공학석사 학위논문

신뢰할 수 있는 이상치 감지를 위한 에너지 기반 딥 앙상블 방법

Energy-Based Deep Ensembles for Reliable Out-of-distribution Detection

2022년 8월

서울대학교 대학원

기계공학부

권 혁 준

신뢰할 수 있는 이상치 감지를 위한 에너지 기반 딥 앙상블 방법

Energy-Based Deep Ensembles for Reliable Out-of-distribution Detection

지도교수 박 종 우

이 논문을 공학석사 학위논문으로 제출함

2022 년 4 월

서울대학교 대학원

기계공학부

권 혁 준

권 혁 준의 공학석사 학위논문을 인준함

2022 년 6 월

위 원 장 : 조 규 진 (인)

부위원장 : 박 종 우 (인)

위 원 : 김 아 영 (인)

ABSTRACT

Energy-Based Deep Ensembles for Reliable Out-of-distribution Detection

by

Hyeokjun Kwon

Department of Mechanical Engineering
Seoul National University

Models used for prediction tasks like classification and regression should ideally possess two properties: (i) any model should be able to determine whether an input x is in-distribution (an inlier, i.e., drawn from the data distribution $p(x)$), or out-of-distribution (an outlier, or OOD), and (ii) once outliers have been filtered out, the model should offer a quantifiable measure of predictive uncertainty. Deep Ensemble (DE) methods that employ multiple probabilistic prediction models, together with their recent variants, are widely used precisely because of these

two properties. Yet for OOD detection tasks, the performance of existing ensemble methods leaves much to be desired; we claim that this is because the models do not learn $p(x)$. We propose in this thesis *Energy-Based Deep Ensemble (EBDE)* methods, in which the prediction model $p(y|x)$ and input data distribution $p(x)$ are learned simultaneously. Specifically, $p(x)$ is formulated as an energy-based model, with the ensemble disagreement (i.e., the collective variance of the predictions made by each model) used as the energy function. Experiments involving a wide range of datasets confirm that EBDE significantly outperforms existing DEs for OOD detection tasks, while achieving comparable performance in prediction and uncertainty quantification.

Keywords: Deep Ensembles, Out-of-distribution detection, Uncertainty quantification, Energy-based models, Ensemble diversification.

Student Number: 2020-22845

Contents

List of Figures

1 Introduction

Deep neural networks (DNNs) have recently shown remarkable performance in prediction tasks [1, 2, 3, 4, 5], e.g., classification and regression, in which models are used to predict the label y given an input x. Besides having good prediction performance, any prediction model should ideally possess two additional properties. First, given an input x, the model should be able to determine whether it is in-distribution (inlier) or out-of-distribution (OOD, outlier) [6] (by an outlier we mean an input x that is unlikely to be drawn from the input training data distribution $p(x)$). Secondly, after outliers have been filtered out, the models should offer a quantifiable measure of the predictive uncertainty of inliers [7, 8]. These are essential features of any trustworthy machine learning system.

Among existing approaches [9, 10, 11, 12, 13] that attempt to design prediction models with the two aforementioned properties, *Deep Ensemble* (DE) methods [14] that employ multiple probabilistic DNN models $\{p(y|x;\theta_i)\}_{i=1}^{M}$ are widely used precisely because of these two properties. DE uses *ensemble disagreement*, the

degree to which each model in the ensemble disagrees, to detect whether an input is an inlier or outlier – if the variance of the models' prediction values for an input x is high, then x is classified as an outlier – and the predictive uncertainty is computed by using the average of the probabilistic model predictions $1/M \sum_{i=1}^{M} p(y|x;\theta_i)$ (e.g., the entropy of the mean prediction model).

As an OOD detector, the vanilla version of DE, in which each model is trained individually, is fundamentally limited. It relies solely on the stochasticity of the training procedure (e.g., random network initialization) and naively expects each model to predict a different value y for an outlier x. However, the vanilla DE often does not work in the expected way [15, 16, 17].

