
<thead>
<tr>
<th>Model</th>
<th>SVHN</th>
<th>CIFAR100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax</td>
<td>96.6</td>
<td>72.6</td>
</tr>
<tr>
<td>JEM</td>
<td>96.7</td>
<td>72.2</td>
</tr>
<tr>
<td>GMMC (Dis)</td>
<td>97.1</td>
<td>75.4</td>
</tr>
<tr>
<td>GMMC (Gen)</td>
<td>97.2</td>
<td>73.9</td>
</tr>
</tbody>
</table>

The results on CIFAR10, CIFAR100 and SVHN are shown in Table 3.1 and Table 3.2, respectively. It can be observed that GMMC outperforms the state-of-the-art hybrid models in terms of accuracy (94.08%) and FID score (37.0), while being slightly worse in IS score. Since no IS and FID scores are commonly reported on SVHN and CIFAR100, we present the classification accuracies and generated samples on these two benchmarks. Our GMMC models achieve 97.2% and 73.9% accuracy on SVHN and CIFAR100, respectively, outperforming the softmax classifier and JEM by notable margins. Example images generated by GMMC for CIFAR10 are shown in Fig. 3.3. Additional GMMC generated images for CIFAR100 and SVHN can be found in Appendix F.
While modern deep models have grown more accurate in the past few years, recent researches have shown that their predictions could be over-confident (Guo et al. 2017a). Outputting an incorrect but confident decision can have catastrophic consequences. Hence, calibration of uncertainty for DNNs is a critical research topic. Here, the confidence is defined as $\max_y p(y|x)$ which is used to decide when to output a prediction. A well-calibrated, but less accurate model can be considerably more useful than a more accurate but less-calibrated model.

We train GMMC on the CIFAR10 dataset, and compare its Expected Calibration Error (ECE) score (Guo et al. 2017a) to that of the standard softmax classifier and JEM. Results are shown in Fig. 3.4 with additional results on SVHN and CIFAR100 provided in Appendix D.1. We find that the model trained by GMMC (Gen) achieves a much smaller ECE (1.33% vs. 4.18%), demonstrat-
ing GMMC’s predictions are better calibrated than the competing methods.

![Figure 3.4 Calibration Results on CIFAR10. The smaller ECE is, the better.](image)

### 3.5.3 Out-Of-Distribution Detection

The OOD detection is a binary classification problem, which outputs a score $s_\theta(x) \in \mathbb{R}$ for a given query $x$. The model should be able to assign lower scores to OOD examples than to in-distribution examples, such that it can be used to distinguish two sets of examples. Following the settings of JEM (Grathwohl et al. 2020a), we use the Area Under the Receiver-Operating Curve (AUROC) (Hendrycks and Gimpel 2016) to evaluate the performance of OOD detection. In our experiments, three score functions are considered: the input density $p_\theta(x)$ (Nalisnick et al. 2018), the predictive distribution $p_\theta(y|x)$ (Hendrycks and Gimpel 2016), and the approximate mass $\left\| \frac{\partial \log p_\theta(x)}{\partial x} \right\|$ (Grathwohl et al. 2020a).

#### (1) Input Density

A natural choice of $s_\theta(x)$ is the input density $p_\theta(x)$. For OOD detection, intuitively we consider examples with low $p(x)$ to be OOD. Quantitative results can be found in Table 3.3 (top). The corresponding distributions of scores are visualized in Table 3.4. The GMMC model assigns higher likelihoods to in-distribution data than to the OOD data, outperforming all the other models by a
significant margin.

(2) Approximate Mass

Recent work of (Nalisnick et al. 2019a) has found that likelihood may not be enough for OOD detection in high-dimensional space. Real samples from a distribution form the area of high probability mass. But a point may have a high density while the surrounding area has a very low density, which indicates the density can change rapidly around it and that point is likely not a sample from the real data distribution. Thus, the norm of gradient of the log-density will be large compared to examples in the area mass. Based on this reasoning, Grathwohl et al. propose a new OOD score: 

$$\theta(x) = -\|\frac{\partial \log p_\theta(x)}{\partial x}\|_2.$$ 

Adopting this score function, we find that our model still outperforms the other competing methods (JEM and IGEBM), as shown in Table 3.3 (bottom).

(3) Predictive Distribution

Another useful OOD score is the maximum probability from a classifier’s predictive distribution:

$$s_\theta(x) = \max_y p_\theta(y|x).$$ 

Hence, OOD performance using this score is highly correlated with a model’s classification accuracy. The results can be found in Table 3.3 (middle). Interestingly, with this score function, there is no clear winner over four different benchmarks consistently, while GMMC performs similarly to JEM in most of the cases.

In summary, among all three different OOD score functions, GMMC outperforms the competing methods by a notable margin with two of them, while being largely on par with the rest one. The improved performance of GMMC on OOD detection is likely due to its explicit generative modeling of $p_\theta(x, y)$, which improves the evaluation of $p_\theta(x)$ over other methods.
Table 3.3 OOD Detection Results. Models are trained on CIFAR10. Values are AUROC.

<table>
<thead>
<tr>
<th>(s_\theta(x))</th>
<th>Model</th>
<th>SVHN</th>
<th>CIFAR10</th>
<th>Interp</th>
<th>CIFAR100</th>
<th>CelebA</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\log p_\theta(x))</td>
<td>Unconditional Glow</td>
<td>.05</td>
<td>.51</td>
<td>.55</td>
<td>.57</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Class-Conditional Glow</td>
<td>.07</td>
<td>.45</td>
<td>.51</td>
<td>.53</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IGEBM</td>
<td>.63</td>
<td>.70</td>
<td>.50</td>
<td>.70</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM</td>
<td>.67</td>
<td>.65</td>
<td>.67</td>
<td>.75</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GMMC (Gen)</td>
<td>.84</td>
<td>.75</td>
<td>.84</td>
<td>.86</td>
<td></td>
</tr>
<tr>
<td>(\max_y \log p_\theta(y</td>
<td>x))</td>
<td>Wide-ResNet</td>
<td>.93</td>
<td>.77</td>
<td>.85</td>
<td>.62</td>
</tr>
<tr>
<td></td>
<td>Class-Conditional Glow</td>
<td>.64</td>
<td>.61</td>
<td>.65</td>
<td>.54</td>
<td>.69</td>
</tr>
<tr>
<td></td>
<td>IGEBM</td>
<td>.43</td>
<td>.69</td>
<td>.54</td>
<td>.54</td>
<td>.69</td>
</tr>
<tr>
<td></td>
<td>JEM</td>
<td>.89</td>
<td>.75</td>
<td>.87</td>
<td>.79</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GMMC (Gen)</td>
<td>.84</td>
<td>.72</td>
<td>.81</td>
<td>.31</td>
<td></td>
</tr>
<tr>
<td>(| \frac{\partial \log p_\theta(x)}{\partial x} |)</td>
<td>Unconditional Glow</td>
<td>.95</td>
<td>.27</td>
<td>.46</td>
<td>.29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Class-Conditional Glow</td>
<td>.47</td>
<td>.01</td>
<td>.52</td>
<td>.59</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IGEBM</td>
<td>.84</td>
<td>.65</td>
<td>.55</td>
<td>.66</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM</td>
<td>.83</td>
<td>.78</td>
<td>.82</td>
<td>.79</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GMMC (Gen)</td>
<td>.88</td>
<td>.79</td>
<td>.85</td>
<td>.87</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4 Histograms of \(\log_\theta p(x)\) for OOD detection. Green corresponds to in-distribution dataset, while red corresponds to OOD dataset.

3.5.4 Robustness

DNNs have demonstrated remarkable success in solving complex prediction tasks. However, recent works (Szegedy et al. 2014a; Goodfellow et al. 2015a; Kurakin et al. 2017) have shown that
they are particularly vulnerable to adversarial examples, which are in the form of small perturbations to inputs but can lead DNNs to predict incorrect outputs. Adversarial examples are commonly generated through an iterative optimization procedure, which resembles the iterative sampling procedure of SGLD in Eq. 3.3. GMMC (and JEM) further utilizes sampled data along with real data for model training (see Eq. 3.7). This again shares some similarity with adversarial training (Goodfellow et al. 2015a), which has been proved to be the most effective method for adversarial defense. In this section, we show that GMMC achieves considerable robustness compared to other methods thanks to the MMC modeling (Pang et al. 2020b) and its generative training.

Table 3.5 Classification accuracies when models are under $L_\infty$ PGD attack with different $\epsilon$’s. All models are trained on CIFAR10.

<table>
<thead>
<tr>
<th>Model</th>
<th>Clean (%)</th>
<th>PGD-40 $\epsilon = 4/255$</th>
<th>PGD-40 $\epsilon = 8/255$</th>
<th>PGD-40 $\epsilon = 16/255$</th>
<th>PGD-40 $\epsilon = 32/255$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax</td>
<td>93.56</td>
<td>19.05</td>
<td>4.95</td>
<td>0.57</td>
<td>0.06</td>
</tr>
<tr>
<td>JEM</td>
<td>92.83</td>
<td>34.39</td>
<td>21.23</td>
<td>5.82</td>
<td>0.50</td>
</tr>
<tr>
<td>GMMC (Dis)</td>
<td>94.34</td>
<td>44.83</td>
<td>42.22</td>
<td>41.76</td>
<td>38.69</td>
</tr>
<tr>
<td>GMMC (Gen)</td>
<td>94.13</td>
<td><strong>56.29</strong></td>
<td><strong>56.27</strong></td>
<td><strong>56.13</strong></td>
<td><strong>55.02</strong></td>
</tr>
</tbody>
</table>

(1) PGD Attack

We run the white-box PGD attack (Madry et al. 2018) on the models trained by standard softmax, JEM and GMMC. We use the same number of steps (40) with different $\epsilon$’s as the settings of JEM for PGD. Table 3.5 reports the test accuracies of different methods. It can be observed that GMMC achieves much higher accuracies than standard softmax and JEM under all different attack strengths. The superior defense performance of GMMC (Dis) over JEM mainly attributes to the MMC modeling (Pang et al. 2020b), while GMMC (Gen) further improves its robustness over
GMMC (Dis) due to the generative training.

(2) C&W Attack

Pang et al. (Pang et al. 2018) reveal that when applying the C&W attack (Carlini and Wagner 2017) on their trained networks some adversarial noises have clearly interpretable semantic meanings to the original images. Tsipras et al. (Tsipras et al. 2018) also discover that the loss gradient of adversarially trained robust model aligns well with human perception. Interestingly, we observe similar phenomenon from our GMMC model. Fig. 3.6 shows some examples of adversarial noises generated from the GMMC (Gen) model under the C&W attack, where the noises are calculated as $(x_{\text{adv}} - x)/2$ to keep the pixel values in $[-0.5, 0.5]$. We observe around 5% of adversarial noises have clearly interpretable semantic meanings to their original images. These interpretable adversarial noises indicate that GMMC (Gen) can learn robust features such that the adversarial examples found by the C&W attack have to weaken the features of the original images as a whole, rather than generating salt-and-pepper like perturbations as for models of lower robustness.

To have a quantitative measure of model robustness under C&W attack, we apply the C&W attack to the models trained by the standard softmax, JEM and GMMC. In terms of classification accuracy, all of them achieve almost 100% error rate under the C&W attack. However, as shown in Table 3.7, the adversarial noises to attack the GMMC (Gen) model have a much larger $L_2$ norm than that of adversarial noises for other models. This indicates that to attack GMMC (Gen), the C&W attack has to add much stronger noises in order to successfully evade the network. In addition, GMMC (Dis) achieves a similar robustness as JEM under the C&W attack, while GMMC (Gen) achieves an improved robustness over GMMC (Dis) likely due to the generative training of
Table 3.6 Example adversarial noises generated from the GMMC (Gen) model under C&W attack on CIFAR10.

Table 3.7 $L_2$ norms of adversarial perturbations under C&W attack on CIFAR10.

<table>
<thead>
<tr>
<th>Model</th>
<th>Untarget iter=100</th>
<th>Target iter=1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax</td>
<td>0.205</td>
<td>0.331</td>
</tr>
<tr>
<td>JEM</td>
<td>0.514</td>
<td>0.905</td>
</tr>
<tr>
<td>GMMC (Dis)</td>
<td>0.564</td>
<td>0.784</td>
</tr>
<tr>
<td>GMMC (Gen)</td>
<td>1.686</td>
<td>1.546</td>
</tr>
</tbody>
</table>

3.5.5 Training Stability

Compared to JEM, GMMC has a well-formulated latent feature distribution, which fits well with the generative process of image synthesis. One advantage we observed from our experiments is that GMMC alleviates most of the instability issues of JEM. Empirically, we find that JEM can train less than 60 epochs before running into numerical issues, while GMMC can run 150 epochs smoothly without any numerical issues in most of our experiments.

3.5.6 Joint Training

Finally, we compare the joint training of JEM and GMMC on CIFAR10. The results show that joint training of GMMC is quite stable in most of our experiments, while JEM experiences substantial numerical instability issues. However, the quality of generated images from joint training
of GMMC is not as good as generative training of GMMC from scratch. Due to page limit, details of the comparison are relegated to Appendix E.

### 3.6 Conclusion and Future work

In this paper, we propose GMMC by reinterpreting the max-Mahalanobis classifier (Pang et al. 2020b) as an EBM. Compared to the standard softmax classifier utilized in JEM, GMMC models the latent feature space explicitly as the max-Mahalanobis distribution, which aligns well with the generative process of image synthesis. We show that GMMC can be trained discriminative, generatively or jointly with reduced complexity and improved stability compared to JEM. Extensive experiments on the benchmark datasets demonstrate that GMMC can achieve state-of-the-art discriminative and generative performances, and improve calibration, out-of-distribution detection and adversarial robustness.

As for future work, we plan to investigate the GMMC models trained by different methods: discriminative vs. generative. We are interested in the differences between the features learned by different methods. We also plan to investigate the joint training of GMMC to improve the quality of generated images further because joint training speeds up the learning of GMMC significantly and can scale up GMMC to large-scale benchmarks, such as ImageNet.
CHAPTER 4
JEM++: IMPROVED TECHNIQUES FOR TRAINING JEM

4.1 Abstract

Joint Energy-based Model (JEM) (Grathwohl et al. 2020a) 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 (Zhang et al. 2019) 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

4.2 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 (Krizhevsky and Hinton 2009; He et al. 2016; Brock et al. 2019a; Brown et al.)

¹Code: https://github.com/sndnyang/JEMPP
2020). However, prior works on discriminative models and generative models are largely separated. Even though a few researches (e.g., (Chapelle et al. 2009; Dempster et al. 1977)) have shown that generative training is beneficial to discriminative models, most recent works on generative models focus primarily on qualitative sample quality (Brock et al. 2019b; Santurkar et al. 2019; Vahdat and Kautz 2020), and the discriminative performances of state-of-the-art generative models are still far behind discriminative ones (Behrmann et al. 2018; Chen et al. 2019; Du and Mordatch 2019).

Among different discriminative and generative models, energy-based models (EBMs) (LeCun et al. 2006) are an appealing class of probabilistic models, which can be viewed as hybrid models with both discriminative and generative powers (Grathwohl et al. 2020a). Compared to the popular generative models, such as VAE (Kingma and Welling 2014) and GAN (Goodfellow et al. 2014), 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 (Xie et al. 2016; Han et al. 2019; Du and Mordatch 2019; Grathwohl et al. 2020a). Among them, JEM (Grathwohl et al. 2020a) is one of the most representative ones, which reinterprets the modern CNN classifier (e.g., Wide-ResNet (Zagoruyko and Komodakis 2016)) 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 (Zhao et al. 2020; Grathwohl et al. 2020b; Gao et al. 2020; Grathwohl et al. 2021).

However, training EBMs is still a challenging task. As shown in Table 4.1, existing methods demonstrate a great deal of tradeoffs among different algorithmic features in the quest of improved training algorithms. Most of the works (Nijkamp et al. 2019b; Du and Mordatch 2019; Grathwohl et al. 2020a) adopt the SGLD sampling (Welling and Teh 2011) 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 (Han et al. 2020b; Xie et al. 2020b; Grathwohl et al. 2021) or use special architectures (Grathwohl et al. 2020b; Vincent 2011) to amortize the SGLD sampling or improve its stability. Given the architectural simplicity of the SGLD-based methods, especially JEM (Grathwohl et al. 2020a), 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?

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
Table 4.1 Characteristics of different EBM training methods.

<table>
<thead>
<tr>
<th>Training Method</th>
<th>Fast</th>
<th>Stable</th>
<th>High dimensional</th>
<th>No aux model</th>
<th>Unrestricted architecture</th>
<th>Approximates likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGLD-based</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Score Matching</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Noise Contrastive</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Regularized Generator</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>JEM++ (ours)</td>
<td>↑</td>
<td>↑</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

\(L_p\)-norm ball of previous samples. Secondly, JEM does not support modern architecture features such as batch norm (Ioffe and Szegedy 2015)\(^2\). 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 (Zhang et al. 2019), a general framework for PGD (Madry et al. 2018) 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.

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

4.3 Energy-Based Models

Energy-based models (EBMs) (LeCun et al. 2006) define an energy function that assigns low energy values to samples drawn from data distribution and high values otherwise, such that any probability density $p_\theta(x)$ can be expressed via a Boltzmann distribution as

$$p_\theta(x) = \frac{\exp \left( - E_\theta(x) \right)}{Z(\theta)}, \quad (4.1)$$

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

The key challenge of training EBMs lies in estimating the partition function $Z(\theta)$, which is notoriously intractable. The standard maximum likelihood estimation of parameters $\theta$ 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 $x \in \mathcal{X}$ w.r.t. $\theta$ can
be expressed as

$$\frac{\partial \log p_\theta(x)}{\partial \theta} = \mathbb{E}_{p_\theta(x')} \frac{\partial E_\theta(x')}{\partial \theta} - \frac{\partial E_\theta(x)}{\partial \theta}, \quad (4.2)$$

where the expectation is over the density function $p_\theta(x')$, sampling from which is challenging due to the intractable $Z(\theta)$. Therefore, MCMC and Gibbs sampling (Hinton 2002b) have been proposed previously to estimate the expectation efficiently. To speed up the mixing for effective sampling, recently Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh 2011) has been employed to train EBMs by using the gradient information (Nijkamp et al. 2019b; Du and Mordatch 2019; Grathwohl et al. 2020a). Specifically, to sample from $p_\theta(x)$, SGLD follows

$$x^0 \sim p_0(x),$$

$$x^{t+1} = x^t - \frac{\alpha}{2} \frac{\partial E_\theta(x^t)}{\partial x^t} + \alpha \epsilon^t, \quad \epsilon^t \sim N(0, 1), \quad (4.3)$$

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

Prior works (Du and Mordatch 2019; Grathwohl et al. 2020a; Nijkamp et al. 2019a) 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. (Nijkamp et al. 2019a) 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 gener-
ate samples from SGLD chain after it converges. This requires the step-size $\alpha$ to decay with a polynomial schedule and an infinite number of sampling steps, which is not realistic in practical applications. Instead, JEM (Grathwohl et al. 2020a) uses a constant step-size $\alpha$ 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.

4.4 JEM++: The Improved Training of JEM

We first give a brief introduction of JEM (Grathwohl et al. 2020a) and then discuss a variety of new training procedures to improve its accuracy, stability and speed.

Joint Energy-based Models (JEM) (Grathwohl et al. 2020a) reinterprets modern CNN classifiers as EBMs. Considering a CNN classifier of parameters $\theta$, given an input $x$ the classifier first maps the input to a vector of $C$ real-valued numbers (or logits): $f_\theta(x)[y], \forall y \in [1, \cdots, C]$, where $C$ is the number of classes; the logits are then normalized via the softmax function to yield a probability vector: $p_\theta(y|x) = e^{f_\theta(x)[y]} / \sum_{y'} e^{f_\theta(x)[y']}. \text{Interestingly, the same vector of logits } f_\theta(x)[y] \text{ can also be used to define an EBM for the joint density: } p_\theta(x, y) = e^{f_\theta(x)[y]} / Z(\theta), \text{ where } Z(\theta) \text{ is an unknown normalizing constant (regardless of } x \text{ or } y). \text{ Then a marginal density of } x \text{ can be achieved by marginalizing the joint density as: } p_\theta(x) = \sum_y p_\theta(x, y) = \sum_y e^{f_\theta(x)[y]} / Z(\theta). \text{ Comparing this density with Eq. 4.1, it is readily to show that the corresponding energy function}
of $x$ is defined as

$$E_\theta(x) = -\log \sum_y e^{f_\theta(x)|y} = -\text{LSE}(f_\theta(x)),$$

(4.4)

where LSE(·) denotes the Log-Sum-Exp function.

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

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

(4.5)

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. 4.2 with the SGLD sampling defined in (4.3). 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.

### 4.4.1 Training EBM as a Minimax Optimization

In practice, when we employ the maximum likelihood estimate of model parameters $\theta$ with Eq. 4.2, a minibatch of $B$ samples $\{x_1, x_2, \ldots, x_B\} \sim p_\theta(x)$ and a minibatch of $B$ real data samples $\{x_1^r, x_2^r, \ldots, x_B^r\} \sim X$ 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. (Nijkamp et al. 2019a) who have found the insignificance of noise
term in the SGLD sampling (4.3), our empirical study also confirms this observation. Thus, we ignore the noise term in Eq. 4.3 and treat it as an artifact that generates some stochasticity in the sampling process to facilitate the optimization. Under this assumption, the SGLD sampling (4.3) can be reinterpreted approximately as an SGD iteration, with a learning rate of $\alpha/2$, initialized from a random sample of $p_0(x)$. Assume the convergence can be achieved, the objective of the SGLD sampling (4.3) is to solve the following optimization problem approximately \(^3\)

$$x^* = \arg\min_x E_\theta(x). \quad (4.6)$$

Therefore, the maximum likelihood learning of EBM with Eq. 4.2 is to approximately solve the following minimax game

$$\max_{\theta} \left[ \min_x E_\theta(x) - E_\theta(x^r) \right]. \quad (4.7)$$

To have a robust convergence behavior, we can solve the inner minimization problem of (4.7) by using the Proximal Point Method (Parikh and Boyd 2014). We can further treat the minimax optimization problem (4.7) as a multi-step differential game and extend YOPO (Zhang et al. 2019), a general framework of accelerating PGD, to speed up the training of EBM. Next, we describe these new training procedures in details.

\(^3\)The entire pipeline is still a stochastic sampler since samples are generated by running finite-length stochastic gradient decent with random initialization.
4.4.2 Proximal SGLD

Prior works on EBMs reveal the tradeoff between training stability and computational time of SGLD-based approaches (Nijkamp et al. 2019b; Du and Mordatch 2019; Grathwohl et al. 2020a). 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 (Rockafellar 1976; Parikh and Boyd 2014). To solve the inner minimization problem of (4.7), the algorithm generates a sequence \( \{x^t\} \) by the following proximal point iteration:

\[
x^{t+1} = \arg\min_{x} E_{\theta}(x) \quad s.t. \quad ||x - x^t|| < \varepsilon,
\] (4.8)

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. 4.8 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 (Rockafellar 1976). Thus, if we solve each minimization problem (4.8) inexacty with one
step of SGD, we obtain an inexact proximal point iteration

\[ x^{t+1} = x^t - \frac{\alpha}{2} L_p(\nabla_x E_\theta(x^t), \varepsilon), \]  

(4.9)

where \( L_p(\cdot, \varepsilon) \) projects the gradient to an \( L_p \)-norm ball of a radius \( \varepsilon \). Empirically, we find the \( L_\infty \)-norm works well across different architectures and datasets. Hence, we only consider the \( L_\infty \)-norm in the rest of the paper. With an \( L_\infty \)-norm, Eq. 4.9 can be rewritten as

\[ x^{t+1} = x^t - \frac{\alpha}{2} \text{clamp}(\nabla_x E_\theta(x^t), \varepsilon) + \alpha \epsilon^t, \]  

(4.10)

where the \( \text{clamp}(\cdot, \varepsilon) \) 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. 4.10, which resembles the original SGLD sampling (4.3) but with a gradient clamping operator used to enforce the proximity constraint.

### 4.4.3 Training EBM as a Differential Game

As discussed in Section 4.4.1, the maximum likelihood learning of EBM (4.7) solves a minimax game approximately. This objective has a close relationship to adversarial training with the PGD attack (Madry et al. 2018). 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 (Pontryagin 1987), a general framework in optimal control, Zhang et al. (Zhang et al. 2019) propose an optimization method called YOPO
(You-Propogate Only Once) 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 (4.7) can also be treated as a multi-step differential game and the sampled image $x$ from proximal SGLD (4.10) is only coupled with the first layer’s weights. By inserting the energy function (4.4) into (4.7), we can rewrite the minimax objective as:

$$\max_{\theta} \left[ \min_x -\text{LSE}(g_{\tilde{\theta}}(f_0(x, \theta_0))) - E_{\theta}(x^+) \right]$$

(4.11)

where $f_0$ denotes the first layer of a CNN-based EBM, $g_{\tilde{\theta}} = f^{\theta_{T-1}}_{T-1} \circ f^{\theta_{T-2}}_{T-2} \circ \cdots f^\theta_1$ denotes the network without the first layer, such that $f_\theta(x) = g_{\tilde{\theta}}(f_0(x, \theta_0))$. Given a sample $x$, the gradient of energy function (4.4) can be calculated by chain rule as:

$$\frac{\partial E_{\theta}(x)}{\partial x} = -\nabla_{g_{\tilde{\theta}}} \text{LSE}(g_{\tilde{\theta}}(f_0(x, \theta_0)))$$

$$\cdot \nabla_{f_0} g_{\tilde{\theta}}(f_0(x, \theta_0)) \cdot \nabla_x f_0(x, \theta_0).$$

(4.12)

Proximal SGLD (4.10) conducts $K$ sweeps of full forward and backward propagations for each update of $\theta$. 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
Figure 4.1 Comparison between SGLD-$K$ and PYLD-$M \times N$.

YOPO and introduce a slack variable:

$$p = -\nabla_{\theta} \text{LSE}\left(g_{\bar{\theta}}(f_{0}(x, \theta_0))\right) \cdot \nabla_{\theta} g_{\bar{\theta}}(f_{0}(x, \theta_0)),$$  \hspace{1cm} (4.13)

and freeze it as a constant in the inner loop of the sample update. We call our accelerated Proximial SGLD algorithm PYLD-$M \times N$ (Proximal-YOPO-SGLD) with $M$ outer loops and $N$ inner loops. Figure 4.1 demonstrates a conceptual comparison between SGLD-$K$ and PYLD-$M \times N$. SGLD-$K$ accesses the data $K$ times requiring $K$ full forward and backward propagations. On the contrary, PYLD-$M \times N$ 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 (Zhang et al. 2019), when $M \times N \approx K$, PYLD can achieve a similar sample quality as SGLD. But PYLD-$M$-
Algorithm 2 PYLD-M-N sampling: Given network \( g_\theta \) and \( f_0 \) with \( \theta_0 \), step-size \( \alpha \), number of steps \( M \) and \( N \)

1: \( x^0 \sim p_0(x) \)  
2: \textbf{for} \( t \in [0, 1, \cdots, M - 1] \) \textbf{do}  
3: \hspace{1em} \% calculate the slack variable  
4: \hspace{2em} \( p = -\nabla_{\theta} \text{LSE}(g_\theta(f_0(x^t, \theta_0))) \cdot \nabla_{\theta} g_\theta(f_0(x^t, \theta_0)) \)  
5: \hspace{1em} \( x^{t,0} = x^t \)  
6: \hspace{1em} \textbf{for} \( s \in [0, 1, \cdots, N - 1] \) \textbf{do}  
7: \hspace{2em} \( \gamma = \text{clamp}(p \cdot \nabla_{x^{t,s}} f_0(x^{t,s}, \theta_0), \varepsilon) \)  
8: \hspace{2em} \( x^{t,s+1} = x^{t,s} - \alpha/2 \cdot \gamma \)  
9: \hspace{1em} \textbf{end for}  
10: \hspace{1em} \( x^{t+1} = x^{t,N} + \alpha \varepsilon_t \)  
11: \hspace{1em} \textbf{end for}  
12: \textbf{return} \( x^M \)

\( N \) 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 2. For more details of YOPO, we refer the readers to (Zhang et al. 2019).

4.4.4 Informative Initialization

The initial sampling distribution \( p_0(x) \) also plays an important role in the training of EBM. Nijkamp et al. (Nijkamp et al. 2019a) summarize two main types of SGLD initializations for \( x^0 \): non-informative initialization and informative initialization. The former initializes the sample \( x^0 \) 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) (Hinton 2002b). Based on this, Tieleman (Tieleman 2008) 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. (Nijkamp et al. 2019b) 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. (Xie et al. 2016) 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 $\rho$) reinitializes the chains from random noise. This is also the sampling approach adopted by IGEBM (Du and Mordatch 2019) and JEM (Grathwohl et al. 2020a), 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

$$p_0(x) = \sum_y \pi_y N(\mu_y, \Sigma_y)$$  \hspace{1cm} (4.14)

with

$$\pi_y = |\mathcal{D}_y| / \sum_{y'} |\mathcal{D}_{y'}|,$$  

$$\mu_y = \mathbb{E}_{x \sim \mathcal{D}_y}[x],$$  

$$\Sigma_y = \mathbb{E}_{x \sim \mathcal{D}_y} \left[ (x - \mu_y)(x - \mu_y)^\top \right],$$

where $\mathcal{D}_y$ denotes the set of training samples with label $y$. As an example, Figure 4.2 visualizes the $\{\mu_1, \mu_2, \cdots, \mu_{10}\}$ (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 $x^0$ to guide the SGLD chain to converge faster than from a random noise since the initial sample $x^0$ 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 (Ioffe and Szegedy 2015), a modern architecture feature of DNNs, that is excluded by IGEBM and JEM due to the training difficulty introduced by batch norm.

**4.4.5 Batch Normalization and Learning Rate**

Batch norm (Ioffe and Szegedy 2015) 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 (Du and Mordatch 2019) and JEM (Grathwohl et al. 2020a), 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 \( x^0 \) are
sampled from a uniform or Gaussian distribution as in IGEBM and JEM, \( x^0 \) and real data samples
have different underlying distributions, violating the assumption of batch norm.

Similar phenomenon has also been observed by Xie et al. (Xie et al. 2020a) 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 \( p_0(x) \), we can use the informative
initialization discussed above to enable batch norm in the EBM training. Since the Gaussian mix-
ture distribution (4.14) 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 3 provides the pseudo-code for JEM++ training, which follows a sim-
ilar design of JEM (Grathwohl et al. 2020a) and IGEBM (Du and Mordatch 2019) with a replay
buffer. For brevity, only one real sample \( (x^r, y^r) \sim D \) and one generated sample \( x^M \sim p_\theta(x) \) are
used to optimize the parameter \( \theta \). It is straightforward to generalize the pseudo-code above to a
Algorithm 3 Training JEM++: Given network $f_{\theta}$, step-size $\alpha$, replay buffer $B$, number of steps $M$ and $N$, reinitialization frequency $\rho$, and number of classes $C$

1: while not converged do
2: Sample $(x^r, y^r) \sim D$
3: Sample $x^0 \sim B$ with probability $1 - \rho$, else $x^0 \sim \mathcal{N}(\mu_y, \Sigma_y)$, $y \sim p(y) = \pi$
4: Apply PYLD in Algo. 2 to sample $x^M$ from $x^0$
5: Calculate gradient with Eq. 4.2 from $x^r$ and $x^M$, and gradient of CE loss from $(x^r, y^r)$, and update model parameter $\theta$
6: Add / replace sample $x^M$ back to $B$
7: end while

mini-batch setting, which we use in the experiments.

4.5 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 (Grathwohl et al. 2020a), 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.

4.5.1 Hybrid Modeling

We train JEM++ on three benchmark datasets: CIFAR10, CIFAR100 (Krizhevsky and Hinton 2009) and SVHN (Netzer et al. 2011), and compare it to the state-of-the-art hybrid models, as

\(^4\)https://github.com/wgrathwohl/JEM
Table 4.2 Hybrid Modeling Results on CIFAR10.

<table>
<thead>
<tr>
<th>Class</th>
<th>Model</th>
<th>Acc %</th>
<th>IS</th>
<th>FID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Residual Flow (Chen et al. 2019)</td>
<td>70.3</td>
<td>3.60</td>
<td>46.4</td>
</tr>
<tr>
<td></td>
<td>Glow (Kingma and Dhariwal 2018)</td>
<td>67.6</td>
<td>3.92</td>
<td>48.9</td>
</tr>
<tr>
<td></td>
<td>IGEBM (Du and Mordatch 2019)</td>
<td>49.1</td>
<td>8.30</td>
<td>37.9</td>
</tr>
<tr>
<td>Single</td>
<td>JEM (K=20) (Grathwohl et al. 2020a) 1×</td>
<td>92.9</td>
<td>8.76</td>
<td>38.4</td>
</tr>
<tr>
<td>Hybrid</td>
<td>JEM++ (M=5) 2.4×</td>
<td>91.1</td>
<td>7.81</td>
<td>37.9</td>
</tr>
<tr>
<td>Model</td>
<td>JEM++ (M=10) 1.5×</td>
<td>93.5</td>
<td>8.29</td>
<td>37.1</td>
</tr>
<tr>
<td></td>
<td>JEM++ (M=20) .92×</td>
<td>94.1</td>
<td>8.11</td>
<td>38.0</td>
</tr>
<tr>
<td>Reg</td>
<td>VERA (α=100) 2.8×</td>
<td>93.2</td>
<td>8.11</td>
<td>30.5</td>
</tr>
<tr>
<td>Gen.</td>
<td>VERA (Grathwohl et al. 2021) (α=1) 2.8×</td>
<td>76.1</td>
<td>8.00</td>
<td>27.5</td>
</tr>
<tr>
<td>Disc.</td>
<td>WRN w/ BN</td>
<td>95.8</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Gen.</td>
<td>SNGAN (Miyato et al. 2018)</td>
<td>N/A</td>
<td>8.59</td>
<td>25.5</td>
</tr>
<tr>
<td></td>
<td>NCSN (Song and Ermon 2019)</td>
<td>N/A</td>
<td>8.91</td>
<td>25.3</td>
</tr>
</tbody>
</table>

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 4.3 Test Accuracy (%) on SVHN and CIFAR100.

<table>
<thead>
<tr>
<th>Model</th>
<th>SVHN</th>
<th>CIFAR100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax (w/ BN)</td>
<td>97.0</td>
<td>78.9</td>
</tr>
<tr>
<td>VERA (Grathwohl et al. 2021)</td>
<td>96.8</td>
<td>72.2</td>
</tr>
<tr>
<td>JEM (K=20)</td>
<td>96.7</td>
<td>72.2</td>
</tr>
<tr>
<td>JEM++ (M=5)</td>
<td>96.7</td>
<td>72.0</td>
</tr>
<tr>
<td>JEM++ (M=10)</td>
<td><strong>96.9</strong></td>
<td><strong>74.5</strong></td>
</tr>
</tbody>
</table>

well as standalone generative and discriminative models. Following the settings of JEM (Grathwohl et al. 2020a), all our experiments are based on the Wide-ResNet architecture (Zagoruyko and Komodakis 2016), 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 eval-

---

$^5$JEM cannot use a learning rate larger than 0.0001. Otherwise, it is extremely unstable and diverges easily at early epochs.
uate the quality of generated images, we adopt the Inception Score (IS) (Salimans et al. 2016) and Fréchet Inception Distance (FID) (Heusel et al. 2017).

The results on CIFAR10, CIFAR100 and SVHN are reported in Table 4.2 and 4.3, 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 ($1.5 \times$), 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 and CIFAR100 are shown in Figure 4.3 and 4.4, 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.

### 4.5.2 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
Figure 4.3 JEM++ generated CIFAR10 samples.

before 60 epochs. Prior works (Du and Mordatch 2019; Grathwohl et al. 2020a), 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 4.5 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 4.4, the informative initialization improves JEM’s stability because the initial samples $x^0$ 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 4.5.

We further compare the training speed between JEM and JEM++ in terms of run-time per
Figure 4.4 JEM++ generated class-conditional samples of SVHN and CIFAR100. Each row corresponds to one class.

each epoch. The results are reported in Table 4.4, 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 4.5) and training speed (Table 4.4), $M = 10$ and $N = 5$ achieves a good balance between the two criteria and therefore is our default configuration of JEM++. 
Figure 4.5 The learning curves of JEM++ trained on CIFAR10 with (1) Number of steps, and (2) the proximity constraint.

Table 4.4 Run-time comparison of JEM and JEM++ on CIFAR10.

<table>
<thead>
<tr>
<th>Model</th>
<th>Minutes per epoch</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>JEM</td>
<td>30.1</td>
<td>1×</td>
</tr>
<tr>
<td>JEM++, $M = 5$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N = 5$</td>
<td>12.5</td>
<td>2.41×</td>
</tr>
<tr>
<td>$N = 10$</td>
<td>12.6</td>
<td>2.39×</td>
</tr>
<tr>
<td>$N = 20$</td>
<td>13.0</td>
<td>2.31×</td>
</tr>
<tr>
<td>JEM++, $M = 10$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N = 5$</td>
<td>20.1</td>
<td>1.49×</td>
</tr>
<tr>
<td>$N = 10$</td>
<td>20.3</td>
<td>1.48×</td>
</tr>
<tr>
<td>$N = 20$</td>
<td>20.4</td>
<td>1.47×</td>
</tr>
<tr>
<td>JEM++, $M = 20$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N = 5$</td>
<td>32.5</td>
<td>.93×</td>
</tr>
<tr>
<td>$N = 10$</td>
<td>32.7</td>
<td>.92×</td>
</tr>
<tr>
<td>$N = 20$</td>
<td>32.9</td>
<td>.91×</td>
</tr>
</tbody>
</table>
4.5.3 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 4.5. 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 (Kingma and Ba 2015) 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 4.5 Ablation study of different components of JEM++. All the models are trained on CIFAR10 with $M = 10$ and $N = 5$.

<table>
<thead>
<tr>
<th>Ablation</th>
<th>Acc %</th>
<th>IS</th>
<th>FID</th>
</tr>
</thead>
<tbody>
<tr>
<td>JEM++</td>
<td>93.5</td>
<td>8.29</td>
<td>37.1</td>
</tr>
<tr>
<td>w/o Proximity</td>
<td>92.9</td>
<td>7.92</td>
<td>36.0</td>
</tr>
<tr>
<td>w/ Adam</td>
<td>92.5</td>
<td>7.65</td>
<td>42.7</td>
</tr>
<tr>
<td>random init (w/ BN)$^1$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>random init (w/o BN)$^2$</td>
<td>88.6</td>
<td>7.64</td>
<td>35.1</td>
</tr>
<tr>
<td>informative init (w/o BN)$^3$</td>
<td>91.1</td>
<td>7.92</td>
<td>39.8</td>
</tr>
</tbody>
</table>

$^1$ 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.

4.5.4 Classification Accuracy vs. Image Quality

One interesting phenomenon we observed from our experiments is the tradeoff between classification accuracy and image quality. Figure 4.6 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 (4.7), in which the classifier and the implicit generator compete with each other to achieve an equilibrium. Compared to the standard GANs (Goodfellow et al. 2014), 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 4.6 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.

4.6 Applications

In the main text, we compared JEM and JEM++ in terms of classification accuracy, image quality, training stability and speed. Here we compare JEM and JEM++ in other downstream applications, such as adversarial robustness, calibration and out of distribution (OOD) detection.

4.6.1 Robustness

It’s well known that DNNs are particularly vulnerable to adversarial examples (Szegedy et al. 2014b; Goodfellow et al. 2015b) in the form of small perturbations to inputs that lead DNNs to predict incorrect outputs. Specially, the widely explored adversarial examples are defined as perturbed inputs $\tilde{x} = x + \delta$ under an $L_p$-norm constraint $\|\delta\|_p < \varepsilon$. To overcome the security
threat posed by adversarial examples, a variety of defense algorithms have been proposed in the past few years to improve the robustness of models (Goodfellow et al. 2015a; Dziugaite et al. 2016; Guo et al. 2017b; Akhtar et al. 2018; Madry et al. 2018; Chiang et al. 2020). Among them, adversarial training (Goodfellow et al. 2015a; Madry et al. 2018) has been proved to be the most effective one to defend adversarial examples.

As we discussed in Section 3.1, there is a close relationship between the maximum likelihood learning of EBM (7) and adversarial training with PGD (Madry et al. 2018) as both solve a similar minimax objective. Therefore, the maximum likelihood trained EBMs should be more robust to adversarial examples than the standard trained softmax classifiers, and this has been empirically verified by recent works (e.g., (Du and Mordatch 2019; Grathwohl et al. 2020a)). Since JEM++ improves JEM’s accuracy, training stability and speed, it’s interesting to check if JEM++ can improve model robustness as well.

![Figure 4.7 Adversarial robustness under the PGD attacks.](image)

To evaluate the robustness of a given model, we run a white-box PGD attack (Madry et al. 2018).
under an $L_\infty$ or $L_2$ constraint using foolbox (Rauber et al. 2017), with the results reported in Figure 4.7. It can be observed that JEM++ achieves a similar robustness with JEM under the $L_\infty$ and $L_2$ PGD attacks, while both are more robust than the standard softmax classifiers. The adversarial training with PGD (Madry et al. 2018; Santurkar et al. 2019) achieves the highest robustness since it is trained and test under the same PGD attacks, while JEM/JEM++ are trained on real and generated samples from the energy function, without the access to the PGD samples for training.

4.6.2 Calibration

Recent researches have shown that the predictions from modern DNNs could be over-confident (Guo et al. 2017a), i.e., they often output incorrect but confident predictions, which could have catastrophic consequences. Hence, calibration of uncertainty for DNNs is a critical task with an enormous practical impact nowadays. Here, the confidence is defined as $\max_y p(y|x)$ which is used to decide when to output a prediction. In this section, we compare the calibration qualities of models trained by JEM and JEM++ as well as the standard softmax classifiers on the CIFAR10/100 dataset.

Expected Calibration Error (ECE) is a standard metric to evaluate the calibration quality of a classifier (Guo et al. 2017a). It firstly computes the confidence of the model, $\max_y p(y|x_i)$, for each $x_i$ in the dataset. Then it groups the predictions into equally spaced buckets $\{B_1, B_2, \cdots, B_M\}$ based on the confidence scores. For example, if $M = 20$, then $B_1$ would represent all examples for which the model's
confidence scores were between 0 and 0.05. Then ECE is calculated as

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left| \text{acc}(B_m) - \text{conf}(B_m) \right|, \quad (4.15)$$

where $n$ is the number of data in the dataset, acc$(B_m)$ is the average accuracy of the model on all the examples in $B_m$ and conf$(B_m)$ is the average confidence on all the examples in $B_m$. In our experiments, we set $M = 20$. For a perfectly calibrated model, the ECE will be 0 for any $M$.

Figures 4.8 and 4.9 report the results on CIFAR10 and CIFAR100, respectively. As we can see, the models trained by JEM and JEM++ are better calibrated than the standard softmax classifiers, while JEM++ achieves better calibration qualities than JEM on CIFAR10 (2.35% vs. 4.2%) and CIFAR100 (3.3% vs. 4.87%) with notable margins.

![Figure 4.8 Calibration results on CIFAR10. The smaller ECE is, the better.](a) Standard Softmax (b) JEM ($K=20$) (c) JEM++ ($M=5$) (d) JEM++ ($M=10$)](a) Standard Softmax (b) JEM ($K=20$) (c) JEM++ ($M=5$) (d) JEM++ ($M=10$)

![Figure 4.9 Calibration results on CIFAR100. The smaller ECE is, the better.](a) Standard Softmax (b) JEM ($K=20$) (c) JEM++ ($M=5$) (d) JEM++ ($M=10$)
4.6.3 Out-Of-Distribution Detection

The OOD detection is a binary classification problem, which outputs a score $s_\theta(x) \in \mathbb{R}$ for a given query $x$. The model should be able to assign lower scores to OOD examples than to in-distribution examples, such that it can be used to distinguish two sets of examples. Following the settings of JEM (Grathwohl et al. 2020a), we use the Area Under the Receiver-Operating Curve (AUROC) (Hendrycks and Gimpel 2016) to evaluate the performance of OOD detection. In our experiments, two standard score functions are considered: the input density $p_\theta(x)$ (Nalisnick et al. 2018) and the predictive distribution $p_\theta(y|x)$ (Hendrycks and Gimpel 2016).

Input Density

A natural choice of $s_\theta(x)$ is the input density $p_\theta(x)$. For OOD detection, intuitively we consider examples with low $p(x)$ to be OOD. Quantitative results can be found in Table 4.6 (top row), where CIFAR10 is the in-distribution data and SVHN, an interpolated CIFAR10, CIFAR100 and CelebA are treated as out-of-distribution data, respectively. Moreover, the corresponding distributions of scores are visualized in Table 4.7. As can be seen, the JEM++ model assigns higher likelihoods to in-distribution data than to the OOD data, outperforming JEM and all the other models by significant margins.

Predictive Distribution

Another useful OOD score is the maximum probability from a classifier’s predictive distribution: $s_\theta(x) = \max_y p_\theta(y|x)$. Hence, OOD performance using this score is highly correlated with a model’s classification accuracy. The results can be found in Table 4.6 (bottom row). Again,
JEM++ outperforms JEM and all the other models by notable margins.

Table 4.6 OOD detection results. Models are trained on CIFAR10. Values are AUROC.

<table>
<thead>
<tr>
<th>$s_\theta(x)$</th>
<th>Model</th>
<th>SVHN</th>
<th>CIFAR10</th>
<th>Interp</th>
<th>CIFAR100</th>
<th>CelebA</th>
</tr>
</thead>
<tbody>
<tr>
<td>log $p_\theta(x)$</td>
<td>Uncond Glow</td>
<td>.05</td>
<td>.51</td>
<td>.55</td>
<td>.57</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IGEBM</td>
<td>.63</td>
<td>.70</td>
<td>.50</td>
<td>.70</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM (K=20)</td>
<td>.67</td>
<td>.65</td>
<td>.67</td>
<td>.75</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM++ (M=5)</td>
<td>.89</td>
<td>.73</td>
<td>.81</td>
<td>.74</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM++ (M=10)</td>
<td>.63</td>
<td>.68</td>
<td>.64</td>
<td>.59</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM++ (M=20)</td>
<td>.85</td>
<td>.57</td>
<td>.68</td>
<td>.89</td>
<td></td>
</tr>
<tr>
<td>max$<em>y p</em>\theta(y</td>
<td>x)$</td>
<td>WideResNet</td>
<td>.93</td>
<td>.77</td>
<td>.85</td>
<td>.62</td>
</tr>
<tr>
<td></td>
<td>IGEBM</td>
<td>.43</td>
<td>.69</td>
<td>.54</td>
<td>.69</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM (K=20)</td>
<td>.89</td>
<td>.75</td>
<td>.87</td>
<td>.79</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM++ (M=5)</td>
<td>.88</td>
<td>.78</td>
<td>.86</td>
<td>.78</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM++ (M=10)</td>
<td>.91</td>
<td>.78</td>
<td>.88</td>
<td>.82</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM++ (M=20)</td>
<td>.94</td>
<td>.77</td>
<td>.88</td>
<td>.90</td>
<td></td>
</tr>
</tbody>
</table>

4.7 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 4.6. 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., (Grathwohl et al. 2021; Ardizzone et al. 2020)). We also plan to explore JEM++ to large-scale benchmarks, such as ImageNet, and its application to other domains, such as NLP.
<table>
<thead>
<tr>
<th>JEM</th>
<th>JEM++(M=5)</th>
<th>JEM++(M=10)</th>
<th>JEM++(M=20)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Table 4.7 Histograms of $\log_\theta p(x)$ for OOD detection. Green corresponds to in-distribution dataset, while red corresponds to OOD dataset.
CHAPTER 5

MA-EBM: TOWARDS UNDERSTANDING THE MANIFOLD OF ENERGY-BASED MODELS

5.1 Abstract

Energy-based models (EBMs) exhibit a variety of desirable properties in predictive tasks, such as
generality, simplicity and compositionality. However, training EBMs on high-dimensional datasets
remains unstable and expensive. In this paper, we present a Manifold-Aware EBM (MA-EBM) to
boost the overall performance of unconditional EBM and Joint Energy-based Model (JEM) (Grath-
wohl et al. 2020a). Despite its simplicity, MA-EBM significantly improves unconditional EBMs
in image generation quality, training stability and speed on a host of benchmark datasets, such as
CIFAR10, CIFAR100, CelebA-HQ, and ImageNet 32x32. Once class labels are available, label-
incorporated MA-EBM (MA-JEM) further surpasses MA-EBM in image generation quality with
an over 40% FID improvement, while enjoying improved accuracy. To the best of our knowledge,
MA-EBM is currently the most stable EBM with the least number of MCMC steps $K$ (e.g., $K = 2$)
without divergence issues, and MA-JEM is the first algorithm that demonstrates the utility of labels
that can dramatically improve the image generation quality of EBMs.

5.2 Introduction

Energy-Based Models (EBMs) are an appealing class of probabilistic models, which are widely
applicable in image generation, out of distribution detection, adversarial robustness, and hybrid
discriminative-generative modeling (Nijkamp et al. 2019b; Du and Mordatch 2019; Du et al. 2021;
Gao et al. 2021; Grathwohl et al. 2020a; Yang and Ji 2021; Grathwohl et al. 2021; Xiao et al.
Despite a number of desirable properties, training EBMs on high-dimensional datasets remains very challenging. Most of the works utilize the Markov Chain Monte Carlo (MCMC) sampling (Welling and Teh 2011) to generate samples from the model distribution represented by an EBM. Specifically, they require $K$-step Langevin Dynamics sampling (Welling and Teh 2011) to generate samples from the model distribution in every iteration, which can be extremely expensive when using a large number of sampling steps, or highly unstable with a small number of steps. The trade-off between the training time and stability prevents the MCMC sampling based EBMs from scaling to large-scale datasets.

![Generated samples of CelebA-HQ 128x128 from our MA-EBM.](image)

Recently, there are a flurry of works on training and improving EBMs. The most recent studies (Du et al. 2021; Gao et al. 2021) on the MCMC-based approach focus on improving the generation quality and stability. However, they still resort to a long sampling chain and requires expensive training. Another branch of works (Grathwohl et al. 2021; Xiao et al. 2021) augment the EBM with a regularized generator in a GAN-style training to improve the stability and speed, sacrificing the desired property of learning a single object. Moreover, JEM (Grathwohl et al. 2020a) proposes an elegant framework to reinterpret the modern CNN classifier as an EBM and achieves impressive performances in image classification and generation simultaneously. However, it also suffers from
the divergence issue of the MCMC-based sampling, and its generative performance falls behind state-of-the-art EBMs. Tackling the limitations of JEM, JEM++ (Yang and Ji 2021) introduces a variety of training procedures and architecture features to improve JEM in terms of accuracy, speed and stability altogether. Furthermore, JEM++ demonstrates a trade-off between classification accuracy and image quality, but it still cannot improve image generation quality notably.

In this paper, we introduce three simple yet effective training techniques to improve unconditional EBM and JEM in terms of image generation quality, training stability and speed altogether. First, the informative initialization introduced in JEM++ dramatically improves the training stability and reduces the required MCMC sampling steps. However, it’s not scalable for high-resolution and large-scale datasets. Hence, we introduce a simplified informative initialization that is suitable for unconditional EBM and JEM for high-resolution images and large number of classes (e.g., 128x128 CelebA-HQ and 1000-class ImageNet 32x32 datasets). Second, we find the $L_2$ regularization of the energy magnitude does not work with the energy function utilized in JEM. To enable $L_2$ regularization and improve the training stability, we augment the standard softmax classifier with a new energy head, which is then $L_2$ regularized. We call our new unconditional EBM as Manifold-Aware EBM (MA-EBM) since the gap between the manifolds of initial samples and real data is much smaller than that of the original unconditional EBM and the energy manifold is further $L_2$ regularized. Finally, observing that data augmentation hurts the generative quality but improves the classification accuracy in JEM, we train our label-incorporated MA-EBM (MA-JEM) by using two mini-batches: one with data augmentation for classification, and the other one without data augmentation for maximum likelihood estimation of EBM. All of these techniques allow us to
improve image generation quality of EBM dramatically, while retaining or sometimes improving classification accuracy of prior state-of-the-art EBMs. Example generated samples of CelebA-HQ 128x128 from our MA-EBM are illustrated in Fig. 5.1, the quality of which is much higher than that of the original EBM as we will discuss in the experiments.

Throughout the history of research, we have learned that the speed of scientific advancement is directly proportional to the number of researchers, making it essential to enable researchers to participate in all areas of research.

Our main contributions are summarised as follows:

1. We propose a simplified informative initialization to initialize the SGLD chain, which stabilizes and accelerates the training of unconditional EBM and JEM, while being scalable for high resolution and large-scale datasets.

2. Adding an $L_2$-regularized energy head on top of a CNN feature extractor to represent an energy function stabilizes the training of JEM significantly.

3. We improve image generation quality of MA-JEM by using two mini-batches: one with data augmentation for classification, and the other one without data augmentation for maximum likelihood estimation of EBMs.

4. We conduct extensive experiments on four benchmark datasets. MA-EBM matches or outperforms prior state-of-the-art unconditional EBMs, while significantly improving training stability and speed. On CIFAR10, CIFAR100 and Imagenet 32x32, MA-EBM only requires $K = 2$ to stabilize the training. Moreover, MA-JEM improves JEM’s training stability and
speed, image generation quality, and classification accuracy altogether, while outperforming MA-EBM in image generation quality by a notable margin.

5. To the best of our knowledge, our models are currently the most stable EBMs with the least number of MCMC sampling steps (e.g., $K = 2$), and MA-JEM is the first algorithm that demonstrates the utility of labels that can dramatically improve the image generation quality of EBMs.

5.3 Background

Energy-based Models (EBMs) (LeCun et al. 2006) utilizes the idea that any probability density $p_\theta(x)$ can be expressed via a Boltzmann distribution as

$$ p_\theta(x) = \frac{\exp(-E_\theta(x))}{Z(\theta)}, \quad (5.1) $$

where $E_\theta(x)$ is an energy function that maps $x \in \mathcal{X}$ to a scalar, and $Z(\theta) = \int_{\mathcal{X}} \exp(-E_\theta(x)) \, dx$ is the normalizing constant w.r.t $x$ (also known as the partition function). Ideally, an energy function should assign low energy values to samples drawn from data distribution and high values otherwise.

The key challenge of EBM training is estimating the intractable partition function $Z(\theta)$, and the maximum likelihood estimation of parameters $\theta$ is not straightforward. A number of sampling-based approaches have been proposed to approximate the partition function effectively. Specifi-
cally, the derivative of the log-likelihood of \( x \in X \) w.r.t. \( \theta \) can be expressed as

\[
\frac{\partial \log p_\theta(x)}{\partial \theta} = \mathbb{E}_{p_\theta(x')} \left[ \frac{\partial E_\theta(x')}{\partial \theta} \right] - \mathbb{E}_{p_d(x)} \left[ \frac{\partial E_\theta(x)}{\partial \theta} \right],
\]

(5.2)

where the first expectation is over the model density \( p_\theta(x') \), which is challenging due to the intractable \( Z(\theta) \).

To estimate it efficiently, MCMC and Gibbs sampling (Hinton 2002b) have been proposed. Moreover, to speed up the sampling, Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh 2011) is recently employed to train EBMs (Nijkamp et al. 2019b; Du and Mordatch 2019; Grathwohl et al. 2020a). Specifically, to sample from \( p_\theta(x) \), the SGLD follows

\[
x^0 \sim p_0(x), \\
x^{t+1} = x^t - \alpha \frac{\partial E_\theta(x^t)}{\partial x^t} + \alpha \epsilon^t, \quad \epsilon^t \sim \mathcal{N}(0, 1),
\]

(5.3)

where \( p_0(x) \) is typically a uniform distribution over \([-1, 1]\), whose samples are refined via a noisy gradient decent with step-size \( \alpha \) over a sampling chain.

Prior works (Nijkamp et al. 2019a,b; Du and Mordatch 2019; Grathwohl et al. 2020a) have investigated the effect of hyper-parameters in SGLD sampling and showed that the SGLD-based approaches suffer from poor stability and prolonged computation of sampling at every iteration. Nijkamp et al. (Nijkamp et al. 2019a) find that it’s desirable to generate samples from the SGLD chain after it converges. The convergence requires the step-size \( \alpha \) to decay with a polynomial schedule and infinite sampling steps, which is impractical. Therefore, Short-Run and Long-Run
MCMC samplings are utilized for EBM training. Moreover, most of works (Du and Mordatch 2019; Grathwohl et al. 2020a; Du et al. 2021) use a constant step-size $\alpha$ during sampling and approximate the samples with a sampler that runs only for a finite number of steps, which is still computationally very expensive. Another recent work (Gao et al. 2021) combines the SGLD-based approach with diffusion models (Ho et al. 2020) under a framework of conditional EBMs. They achieve state-of-the-art image generation quality and obtain a faithful energy potential.

Joint Energy-based Models (JEM) (Grathwohl et al. 2020a) demonstrates that standard softmax-based classifiers can be trained as EBMs. Given an input $x \in \mathbb{R}^D$, a classifier of parameters $\theta$ maps the input to a vector of $C$ real-valued numbers (known as logits): $f_{\theta}(x)[y], \forall y \in [1, \cdots, C]$, where $C$ is the number of classes. Then the softmax function is employed to convert the logits into a categorical distribution: $p_{\theta}(y|x) = e^{f_{\theta}(x)[y]} / \sum_{y'} e^{f_{\theta}(x)[y']}$. The authors reuse the logits to define an energy function for the joint density: $p_{\theta}(x, y) = e^{f_{\theta}(x)[y]} / Z(\theta)$. Then a marginal density of $x$ can be achieved by marginalizing out $y$ as: $p_{\theta}(x) = \sum_y p_{\theta}(x, y) = \sum_y e^{f_{\theta}(x)[y]} / Z(\theta)$. As a result, the corresponding energy function of $x$ is defined as

$$E_{\theta}(x) = -\log \sum_y e^{f_{\theta}(x)[y]} = -\text{LSE}(f_{\theta}(x)), \quad (5.4)$$

where LSE(·) denotes the Log-Sum-Exp function. The advantage of this LSE energy function is that an additional degree of freedom in the scale of the logit vector now can model the data distribution.

To optimize the model parameter $\theta$, JEM maximizes the logarithm of joint density function
\( p_\theta(x, y): \)

\[
\log p_\theta(x, y) = \log p_\theta(y|x) + \log p_\theta(x), \quad (5.5)
\]

where the first term denotes the cross-entropy objective for classification, and the second term can be optimized by the maximum likelihood learning of EBM as shown in Eq. 5.2. We can also interpret the second term as an unsupervised regularization on model parameter \( \theta \).

## 5.4 Method

### 5.4.1 Informative Initialization and MA-EBM

As shown in Eq. 5.3, the SGLD sampling starts from an initial distribution \( p_0(x) \). To train the EBM as a generative model, Short-Run MCMC sampling (Nijkamp et al. 2019b) utilizes an MCMC sampler that starts from a random noise distribution such as a uniform distribution. A concurrent work IGEBM (Du and Mordatch 2019) proposes an initialization approach with a sample replay buffer in which they store past generated samples and draw either samples from replay buffer or uniform random noise to initialize the Langevin dynamics procedure. This is also the sampling approach adopted by (Grathwohl et al. 2020a; Zhu et al. 2021). Furthermore, JEM++ (Yang and Ji 2021) introduces an informative initialization with the replay buffer by using a Gaussian mixture distribution estimated from the training images, which significantly reduces the number of sampling steps required by SGLD while improving its training stability.

However, the class-dependent Gaussian mixture distribution is not available for unconditional case or unsupervised datasets. Furthermore, the per-class covariance matrices of the Gaussian
mixture distribution utilized by JEM++ can be huge for high-resolution image datasets with large number of classes. Hence, we estimate a single Gaussian distribution for the whole training dataset, instead of each category. That is, we estimate the initial sampling distribution as

\[ p_0(x) = \mathcal{N}(\mu, \Sigma) \]

with \( \mu = \mathbb{E}_{x \sim \mathcal{D}}[x], \quad \Sigma = \mathbb{E}_{x \sim \mathcal{D}}[(x - \mu)(x - \mu)^\top] \),

where \( \mathcal{D} \) denotes the whole training set. The visualization of the estimated centers and samples from \( p_0(x) \) of different datasets are provided in the appendix. Since only one Gaussian distribution is estimated from the whole training set, we can apply it for unconditional datasets such as CelebA, and reduce the memory and space required for the large covariance matrices\(^1\). Although \( \mu \) and \( \Sigma \) can be well estimated with sufficient samples, they still lead to a biased initialization with higher variance, compared to the Gaussian mixture initialization utilized in JEM++. But our empirical study also shows that our simplified initialization won’t deteriorate the performance than bias-reduced Gaussian mixture initialization.

Since the manifold of \( x_0 \) from our informative initialization is much closer to the real data manifold than that of uniform initialization, this informative initialization reduces the required sampling steps (and thus accelerates training), and also improves training stability as we will demonstrate in the experiments. We therefore call the EBM with this simplified informative initialization as MA-EBM throughout this work.

\(^1\)One covariance matrix of CIFAR10 requires \( (3 \times 32 \times 32)^2 \approx 9.4M = 37.6\text{MB} \). \( k \)-class takes \( 37.6^k \text{MB} \).
Figure 5.2 The architecture of MA-JEM. A energy head $f_e$ is augmented for energy magnitude regularization and two mini-batches are used for the training of classifier and the maximum likelihood estimate of EBM, respectively.

### 5.4.2 Energy Function Regularization

IGEBM (Du and Mordatch 2019) finds that constraining the Lipschitz constant of the energy network can ease the instability issue in Langevin dynamics. Hence, they weakly $L_2$ regularize energy magnitudes for both positive and negative samples to the contrastive divergence as:

$$
\mathcal{L} = \frac{1}{B} \sum_{i=1}^{B} \left( E_i^+ - E_i^- + \alpha (E_i^{+2} + E_i^{-2}) \right),
$$

where $E^+ = E_\theta(x^+)$ with $x^+$ sampled from the data distribution $p_d$, and $E^- = E_\theta(x^-)$ with $x^-$ sampled from the model distribution $p_\theta(x)$. The effect of $L_2$ regularization can be viewed as Fig 5.6b. However, since $L_2$ regularization would force the vector of logits $f_\theta(x)$ to be uniform, while maximizing $p_\theta(y|x)$ boosts $f_\theta(x)[y]$. Hence, the $L_2$ regularization is incompatible with Eq. 5.4 and cannot be directly applied to vanilla JEM.

To incorporate $L_2$ regularization to JEM, we propose to augment the standard CNN softmax
classifier with an extra fully connected layer, called Energy Head, as shown in Fig. 5.2. Then the $L_2$ regularization is applied on the energy head (instead of the LSE classification head) to improve the training stability.

### 5.4.3 MA-JEM

Existing work (Nijkamp et al. 2019b) studied the effect of injected noise on training stability via smoothing $p_{\text{data}}$ with additive noises $\mathbf{x} \leftarrow \mathbf{x} + \epsilon \sim N(0, \sigma^2 I)$. Their results demonstrated that the fidelity of the examples in terms of IS and FID improves, when lowering $\sigma^2$. And they depict the tradeoff between the sampling steps $K$ and the level of injected noise, indicating the training time and the stability/fidelity. JEM has similar observation that increasing $\sigma$ makes training more stable but hurts generation quality. Gao et al. (Gao et al. 2021) propose to learn a sequence of EBMs for the distributions of the diffusion process (Ho et al. 2020). By doing so, their method can avoid the injected noise and achieve STOA generation quality of EBMs.

The data augmentation is known to improve the classification accuracy and the performance of self-supervised representation learning (Chen et al. 2020). We use $T$ to denote the augmentation function for classification. Hence, the actual objective of JEM is

$$\log p_\theta(\mathbf{x}, y) = \log p_\theta(y|T(\mathbf{x})) + \log p_\theta(T(\mathbf{x})), \quad (5.8)$$

which indicates that JEM and JEM++ perform the maximum likelihood estimate of $p_\theta(T(\mathbf{x}))$, rather than $p_\theta(\mathbf{x})$. Motivated by it, we remove the data augmentation for maximum likelihood modeling and only keep the data augmentation for classification. Then our final objective function
is:

\[
\log p_\theta(x, y) = \log p_\theta(y|T(x)) + \log p_\theta(x),
\]

(5.9)

where the first term is calculated with the classification head, and the second term is evaluated with the energy head \( E_\theta(x) = f_c(f_h(x)) \), instead of the LSE energy function defined in Eq. 4.4. Fig. 5.2 presents the overall architecture of MA-JEM. We can also view MA-JEM as a variant of MA-EBM augmented with a linear classification head when class labels are available for training. More interestingly, we empirically find optimizing Eq. 5.9 with \( p_\theta(x) \) is very stable without injected noise. In contrast, the additive Gaussian noise is crucial to stabilize the training of Eq. 5.8 with \( p_\theta(T(x)) \). We leave the explanation as an exciting direction of future work.

In summary, Algorithm 4 provides the pseudo-code for MA-EBM/JEM training, which follows a similar design of IGEBM (Du and Mordatch 2019), JEM (Grathwohl et al. 2020a) and JEM++ (Yang and Ji 2021) with a replay buffer. For brevity, only one real sample and one generated sample are used to optimize the model parameter \( \theta \). It is straightforward to generalize the pseudo-code below to a mini-batch setting, which we use in the experiments.

5.5 Experiments

In this section, we first evaluate the generative performance of MA-EBM on multiple datasets, including CIFAR10, CIFAR100, CelebA-HQ 128x128 and ImageNet 32x32. Then, we investigate the efficacy of the MA-JEM on CIFAR10 and CIFAR100. Third, the training speeds of different EBMs are evaluated to demonstrate the efficiency of our methods. Finally, the ablation study
Algorithm 4 MA-EBM/JEM Training: Given network $f_{\theta}$, SGLD step-size $\alpha$, SGLD noise $\sigma$, replay buffer $B$, SGLD steps $K$, reinitialization frequency $\rho$

1: while not converged do
2: Sample $x^+$ and $y$ from dataset
3: Sample $\hat{x}_0 \sim B$ with probability $1 - \rho$, else $\hat{x}_0 \sim p_0(x)$ as Eq. 5.6
4: for $t \in [1, 2, \ldots, K]$ do
5: $\hat{x}_t = \hat{x}_{t-1} - \alpha \cdot \frac{\partial E(\hat{x}_{t-1})}{\partial \hat{x}_{t-1}} + \sigma \cdot \mathcal{N}(0, I)$
6: end for
7: $x^- = \text{StopGrad}(\hat{x}_K)$
8: $L_{\text{gen}}(\theta) = E(x^+) - E(x^-) + \alpha \left( E(x^+)^2 + E(x^-)^2 \right)$ as Eq. 5.7.
9: $L(\theta) = L_{\text{gen}}(\theta)$ for MA-EBM
10: $L(\theta) = L_{\text{clf}}(\theta) + L_{\text{gen}}(\theta)$ with $L_{\text{clf}}(\theta) = \text{xent}(f_{\theta}(x), y)$ for MA-JEM
11: Calculate gradient $\frac{\partial L(\theta)}{\partial \theta}$ to update $\theta$
12: Add $x^-$ to $B$
13: end while

and visualization of the differences between trained EBM and JEM are provided to analyze their generative capability and stability.

Our code is largely built on top of JEM (Grathwohl et al. 2020a)\(^2\). For a fair comparison with JEM, we update each model with 390 iterations in 1 epoch. Empirically, we find a batch size of 128 for $p_{\theta}(y|x)$ achieves the best classification accuracy on CIFAR10, while we use 64, the same batch size as in JEM, for $p_{\theta}(x)$. All our experiments are performed with PyTorch on Nvidia GPUs. For CIFAR10 and CIFAR100, we train the backbone Wide-ResNet 28-10 (Zagoruyko and Komodakis 2016) on a single GPU. Due to limited computational resources, we use Wide-ResNet 28-2 for ImageNet 32x32 on a single GPU, and Wide-ResNet 28-5 for CelebA-HQ 128x128 on 2 GPUs.

For reproducibility, our source code is also provided as a part of the supplementary material.

\(^2\)https://github.com/wgrathwohl/JEM
5.5.1 MA-EBM

We first evaluate the performance of MA-EBM on CIFAR10, CIFAR100, CelebA-HQ 128 and ImageNet 32x32. We utilize the Inception Score (IS) (Salimans et al. 2016) and Fréchet Inception Distance (FID) (Heusel et al. 2017) to evaluate the quality of generated images.

Table 5.1 Inception and FID scores of MA-EBM on CIFAR10.

<table>
<thead>
<tr>
<th>Model</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>MA-EBM(K=1)*</td>
<td>6.02</td>
<td>35.7</td>
</tr>
<tr>
<td>MA-EBM(K=2)</td>
<td>6.72</td>
<td>27.1</td>
</tr>
<tr>
<td>MA-EBM(K=5)</td>
<td>7.14</td>
<td>22.7</td>
</tr>
<tr>
<td>MA-EBM(K=10)</td>
<td>7.08</td>
<td>20.4</td>
</tr>
<tr>
<td>MA-EBM(K=20)</td>
<td>7.20</td>
<td>21.1</td>
</tr>
<tr>
<td>Explicit EBM(Unconditional)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShortRun(K=100)</td>
<td>6.72</td>
<td>32.1</td>
</tr>
<tr>
<td>IGEBM(K=60) (Du and Mordatch 2019)</td>
<td>6.78</td>
<td>38.2</td>
</tr>
<tr>
<td>CF-EBM(K=50) (Zhao et al. 2021)</td>
<td></td>
<td>16.7</td>
</tr>
<tr>
<td>KL-EBM(K=40) (Du et al. 2021)</td>
<td>7.85</td>
<td>25.1</td>
</tr>
<tr>
<td>DiffuRecov(K=30) (Gao et al. 2021)+</td>
<td>8.31</td>
<td>9.58</td>
</tr>
<tr>
<td>Regularized Generator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GEBM (Arbel and Zhou 2021)</td>
<td>-</td>
<td>23.02</td>
</tr>
<tr>
<td>VAEBM(K=6) (Xiao et al. 2021)†</td>
<td>8.43</td>
<td>12.19</td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNGAN (Miyato et al. 2018)</td>
<td>8.59</td>
<td>21.7</td>
</tr>
<tr>
<td>NCSN (Song and Ermon 2019)</td>
<td>8.91</td>
<td>25.3</td>
</tr>
<tr>
<td>StyleGAN2-ADA (Karras et al. 2020)</td>
<td>9.74</td>
<td>2.92</td>
</tr>
<tr>
<td>DDPM (Ho et al. 2020)</td>
<td>9.46</td>
<td>3.17</td>
</tr>
</tbody>
</table>

* MA-EBM diverges with $K = 1$, and we report the best FID before diverging.
+ DiffuRecov diverges with $K = 10$.
† VAEBM eventually explodes around 25,000 iterations (or 16 epochs) on CIFAR10.

The results are reported in Table 5.1 and 5.2, respectively. It can be observed that our method consistently surpasses existing methods in terms of sampling steps by a significant margin. On CIFAR10, MA-EBM outperforms many EBM approaches and SNGAN in terms of FID, while the performance is slightly worse than SNGAN on CIFAR100. Some EBM approaches show better
Table 5.2 FID results of MA-EBM on CIFAR100, CelebA-HQ 128, and ImageNet 32x32.

<table>
<thead>
<tr>
<th>Model</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR100 Unconditional</td>
<td></td>
</tr>
<tr>
<td>MA-EBM(K=1)*</td>
<td>45.5</td>
</tr>
<tr>
<td>MA-EBM(K=2)</td>
<td>26.2</td>
</tr>
<tr>
<td>MA-EBM(K=5)</td>
<td>27.2</td>
</tr>
<tr>
<td>MA-EBM(K=10)</td>
<td>26.9</td>
</tr>
<tr>
<td>SNGAN (Miyato et al. 2018)</td>
<td>22.4</td>
</tr>
<tr>
<td>LeCAM (Tseng et al. 2021)</td>
<td><strong>2.99</strong></td>
</tr>
<tr>
<td>CelebA-HQ 128x128 Unconditional</td>
<td></td>
</tr>
<tr>
<td>MA-EBM(K=5)*</td>
<td>57.76</td>
</tr>
<tr>
<td>MA-EBM(K=10)</td>
<td>39.87</td>
</tr>
<tr>
<td>KL-EBM(K=40) (Du et al. 2021)</td>
<td>28.78</td>
</tr>
<tr>
<td>SNGAN (Miyato et al. 2018)</td>
<td><strong>24.36</strong></td>
</tr>
<tr>
<td>ImageNet 32x32 Unconditional</td>
<td></td>
</tr>
<tr>
<td>MA-EBM(K=2)</td>
<td>54.52</td>
</tr>
<tr>
<td>MA-EBM(K=5)</td>
<td>52.71</td>
</tr>
<tr>
<td>IGEBM(K=60) (Du and Mordatch 2019)</td>
<td>62.23</td>
</tr>
<tr>
<td>KL-EBM(K=40) (Du et al. 2021)</td>
<td><strong>32.48</strong></td>
</tr>
</tbody>
</table>

* Our models diverge during training with given $K$, and we report the best FID before diverging.

performance, such VAEBM, CF-EBM and DiffuRecov. However, they requires an extra pretrained generator, or special architecture, or much larger sampling steps, while MA-EBM can train on a classical architecture as the backbone with least $K$. On ImageNet 32x32, we note that MA-EBM with $K = 2$ is incredibly stable and achieves FID 54.52 within 30 epochs and outperforms IGEBM. In addition, increasing sampling steps $K$ further doesn’t have an obvious improvement. Finally, on CelebA-HQ, MA-EBM is worse than baseline methods as we find it is less stable and requires more sampling steps due to the high resolution of CelebA-HQ. Nevertheless, our method builds a new solid baseline on different large-scale benchmarks for further investigations of EBM training in these more challenging tasks. Samples generated by MA-EBMs for CIFAR10, CIFAR100, and CelebA-HQ are shown in Fig. 5.3 and Fig. 5.1, respectively. The generated samples of ImageNet
Figure 5.3 MA-EBM generated samples of CIFAR10 and CIFAR100.

32x32 can be found in the appendix.

5.5.2 MA-JEM

We train MA-JEM on two benchmark datasets: CIFAR10 and CIFAR100, and compare its performance to the state-of-the-art hybrid models and some representative generative models. Table 5.3 and 5.4 report results on CIFAR10 and CIFAR100, respectively. As we can see, MA-JEM improves JEM’s image generation quality, stability, speed, and accuracy by a notable margin. It also boosts the IS and FID scores over MA-EBM. To the best of our knowledge, MA-JEM is the first EBM algorithm that demonstrates the utility of labels, which can dramatically improve the image generation quality of EBMs. On CIFAR100, IS and FID scores are not commonly reported by the state-of-the-art hybrid models, such as JEM (Grathwohl et al. 2020a), VERA (Grathwohl et al. 2021), and JEM++ (Yang and Ji 2021). Hence, our work again builds the first solid baseline for
hybrid modeling on CIFAR100 with a decent classification accuracy and image generation quality for future investigations. Images generated by MA-JEM for CIFAR10 and CIFAR100 are can be found in Fig. 5.4.

5.5.3 Analysis

5.5.3.1 Is Energy Head better than LSE?

To evaluate the effect of the energy head, we conduct an experiment comparing MA-JEM (with energy head) and LSE-JEM (without energy head) on CIFAR100. Fig. 5.5 shows that MA-JEM achieves much higher classification accuracy, comparable FID but lower Inception Score than LSE-JEM. However, we empirically find LSE-JEM is less stable than MA-JEM which leads us to analyze the manifolds learned by different models.
Table 5.3 Hybrid Modeling Results on CIFAR10.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc %</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>MA-JEM(K=1)*</td>
<td>78.4</td>
<td>7.91</td>
<td>29.8</td>
</tr>
<tr>
<td>MA-JEM(K=2)*</td>
<td>86.5</td>
<td>8.64</td>
<td>19.3</td>
</tr>
<tr>
<td>MA-JEM(K=5)</td>
<td>93.1</td>
<td>8.71</td>
<td>12.1</td>
</tr>
<tr>
<td>MA-JEM(K=10)</td>
<td>93.8</td>
<td>8.52</td>
<td><strong>11.5</strong></td>
</tr>
<tr>
<td>MA-JEM(K=20)</td>
<td><strong>94.2</strong></td>
<td>8.72</td>
<td>12.2</td>
</tr>
</tbody>
</table>

**Single Hybrid Model**

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc %</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residual Flow</td>
<td>70.3</td>
<td>3.60</td>
<td>46.4</td>
</tr>
<tr>
<td>IGEBM(K=60)</td>
<td>49.1</td>
<td>8.30</td>
<td>37.9</td>
</tr>
<tr>
<td>JEM(K=20)+</td>
<td>92.9</td>
<td>8.76</td>
<td>38.4</td>
</tr>
<tr>
<td>JEM++ (M=5)</td>
<td>91.1</td>
<td>7.81</td>
<td>37.9</td>
</tr>
<tr>
<td>JEM++ (M=10)</td>
<td>93.5</td>
<td>8.29</td>
<td>37.1</td>
</tr>
<tr>
<td>JEM++ (M=20)</td>
<td>94.1</td>
<td>8.11</td>
<td>38.0</td>
</tr>
<tr>
<td>JEAT</td>
<td>85.2</td>
<td>8.80</td>
<td>38.2</td>
</tr>
</tbody>
</table>

**Regularized Generator**

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc %</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERA (α=100)</td>
<td>93.2</td>
<td>8.11</td>
<td>30.5</td>
</tr>
<tr>
<td>VERA (α=1)</td>
<td>76.1</td>
<td>8.00</td>
<td>27.5</td>
</tr>
</tbody>
</table>

* MA-JEM diverges with given K. We report the best performance before the diverging of training.
+ JEM(K = 20) still suffers from high instability and regularly diverges before 60 epochs.

5.5.3.2 Manifold Analysis

To facilitate better understanding of different approaches, we utilize the t-SNE visualization for manifold analysis shown in Fig. 5.6a and 5.6b. For Fig 5.6a, three CIFAR10-trained MA-EBM, MA-JEM, and LSE-JEM with K = 10 are involved. We further conduct the comparison between MA-EBMs(K = 5) with and without energy $L_2$ regularization in Fig 5.6b. To have a fair comparison, we pick fixed samples from CIFAR10 as $x^+$, initialize samples from $p_0(x)$ as $x^0$, and randomly select samples from the replay buffer of each pre-trained models as $x^-$. We compute the output features of the penultimate layer and apply the t-SNE technique.

As we can observe in Fig. 5.6, MA-JEM with label information forms much denser manifolds for real data $x^+$ and generated samples $x^-$ than MAEBM. It gives us the potential explanation...
Table 5.4 Hybrid Modeling Results on CIFAR100.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc % ↑</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax (w/ BN)</td>
<td>78.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LeCAM</td>
<td>-</td>
<td>-</td>
<td>2.99</td>
</tr>
<tr>
<td>SNGAN(Cond)</td>
<td>-</td>
<td>9.30</td>
<td>15.6</td>
</tr>
<tr>
<td>aw-SNGAN(Cond)</td>
<td>-</td>
<td>9.48</td>
<td>14.42</td>
</tr>
<tr>
<td>BigGAN(Cond)</td>
<td>-</td>
<td>11.0</td>
<td>11.73</td>
</tr>
<tr>
<td>aw-BigGAN(Cond)</td>
<td>-</td>
<td>10.2</td>
<td>10.23</td>
</tr>
<tr>
<td>JEM(K=20)*</td>
<td>70.4</td>
<td>10.32</td>
<td>51.7</td>
</tr>
<tr>
<td>JEM(K=30)*</td>
<td>72.8</td>
<td>10.84</td>
<td>34.2</td>
</tr>
<tr>
<td>JEM++(K=5)*</td>
<td>72.0</td>
<td>8.19</td>
<td>37.7</td>
</tr>
<tr>
<td>JEM++(K=10)*</td>
<td>74.5</td>
<td>10.23</td>
<td>32.9</td>
</tr>
<tr>
<td>VERA(α=100)*</td>
<td>72.2</td>
<td>8.14</td>
<td>28.2</td>
</tr>
<tr>
<td>VERA(α=1)*</td>
<td>48.7</td>
<td>7.97</td>
<td>26.6</td>
</tr>
<tr>
<td>MA-JEM(K=1)+</td>
<td>46.5</td>
<td>8.71</td>
<td>26.2</td>
</tr>
<tr>
<td>MA-JEM(K=2)+</td>
<td>63.5</td>
<td>11.22</td>
<td>15.1</td>
</tr>
<tr>
<td>MA-JEM(K=5)</td>
<td>73.5</td>
<td><strong>11.95</strong></td>
<td>13.5</td>
</tr>
<tr>
<td>MA-JEM(K=10)</td>
<td><strong>75.1</strong></td>
<td>11.72</td>
<td><strong>12.7</strong></td>
</tr>
</tbody>
</table>

* No official IS and FID scores are reported.
+ We report the best FID before diverging.

about why label information can improve the generation quality. What’s more, the latent feature spaces of MA-JEM is better formulated than LSE-JEM, and there’s much less overlap between $x^0$ and $x^+$ which is desirable since $x^0$ and $x^+$ should have different energies. The distance between $x^0$ and $x^+$ shown in MA-EBM with regularization and MA-JEM indicates that we need a certain number of sampling steps $K$ to transit from $x^0$ to $x^+$, otherwise it is prone to diverge. However, we can also observe that the features of $x^0$ and $x^+$ are roughly mix together under the cases of an MA-EBM without regularization and a LSE-JEM. The mixing manifolds demonstrate that the instability can be caused by other mechanisms and we leave it for future exploration.

5.5.4 Training Speed

We report the empirical training speeds of our MA-JEM and baseline methods on a single Titan GPU in Table 4.4. As discussed previously, two mini-batches are utilized in MA-JEM: one for
Figure 5.5 Comparison between LSE-JEM and MA-JEM on CIFAR100. LSE-JEM diverges before 50 epochs while MA-JEM is more stable. LSE-JEM is JEM with our simplified initialization.

training of EBMs and the other one for training of classifiers. In our experiments, we set the mini-batch size to 64 for EBM training, but use 128 for the classification batch because it achieves the best accuracy. It can be observed that MA-EBM/JEM are the most stable EBM algorithms with the least number of MCMC sampling steps (e.g., $K = 2$ and $K = 5$) without divergence issues, leading to 4.87x and 2.22x speedups over the original JEM, respectively. Here, the acceleration comes from the reduced MCMC sampling steps without stability drop. Notably, Diffusion-based EBMs (Gao et al. 2021) and VAEBM (Xiao et al. 2021), requiring 8 GPUs/TPUs for days of training, are too expensive to be practical for a research lab in academia to afford.
Figure 5.6 t-SNE visualization of the latent feature spaces learned by different models trained on CIFAR10. a) shows the manifolds from MA-JEM are denser than MA-EBM and better formulated than LSE-JEM. b) shows The manifolds with regularization are denser and separate $x^+$ and $x^0$.

5.6 Conclusion

In this paper, we propose three training techniques to improve the image generation quality, classification accuracy, as well as training speed and stability of unconditional EBM and JEM altogether. The experimental results demonstrate that our models surpass prior state-of-the-arts significantly and enable us to scale the MCMC-based EBM learning to high resolution large scale image datasets, such as CelebA-HQ 128x128 and ImageNet 32x32.

As of future work, we are interested in building a theoretical understanding of the instability of EBM. We plan to explore the mechanism that hinders the accuracy of MA-JEM from the standard classifier. We are also interested in its application to other domains, such as NLP, time series, and discrete distribution.
Table 5.5 Run-time comparison on CIFAR10. We set 390 iterations as one epoch and the training epochs are 200 epochs. The batch size is 64 for all models except DiffuRecov and VAEBM.

<table>
<thead>
<tr>
<th>Model</th>
<th>Minutes per Epochs</th>
<th>Runtime (Hours)</th>
<th>Actual Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifier</td>
<td>0.77</td>
<td>2.6</td>
<td></td>
</tr>
<tr>
<td>JEM(K=20)*</td>
<td>15.1</td>
<td>50.3</td>
<td>1×</td>
</tr>
<tr>
<td>JEM++(K=5)</td>
<td>6.3</td>
<td>20.9</td>
<td>2.39×</td>
</tr>
<tr>
<td>JEM+++(K=10)</td>
<td>10.2</td>
<td>33.9</td>
<td>1.48×</td>
</tr>
<tr>
<td>VERA</td>
<td>9.6</td>
<td>32.2</td>
<td>2.8×†</td>
</tr>
<tr>
<td>MA-EBM (K=1)</td>
<td>2.4</td>
<td>7.9</td>
<td>6.29×</td>
</tr>
<tr>
<td>MA-EBM (K=2)</td>
<td>3.1</td>
<td>10.3</td>
<td>4.87×</td>
</tr>
<tr>
<td>MA-EBM (K=5)</td>
<td>5.4</td>
<td>18.0</td>
<td>2.79×</td>
</tr>
<tr>
<td>MA-EBM (K=10)</td>
<td>9.0</td>
<td>30.0</td>
<td>1.67×</td>
</tr>
<tr>
<td>MA-JEM (K=1)</td>
<td>3.8</td>
<td>12.7</td>
<td>3.97×</td>
</tr>
<tr>
<td>MA-JEM (K=2)</td>
<td>4.6</td>
<td>15.3</td>
<td>3.28×</td>
</tr>
<tr>
<td>MA-JEM (K=5)</td>
<td>6.8</td>
<td>22.7</td>
<td>2.22×</td>
</tr>
<tr>
<td>MA-JEM (K=10)</td>
<td>10.5</td>
<td>35.0</td>
<td>1.43×</td>
</tr>
<tr>
<td>IGEBM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KL-EBM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDPM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DiffuRecov</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VAEBM†</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* JEM(K=20) is much less stable than MA-EBM(K=2) and MA-JEM(K=5).
† VERA reports a 2.8× speedup while we run the official code and report a fair comparison results.
+ The runtime is for pretraining NVAE only. For VAEBM, They report the training takes around 25,000 iterations (or 16 epochs) on CIFAR-10 using one 32-GB V100 GPU. Then they cannot generate realistic samples anymore. No detail about the runtime.
CHAPTER 6
SHARPNESS-AWARE JOINT ENERGY-BASED MODELS

6.1 Abstract

Can we train a hybrid discriminative-generative model within a single network? This question has recently been answered in the affirmative, introducing the field of Joint Energy-based Model (JEM) (Grathwohl et al. 2020a; Yang and Ji 2021), which achieves high classification accuracy and image generation quality simultaneously. Despite recent advances, there remain two performance gaps: the accuracy gap to the standard softmax classifier, and the generation quality gap to state-of-the-art generative models. In this paper, we introduce a variety of training techniques to bridge the accuracy gap and the generation quality gap of JEM. 1) We incorporate a recently proposed sharpness-aware minimization (SAM) framework to train JEM, which promotes the energy landscape smoothness and the generalizability of JEM. 2) We exclude data augmentation from the maximum likelihood estimate pipeline of JEM, and mitigate the negative impact of data augmentation to image generation quality. Extensive experiments on multiple datasets demonstrate that our SAJEM achieves state-of-the-art performances and outperforms JEM in image classification, image generation, calibration, out-of-distribution detection and adversarial robustness by a notable margin.

6.2 Introduction

In recent years, deep neural networks (DNNs) have achieved state-of-the-art performances on a wide range of learning tasks, including image classification, image generation, object detection,
and language understanding (Krizhevsky and Hinton 2009; He et al. 2016). Among them, energy-based models (EBMs) have seen a flurry of interest, partly inspired by the impressive results of IGEBM (Du and Mordatch 2019) and JEM (Grathwohl et al. 2020a), which exhibit the capability of training generative models with a discriminative framework. Specifically, JEM (Grathwohl et al. 2020a) reinterprets the standard softmax classifier as an EBM and achieves impressive performances in image classification and generation simultaneously. Furthermore, these EBMs enjoy improved performance on out-of-distribution detection, calibration, and adversarial robustness. The follow-up works (e.g., (Yang and Ji 2021; Grathwohl et al. 2021)) further improve the training in terms of speed, stability and accuracy.

Figure 6.1 Visualizing the energy landscapes (Li et al. 2018) of different models trained on CIFAR10. Please note the dramatic differences of y-axis scales, indicating that SAJEM identifies the smoothest local optimal among all methods considered.

Despite the recent advances and the appealing property of training a single network for hybrid modeling, training JEM is still challenging on complex high-dimensional data since it requires an expensive MCMC sampling. In addition, JEM trained models still have an accuracy gap to the standard softmax classifier and a generation quality gap to the GAN-based approaches.

In this paper, we introduce a few simple yet effective training techniques to bridge the accu-
racy gap and generation quality gap of JEM. Our hypothesis is that both performance gaps are the symptoms of the generalizability of JEM trained models. We therefore analyze the trained models under the lens of loss geometry. Figure 6.1 visualizes the energy landscapes of different models by the technique introduced in (Li et al. 2018). Since different models are trained with different loss functions, visualizing their loss functions is not meaningful for the purpose of comparison. Therefore, the LSE energy functions (i.e., Eq. 6.4) of different models are visualized. Comparing Figure 6.1(a) and (b), we find that JEM converges to an extremely sharp local maxima of the energy landscape as manifested by the significantly large y-axis scale. By incorporating a recently proposed sharpness-aware optimizer (SAM) (Foret et al. 2021) to JEM, the energy landscape of trained model (JEM+SAM) becomes much smoother as shown in Figure 6.1(c). This also substantially improves the image classification accuracy and generation quality. To further improve the energy landscape smoothness, we exclude data augmentation from the maximum likelihood estimate pipeline of JEM, and visualize the energy landscape of SAJEM in Figure 6.1(d), which achieves the smoothest landscape among all the models considered. This further improves image generation quality dramatically while retaining or sometimes improving classification accuracy. Since our method improves the performance of JEM and JEM++ primarily in the framework of sharpness-aware optimization, we refer it as SAJEM, a Sharpness-Aware Joint Energy-based Model.

Our main contributions are summarized as follows:

1. We investigate the energy landscapes of different models and find that JEM leads to the sharpest one, which potentially undermines the generalizability of trained models.
2. We incorporate the sharpness-aware minimization (SAM) framework to JEM to promote the energy landscape smoothness, and thus model generalizability.

3. We recognize the negative impact of data augmentation in the training pipeline of JEM, and introduce two data loaders for image classification and image generation separately, which improves image generation quality significantly.

4. Extensive experiments on multiple datasets demonstrate that SAJEM achieves state-of-the-art discriminative and generative performances, while outperforming JEM in calibration, out-of-distribution detection and adversarial robustness by a notable margin.

6.3 Related Work

6.3.1 Energy-Based Models

Energy-based Models (EBMs) (LeCun et al. 2006) stem from the observation that any probability density function \( p_\theta(x) \) can be expressed via a Boltzmann distribution as

\[
p_\theta(x) = \frac{\exp \left( -E_\theta(x) \right)}{Z(\theta)},
\]

where \( E_\theta(x) \) is an energy function that maps input \( x \in \mathcal{X} \) to a scalar, and \( Z(\theta) = \int_{\mathcal{X}} \exp \left( -E_\theta(x) \right) \) is the normalizing constant w.r.t. \( x \) (also known as the partition function). Ideally, an energy function should assign low energy values to the samples drawn from data distribution, and high values otherwise.

The key challenge of EBM training is to estimate the intractable partition function \( Z(\theta) \), and thus the maximum likelihood estimate of parameters \( \theta \) is not straightforward. Prior works have de-
veloped a number of sampling-based approaches to approximate the partition function effectively. Specifically, the derivative of the log-likelihood of $x \in X$ w.r.t. $\theta$ can be expressed as

$$\frac{\partial \log p_{\theta}(x)}{\partial \theta} = \mathbb{E}_{p_{\theta}(x)} \left[ \frac{\partial E_{\theta}(x)}{\partial \theta} \right] - \mathbb{E}_{p_{d}(x)} \left[ \frac{\partial E_{\theta}(x)}{\partial \theta} \right], \quad (6.2)$$

where $p_{d}(x)$ is the real data distribution (i.e., training dataset), and $p_{\theta}(x)$ is the estimated probabilistic density function, sampling from which is challenging due to the intractable $Z(\theta)$.

To sample from $p_{\theta}(x)$ efficiently, MCMC and Gibbs sampling (Hinton 2002b) have been proposed. Furthermore, by utilizing the gradient information, recently Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh 2011) has been employed to speed up the sampling from EBM (Nijkamp et al. 2019b; Du and Mordatch 2019; Grathwohl et al. 2020a). Specifically, to sample from $p_{\theta}(x)$, the SGLD follows

$$x^{0} \sim p_{0}(x), \quad x^{t+1} = x^{t} - \alpha \frac{\partial E_{\theta}(x^{t})}{\partial x^{t}} + \alpha \epsilon^{t}, \quad \epsilon^{t} \sim \mathcal{N}(0, 1), \quad (6.3)$$

where $p_{0}(x)$ is typically a uniform distribution over $[-1, 1]$, whose samples are refined via a noisy gradient decent with step-size $\alpha$ over a sampling chain. A recently emergent direction, called Score matching (Hyvärinen 2005a; Swersky et al. 2011; Song and Ermon 2019; Song et al. 2020) learns EBM by matching the gradient of the log probability density of the model distribution to that of the data distribution, and also explores the behaviour of flat minima in generative modelling.
6.3.2 Joint Energy-based Models

Joint Energy-based Model (JEM) (Grathwohl et al. 2020a) reinterprets the standard softmax classifier as an EBM and trains a single network for hybrid discriminative-generative modeling. Specifically, Grathwohl et. al (Grathwohl et al. 2020a) were the first to recognize that the logits \( f_{\theta}(x)[y] \) from a standard softmax classifier can be considered as an energy function over \((x, y)\), and thus the joint density can be defined as \( p_{\theta}(x, y) = e^{f_{\theta}(x)[y]} / Z(\theta) \), where \( Z(\theta) \) is an unknown normalizing constant (regardless of \( x \) or \( y \)). Then the density of \( x \) can be derived by marginalizing over \( y \):

\[
p_{\theta}(x) = \sum_y p_{\theta}(x, y) = \sum_y e^{f_{\theta}(x)[y]} / Z(\theta).
\]

Subsequently, the corresponding energy function of \( x \) can be identified as

\[
E_{\theta}(x) = -\log \sum_y e^{f_{\theta}(x)[y]} = -\text{LSE}(f_{\theta}(x)), \tag{6.4}
\]

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

To optimize the model parameter \( \theta \), JEM maximizes the logarithm of joint density function \( p_{\theta}(x, y) \):

\[
\log p_{\theta}(x, y) = \log p_{\theta}(y|x) + \log p_{\theta}(x), \tag{6.5}
\]

where the first term denotes the cross-entropy objective for classification, and the second term can be optimized by the maximum likelihood learning of EBM as shown in Eq. 6.2.

However, JEM suffers from high training instability even with a large number of SGLD sampling steps \( K \) (e.g., \( K = 20 \)). After divergence, JEM requires to restart the SGLD sampling with a
doubled $K$. Recently, JEM++ (Yang and Ji 2021) proposes a number of new training techniques to improve JEM’s accuracy, training stability and speed altogether, including the proximal gradient clipping, YOPO-based SGLD sampling acceleration, and informative initialization. Furthermore, JEM++ enables batch norm (Ioffe and Szegedy 2015) in the backbone models, while IGEBM and JEM have to exclude batch norm due to the high training instability incurred by it.

### 6.3.3 Flat Minima and Generalization

A great number of previous works have investigated the relationship between the flatness of local minima and the generalizability of learned models (Li et al. 2018; Keskar et al. 2017; Wei et al. 2020; Chen et al. 2022; Foret et al. 2021; Kwon et al. 2021). Now it is widely accepted and empirically verified that flat minima tend to give better generalization performance. Based on these observations, several recent regularization techniques are proposed to search for the flat minima of loss landscapes (Wei et al. 2020; Chen et al. 2022; Foret et al. 2021; Kwon et al. 2021). Among them, the sharpness-aware minimization (SAM) (Foret et al. 2021) is a recently introduced optimizer that demonstrates promising performance across all kinds of models and tasks, such as ResNet (He et al. 2016), Vision Transformer (ViT) (Chen et al. 2022) and Language Modeling (Bahri et al. 2022). To the best of our knowledge, we are the first to explore the sharpness-aware optimization to improve the discriminative and generative performance of EBM.
6.4 SAJEM

6.4.1 Sharpness-Aware Minimization

To train a generalizable model, SAM (Foret et al. 2021) proposes to find model parameters $\theta$ whose entire neighborhoods have uniformly low training loss $L_{\text{train}}$ by optimizing a minimax objective:

$$
\min_{\theta} \max_{\|\epsilon\|_2 \leq \rho} L_{\text{train}}(\theta + \epsilon) + \lambda \|\theta\|_2^2,
$$

(6.6)

where $\rho$ is the size of the $L_2$-ball centered at model parameter $\theta$, and $\lambda$ is a hyperparameter for $L_2$ regularization on $\theta$. To solve the inner maximization problem, SAM employs the Taylor expansion to develop an efficient first-order approximation to the optimal $\epsilon^*$ as:

$$
\hat{\epsilon}(\theta) = \arg\max_{\|\epsilon\|_2 \leq \rho} L_{\text{train}}(\theta) + \epsilon^T \nabla_{\theta} L_{\text{train}}(\theta) = \rho \nabla_{\theta} L_{\text{train}}(\theta) / \|\nabla_{\theta} L_{\text{train}}(\theta)\|_2,
$$

(6.7)

which is a normalized gradient at the current model parameter $\theta$. After $\hat{\epsilon}$ is determined, SAM updates $\theta$ based on the gradient $\nabla_{\theta} L_{\text{train}}(\theta)|_{\theta + \hat{\epsilon}(\theta)} + 2\lambda \theta$ at an updated parameter location $\theta + \hat{\epsilon}$.

More recently, Kwon et al. (Kwon et al. 2021) propose the Adaptive SAM (ASAM) with the objective:

$$
\min_{\theta} \max_{\|\epsilon\|_2 \leq \rho} L_{\text{train}}(\theta + \epsilon) + \lambda \|\theta\|_2^2,
$$

(6.8)
where $T_\theta$ is an element-wise operator $T_\theta = \text{diag}(|\theta_1|, ..., |\theta_k|)$ with $\theta = [\theta_1, \theta_2, ..., \theta_k]$. Similar to SAM, the Taylor expansion is leveraged in ASAM to derive a first-order approximation to the optimal $\epsilon^*$ with $\hat{\epsilon}(\theta) = \rho T_\theta \text{sign} (\nabla L_{\text{train}}(\theta))$.

As we observed from Figure 6.1(a) and (b), JEM trained models converge to very sharp local optima, which potentially undermines the generalizability of JEM. We therefore incorporate the sharpness-aware optimization (SAM) framework to the original training pipeline of JEM (Grathwohl et al. 2020a) in order to improve the generalizability of trained models. Specifically, instead of a traditional maximum likelihood training, we optimize the joint density function of JEM in a minimax objective:

$$
\max_{\theta} \min_{\|\epsilon\|_2 \leq \rho} \log p_\theta(x, y) + \lambda \|\theta\|^2_2.
$$

For the outer maximization that involves $\log p_\theta(x)$, SGLD is again used to sample from $p_\theta(x)$ as in the original JEM.

### 6.4.2 Image Generation without Data Augmentation

Data augmentation is a critical technique for supervised deep learning and self-supervised contrastive learning (Krizhevsky et al. 2012; Chen et al. 2020). Not surprisingly, JEM also utilizes data augmentation in its training pipeline, such as horizontal flipping, random cropping, and padding. Specifically, let $T$ denote a data augmentation operator. The actual objective function of JEM is
which shows that JEM maximizes the likelihood function $p_\theta(T(x))$ rather than $p_\theta(x)$. From our empirical study, horizontal flipping has little impact on the FID scores, while cropping and padding play a bigger role because the generated images contain cropping and padding effects, which hurt the FID scores. This is consistent with GANs (Goodfellow et al. 2014), which observed that any augmentation that is applied to the training dataset will get inherited in the generated images. Based on this observation, we exclude the data augmentation from $p_\theta(T(x))$ and only retain the data augmentation for classification given its pervasive success in image classification. To this end, our final objective function of SAJEM becomes:

$$\log p_\theta(x, y) = \log p_\theta(y|T(x)) + \log p_\theta(T(x)),$$

where the first term is calculated using a mini-batch with data augmentation, and the second term is calculated using the mini-batch without data augmentation, which can be implemented efficiently by using two data loaders.

Algorithm 5 provides the pseudo-code of SAJEM training, which follows a similar design of JEM (Grathwohl et al. 2020a) and JEM++ (Yang and Ji 2021) with a replay buffer. For brevity, only one real sample and one generated sample are used to optimize the model parameter $\theta$. It is straightforward to generalize the pseudo-code below to a mini-batch setting, which we use in
Algorithm 5 SAJEM Training: Given network $f_\theta$, SGLD step-size $\alpha$, SGLD noise $\sigma$, SGLD steps $K$, replay buffer $B$, reinitialization frequency $\gamma$, SAM noise bound $\rho$, and learning rate $lr$

1: while not converged do
2: Sample $x^+$ and $y$ from dataset
3: Sample $\tilde{x}_0 \sim B$ with probability $1 - \gamma$, else $\tilde{x}_0 \sim p_0(x)$
4: for $t \in [1, 2, \ldots, K]$ do
5: $\tilde{x}_t = \tilde{x}_{t-1} - \alpha \cdot \frac{\partial E(\tilde{x}_{t-1})}{\partial \tilde{x}_{t-1}} + \sigma \cdot N(0, I)$
6: end for
7: $x^- = \text{StopGrad}(\tilde{x}_K)$
8: $L_{\text{gen}}(\theta) = E(x^+) - E(x^-)$
9: $L(\theta) = L_{\text{clf}}(\theta) + L_{\text{gen}}(\theta)$ with $L_{\text{clf}}(\theta) = \text{xent}(f_\theta(x), y)$
10: # Apply SAM optimizer as following:
11: Compute gradient $\nabla_{\theta} L(\theta)$ of the training loss
12: Compute $\hat{\epsilon}(\theta) = \rho$ as in Eq. 6.7
13: Compute gradient $g = \nabla_{\theta} L(\theta)|_{\theta + \hat{\epsilon}(\theta)}$
14: Update model parameters: $\theta = \theta - lr \cdot g$
15: Add $x^-$ to $B$
16: end while

the experiments. It’s worth mentioning that we adopt the Informative Initialization in JEM++ to initialize the Markov chain from $p_0(x)$, which enables the batch norm and plays a crucial role in the tradeoff between the number of SGLD sampling steps $K$ and overall performance, including the classification accuracy and training stability.

6.5 Experiments

In this section, we investigate the efficacy of SAJEM on several discriminative and generative tasks, including image classification, generation, calibration, out-of-distribution (OOD) detection, adversarial robustness and more. Our code is built on top of JEM++ (Yang and Ji 2021)\(^1\) (given its improved performance over JEM) and SAM\(^2\). We train SAJEM on two benchmark datasets, CIFAR10 and CIFAR100, with the backbone Wide-ResNet 28-10 (Zagoruyko and Komodakis

\(^1\)https://github.com/sndnyang/jempp
\(^2\)https://github.com/davda54/sam
For a fair comparison, our experiments closely follow the settings of JEM and JEM++. All our experiments are implemented using PyTorch on a single Nvidia RTX GPU. More details of the experimental setup are provided in the appendix. Our source code is also provided as a part of supplementary material.

### 6.5.1 Hybrid Modeling

Table 6.1 Results on CIFAR10.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc %</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAJEM (K=5)</td>
<td>93.8</td>
<td>8.90</td>
<td>11.2</td>
</tr>
<tr>
<td>SAJEM (K=10)</td>
<td>94.8</td>
<td>8.63</td>
<td>11.2</td>
</tr>
<tr>
<td>SAJEM (K=20)</td>
<td>95.9</td>
<td>8.40</td>
<td>13.1</td>
</tr>
</tbody>
</table>

Single Hybrid Model

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc %</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>IGEBM (K=60)</td>
<td>49.1</td>
<td>8.30</td>
<td>37.9</td>
</tr>
<tr>
<td>JEM (K=20)</td>
<td>92.9</td>
<td>8.76</td>
<td>38.4</td>
</tr>
<tr>
<td>JEM++ (M=5)</td>
<td>91.1</td>
<td>7.81</td>
<td>37.9</td>
</tr>
<tr>
<td>JEM++ (M=10)</td>
<td>93.5</td>
<td>8.29</td>
<td>37.1</td>
</tr>
<tr>
<td>JEM++ (M=20)</td>
<td>94.1</td>
<td>8.11</td>
<td>38.0</td>
</tr>
<tr>
<td>JEAT</td>
<td>85.2</td>
<td>8.80</td>
<td>38.2</td>
</tr>
</tbody>
</table>

Other EBMs

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc %</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShortRun (K=100)</td>
<td>-</td>
<td>6.72</td>
<td>32.1</td>
</tr>
<tr>
<td>CF-EBM (K=50)</td>
<td>-</td>
<td>-</td>
<td>16.7</td>
</tr>
<tr>
<td>ImCD (K=40)</td>
<td>-</td>
<td>7.85</td>
<td>25.1</td>
</tr>
<tr>
<td>DiffuRecov (K=30)</td>
<td>-</td>
<td>8.31</td>
<td>9.58</td>
</tr>
<tr>
<td>VAEBM (K=6)</td>
<td>-</td>
<td>8.43</td>
<td>12.2</td>
</tr>
<tr>
<td>VERA</td>
<td>93.2</td>
<td>8.11</td>
<td>30.5</td>
</tr>
</tbody>
</table>

Other Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc %</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax</td>
<td>96.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Softmax + SAM</td>
<td>97.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SNGAN</td>
<td>-</td>
<td>8.59</td>
<td>21.7</td>
</tr>
<tr>
<td>StyleGAN2-ADA</td>
<td>-</td>
<td>9.74</td>
<td>2.92</td>
</tr>
<tr>
<td>DDPM</td>
<td>-</td>
<td>9.46</td>
<td>3.17</td>
</tr>
</tbody>
</table>

Figure 6.2 Generated CIFAR10 samples from SAJEM.
criminative and generative models on CIFAR10 and CIFAR100. Inception Score (IS) (Salimans et al. 2016) and Fréchet Inception Distance (FID) (Heusel et al. 2017) are employed to evaluate the quality of generated images. The results on CIFAR10 and CIFAR100 are shown in Tables 6.1 and 6.2, respectively. SAJEM ($K = 10$) outperforms JEM ($K = 20$) and JEM++ ($M = 20$) in terms of accuracy (94.9%) and FID score (11.7) on CIFAR10. The improvement on CIFAR100 over JEM and JEM++ is also significant. Moreover, SAJEM is superior in training stability. SAJEM ($K = 5$) matches or outperforms the classification accuracy of JEM++ ($M = 20$), and in the meantime it exhibits high stability during training (let alone with larger $K$s) while JEM ($K = 20$) and JEM++ ($M = 5$) would be easily diverged at early epochs. One interesting phenomenon we observed from our experiments is that there is a trade-off between classification accuracy and image quality when increasing $K$, similar to the trade-off observed by JEM++ (Yang and Ji 2021). Example images generated by SAJEM for CIFAR10 and CIFAR100 are provided in figures/sajem 6.2 and 6.3, respectively.
Table 6.2 Hybrid Modeling Results on CIFAR100.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc % ↑</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAJEM (K=5)</td>
<td>75.0</td>
<td>11.63</td>
<td>14.4</td>
</tr>
<tr>
<td>SAJEM (K=10)</td>
<td>76.4</td>
<td>10.95</td>
<td>15.1</td>
</tr>
<tr>
<td>SAJEM (K=20)</td>
<td><strong>77.3</strong></td>
<td>10.78</td>
<td>19.9</td>
</tr>
<tr>
<td>JEM (K=20)*</td>
<td>72.2</td>
<td>10.22</td>
<td>38.1</td>
</tr>
<tr>
<td>JEM++ (M=5)*</td>
<td>72.1</td>
<td>8.05</td>
<td>38.9</td>
</tr>
<tr>
<td>JEM++ (M=10)*</td>
<td>74.2</td>
<td>9.97</td>
<td>34.5</td>
</tr>
<tr>
<td>JEM++ (M=20)*</td>
<td>75.9</td>
<td>10.07</td>
<td>33.7</td>
</tr>
<tr>
<td>VERA (α=100)*</td>
<td>72.2</td>
<td>8.25</td>
<td>29.5</td>
</tr>
<tr>
<td>VERA (α=1)*</td>
<td>48.7</td>
<td>7.84</td>
<td>25.1</td>
</tr>
<tr>
<td>Softmax</td>
<td>81.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Softmax + SAM</td>
<td>83.4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SNGAN</td>
<td>-</td>
<td>9.30</td>
<td>15.6</td>
</tr>
<tr>
<td>BigGAN</td>
<td>-</td>
<td>11.0</td>
<td><strong>11.7</strong></td>
</tr>
</tbody>
</table>

* No official IS and FID scores are reported. We run the official code with the default settings and report the results.

6.5.2 Calibration

While modern classifiers are growing more accurate, recent works show that their predictions could be over-confident due to increased model capacity (Guo et al. 2017a). Typically, the confidence of a model’s prediction can be defined as \( \max_y p(y|x) \) and is used to decide whether to output a prediction. However, incorrect but confident predictions can be catastrophic for safety-critical applications, which necessitates calibration of uncertainty especially for models of large capacity. As such, a well-calibrated but less accurate model can be considerably more useful than a more accurate but less-calibrated model.

In this experiment, all models are trained on the CIFAR10 dataset for a fair comparison. We compare the Expected Calibration Error (ECE) score (Guo et al. 2017a) of SAJEM to that of the
standard softmax classifier and JEM. We utilize the reliability diagram to visualize the discrepancy between the true probability and the confidence, with the results shown in Figure 4.8. We find that the model trained by SAJEM ($K = 10$) achieves a much smaller ECE (1.36% vs. 4.2% of JEM and 5.5% of softmax classifier), demonstrating SAJEM’s predictions are better calibrated than the competing methods. We further note that larger $K$ hurts the calibration quality slightly. Due to page limit, more results are relegated to the appendix.

![Figure 6.4 Calibration results on CIFAR10. The smaller ECE is, the better.](image)

Table 6.3 Histograms of $\log_\theta p(x)$ for OOD detection. Green corresponds to in-distribution dataset, while red corresponds to OOD dataset.
6.5.3 Out-Of-Distribution Detection

Formally, the OOD detection is a binary classification problem, which outputs a score \( s_\theta(x) \in \mathbb{R} \) for a given query \( x \). The model should be able to assign lower scores to OOD examples than to in-distribution examples such that it can be used to distinguish OOD examples from in-distribution ones. Following the settings of JEM (Grathwohl et al. 2020a), we use the Area Under the Receiver-Operating Curve (AUROC) (Hendrycks and Gimpel 2016) to evaluate the performance of OOD detection. In our experiments, two score functions are considered: the input density \( p_\theta(x) \) (Nalisnick et al. 2018), the predictive distribution \( p_\theta(y|x) \) (Hendrycks and Gimpel 2016).

Table 6.4 OOD detection results. Models are trained on CIFAR10. Values are AUROC.

<table>
<thead>
<tr>
<th>( s_\theta(x) )</th>
<th>Model</th>
<th>SVHN</th>
<th>CIFAR10</th>
<th>Interp</th>
<th>CIFAR100</th>
<th>CelebA</th>
</tr>
</thead>
<tbody>
<tr>
<td>log ( p_\theta(x) )</td>
<td>WideResNet</td>
<td>.91</td>
<td>-</td>
<td>.87</td>
<td>.78</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IGBM</td>
<td>.63</td>
<td>.70</td>
<td>.50</td>
<td>.70</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM (K=20)</td>
<td>.67</td>
<td>.65</td>
<td>.67</td>
<td>.75</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM++ (M=20)</td>
<td>.85</td>
<td>.57</td>
<td>.68</td>
<td>.80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VERA</td>
<td>.83</td>
<td>.86</td>
<td>.73</td>
<td>.33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ImCD</td>
<td>.91</td>
<td>.65</td>
<td>.83</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SAJEM (K=5)</td>
<td>.91</td>
<td>.79</td>
<td>.90</td>
<td>.82</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SAJEM (K=10)</td>
<td>.95</td>
<td>.81</td>
<td>.90</td>
<td>.88</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SAJEM (K=20)</td>
<td>.98</td>
<td>.83</td>
<td>.92</td>
<td>.95</td>
<td></td>
</tr>
<tr>
<td>max_{y} p_\theta(y</td>
<td>x)</td>
<td>WideResNet</td>
<td>.93</td>
<td>.77</td>
<td>.85</td>
<td>.62</td>
</tr>
<tr>
<td></td>
<td>IGBM</td>
<td>.43</td>
<td>.69</td>
<td>.54</td>
<td>.69</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM (K=20)</td>
<td>.89</td>
<td>.75</td>
<td>.87</td>
<td>.79</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM++ (M=20)</td>
<td>.94</td>
<td>.77</td>
<td>.88</td>
<td>.90</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SAJEM (K=5)</td>
<td>.92</td>
<td>.77</td>
<td>.88</td>
<td>.81</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SAJEM (K=10)</td>
<td>.93</td>
<td>.78</td>
<td>.89</td>
<td>.78</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SAJEM (K=20)</td>
<td>.96</td>
<td>.80</td>
<td>.91</td>
<td>.84</td>
<td></td>
</tr>
</tbody>
</table>

Input Density

We can use the input density \( p_\theta(x) \) as \( s_\theta(x) \). Intuitively, examples with low \( p(x) \) are considered to be OOD samples. Quantitative results can be found in Table 6.4 (top row), where CIFAR10 is
the in-distribution data and SVHN, an interpolated CIFAR10, CIFAR100 and CelebA are out-of-distribution data, respectively. Moreover, the corresponding visualization are shown in Table 6.3. As we can seen, SAJEM performs better in distinguishing the in-distribution samples from out-of-distribution samples, outperforming JEM, JEM++ and all the other models by significant margins.

Predictive Distribution

Another useful OOD score function is the maximum probability from a classifier’s predictive distribution: $s_\theta(x) = \max_y p_\theta(y|x)$. Hence, OOD performance using this score is highly correlated with a model’s classification accuracy. The results can be found in Table 6.4 (bottom row). Again, our SAJEM outperforms JEM and all the other models in majority of cases.

Table 6.4 also shows that JEM and JEM++ are even worse than a standard classifier in terms of OOD detection. Our observations suggest that when maximizing $p_\theta(T(x))$ an data augmentation $T$ is introduced in the training pipeline of JEM and JEM++, which is undesirable as it enlarges the span of the predicted in-distribution. In contrast, VERA, ImCD, and SAJEM exclude the data augmentation from their training pipelines that involve $p_\theta(x)$. Consistently, they all demonstrate improved OOD detection performance over JEM and JEM++.

6.5.4 Robustness

DNNs are known to be vulnerable to adversarial examples (Szegedy et al. 2014b; Goodfellow et al. 2015b) in the form of tiny but sensitive perturbations to the inputs leading the model to give incorrect predictions. To overcome the security threat posed by adversarial examples, a variety of defense algorithms have been proposed in the past few years to improve the robustness of mod-
els (Goodfellow et al. 2015a; Dziugaite et al. 2016; Guo et al. 2017b; Akhtar et al. 2018; Madry et al. 2018; Chiang et al. 2020). Existing works (Grathwohl et al. 2020a; Hill et al. 2021) have verified empirically that JEM is more robust than the standard trained softmax classifiers. Since SAJEM promotes the smoothness of energy landscape, it’s interesting to check if SAJEM can improve model robustness as well.

The white-box PGD attack (Madry et al. 2018) under an $L_\infty$ or $L_2$ constraint is the most common approach to evaluate the robustness of a given classifier. However, Athalye et al. (Athalye et al. 2018) find that the defense methods using gradient obfuscation always report overrated robustness, while the defense can be overcome with minor adjustments to the standard PGD attack. To evaluate the robustness of energy-based models, Mitch Hill et al. (Hill et al. 2021) propose the specific Expectation-Over-Transformation (EOT) attack and Backward Pass Differentiable Approximation (BPDA) attack.

In this section, we perform the same BPDA+EOT attack as in (Hill et al. 2021) to evaluate the model robustness, with the results reported in Figure 6.5. As we can see, SAJEM achieves a similar robustness as JEM under the $L_\infty$ and $L_2$ attacks, while both are more robust than the standard softmax classifiers. Moreover, we find that larger $K$ hurts the robustness significantly, although the clean accuracy is improved. Similar observation has been reported by Yao et al. (Zhu et al. 2021), who found that EBM learns a smooth energy function around real data by increasing the energy of SGLD-sampled points; however, a larger $K$ can generate samples of lower energy which are closer to real data distribution, and thus lead to a sharper energy landscape around real data after optimizing on both real data and generated data. As such, the models trained with larger
$K$ are less robust than the ones with smaller $K$.

Figure 6.5 Adversarial robustness under the PGD attacks.

![Figure 6.5 Adversarial robustness under the PGD attacks.](image)

(a) $L_{\infty}$ Robustness  
(b) $L_2$ Robustness

### 6.5.5 Mode Coverage

Interpolation

We show the interpolation between generated examples. Following the same procedure used in (Nijkamp et al. 2019b), we initialize $z_1, z_2$ from $p_0(z)$ and generate the sequence of interpolated noise $z_\alpha = \alpha z_1 + \sqrt{1 - \alpha^2} z_2$ where $\alpha \in [0, 1]$. Then we apply SGLD sampling with a CIFAR10-trained SAJEM. The top two rows in Figure 6.6 shows that the interpolation can generate realistic samples of different categories, which demonstrates the intra-class smoothness and inter-class mode coverage.
Overfitting

To further test mode coverage in SAJEM, we investigate the SGLD sampling on corrupted and clean CIFAR10 images. In other word, we show the capability of SAJEM for the inpainting and refinement tasks. The bottom two groups in Figure 6.6 show the final generated samples. By running Langevin dynamics on the images, we find the diversity of completions on images, indicating low overfitting on the training set and diversity characterized by likelihood models.

In summary, Figure 6.6 shows that SAJEM has good mode coverage and manifold smoothness. However, a large number of steps of sampling still lead to more saturated images. More details about the evolution of these experiments can be found in the appendix.
6.5.6 Ablation Study

In this section, we study the effect of sharpness-aware optimization (SAM) to SAJAM’s performance on image classification and image generation. The results are reported in Table 6.5. It can be observed that SAM can improve both the accuracy and generation quality of JEM/JEM++, while the improvements on classification accuracy are more pronounced. Secondly, by further excluding data augmentation operator \( T \) from \( p_\theta(T(x)) \) of JEM/JEM++, which leads to SAJAM, the FID scores have been improved dramatically from 35.0 to 11.2. Prior works on EBMs (Nijkamp et al. 2019b; Grathwohl et al. 2020a; Yang and Ji 2021) add data augmentation to their training pipelines to stabilizing the training. However, the data augmentation introduces the noise to the training images, which leads to foggy synthesized examples. What isn’t (and cannot be easily) shown here is that SAM also improves the training stability. As such, after excluding data augmentation, SAJEM is still very stable, whilst optimizing on \( p_\theta(x) \) without data augmentation improves image generation quality significantly. Then we study the effect of different \( \rho \)s for SAJEM in Table 6.6. After fine-tuning \( \rho \), we can get slightly better accuracy and FID scores.

<table>
<thead>
<tr>
<th>Ablation</th>
<th>Acc % ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>JEM</td>
<td>89.5</td>
<td>36.2</td>
</tr>
<tr>
<td>JEM + SAM</td>
<td>90.1</td>
<td>35.0</td>
</tr>
<tr>
<td>JEM++</td>
<td>93.5</td>
<td>37.1</td>
</tr>
<tr>
<td>JEM++ + SAM</td>
<td>94.1</td>
<td>36.6</td>
</tr>
<tr>
<td>JEM++ w/o DA</td>
<td>93.6</td>
<td>12.9</td>
</tr>
<tr>
<td>JEM++ w/o DA (L_2)*</td>
<td>93.4</td>
<td>-</td>
</tr>
<tr>
<td>SAJEM</td>
<td><strong>94.8</strong></td>
<td><strong>11.2</strong></td>
</tr>
</tbody>
</table>

* It fails to generate realistic images after 110 epochs.

Table 6.5 Ablation study of SAJEM. All the models are trained on CIFAR10 with \( K = 10 \).

<table>
<thead>
<tr>
<th>Ablation</th>
<th>Acc % ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASAM (( \rho = 0.5 ))</td>
<td>94.2</td>
<td>12.1</td>
</tr>
<tr>
<td>ASAM (( \rho = 1 ))</td>
<td>94.5</td>
<td>11.9</td>
</tr>
<tr>
<td>ASAM (( \rho = 2 ))</td>
<td>94.8</td>
<td><strong>11.2</strong></td>
</tr>
<tr>
<td>ASAM (( \rho = 4 ))</td>
<td>95.3</td>
<td>11.5</td>
</tr>
<tr>
<td>ASAM (( \rho = 8 ))</td>
<td>Diverged after 2nd epoch</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ablation</th>
<th>Acc % ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAM (( \rho = 0.05 ))</td>
<td>94.8</td>
<td>10.9</td>
</tr>
<tr>
<td>SAM (( \rho = 0.1 ))</td>
<td>95.3</td>
<td>11.4</td>
</tr>
<tr>
<td><strong>SAM (( \rho = 0.2 ))</strong></td>
<td><strong>96.1</strong></td>
<td>12.7</td>
</tr>
<tr>
<td>SAM (( \rho = 0.4 ))</td>
<td>95.1</td>
<td>14.1</td>
</tr>
<tr>
<td>SAM (( \rho = 0.8 ))</td>
<td>91.9</td>
<td>19.5</td>
</tr>
</tbody>
</table>

Table 6.6 Ablation Study of \( \rho \). All the models are trained on CIFAR10 with \( K = 10 \).
6.6 Limitations

It is challenging to train MCMC-based EBMs, including JEM, JEM++ and SAJEM, on complex high-dimensional data. IGEBM, JEM and many prior works have investigated methods to stabilize the training of EBM, but they require an extremely expensive MCMC sampling with a large $K$. Our SAJEM can stabilize the training on CIFAR10 and CIFAR100 with a small $K$ (e.g., $K = 5$). However, when the image resolution is scaling up, SAJEM has to increase $K$ accordingly. Hence, the trade-off between generation quality and computational complexity still limits the application of SAJEM to large-scale benchmarks, including ImageNet. Another negative effect is that our proposed method causes an environmental issue because it requires more computational resources (e.g., more GPU computations) due to using SAM.

6.7 Conclusion

We propose SAJEM to bridge the accuracy gap and the generation quality gap of JEM. By incorporating the sharpness-aware optimization (SAM) framework to JEM and excluding the undesirable data augmentation from the training pipeline of JEM, SAJEM promotes the energy landscape smoothness and hence the generalizability of trained models. Our experiments verify the effectiveness of these techniques with improvements on multiple benchmark datasets and demonstrate the state-of-the-art results in most of the tasks of image classification, generation, uncertainty calibration, OOD detection and adversarial robustness. As for the future work, we are interested in improving the scalability of EBM to large-scale benchmarks, such as ImageNet and NLP tasks.
CHAPTER 7
YOUR VIT IS SECRETLY A HYBRID DISCRIMINATIVE-GENERATIVE DIFFUSION MODEL

7.1 Abstract

Diffusion Denoising Probability Models (DDPM) (Ho et al. 2020) and Vision Transformer (ViT) (Dosovitskiy et al. 2021) have seen significant progress in generative tasks and discriminative tasks, respectively. Thus far these models have largely been developed in two separate domains. In this paper, we establish a direct connection between DDPM and ViT by integrating the ViT architecture into DDPM, and introduce a new generative model called Generative ViT (GenViT). The modeling flexibility of ViT enables us to further extend GenViT to hybrid discriminative-generative modeling, and introduce a Hybrid ViT (HybViT). Our work is among the first to exploit a single ViT for image generation and classification jointly. We conduct a series of experiments to analyze the performance of proposed models and demonstrate their superior performances over prior state-of-the-arts in both generative and discriminative tasks.

7.2 Introduction

Discriminative models and generative models based on the Convolutional Neural Network (CNN) (LeCun et al. 1989) architectures, such as GAN (Goodfellow et al. 2014) and ResNet (He et al. 2016), have achieved state-of-the-art performance in a wide range of learning tasks. Thus far they have largely been developed in two separate domains. In recent years, ViTs have started to rival CNNs in many vision tasks. Unlike CNNs, ViTs can capture the features from an entire image by self-attention, and they have demonstrated superiority in modeling non-local contextual dependencies
as well as their efficiency and scalability to achieve comparable classification accuracy with smaller computational budgets (measured in FLOPs). Since the inception, ViTs have been exploited in various tasks such as object detection (Carion et al. 2020), video recognition (Bertasius et al. 2021), multi-modal pre-training (Kim et al. 2021), and image generation (Jiang et al. 2021; Lee et al. 2022). Especially, VQ-GAN (Esser et al. 2021), TransGAN (Jiang et al. 2021) and ViTGAN (Lee et al. 2022) investigate the application of ViT in image generation. However, VQ-GAN is built upon an extra CNN-based VQ-VAE, and the latter two require two ViTs to construct a GAN for generation tasks. Therefore we ask the following question: is it possible to train a generative model using a single ViT?

DDPM is a class of generative models that matches a data distribution by learning to reverse a multi-step diffusion process. It has recently been shown that DDPMs can even outperform prior SOTA GAN-based generative models (Dhariwal and Nichol 2021; Brock et al. 2019a; Karras et al. 2020). Unlike GAN which needs to train with two competing networks, DDPM utilizes a UNet (Ronneberger et al. 2015) as a backbone for image generation and is trained to optimize maximum likelihood to avoid the notorious instability issue in GAN (Miyato et al. 2018; Brock et al. 2019a) and EBM (Du and Mordatch 2019; Grathwohl et al. 2020a).

In this paper, we establish a direct connection between DDPM and ViT for the task of image generation and classification. Specifically, we answer the question whether a single ViT can be trained as a generative model. We design Genenerative ViT (GenViT) for pure generation tasks, as well as Hybrid ViT (HybViT) that extends GenViT to a hybrid model for both image classification and generation. As shown in Fig 7.2 and 7.3, the reconstruction of image patches and the classifi-
cation are two routines independent to each other and train a shared set of features together. Our experiments show that HybViT outperforms previous state-of-the-art hybrid models. In particular, the Joint Energy-based Model (JEM), the previous state-of-the-art proposed by (Grathwohl et al. 2020a; Yang and Ji 2021), requires extremely expensive MCMC sampling, which introduce instability and causes the training processes to fail for large-scale datasets due to the long training procedures required. To the best of our knowledge, GenViT is the first model that utilizes a single ViT as a generative model, and HybViT is a new type of hybrid model without the expensive MCMC sampling during training. Compared to existing methods, our new models demonstrate a number of conceptual advantages (Du and Mordatch 2019): 1) Our methods provide simplicity and stability similar to DDPM, and are less prone to collapse compared to GANs and EBM. 2) The generative and discriminative paths of our model are trained with a single objective which enables sharing of statistical strengths. 3) Advantageous computational efficiency and scalability to growing model and data sizes inherited from the ViT backbone.

Our contributions can be summarized as following:

1. We propose GenViT, which to the best of our knowledge, is the first approach to utilize a single ViT as an alternative to the UNet in DDPM.

2. We introduce HybViT, a new hybrid approach for image classification and generation leveraging ViT, and show that HybViT considerably outperforms the previous state-of-the-art hybrid models on both classification and generation tasks while at the same time optimizes more effectively than MCMC-based models such as JEM/JEM++. 
3. We perform comprehensive analysis on model characteristics including adversarial robustness, uncertainty calibration, likelihood and OOD detection, comparing GenViT and Hybrid ViT with existing benchmarks.

7.3 Related Work

7.3.1 Denoising Diffusion Probabilistic Models

We first review the derivation of DDPM (Ho et al. 2020). DDPM is built upon the theory of Nonequilibrium Thermodynamics (Sohl-Dickstein et al. 2015) with a few simple yet effective assumptions. It assumes diffusion is a noising process $q$ that accumulates isotropic Gaussian noises over timesteps (Figure 7.1).

![Figure 7.1 A graphical model of diffusion process.](image)

Starting from the data distribution $x_0 \sim q(x_0)$, the diffusion process $q$ produces a sequence of latents $x_1$ through $x_T$ by adding Gaussian noise at each time $t \in [0, \ldots, T - 1]$ with variance $\beta_t \in (0, 1)$ as follows:

$$q(x_1, \ldots, x_T | x_0) := \prod_{t=1}^{T} q(x_t | x_{t-1})$$ (7.1)

$$q(x_t | x_{t-1}) := \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t \mathbf{I})$$ (7.2)
Then, the process in reverse aims to get a sample in $q(x_0)$ from sampling $x_T \sim \mathcal{N}(0, \mathbf{I})$ by using a neural network:

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

(7.3)

With the approximation of $q$ and $p$, (Ho et al. 2020) get a variational lower bound (VLB) as follows:

$$\log p_\theta(x_0) \geq \log p_\theta(x_0) - D_{KL}(q(x_{1:T}|x_0) \| p_\theta(x_0:T))$$

$$= -\mathbb{E}_q \left[ \frac{q(x_{1:T}|x_0)}{p_\theta(x_0:T)} \right]$$

(7.4)

Then they derive a loss for VLB as:

$$L_{vlb} = L_0 + L_1 + \ldots + L_{T-1} + L_T$$

(7.5)

$$L_0 = -\log p_\theta(x_0|x_1)$$

(7.6)

$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t))$$

(7.7)

$$L_T = D_{KL}(q(x_T|x_0) \| p(x_T))$$

(7.8)

where $L_0$ is modeled by an independent discrete decoder from the Gaussian $\mathcal{N}(x_0; \mu_\theta(x_1, 1), \sigma_1^2 \mathbf{I})$, and $L_T$ is constant and can be ignored.

As noted in (Ho et al. 2020), the forward process can sample an arbitrary timestep $x_t$ directly conditioned on the input $x_0$ in a closed form. With the nice property, we define $\alpha_t := 1 - \beta_t$ and
Then we have

\[
q(x_t | x_0) = N(x_t; \sqrt{\alpha_t}x_0, (1 - \alpha_t)I) \tag{7.9}
\]

\[
x_t = \sqrt{\alpha_t}x_0 + \sqrt{1 - \alpha_t}\epsilon \tag{7.10}
\]

where \(\epsilon \sim N(0, I)\) using the reparameterization. Then using Bayes theorem, we can calculate the posterior \(q(x_{t-1} | x_t, x_0)\) in terms of \(\tilde{\beta}_t\) and \(\tilde{\mu}_t(x_t, x_0)\) as follows:

\[
q(x_{t-1} | x_t, x_0) = N(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t I) \tag{7.11}
\]

\[
\tilde{\mu}_t(x_t, x_0) := \sqrt{\frac{\alpha_{t-1}}{1 - \alpha_t}}x_0 + \frac{\sqrt{\alpha_t(1 - \alpha_{t-1})}}{1 - \alpha_t}x_t \tag{7.12}
\]

\[
\tilde{\beta}_t := \frac{1 - \alpha_{t-1}}{1 - \alpha_t} \beta_t \tag{7.13}
\]

As we can observe, the objective in Eq. 7.5 is a sum of independent terms \(L_{t-1}\). Using Eq. 7.10, we can sample from an arbitrary step of the forward diffusion process and estimate \(L_{t-1}\) efficiently. Hence, (Ho et al. 2020) uniformly sample \(t\) for each sample in each mini-batch to approximate the expectation \(E_{x_0, t, \epsilon}[L_{t-1}]\) to estimate \(L_{\text{vib}}\).

To parameterize \(\mu_\theta(x_t, t)\) for Eq. 7.12, we can predict \(\mu_\theta(x_t, t)\) directly with a neural network. Alternatively, we can first use Eq. 7.10 to replace \(x_0\) in Eq. 7.12 to predict the noise \(\epsilon\) as

\[
\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \epsilon_\theta(x_t, t) \right), \tag{7.14}
\]
(Ho et al. 2020) found that predicting the noise $\epsilon$ worked best with a reweighted loss function:

$$L_{\text{simple}} = E_{t,x_0,\epsilon} \left[ ||\epsilon - \epsilon_\theta(x_t, t)||^2 \right].$$  \hfill (7.15)

This objective can be seen as a reweighted form of $L_{\text{vib}}$ (without the terms affecting $\Sigma_\theta$). For more details of the training and inference, we refer the readers to (Ho et al. 2020). A closely related branch is called score matching (Song and Ermon 2019; Song et al. 2020), which builds a connection bridging DDPMs and EBMs. Our work is mainly built upon DDPM, but it’s straightforward to substitute DDPM with a score matching method.

### 7.4 Vision Transformers

Transformers (Vaswani et al. 2017) have made huge impacts across many fields of deep learning (Han et al. 2020a) due to their prediction power and flexibility. They are based on the concept of self-attention, a function that allows interactions with strong gradients between all inputs, irrespective of their spatial relationships. The self-attention layer (Eq. 7.16) encodes inputs as key-value pairs, where values $V$ represent embedded inputs and keys $K$ act as an indexing method, and subsequently, a set of queries $Q$ are used to select which values to observe. Hence, a single self-attention head is computed as:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V.$$  \hfill (7.16)

where $d_k$ is the dimension of $K$. 
Vision transformers (ViT) ViT2021 has emerged as a famous architecture that outperforms CNNs in various vision domains. The transformer encoder is constructed by alternating layers of multi-headed self-attention (MSA) and MLP blocks (Eq. 7.18, 7.19), and layernorm (LN) is applied before every block, followed by residual connections after every block (Wang et al. 2019; Baevski and Auli 2019). The MLP contains two layers with a GELU non-linearity. The 2D image \( x \in \mathbb{R}^{H \times W \times C} \) is flattened into a sequence of image patches, denoted by \( x_p \in \mathbb{R}^{L \times (P^2 \cdot C)} \), where \( L = \frac{H \times W}{P^2} \) is the effective sequence length and \( P \times P \times C \) is the dimension of each image patch.

Following BERT (Devlin et al. 2019), we prepend a learnable classification embedding \( x_{\text{class}} \) to the image patch sequence, then the 1D positional embeddings \( E_{\text{pos}} \) are added to formulate the
patch embedding $z_0$. The overall pipeline of ViT is shown as follows:

$$z_0 = [x_{\text{class}}; x^1_p E; x^2_p E; \cdots; x^N_p E] + E_{\text{pos}},$$  

(7.17)

$$E \in \mathbb{R}^{(P^2 \cdot C) \times D}, E_{\text{pos}} \in \mathbb{R}^{(N+1) \times D}$$

$$z'_\ell = \text{MSA}(\text{LN}(z_{\ell-1})) + z_{\ell-1}, \ \ell = 1 \ldots L$$  

(7.18)

$$z_\ell = \text{MLP}(\text{LN}(z'_\ell)) + z'_\ell, \ \ell = 1 \ldots L$$  

(7.19)

$$y = \text{LN}(z_0^L)$$  

(7.20)

ViT have made significant breakthroughs in various discriminative tasks and generative tasks, including image classification, multi-modal, and high-quality image and text generation (Dosovitskiy et al. 2021; Dou et al. 2022; Lee et al. 2022). Inspired by the parallelism between patches/embeddings of ViT, we experiment with applying a standard ViT directly to generative modeling, with the minimal possible modifications.

7.4.1 Hybrid models

Hybrid models (Raina et al. 2004) commonly model the density function $p(x)$ and perform discriminative classification jointly using shared features. Notable examples are (Diederik Kingma and Welling 2014; Chongxuan et al. 2017; Nalisnick et al. 2019b; Grathwohl et al. 2020a, 2021; Arbel and Zhou 2021).

Hybrid models can utilize two or more classes of generative model to balance the trade-off such as slow sampling and poor scalability with dimension. For example, VAE can be increased by applying a second generative model such as a Normalizing Flow (Kingma et al. 2016; Grathwohl
et al. 2019; Vahdat and Kautz 2020) or EBM (Pang et al. 2020a) in latent space. Alternatively, a second model can be used to correct samples (Xiao et al. 2021). In our work, we focus on training a single ViT as a hybrid model without the auxiliary model.

7.4.2 Energy-Based Models

Energy-based models (EBMs) are an appealing family of models to represent data as they permit unconstrained architectures. Implicit EBMs define an unnormalized distribution over data typically learned through contrastive divergence (Du and Mordatch 2019; Hinton 2002b).

Joint Energy-based Model (JEM) (Grathwohl et al. 2020a) reinterprets the standard softmax classifier as an EBM and trains a single network to achieve impressive hybrid discriminative-generative performance. Beyond that, JEM++ (Yang and Ji 2021) proposes several training techniques to improve JEM’s accuracy, training stability, and speed, including proximal gradient clipping, YOPO-based SGLD sampling, and informative initialization. Unfortunately, training EBMs using SGLD sampling is still impractical for high-dimensional data.

7.5 Method

7.5.1 A Single ViT is a Generative Model

We propose GenViT by substituting UNet the backbone of DDPM with a single ViT. In our model design, we follow the standard ViT (Dosovitskiy et al. 2021) as close as possible. An overview of the architecture of the proposed GenViT is depicted in Fig 7.2.

Given the input $x_t$ from DDPM, we follow the raster scan to get a sequence of image patches
\( x_p \), which is fed into GenViT as:

\[
\begin{align*}
  h_0 &= [x_{\text{class}}; x_p^1E; x_p^2E; \cdots; x_p^NE] + E_{\text{pos}}, \\
  E &\in \mathbb{R}^{(P^2-C) \times D}, \ E_{\text{pos}} \in \mathbb{R}^{(N+1) \times D} \\
  h_\ell' &= \text{MSA}(\text{LN}(M(h_{\ell-1}, A))) + h_{\ell-1}, \ \ell = 1, \ldots, L \\
  h_\ell &= \text{MLP}(\text{LN}(M(h_\ell', A))) + h_\ell', \ \ell = 1, \ldots, L \\
  y &= h_L = [y^1, \cdots, y^N] \\
  x' &= [x_p^1, \cdots, x_p^N] = [f_r(y^1), \cdots, f_r(y^N)], \\
  x'_p &\in \mathbb{R}^{P^2 \times C}, x' \in \mathbb{R}^{H \times W \times C}.
\end{align*}
\]

Different from ViT, GenViT takes the embedding of \( t \) as input to control the hidden features \( h_\ell \) every layer, and finally reconstruct the \( L \)-th layer output \( h_L \in \mathbb{R}^{(N+1) \times D} \) to an image \( x' \). Following the design of UNet in DDPM, we first compute the embedding of \( t \) using an MLP \( A = \text{MLP}_t(t) \). Then we compute \( M(h_\ell, A) = h_\ell * (\mu_\ell(A) + 1) + \sigma_\ell(A) \) for each layer, where \( \mu_\ell(A) = \text{MLP}_\ell(A) \).

### 7.5.2 ViT as a Hybrid Model

JEM reinterprets the standard softmax classifier as an EBM and trains a single network for hybrid discriminative-generative modeling. Specifically, JEM maximizes the logarithm of joint density function \( p_\theta(x, y) \):

\[
\log p_\theta(x, y) = \log p_\theta(y|x) + \log p_\theta(x), \quad (7.22)
\]
where the first term is the cross-entropy classification objective, and the second term can be optimized by the maximum likelihood learning of EBM using contrastive divergence and MCMC sampling. However, MCMC-based EBM is notorious due to the expensive $K$-step MCMC sampling that requires $K$ full forward and backward propagations at every iteration. Hence, removing the MCMC sampling in training is a promising direction (Grathwohl et al. 2021).

We propose Hybrid ViT (HybViT), a simple framework to extend GenViT for hybrid modeling. We substitute the optimization of $\log p_\theta(x)$ in Eq. 4.5 with the VLB of GenViT as Eq. 7.4. Hence, we can train $p(y|x)$ using standard cross-entropy loss and optimize $p(x)$ using $L_{\text{simple}}$ loss in Eq 7.15. The final loss of our HybViT is

$$L = L_{\text{CE}} + \alpha L_{\text{simple}}$$

$$= E_{x_0,y} [H(x_0, y)] + \alpha E_{t,x_0,\epsilon} [||\epsilon - \epsilon_\theta(x_t, t)||^2]$$

We empirically find that a larger $\alpha = 100$ improves the generation quality while retaining comparable classification accuracy. The training pipeline can be viewed in Fig 7.3.

### 7.6 Experiments

This section evaluates the discriminative and generative performance on multiple benchmark datasets, including CIFAR10, CIFAR100, STL10, CelebA-HQ-128, Tiny-ImageNet, and ImageNet 32x32.

Our code is largely built on top of ViT (Lee et al. 2021)

1. https://github.com/aanna0701/SPT_LSA_ViT
Figure 7.3 The pipeline of HybViT.

390 iterations for CNN-based methods\(^3\). Most experiments of ViTs run for 500 epochs, but 2500 epochs for STL10 and 100 epochs for ImageNet 32x32. Thanks to the memory efficiency of ViT, all our experiments can be performed with PyTorch on a single Nvidia GPU. For reproducibility, our source code is provided in the supplementary material.

Figure 7.4 GenViT Generated samples of CIFAR10 and CelebA 128.

\(^3\)ViT-based models use 3× repeated augmentations (Touvron \textit{et al.} 2021)
7.6.1 Hybrid Modeling

We first compare the performance with the state-of-the-art hybrid models, stand-alone discriminative and generative models on CIFAR10. We use accuracy, Inception Score (IS) (Salimans et al. 2016) and Fréchet Inception Distance (FID) (Heusel et al. 2017) as evaluation metrics. IS and FID are employed to evaluate the quality of generated images. The results on CIFAR10 are shown in Tables 7.1. HybViT outperforms other hybrid models like JEM ($K = 20$) and JEM++ ($M = 20$) on accuracy (95.9%) and FID score (26.4) on CIFAR10, when the original ViT achieves comparable accuracy to WideResNet(WRN) 28-10. Moreover, GenViT and HybViT are superior in training stability. HybViT matches or outperforms the classification accuracy of JEM++ ($M = 20$), and in the meantime, it exhibits high stability during training while JEM ($K = 20$) and JEM++ ($M = 5$) would easily diverge at early epochs. The comparison results on more benchmark datasets, including CIFAR100, STL10, CelebA-128, Tiny-ImageNet, ImageNet 32x32 are shown in Table 7.2. Example images generated by GenViT and HybViT are provided in Figures 7.4 and 7.5, respectively. More generated images can be found in the appendix.
Table 7.1 Results on CIFAR10.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc %</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViT</td>
<td>96.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GenViT</td>
<td>-</td>
<td>8.17</td>
<td>20.2</td>
</tr>
<tr>
<td>HybViT</td>
<td>95.9</td>
<td>7.68</td>
<td>26.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Single Hybrid Model</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IGEBM</td>
<td>49.1</td>
<td>8.30</td>
<td>37.9</td>
</tr>
<tr>
<td>JEM</td>
<td>92.9</td>
<td>8.76</td>
<td>38.4</td>
</tr>
<tr>
<td>JEM++ (M=20)</td>
<td>94.1</td>
<td>8.11</td>
<td>38.0</td>
</tr>
<tr>
<td>JEAT</td>
<td>85.2</td>
<td>8.80</td>
<td>38.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generative Models</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SNGAN</td>
<td>-</td>
<td>8.59</td>
<td>21.7</td>
</tr>
<tr>
<td>StyleGAN2-ADA</td>
<td>-</td>
<td><strong>9.74</strong></td>
<td><strong>2.92</strong></td>
</tr>
<tr>
<td>DDPM</td>
<td>-</td>
<td>9.46</td>
<td>3.17</td>
</tr>
<tr>
<td>DiffuEBM</td>
<td>-</td>
<td>8.31</td>
<td>9.58</td>
</tr>
<tr>
<td>VAEBM</td>
<td>-</td>
<td>8.43</td>
<td>12.2</td>
</tr>
<tr>
<td>FlowEBM</td>
<td>-</td>
<td>-</td>
<td>78.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other Models</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WRN-28-10</td>
<td>96.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VERA (w/ generator)</td>
<td>93.2</td>
<td>8.11</td>
<td>30.5</td>
</tr>
</tbody>
</table>

We compare with results reported by SNGAN (Miyato et al. 2018), StyleGAN2-ADA (Karras et al. 2020), DDPM (Ho et al. 2020), DiffuEBM (Gao et al. 2021), VAEBM (Xiao et al. 2021), VERA (Grathwohl et al. 2021), FlowEBM (Nijkamp et al. 2022).

It’s worth mentioning that the overall quality of synthesis is worse than UNet-based DDPM. In particular, our methods don’t generate realistic images for complex and high-resolution data. ViT is known to model global relations between patches and lack of local inductive bias. We hope advances in ViT architectures and DDPM may address these issues in future work, such as Performer (Choromanski et al. 2021), Swin Transformer (Liu et al. 2021), CvT (Wu et al. 2021).
Figure 7.5 HybViT Generated samples of CIFAR10 and STL10.

and Analytic-DPM (Bao et al. 2022).
Table 7.2 Results on STL10, CelebA 128, Tiny-ImageNet, and ImageNet 32x32.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Acc % ↑</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR100</td>
<td>ViT</td>
<td>77.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>GenViT</td>
<td>8.19</td>
<td>26.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HybViT</td>
<td>77.4</td>
<td>7.45</td>
<td>33.6</td>
</tr>
<tr>
<td></td>
<td>WRN-28-10</td>
<td>79.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SNGAN</td>
<td>-</td>
<td>9.30</td>
<td>15.6</td>
</tr>
<tr>
<td></td>
<td>BigGAN</td>
<td>-</td>
<td>11.0</td>
<td>11.7</td>
</tr>
<tr>
<td>Tiny-ImageNet</td>
<td>ViT</td>
<td>57.6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>GenViT</td>
<td>-</td>
<td>7.81</td>
<td>66.7</td>
</tr>
<tr>
<td></td>
<td>HybViT</td>
<td>56.7</td>
<td>6.79</td>
<td>74.8</td>
</tr>
<tr>
<td></td>
<td>PreactResNet18</td>
<td>55.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>ADC-GAN</td>
<td>-</td>
<td>-</td>
<td>19.2</td>
</tr>
<tr>
<td>STL10</td>
<td>ViT</td>
<td>84.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>GenViT</td>
<td>7.92</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HybViT</td>
<td>80.8</td>
<td>7.87</td>
<td>109</td>
</tr>
<tr>
<td></td>
<td>WRN-16-8</td>
<td>76.6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SNGAN</td>
<td>-</td>
<td>-</td>
<td>9.10</td>
</tr>
<tr>
<td>ImageNet 32x32</td>
<td>ViT</td>
<td>57.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>GenViT</td>
<td>-</td>
<td>7.37</td>
<td>41.3</td>
</tr>
<tr>
<td></td>
<td>HybViT</td>
<td>53.5</td>
<td>6.66</td>
<td>46.4</td>
</tr>
<tr>
<td></td>
<td>WRN-28-10</td>
<td>59.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>IGEBM</td>
<td>-</td>
<td>5.85</td>
<td>62.2</td>
</tr>
<tr>
<td></td>
<td>KL-EBM</td>
<td>-</td>
<td>8.73</td>
<td>32.4</td>
</tr>
<tr>
<td>CelebA 128</td>
<td>GenViT</td>
<td>-</td>
<td>-</td>
<td>22.07</td>
</tr>
<tr>
<td></td>
<td>KL-EBM</td>
<td>-</td>
<td>-</td>
<td>28.78</td>
</tr>
<tr>
<td></td>
<td>SNGAN</td>
<td>-</td>
<td>-</td>
<td>24.36</td>
</tr>
</tbody>
</table>

IGEBM (Du and Mordatch 2019), KL-EBM (Du et al. 2021), SNGAN (Miyato et al. 2018), BigGAN (Brock et al. 2019a), ADC-GAN (Hou et al. 2022), UNet GAN (Schonfeld et al. 2020).

### 7.6.2 Model Evaluation

In this section, we conduct a thorough evaluation of proposed methods beyond the accuracy and generation quality. Note that it is not our intention to propose approaches to match or outperform
the best models in all metrics.

7.6.2.1 Calibration

Recent works show that the predictions of modern convolutional neural networks could be over-confident due to increased model capacity (Guo et al. 2017a). Incorrect but confident predictions can be catastrophic for safety-critical applications. Hence, we investigate ViT and HybViT in terms of calibration using the metric Expected Calibration Error (ECE). Interestingly, Fig 4.8 shows that predictions of both HybViT and ViT look like well-calibrated when trained with strong augmentations, however they are less confident and have worse ECE compared to WRN. More comparison results can be found in the appendix.

![Accuracy: 93.36% ECE: 5.5%](a) WRN  ![Accuracy: 96.56% ECE: 8.53%](b) ViT  ![Accuracy: 95.91% ECE: 9.50%](c) HybViT

Figure 7.6 Calibration results on CIFAR10. The smaller ECE is, the better. However, ViTs are better calibrated.

7.6.2.2 Out-of-Distribution Detection

Determining whether inputs are out-of-distribution (OOD) is an essential building block for safely deploying machine learning models in the open world. The model should be able to assign lower scores to OOD examples than to in-distribution examples such that it can be used to distinguish OOD examples from in-distribution ones. For evaluating the performance of OOD de-
tection, we use a threshold-free metric, called Area Under the Receiver-Operating Curve (AU-ROC) (Hendrycks and Gimpel 2016). Using the input density $p_\theta(x)$ (Nalisnick et al. 2018) as the score, ViTs performs better in distinguishing the in-distribution samples from out-of-distribution samples as shown in Table 7.3.

Table 7.3 OOD detection results. Models are trained on CIFAR10. Values are AUROC.

<table>
<thead>
<tr>
<th>$s_\theta(x)$</th>
<th>Model</th>
<th>SVHN</th>
<th>Interp</th>
<th>C100</th>
<th>CelebA</th>
</tr>
</thead>
<tbody>
<tr>
<td>log $p_\theta(x)$</td>
<td>WRN*</td>
<td>.91</td>
<td>-</td>
<td>.87</td>
<td>.78</td>
</tr>
<tr>
<td></td>
<td>IGEBM</td>
<td>.63</td>
<td>.70</td>
<td>.50</td>
<td>.70</td>
</tr>
<tr>
<td></td>
<td>JEM</td>
<td>.67</td>
<td>.65</td>
<td>.67</td>
<td>.75</td>
</tr>
<tr>
<td></td>
<td>JEM++</td>
<td>.85</td>
<td>.57</td>
<td>.68</td>
<td>.80</td>
</tr>
<tr>
<td></td>
<td>VERA</td>
<td>.83</td>
<td>.86</td>
<td>.73</td>
<td>.33</td>
</tr>
<tr>
<td></td>
<td>KL-EBM</td>
<td>.91</td>
<td>.65</td>
<td>.83</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>ViT</td>
<td>.93</td>
<td>.93</td>
<td>.82</td>
<td>.81</td>
</tr>
<tr>
<td></td>
<td>HybViT</td>
<td>.93</td>
<td>.92</td>
<td>.84</td>
<td>.76</td>
</tr>
</tbody>
</table>

* The result is from (Liu et al. 2020).

7.6.2.3 Robustness

Adversarial examples (Szegedy et al. 2014b; Goodfellow et al. 2015b) tricks the neural networks into giving incorrect predictions by applying minimal perturbations to the inputs, and hence, adversarial robustness is a critical characteristics of the model, which has received an influx of research interest. In this paper, we investigate the robustness of models trained on CIFAR10 using the white-box PGD attack (Madry et al. 2018) under an $L_\infty$ or $L_2$ constraint. Fig 7.7 compares ViT and HybViT with the baseline WRN-based classifier. We can see that ViT and HybViT have similar performance and both outperform WRN-based classifiers.
Figure 7.7 Adversarial robustness under the PGD attacks.

![Adversarial robustness graphs](image)

(a) $L_{\infty}$ Robustness  
(b) $L_2$ Robustness

7.6.2.4 Likelihood

An advantage of DDPM is that it can use the VLB as the approximated likelihood while most EBMs can’t compute the intractable partition function w.r.t $x$. Table 7.4 reports the test negative log-likelihood(NLL) in bits per dimension on CIFAR10. As we can observe, HybViT achieves comparable result to GenViT, and both are worse than other methods.

Table 7.4 NLL measured in bits/dim on CIFAR-10.

<table>
<thead>
<tr>
<th>Model</th>
<th>BPD↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenViT</td>
<td>3.78</td>
</tr>
<tr>
<td>HybViT</td>
<td>3.84</td>
</tr>
<tr>
<td>DDPM (Ho et al. 2020)</td>
<td>3.70</td>
</tr>
<tr>
<td>iDDPM (Nichol and Dhariwal 2021)</td>
<td>2.94</td>
</tr>
<tr>
<td>DiffuEBM(Gao et al. 2021)</td>
<td>3.18</td>
</tr>
<tr>
<td>DistAug (Jun et al. 2020)</td>
<td><strong>2.56</strong></td>
</tr>
</tbody>
</table>
7.6.3 Ablation Study

In this section, we study the effect of different training configurations on the performance of image classification and generation by conducting an exhaustive ablation study on CIFAR10. We investigate the impact of 1) training epochs, 2) the coefficient $\alpha$, and 3) configurations of ViT/HybViT architecture in the main content. Due to page limitations, more results can be found in the appendix.

Table 7.5 Ablation study of epochs.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc %</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViT (epoch=100)</td>
<td>94.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ViT (epoch=300)</td>
<td>96.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ViT (epoch=500)</td>
<td>96.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GenViT(epoch=100)</td>
<td>-</td>
<td>7.25</td>
<td>33.3</td>
</tr>
<tr>
<td>GenViT(epoch=300)</td>
<td>-</td>
<td>7.67</td>
<td>26.2</td>
</tr>
<tr>
<td>GenViT(epoch=500)</td>
<td>-</td>
<td>8.17</td>
<td>20.2</td>
</tr>
<tr>
<td>HybViT(epoch=100)</td>
<td>93.1</td>
<td>7.15</td>
<td>35.0</td>
</tr>
<tr>
<td>HybViT(epoch=300)</td>
<td>95.9</td>
<td>7.59</td>
<td>29.5</td>
</tr>
<tr>
<td>HybViT(epoch=500)</td>
<td>95.9</td>
<td>7.68</td>
<td>26.4</td>
</tr>
</tbody>
</table>

The results are reported in Table 7.5 and 7.6. First, Table 7.5 shows a trade-off between the overall performance and computation time. The gain of classification and generation is relatively large when we prolong the training from 100 epochs to 300. With more training epochs, the accuracy gap between ViT and HybViT decreases. Furthermore, The generation quality can slightly improve after 300 epochs. Then we thoroughly explore the settings of the backbone ViT for GenViT and HybViT in Table 7.6. It can be observed that larger $\alpha$ is preferred with high-quality generation and only small drop in accuracy. The number of heads also has a small effect on the
trade-off between classification accuracy and generation quality. Enlarging the model capacity, depth, or hidden dimensions can improve the accuracy and generation quality.

Table 7.6 Ablation Study on CIFAR10. The default configurations of HybViT are $\alpha = 100, \text{head}=12, \text{depth}=9, \text{dim}=384$.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc %</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>HybViT</td>
<td>95.9</td>
<td>7.68</td>
<td>26.4</td>
</tr>
<tr>
<td>HybViT($\alpha=1$)</td>
<td>96.6</td>
<td>4.74</td>
<td>68.9</td>
</tr>
<tr>
<td>HybViT($\alpha=10$)</td>
<td>97.0</td>
<td>6.40</td>
<td>38.2</td>
</tr>
<tr>
<td>HybViT(head=6)</td>
<td>96.0</td>
<td>7.51</td>
<td>30.0</td>
</tr>
<tr>
<td>HybViT(head=8)</td>
<td>95.9</td>
<td>7.74</td>
<td>28.0</td>
</tr>
<tr>
<td>HybViT(head=16)</td>
<td>95.4</td>
<td>7.79</td>
<td>27.1</td>
</tr>
<tr>
<td>HybViT(depth=6)</td>
<td>94.7</td>
<td>7.39</td>
<td>30.6</td>
</tr>
<tr>
<td>HybViT(depth=12)</td>
<td>96.6</td>
<td>7.78</td>
<td>24.3</td>
</tr>
<tr>
<td>HybViT(dim=192)</td>
<td>94.1</td>
<td>7.06</td>
<td>35.0</td>
</tr>
<tr>
<td>HybViT(dim=768)</td>
<td>96.4</td>
<td>8.04</td>
<td>19.9</td>
</tr>
<tr>
<td>GenViT(dim=192)</td>
<td>-</td>
<td>7.26</td>
<td>32.5</td>
</tr>
<tr>
<td>GenViT(dim=384)</td>
<td>-</td>
<td>8.17</td>
<td>20.2</td>
</tr>
<tr>
<td>GenViT(dim=768)</td>
<td>-</td>
<td>8.32</td>
<td>18.7</td>
</tr>
</tbody>
</table>

While it is challenging for our methods to generate realistic images for complex and high-resolution data, it is beyond the scope of this work to further improve the generation quality for high-resolution data. Thus, it warrants an exciting direction of future work. We suppose the large patch size of the ViT’s architecture is the critical causing factor. Hence, we investigate the impact of different patch sizes on STL10 in Table 7.7. However, even though a smaller patch size can improve the accuracy by a notably margin at the cost of increasing model sizes, but the generation quality for high-resolution images plateaued around $p = 6$. These results indicate that
the bottleneck of image generation comes from other components, such as the linear projections and reconstruction projections, other than the multi-head self-attention. Note that a larger patch size (ps=12) do further deteriorate the generation quality and would lead to critical issues for high-resolution data like ImageNet, since the corresponding patch size is typically set to 14 or larger.

Table 7.7 Ablation Study on STL10. All models are trained for 500 epochs. NoP means Number of Parameters. ps means Patch Size.

<table>
<thead>
<tr>
<th>Model</th>
<th>NoP</th>
<th>Acc %</th>
<th>IS</th>
<th>FID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViT(ps=8)</td>
<td>12.9M</td>
<td>78.7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HybViT(ps=4)</td>
<td>41.1M</td>
<td>87.1</td>
<td>6.90</td>
<td>125.5</td>
</tr>
<tr>
<td>HybViT(ps=6)</td>
<td>17.0M</td>
<td>81.7</td>
<td>7.30</td>
<td>123.6</td>
</tr>
<tr>
<td>HybViT(ps=8)</td>
<td>12.9M</td>
<td>77.5</td>
<td>6.95</td>
<td>125.2</td>
</tr>
<tr>
<td>HybViT(ps=12)</td>
<td>11.4M</td>
<td>69.1</td>
<td>2.55</td>
<td>240.2</td>
</tr>
<tr>
<td>GenViT(dim=384)</td>
<td>12.9M</td>
<td>-</td>
<td>6.95</td>
<td>125.2</td>
</tr>
<tr>
<td>GenViT(dim=576)</td>
<td>26.4M</td>
<td>-</td>
<td>7.02</td>
<td>124.1</td>
</tr>
<tr>
<td>GenViT(dim=768)</td>
<td>45.2M</td>
<td>-</td>
<td>7.01</td>
<td>126.6</td>
</tr>
</tbody>
</table>

7.6.3.1 Training Speed

We report the empirical training speeds of our models and baseline methods on a single GPU for CIFAR10 in Table 4.4 and those for ImageNet 32x32 is in the appendix. As discussed previously, two mini-batches are utilized in HybViT: one for training of $L_{simple}$ and the other for training of the cross entropy loss. Hence, HybViT requires about $2\times$ training time compared to GenViT. One of the advantages of GenViT and HybViT is that even with much more ($7.5\times$) iterations, they still reduce training time significantly compared to EBMs. The results demonstrate that our new methods are much faster and affordable for academia research settings.
Table 7.8 Run-time comparison on CIFAR10. We set 1170 iterations as one epoch for all ViTs, and 390 for WRN-based models. All ViTs are trained for 500 epochs and WRN-based models are trained for 200 epochs.

<table>
<thead>
<tr>
<th>Model</th>
<th>NoP(M)</th>
<th>Min/Epoch</th>
<th>Runtime(Hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ViT-based Models</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ViT(d=384)</td>
<td>11.2</td>
<td>1.72</td>
<td>14.4</td>
</tr>
<tr>
<td>GenViT(d=384)</td>
<td>11.2</td>
<td>2.11</td>
<td>17.6</td>
</tr>
<tr>
<td>HybViT(d=192)</td>
<td>3.2</td>
<td>2.14</td>
<td>17.9</td>
</tr>
<tr>
<td>HybViT(d=384)</td>
<td>11.2</td>
<td>3.71</td>
<td>31.2</td>
</tr>
<tr>
<td>HybViT(d=768)</td>
<td>43.2</td>
<td>9.34</td>
<td>77.8</td>
</tr>
<tr>
<td><strong>WRN-based Models</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRN 28-10</td>
<td>36.5</td>
<td>1.53</td>
<td>5.2</td>
</tr>
<tr>
<td>JEM(K=20)</td>
<td>36.5</td>
<td>30.2</td>
<td>101.3</td>
</tr>
<tr>
<td>JEM++(K=10)</td>
<td>36.5</td>
<td>20.4</td>
<td>67.4</td>
</tr>
<tr>
<td>VERA</td>
<td>40</td>
<td>19.3</td>
<td>64.3</td>
</tr>
<tr>
<td>IGEBM</td>
<td>-</td>
<td></td>
<td>1 GPU for 2 days</td>
</tr>
<tr>
<td>KL-EBM</td>
<td>6.64</td>
<td></td>
<td>1 GPU for 1 day</td>
</tr>
<tr>
<td>VAEBM*</td>
<td>135</td>
<td>400 epochs, 8 GPUs, 55 hours</td>
<td></td>
</tr>
<tr>
<td>DDPM</td>
<td>35.7</td>
<td>800k iter, 8 TPUs, 10.6 hours</td>
<td></td>
</tr>
<tr>
<td>DiffuEBM</td>
<td>34.8</td>
<td>240k iter, 8 TPUs, 40+ hours</td>
<td></td>
</tr>
</tbody>
</table>

* The runtime is for pretraining NVAE only. It further needs 25,000 iterations (or 16 epochs) on CIFAR-10 using one GPU for VAEBM.

### 7.7 Limitations

As shown in previous sections, our models GenViT and HybViT exhibit promising results. However, compared to CNN-based methods, the main limitations are: 1) The generation quality is relatively low compared with pure generation (non-hybrid) SOTA models. 2) They require more training iterations to achieve high classification performance compared with pure classification models. 3) The sampling speed during inference is slow (typically $T \geq 1000$) while GAN only needs one-time forward.

We believe the results presented in this work are sufficient to motivate the community to solve these limitations and improve speed and generative quality.
7.8 Conclusion

In this work, we integrate a single ViT into DDPM to propose a new type of generative model, GenViT. Furthermore, we present HybViT, a simple approach for training hybrid discriminative-generative models. We conduct a series of thorough experiments to demonstrate the effectiveness of these models on multiple benchmark datasets with state-of-the-art results in most of the tasks of image classification, and image generation. We also investigate the intriguing properties, including likelihood, adversarial robustness, uncertainty calibration, and OOD detection. Most importantly, the proposed approach HybViT provides stable training, and outperforms the previous state-of-the-art hybrid models on both discriminative and generation tasks. While there are still challenges in training the models for high-resolution images, we hope the results presented here will encourage the community to improve upon current approaches.
CHAPTER 8
FUTURE WORK

As we discussed previously, JEM, a hybrid model within a single object, is extremely expensive to
train on high dimensional data, and also unstable. To overcome such issues in JEM, we proposed
GMMC, JEM++, MAEBM/MAJEM, and SAJEM. We have seen a huge significant progress to
improve the stability, speed, accuracy, and generative quality altogether. However, the issues still
exist, such as instability, the accuracy gap, poor expansibility to high-resolution data, and the
expensive cost. Then, we proposed a new hybrid framework by combining ViT and Diffusion
Denoising probabilistic Model together. It gets rid of the instability issue and the MCMC sampling.
We can easily apply it to large-scale datasets, however the generation quality is relatively low.

Hence, my future research focus on two directions, the theoretical understanding of JEM and
improving Diffusion-based ViTs, namely GenViT and HybViT.

8.1 Theoretical Analysis of JEM

As shown in Fig. 5.6b, MA-EBM without regularization have a mixing hidden feature space of $x^0$
and $x^+$, which indicates the distance between $x^0$ and $x^+$ is not the only reason for diverge. Hence,
we still don’t have a clear understanding about the instability.

Fig. 5.6b provides a basic but interesting observation about the manifolds learned by different
models. Beyond that, we are more interested in the visualization of the MCMC sampling process.
Fig. 8.1 shows how a trained JEM transits samples from $x_0$ to real data using the SGLD sampling
with a replay buffer. However, Batch Norm introduces stochasticity into the MCMC sampling
process and makes the behaviour more complex. We need better tools for visualization and theory
to explain the origin of instability.

Figure 8.1 The visualization of SGLD sampling process with JEM trained on CIFAR10.

8.2 Improved ViT and Diffusion Model

GenViT and HybViT build a new direction for hybrid discriminative-generative models and maximum likelihood estimation. Two weakness of such diffusion-based ViTs are 1) slower convergence rate and more training iterations required than CNN-based methods, 2) lower generation quality.

Since the proposal Transformer (Vaswani et al. 2017) and ViT (Dosovitskiy et al. 2021), tons of work have been proposed to enhance the performance or efficiency, such as Performer (Choro-
manski et al. 2021), Swin Transformer (Liu et al. 2021). It also happens in the area of DDPM and score matching (Nichol and Dhariwal 2021; Song et al. 2021; Salimans and Ho 2022; Bao et al. 2022). We can explore this promising direction to improve the GenViT and HybViT.
CHAPTER 9
CONCLUSION

In this dissertation proposal, we comprehensively study and propose a series of work on EBM training methods for hybrid discriminative and generative models. In Chapter 3, we propose GMMC, modeling the joint density function of $p(x, y)$ with Max-Mahalanobis Distribution as an EBM. In Chapter 4, we proposed JEM++, a variety of new training procedures and architecture features to improve JEM’s accuracy, training stability, and speed altogether. Chapter 5 introduces the Manifold-Aware EBM/JEM to understand the manifolds. Chapter 6 further incorporates the sharpness-aware minimization technique into JEM++ to smooth the landscape of model and improve the generalizability and generation quality. Most interestingly, in Chapter 7, we propose a new direction for generative model and hybrid model by integrating ViT as the backbone of DDPM. By doing so, the new methods keep away the expensive MCMC sampling and demonstrate some promising future directions.

As future work for this dissertation, we will focus on two directions: 1) The theoretical understanding of the instability and further improvement on the key aspects of EBM training, including the stability, speed, generative quality, and accuracy. 2) The limitation and bottleneck in GenViT and HybViT. We believe the presented work will inspire subsequent research towards EBM, ViT, and hybrid modeling areas.
CHAPTER A
GENERATIVE MAX-MAHALANOBIS CLASSIFIERS FOR IMAGE CLASSIFICATION, GENERATION AND MORE

A Image Classification Benchmarks

The three image classification benchmarks used in our experiments are described below:

1. CIFAR10 (Krizhevsky and Hinton 2009) contains 60,000 RGB images of size $32 \times 32$ from 10 classes, in which 50,000 images are for training and 10,000 images are for test.

2. CIFAR100 (Krizhevsky and Hinton 2009) also contains 60,000 RGB images of size $32 \times 32$, except that it contains 100 classes with 500 training images and 100 test images per class.

3. SVHN (Netzer et al. 2011) is a street view house number dataset containing 73,257 training images and 26,032 test images classified into 10 classes representing digits. Each image may contain multiple real-world house number digits, and the task is to classify the center-most digit.

B Experimental Details

Following JEM (Grathwohl et al. 2020a), all our experiments are based on the Wide-ResNet architecture (Zagoruyko and Komodakis 2016). To ensure a fair comparison, we follow the same configurations of JEM by removing batch normalization and dropout from the network. We use ADAM optimizer (Kingma and Ba 2015) with the initial learning rate of 0.0001 and the decay rate of 0.3, and train all our models for 150 epochs. For CIFAR10 and SVHN, we reduce the learning rate at epoch [30, 50, 80], while for CIFAR100 we reduce the learning rate much later at epoch
The hyperparameters of our generative training of GMMC are listed in Table A.1. Compared to the configurations of IGEBM (Du and Mordatch 2019) and JEM, we use a 10x larger buffer size and set the reinitialization frequency to 2.5%. We note that with these settings GMMC generates images of higher quality. Comparing the two sampling approaches: Staged Sampling and NoiseInjected Sampling, we find that their performances are very similar. Therefore, only the performances of Staged Sampling are reported.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balance factor $\beta$</td>
<td>0.5</td>
</tr>
<tr>
<td>Number of steps $\tau$</td>
<td>20</td>
</tr>
<tr>
<td>Buffer size $</td>
<td>B</td>
</tr>
<tr>
<td>Reinitialization freq. $\rho$</td>
<td>2.5%</td>
</tr>
<tr>
<td>Step-size $\alpha$</td>
<td>1.0</td>
</tr>
</tbody>
</table>

C Pre-designed $\mu$ of MMC

Algorithm 6 describes the construction of a set of means $\mu = \{\mu_y, y = 1, 2, \cdots, C\}$ that satisfy the condition of Max-Mahalanobis Distribution (MMD) (Pang et al. 2018). $C$ is the number of classes, $d$ is the dimension of feature vector $\phi(x) \in \mathbb{R}^d$ extracted by CNN, and $S$ is a hyperparameter, which we set to 10 in all our experiments. For more details of MMD, please refer to (Pang et al. 2018).
**Algorithm 6** GenerateOptMeans($C, d, S$) for MMC (Pang et al. 2018)

**Input:** Number of classes $C$, dimension of feature vector $d$, and hyperparameter $S$. ($C \leq d + 1$)

**Initialization:** Let the $C$ mean vectors be $\mu^*_1 = e_1$ and $\mu^*_i = 0_d$, $i \neq 1$. Here $e_1$ and $0_d$ denote the first unit basis vector and the zero vector in $\mathbb{R}^d$, respectively.

for $i = 2$ to $C$
  for $j = 1$ to $i - 1$
    $\mu^*_i(j) = -[1 + \langle \mu^*_i, \mu^*_j \rangle \cdot (C - 1)]/[\mu^*_j(j) \cdot (C - 1)]$
  end for
  $\mu^*_i(i) = \sqrt{1 - ||\mu^*_i||^2_2}$
end for

for $k = 1$ to $C$
  $\mu^*_k = S \cdot \mu^*_k$
end for

Return: The optimal mean vectors $\mu^*_i$, $i \in [C]$.

**D Calibration**

**Expected Calibration Error**

(ECE) is a commonly adopted metric to measure the calibration of a model. First, it computes the confidence of the model, $\max_y p(y|x_i)$, for each $x_i$ in the dataset. Then it groups the predictions into equally spaced buckets $\{B_1, B_2, \cdots, B_M\}$ based on their confidence scores. For example, if $M = 20$, then $B_1$ would represent all examples for which the model’s confidence scores were between 0 and 0.05. Then ECE is calculated as

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)|,$$

(A.1)

where $n$ is the number of examples in the dataset, acc($B_m$) is the average accuracy of the model on all the examples in $B_m$ and conf($B_m$) is the average confidence on all the examples in $B_m$. In our experiments, we set $M = 20$. For a perfectly calibrated model, the ECE will be 0 for any $M$. 
D.1 Calibration Results on CIFAR100 and SVHN

Fig. A.1 and Table A.2 show the calibration results of GMMC on SVHN and CIFAR100. As we can see, GMMC outperforms the softmax baseline and JEM on SVHN significantly, while JEM works better on CIFAR100.

![GMMC calibration results on SVHN and CIFAR100](image)

Figure A.1 GMMC calibration results on SVHN and CIFAR100.

Table A.2 Calibration results on SVHN and CIFAR100.

<table>
<thead>
<tr>
<th>Model</th>
<th>SVHN</th>
<th>CIFAR100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax</td>
<td>1.91</td>
<td>22.32</td>
</tr>
<tr>
<td>JEM</td>
<td>2.08</td>
<td>4.87</td>
</tr>
<tr>
<td>GMMC</td>
<td><strong>0.93</strong></td>
<td>7.80</td>
</tr>
</tbody>
</table>

E Joint Training

Comparing Eq. 5 and Eq. 7 in the main text, we note that the gradient of Eq. 5 is just the second term of Eq. 7. Hence, we can use (Approach 1) discriminative training to pretrain $\theta$, and then finetune $\theta$ by (Approach 2) generative training. The transition between the two can be achieved by scaling up $\beta$ from 0 to a predefined value (e.g., 0.5). Similar joint training strategy could be
applied to train JEM. However, from our experiments we note that this joint training of JEM is extremely unstable.

Fig. A.2 shows the validation accuracy curves of GMMC trained with discriminative, generative and joint training on CIFAR10. We train each model with 100 epochs, while for the joint training the first 50 epochs are with discriminative training and the rest of 50 epochs are switched to generative training. As we can see, all three models achieve a very similar accuracy of 93.5% in the end. For the joint training, its accuracy drops a little bit around the switching epoch (51th), but finally can catch up to achieve a similar accuracy of 93.5%. Interestingly, we note that the quality of sampled images from joint training, as shown in Fig. A.3, is not as good as that of generative training from scratch. Nevertheless, this results demonstrate that we can potentially use the joint training to speed up the training of GMMC, while generating competitive samples. We leave the improvement of image quality of the joint training of GMMC to future work.

Figure A.2 Validation accuracy curves of GMMC trained with discriminative, generative or joint training.

Similar to the joint training of GMMC, we explore the joint training of JEM, i.e., pretrain the
Figure A.3 Generated CIFAR10 samples (unconditional) by GMMC with joint training. Compared to the samples from GMMC with generative training, these samples are more blurry.

softmax classifier with standard discriminative training and then finetune the softmax classifier with JEM. Table A.3 shows the evolution of energy scores of real data and sampled data before and after the switching epoch. Here the energy function is defined as $E_\theta(x) = -\log \sum_y \exp(f_\theta(x)[y])$ according to JEM. As can be seen, the energy scores on real data and sampled data are quickly exploded within a few iterations, demonstrating the instability of joint training of JEM. We also use different $\tau$s, the number of SGLD steps, to stabilize JEM but with no success.

F Additional GMMC Generated Samples

Additional GMMC generated samples of SVHN and CIFAR100 are provided in Fig. A.4. GMMC generated class-conditional samples of CIFAR10 are provided in Figs A.5-A.14. To evaluate the quality of generated images, we use the same IS and FID code as in IGEBM (Du and Mordatch...
Table A.3 The energy scores of real data and sampled data after switching from discriminative training to JEM on CIFAR10.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Real Data</th>
<th></th>
<th></th>
<th>Sampled Data</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\tau = 20$</td>
<td>$\tau = 50$</td>
<td>$\tau = 100$</td>
<td>$\tau = 20$</td>
<td>$\tau = 50$</td>
<td>$\tau = 100$</td>
</tr>
<tr>
<td>Before transfer</td>
<td>-22.4</td>
<td>-21.2</td>
<td>-21.6</td>
<td>N/A</td>
<td>-2752</td>
<td>-7369</td>
</tr>
<tr>
<td>1</td>
<td>-21.5</td>
<td>-19.3</td>
<td>-20.7</td>
<td>-67</td>
<td>-243</td>
<td>-556</td>
</tr>
<tr>
<td>2</td>
<td>-9.7</td>
<td>-7.6</td>
<td>-8.5</td>
<td>-4</td>
<td>-86</td>
<td>-449</td>
</tr>
<tr>
<td>3</td>
<td>-5.4</td>
<td>-3.9</td>
<td>-3.7</td>
<td>-294</td>
<td>695</td>
<td>1711</td>
</tr>
<tr>
<td>4</td>
<td>-3.2</td>
<td>-2.2</td>
<td>-0.8</td>
<td>4309</td>
<td>3182</td>
<td>6364</td>
</tr>
<tr>
<td>5</td>
<td>-15.4</td>
<td>10.5</td>
<td>13.9</td>
<td>524</td>
<td>693</td>
<td>810</td>
</tr>
<tr>
<td>6</td>
<td>18676</td>
<td>18250</td>
<td>23021</td>
<td>1.04e+7</td>
<td>6.34e+6</td>
<td>1.01e+7</td>
</tr>
<tr>
<td>7</td>
<td>3.11e+9</td>
<td>3.01e+9</td>
<td>5.15e+9</td>
<td>1.04e+7</td>
<td>6.34e+6</td>
<td>1.01e+7</td>
</tr>
</tbody>
</table>

2019) and JEM (Grathwohl et al. 2020a) on the class-conditional samples.

Figure A.4 GMMC generated class-conditional samples of SVHN and CIFAR100. Each row corresponds to one class.
Figure A.5 GMMC generated class-conditional samples of **Plane**

Figure A.6 GMMC generated class-conditional samples of **Car**
Figure A.7 GMMC generated class-conditional samples of **Bird**

Figure A.8 GMMC generated class-conditional samples of **Cat**
Figure A.9 GMMC generated class-conditional samples of Deer

Figure A.10 GMMC generated class-conditional samples of Dog
Figure A.11 GMMC generated class-conditional samples of **Frog**

Figure A.12 GMMC generated class-conditional samples of **Horse**
Figure A.13 GMMC generated class-conditional samples of Ship

Figure A.14 GMMC generated class-conditional samples of Truck
A Experimental Details

To have a fair comparison with JEM (Grathwohl et al. 2020a), all our experiments are based on the Wide-ResNet architecture (Zagoruyko and Komodakis 2016) and follow JEM’s settings whenever possible. As we discussed in the main text, JEM++ enables batch norm (Ioffe and Szegedy 2015) and the SGD optimizer (Robbins and Monro 1951) with a large learning rate, which we find works better than Adam (Kingma and Ba 2015) with a very small learning rate of $1e^{-4}$ that is used by JEM. Specifically, we use SGD with an initial learning rate of 0.1 and a decay rate of 0.2, and train all our models for 150 epochs. We reduce the learning rate at epoch [50, 100, 125]. Table B.1 lists the hyperparameters of JEM++. Note that JEM++ is still highly stable even with $M = 5$. More experimental details can be found in our code, which is publicly available at https://github.com/sndnyang/JEMPP.

Table B.1 Hyperparameters of JEM++ for CIFAR10

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of outer steps $M$</td>
<td>5, 10</td>
</tr>
<tr>
<td>Number of inner steps $N$</td>
<td>5</td>
</tr>
<tr>
<td>Proximity constraint $\varepsilon$</td>
<td>1</td>
</tr>
<tr>
<td>Buffer size $</td>
<td>B</td>
</tr>
<tr>
<td>Reinitialization freq. $\rho$</td>
<td>5%</td>
</tr>
<tr>
<td>PYLD step-size $\alpha$</td>
<td>0.2</td>
</tr>
</tbody>
</table>
B Informative Initialization

In this paper, we introduce a novel informative initialization to start the SGLD chain. Specifically, instead of using a uniform distribution, we sample from a Gaussian mixture distribution estimated from the training data as

\[ p_0(x) = \sum_y \pi_y N(\mu_y, \Sigma_y) \]  

(B.1)

with \( \pi_y = |\mathcal{D}_y| / \sum_{y'} |\mathcal{D}_{y'}| \), \( \mu_y = \mathbb{E}_{x \sim \mathcal{D}_y}[x] \),

\[ \Sigma_y = \mathbb{E}_{x \sim \mathcal{D}_y} \left[ (x - \mu_y)(x - \mu_y)^\top \right] \],
where $\mathcal{D}_y$ denotes the set of training samples with label $y$. Figure B.1 visualizes the categorical centers ($\mu$’s) estimated from the SVHN and CIFAR100 training datasets. Figure B.2 visualizes the categorical centers and the corresponding samples $x_0 \sim p_0(x)$ for CIFAR10. Note that no extra information is used to train JEM++ over JEM.

C Additional Generated Samples

Additional JEM++ generated samples of SVHN and CIFAR100 are provided in Figure B.3. Additional JEM++ generated class-conditional (best and worst) samples of CIFAR10 are provided in Figures B.4-B.13. It is worth noting that the worst images (the lowest $p(x)$ or $p(y|x)$) generated by JEM++ are more visually appealing than JEM generated (see examples in the Appendix of JEM (Grathwohl et al. 2020a)).

Figure B.3 JEM++ generated class-conditional samples of SVHN and CIFAR100. Each row corresponds to one class.
Figure B.4 JEM++ generated class-conditional samples of **Plane**

Figure B.5 JEM++ generated class-conditional samples of **Car**

Figure B.6 JEM++ generated class-conditional samples of **Bird**
Figure B.7 JEM++ generated class-conditional samples of Cat

Figure B.8 JEM++ generated class-conditional samples of Deer

Figure B.9 JEM++ generated class-conditional samples of Dog
Figure B.10 JEM++ generated class-conditional samples of **Frog**

Figure B.11 JEM++ generated class-conditional samples of **Horse**

Figure B.12 JEM++ generated class-conditional samples of **Ship**
Figure B.13 JEM++ generated class-conditional samples of Truck
## A Image Datasets

The image benchmark datasets used in our experiments are described below:

1. CIFAR10 (Krizhevsky and Hinton 2009) contains 60,000 RGB images of size $32 \times 32$ from 10 classes, in which 50,000 images are for training and 10,000 images are for test.

2. CIFAR100 (Krizhevsky and Hinton 2009) also contains 60,000 RGB images of size $32 \times 32$, except that it contains 100 classes with 500 training images and 100 test images per class.

3. CelebA-HQ (Karras et al. 2018) is a human face image dataset. In our experiment, we use the downsampled version with size $128 \times 128$.

4. Imagenet 32x32 (Chrabaszcz et al. 2017) is a downsampled variant of ImageNet with 1,000 classes. It contains the same number of images as vanilla ImageNet, but the image size is $32 \times 32$.

## B Experimental Details

Our code is largely built on top of JEM (Grathwohl et al. 2020a)\(^1\). For a fair comparison with JEM, we update each model with 390 iterations in 1 epoch. Empirically, we find a batch size of 128 for $p_\theta(y|x)$ achieves the best classification accuracy on CIFAR10, while we use 64, the same batch size as in JEM, for the maximum likelihood estimate of $p_\theta(x)$. All our experiments are performed with

[^1]: [https://github.com/wgrathwohl/JEM](https://github.com/wgrathwohl/JEM)
PyTorch on Nvidia GPUs. For CIFAR10 and CIFAR100, we train the backbone Wide-ResNet 28-10 (Zagoruyko and Komodakis 2016) on a single GPU. Due to limited computational resources, we use Wide-ResNet 28-2 for ImageNet 32x32 on a single GPU, and Wide-ResNet 28-5 for CelebA-HQ 128x128 on 2 GPUs.

Table C.1 lists the hyper-parameters of our MA-EBM/JEM algorithms. We train all our models for 200 epochs with the SGD optimizer, a buffer size of 10,000, a reinitialization frequency of 5%. For CIFAR10, we use a larger learning rate of 0.1, while for CIFAR100, CelebA-HQ and Imagenet 32x32 we use an initial learning rate of 0.02.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epochs  $N$</td>
<td>200</td>
</tr>
<tr>
<td>Buffer size $</td>
<td>\mathcal{B}</td>
</tr>
<tr>
<td>Reinitialization freq. $\rho$</td>
<td>5%</td>
</tr>
<tr>
<td>SGLD step-size $\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>SGLD noise $\sigma$</td>
<td>0.001</td>
</tr>
</tbody>
</table>

### C Informative Initialization

In this work, we estimate the initial sampling distribution as

$$p_0(x) = \mathcal{N} (\mu, \Sigma)$$  \hspace{1cm} (C.1)

with $\mu = \mathbb{E}_{x \sim \mathcal{D}} [x]$, $\Sigma = \mathbb{E}_{x \sim \mathcal{D}} \left[ (x - \mu) (x - \mu) \top \right]$, where $\mathcal{D}$ denotes the whole training set. Fig. C.1 illustrate the center (\mu) estimated from each benchmark datatset. Fig. C.2 shows the samples $x_0 \sim p_0(x)$ for each benchmark dataset.
Figure C.1 The centers (μ’s) of CIFAR10, CIFAR100, Imagenet 32 and CelebA-HQ 128.

Figure C.2 Samples \(x_0 \sim p_0(x)\) of CIFAR10, CIFAR100, Imagenet 32x32 and CelebA-HQ 128.

D Generated Images of Imagenet 32x32

Fig. C.3 illustrates example images generated by MA-EBM on Imagenet 32x32.
Figure C.3 MA-EBM generated samples of Imagenet 32x32.
CHAPTER D
SHARPNESS AWARE JOINT ENERGY-BASED MODEL

A Image Datasets

The image benchmark datasets used in our experiments are described below:

1. CIFAR10 (Krizhevsky and Hinton 2009) contains 60,000 RGB images of size $32 \times 32$ from 10 classes, in which 50,000 images are for training, and 10,000 images are for evaluation.

2. CIFAR100 (Krizhevsky and Hinton 2009) also contains 60,000 RGB images of size $32 \times 32$, except that it contains 100 classes with 500 training images and 100 test images per class.

B Experimental Details

To have a fair comparison with JEM (Grathwohl et al. 2020a) and JEM++ (Yang and Ji 2021), all our experiments are based on the Wide-ResNet 28x10 architecture (Zagoruyko and Komodakis 2016) and follow their settings. Then, we use SGD for CIFAR10 and CIFAR100 with an initial learning rate of 0.1 and 0.01, respectively. We set a decay rate of 0.2, and train all our models for 200 epochs. We reduce the learning rate at epoch [60, 120, 180]. Table D.1 lists the hyper-parameters in our experiments.

C Qualitative Analysis of Samples

Generation quality is difficult to qualify. Following the setting of JEM (Grathwohl et al. 2020a), we conduct a qualitative analysis of samples on CIFAR10. In Fig. D.1, the samples that have
Table D.1 Hyper-parameters of SAJEM for CIFAR10 and CIFAR100

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of SGLD steps $K$</td>
<td>5, 10, 20</td>
</tr>
<tr>
<td>Buffer size $</td>
<td>B</td>
</tr>
<tr>
<td>Reinitialization freq. $\gamma$</td>
<td>5%</td>
</tr>
<tr>
<td>SGLD step-size $\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>SGLD noise $\sigma$</td>
<td>0</td>
</tr>
<tr>
<td>SAM noise bound $\rho$</td>
<td>2</td>
</tr>
</tbody>
</table>

higher $p(x)$ exhibit better quality, while samples generated by JEM have white background and centered object (see examples in the Appendix of JEM).

Figure D.1 Each row corresponds to 1 class.

D Energy Landscape

Fig D.3 shows the energy landscapes of different $K$s of SAJEM. As we can see, the overall energy of SAJEM decreases with increasing $K$. We leave exploring this phenomenon for future work.

E Out-of-Distribution Detection

Fig D.2 shows the visualization with different $K$s of SAJEM using the input density $p_\theta(x)$ as $s_\theta(x)$ on CIFAR10.
Figure D.2 Histograms (oriented horizontally for easier visual alignment) of $\text{LSE}(x)$ arranged by class for CIFAR10.

**F Calibration**

Expected Calibration Error

(ECE) is a standard metric to evaluate the calibration quality of a classifier (Guo et al. 2017a). It firstly computes the confidence of the model, $\max_y p(y|x_i)$, for each $x_i$ in the dataset. Then it groups the predictions into equally spaced buckets $\{B_1, B_2, \cdots, B_M\}$ based on the confidence scores. For example, if $M = 20$, then $B_1$ would represent all examples for which the model’s confidence scores were between 0 and 0.05. Then ECE is calculated as

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left| \text{acc}(B_m) - \text{conf}(B_m) \right|,$$

(D.1)

where $n$ is the number of data in the dataset, $\text{acc}(B_m)$ is the average accuracy of the model on all the examples in $B_m$ and $\text{conf}(B_m)$ is the average confidence on all the examples in $B_m$. In
Figure D.3 Energy Landscapes of different models trained on CIFAR10.

In our experiments, we set $M = 20$. For a perfectly calibrated model, the ECE will be 0 for any $M$. Fig D.4 demonstrates the SAM can always improve the calibration and smaller $K$ has better calibration.

G Additional Generated Samples

Additional SAJEM generated class-conditional (best and worst) samples of CIFAR10 are provided in Figures D.5-D.14.
Table D.2 Histograms of $\log_\theta p(x)$ for OOD detection. Green corresponds to in-distribution dataset, while red corresponds to OOD dataset.
Figure D.4 Calibration results on CIFAR10. The smaller ECE is, the better.

Figure D.5 SAJEM generated class-conditional samples of Plane
Figure D.6 SAJEM generated class-conditional samples of **Car**

(a) Samples with highest $p(x)$  
(b) Samples with lowest $p(x)$  
(c) Samples with highest $p(y|x)$  
(d) Samples with lowest $p(y|x)$

Figure D.7 SAJEM generated class-conditional samples of **Bird**

(a) Samples with highest $p(x)$  
(b) Samples with lowest $p(x)$  
(c) Samples with highest $p(y|x)$  
(d) Samples with lowest $p(y|x)$

Figure D.8 SAJEM generated class-conditional samples of **Cat**

(a) Samples with highest $p(x)$  
(b) Samples with lowest $p(x)$  
(c) Samples with highest $p(y|x)$  
(d) Samples with lowest $p(y|x)$
Figure D.9 SAJEM generated class-conditional samples of Deer

Figure D.10 SAJEM generated class-conditional samples of Dog

Figure D.11 SAJEM generated class-conditional samples of Frog
Figure D.12 SAJEM generated class-conditional samples of Horse

Figure D.13 SAJEM generated class-conditional samples of Ship

Figure D.14 SAJEM generated class-conditional samples of Truck
CHAPTER E
YOUR VIT IS SECRETLY A HYBRID DISCRIMINATIVE-GENERATIVE DIFFUSION MODEL

A Image Datasets

The image benchmark datasets used in our experiments are described below:

1. CIFAR10 (Krizhevsky and Hinton 2009) contains 60,000 RGB images of size $32 \times 32$ from 10 classes, in which 50,000 images are for training and 10,000 images are for test.

2. CIFAR100 (Krizhevsky and Hinton 2009) also contains 60,000 RGB images of size $32 \times 32$, except that it contains 100 classes with 500 training images and 100 test images per class.

3. STL10 (Coates et al. 2011) 500 training images from 10 classes as CIFAR10, 800 test images per class.

4. Tiny-ImageNet contains 100000 images of 200 classes (500 for each class) downsized to $64 \times 64$ colored images. Each class has 500 training images, 50 validation images and 50 test images.

5. CelebA-HQ (Karras et al. 2018) is a human face image dataset. In our experiment, we use the downsampled version with size $128 \times 128$.

6. Imagenet 32x32 (Chrabaszcz et al. 2017) is a downsamped variant of ImageNet with 1,000 classes. It contains the same number of images as vanilla ImageNet, but the image size is $32 \times 32$. 
Figure E.1 The evolution of HybViT’s classification accuracy, FID as a function of training epochs on CIFAR10 and ImageNet 32x32.

Figure E.2 The comparison between samples with FID=40 and FID=20. The difference is visually imperceptible for human.

B Model Evaluation

B.1 Qualitative Analysis of Samples

First, we investigate the gap between ViT, GenViT and HybViT in Fig E.1. We select two benchmark datasets CIFAR10 and ImageNet 32x32. It can be observed that the improvement of genera-
tion quality is relatively small after 10% training epochs. The difference is almost visually imperceptible for human between samples with FID=40 and FID=20 as shown in Fig. Hence, we should focus on improving the convergence rates of our models and reducing the training time. Following the setting of JEM (Grathwohl et al. 2020a), we conduct a qualitative analysis of samples on CIFAR10. We define an energy function of $x$ as $p_{\theta}(x) \propto E(x) = \log \sum_y e^{f_{\theta}(x)[y]} = \text{LSE}(f_{\theta}(x))$, the negative of the energy function in (Liu et al. 2020; Grathwohl et al. 2020a). We use a CIFAR10-trained HybViT model to generate 10,000 images from scratch, then feed them back into the HybViT model to compute $E(x)$ and $p(y|x)$. We show the examples and distribution by class in Fig E.3 and Fig E.4. We can observe that the worst examples of Plane can be completely blank. Additional HybViT generated class-conditional (best and worst) samples of CIFAR10 are provided in Figures E.8-E.17.

![Figure E.3 HybViT generated class-conditional (best and worst) samples. Each row corresponds to 1 class.](image)
Figure E.4 Histograms (oriented horizontally for easier visual alignment) of $x$ arranged by class for CIFAR10.

Table E.1 Ablation Study of Data Augmentation on CIFAR10.

<table>
<thead>
<tr>
<th>Model</th>
<th>Aug</th>
<th>Acc % ↑</th>
<th>IS ↑</th>
<th>FID ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViT</td>
<td>Strong</td>
<td>96.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Weak</td>
<td>87.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HybViT</td>
<td>Strong</td>
<td>95.9</td>
<td>7.68</td>
<td>26.4</td>
</tr>
<tr>
<td></td>
<td>Weak</td>
<td>84.6</td>
<td>7.85</td>
<td>24.9</td>
</tr>
</tbody>
</table>

B.2 Data Augmentation

We study the effect of data augmentation. ViT is known to require a too large amount of training data and/or repeated strong data augmentations to obtain acceptable visual representation. Table E.1 compares the performance between strong augmented data and conventional Inception-style pre-processed(namely weak augmentation) data (Szegedy et al. 2016). We can conclude that the strong data augmentation is really essential for high classification performance and the effect
on generation is negative but tiny. Note that the data augmentation is only used for classification, and for DDPM, we don’t apply any data augmentation.

### B.3 Out-of-Distribution Detection

Another useful OOD score function is the maximum probability from a classifier’s predictive distribution: \( s_\theta(x) = \max_y p_\theta(y|x) \). The results can be found in Table E.2 (bottom row).

Table E.2 OOD detection results. Models are trained on CIFAR10. Values are AUROC.

<table>
<thead>
<tr>
<th>( s_\theta(x) )</th>
<th>Model</th>
<th>SVHN</th>
<th>CIFAR10</th>
<th>Interp</th>
<th>CIFAR100</th>
<th>CelebA</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \log p_\theta(x) )</td>
<td>WideResNet</td>
<td>.91</td>
<td>-</td>
<td>.87</td>
<td>.78</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IGEBM</td>
<td>.63</td>
<td>.70</td>
<td>.50</td>
<td>.70</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM</td>
<td>.67</td>
<td>.65</td>
<td>.67</td>
<td>.75</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JEM++</td>
<td>.85</td>
<td>.57</td>
<td>.68</td>
<td>.80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VERA</td>
<td>.83</td>
<td>.86</td>
<td>.73</td>
<td>.33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ImCD</td>
<td>.91</td>
<td>.65</td>
<td>.83</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ViT</td>
<td>.93</td>
<td>.93</td>
<td>.82</td>
<td>.81</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HybViT</td>
<td>.93</td>
<td>.92</td>
<td>.84</td>
<td>.76</td>
<td></td>
</tr>
</tbody>
</table>

| \( \max_y p_\theta(y|x) \) | WideResNet | .93    | .77     | .85    | .62      |        |
|                           | IGEBM     | .43    | .69     | .54    | .69      |        |
|                           | JEM       | .89    | .75     | .87    | .79      |        |
|                           | JEM++     | .94    | .77     | .88    | .90      |        |
|                           | ViT       | .91    | .95     | .82    | .74      |        |
|                           | HybViT    | .91    | .94     | .85    | .67      |        |

### B.4 Robustness

Given ViT models trained with different data augmentations, we can investigate their robustness since weak data augmentations are commonly used in CNNs. Table E.3 shows an interesting phenomena that HybViT with weak data augmentation is much robust than other models, especially under \( L_2 \) attack. We suppose it’s because the noising process feeds huge amount of noisy samples to HybViT, then HybViT learns from the noisy data implicitly to improve the flatness and robustness.
Table E.3 Classification accuracies when models are under $L_\infty$ and $L_2$ PGD attacks with different $\epsilon$’s. All models are trained on CIFAR10.

<table>
<thead>
<tr>
<th>Model</th>
<th>Clean (%)</th>
<th>$L_\infty \epsilon = 1/255$</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>16</th>
<th>22</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViT</td>
<td>96.5</td>
<td>70.8</td>
<td>46.7</td>
<td>21.7</td>
<td>7.0</td>
<td>1.4</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>- Weak Aug</td>
<td>87.1</td>
<td>67.3</td>
<td>41.8</td>
<td>14.8</td>
<td>1.4</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HybViT</td>
<td>95.9</td>
<td>70.4</td>
<td>48.0</td>
<td>21.9</td>
<td>5.5</td>
<td>1.3</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>- Weak Aug</td>
<td>84.6</td>
<td>71.3</td>
<td>55.6</td>
<td>30.3</td>
<td>6.7</td>
<td>0.6</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Clean (%)</th>
<th>$L_2 \epsilon = 50/255$</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>250</th>
<th>300</th>
<th>350</th>
<th>400</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViT</td>
<td>96.5</td>
<td>52.3</td>
<td>9.2</td>
<td>1.1</td>
<td>0.3</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>- Weak Aug</td>
<td>87.1</td>
<td>53.9</td>
<td>21.4</td>
<td>5.5</td>
<td>1.0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HybViT</td>
<td>95.9</td>
<td>58.7</td>
<td>16.3</td>
<td>3.4</td>
<td>1.0</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>- Weak Aug</td>
<td>84.6</td>
<td>65.8</td>
<td>42.3</td>
<td>25.7</td>
<td>13.2</td>
<td>6.4</td>
<td>3.4</td>
<td>1.5</td>
<td>0.7</td>
</tr>
</tbody>
</table>

**B.5 Calibration**

Figures in E.5 provide a comparison of ViT and HybViT with the baselines WRN and JEM, and also corresponding ViTs trained without strong data augmentations. It can be observed that strong data augmentations can better calibrate the predictions of ViT and HybViT, but further make them under-confident.

**B.6 Training Speed**

We further report the empirical training speeds of our models and baseline methods for ImageNet 32x32. Our methods are memory efficient since it only requires a single GPU, and much faster.

Table E.4 Run-time comparison on ImageNet 32x32. All experiments are performed on a single GPU for 100 epochs.

<table>
<thead>
<tr>
<th>Model</th>
<th>NoP(M)</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViT</td>
<td>11.6</td>
<td>3 days</td>
</tr>
<tr>
<td>GenViT</td>
<td>11.6</td>
<td>2 days</td>
</tr>
<tr>
<td>HybViT</td>
<td>11.6</td>
<td>5 days</td>
</tr>
<tr>
<td>IGBM</td>
<td>32 GPUs for 5 days</td>
<td></td>
</tr>
<tr>
<td>KL-EBM</td>
<td>8 GPUs for 3 days</td>
<td></td>
</tr>
</tbody>
</table>
C Additional Generated Samples

Additional generated samples of CIFAR10, CIFAR100, ImageNet 32x32, TinyImageNet, STL10, and CelebA 128 are provided in Figure E.6. We further provide some generated images for ImageNet-128 and vanilla ImageNet-224 are shown in E.7. For ImageNet-224, we set the patch size as 14, then we find higher dimension is required but the generation quality is still low. Due to limited computation resource and low generation quality, we do not conduct experiments on ImageNet-128 and vanilla ImageNet.
Figure E.6 Generated Images

(a) CIFAR10  
(b) CIFAR100  
(c) ImageNet 32x32  
(d) TinyImagenet  
(e) STL10  
(f) CelebA 128
Figure E.7 Generated Images

Figure E.8 HybViT generated class-conditional samples of Plane
Figure E.9 HybViT generated class-conditional samples of \textbf{Car}

Figure E.10 HybViT generated class-conditional samples of \textbf{Bird}

Figure E.11 HybViT generated class-conditional samples of \textbf{Cat}
Figure E.12 HybViT generated class-conditional samples of Deer

Figure E.13 HybViT generated class-conditional samples of Dog

Figure E.14 HybViT generated class-conditional samples of Frog
Figure E.15 HybViT generated class-conditional samples of **Horse**

Figure E.16 HybViT generated class-conditional samples of **Ship**

Figure E.17 HybViT generated class-conditional samples of **Truck**
REFERENCES


Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen.


Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*,
2011.


Tian Han, Erik Nijkamp, Xiaolin Fang, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Diver-
gence triangle for joint training of generator model, energy-based model, and inference model.


Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Sixth Annual Conference on Computational Learning
Theory (COLT), 1993.


Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two pure transformers can make one


Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, and Balaji Lakshminarayanan. Detecting


Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent space


Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and


Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and