To detect outliers, a model should necessarily learn the input data distribution $p(x)$. Any data that has a low probability of being sampled from $p(x)$ can reasonably be considered an outlier; in fact, many unsupervised OOD detection methods attempt to learn $p(x)$ without using labels. As such, for DE methods to be successful in OOD detection tasks, the ensemble disagreement used as the OOD detection criterion in DE should be related to $p(x)$. Specifically, the higher the ensemble disagreement, the lower should be the likelihood. However, both vanilla DE and its recent variants [18, 15, 19, 16] that attempt to diversify models in the ensemble during training, do not learn $p(x)$ (or more specifically, the ensemble disagreement is irrelevant to $p(x)$), thus limiting their performance as OOD detectors.

In this thesis, we propose *Energy-Based Deep Ensemble (EBDE)* methods that simultaneously learn the prediction model and the input data distribution, with the ensemble disagreement used for the energy-based modelling of $p(x)$. Specifically, given a scalar-valued energy function $E_{\text{ed}}(x;\theta_{\text{ens}})$ for $\theta_{\text{ens}} = \{\theta_1, \ldots, \theta_M\}$ that

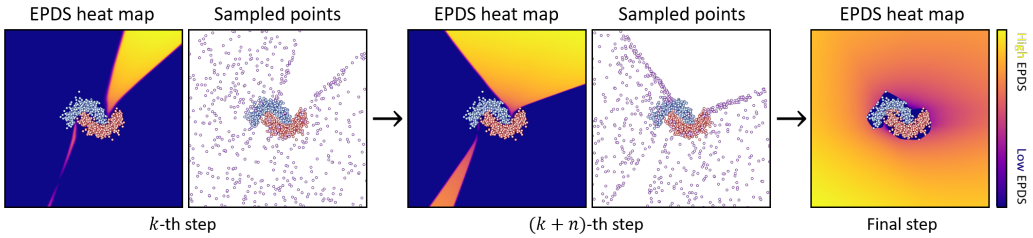

Figure 1.1: Training procedure of EBDE.

measures the degree of ensemble disagreement for x, we first define the energy-based model $p(x;\theta_{\text{ens}}) \propto \exp(-E_{\text{ed}}(x;\theta_{\text{ens}}))$. We then minimize the joint loss function that consists of the original DE loss term and the $p(x;\theta_{\text{ens}})$ learning loss term. Finally, the ensemble disagreement of DE captures $p(x)$ and becomes a valid OOD detection criterion.

We define an *Ensemble Pairwise Divergence Score* (EPDS) for $E_{\text{ed}}(\cdot)$ that enhances stable training of EBDE. As shown in Figure 1.1, during the training of EBDE, outliers that are thought to be inliers, i.e., input data x with low EPDS, are sampled, and the EBDE is trained to increase the EPDS of the sampled data points. Finally, the learned EBDE assigns high EPDS for outliers, i.e, the learned density model $p(x;\theta_{\text{ens}})$ produces low likelihood for outliers.

We use two different types of OOD score functions for EBDE using the learned data density $p(x;\theta_{\text{ens}})$: (i) the negative log-likelihood and (ii) the gradient norm of the log-likelihood. We then compare EBDE's OOD detection performance with existing state-of-the-art deep ensemble methods, e.g., DE, w-WGD, f-WGD, based on both synthetic (1-d regression and 2-D classification) and real data experiments (Image classification and UCI regression benchmarks).

The main contributions of this thesis can be summarized as follows:

1. We propose EBDE, a new family of DE methods that jointly learns $p(y|x)$ and $p(x)$;

2. We propose EPDS, a new measure of ensemble disagreement tailored for EBDE training;

3. We empirically show that EBDE is highly effective for OOD detection even using a relatively small number of models.

2

Background

In this chapter, we give backgrounds for the proposed method, the Energy-Based Deep Ensembles. First, we introduce the problem definition of prediction and predictive uncertainty quantification. Second, we introduce the Deep Ensemble methods and their variants that can perform not only prediction but also predictive uncertainty quantification. Last, we introduce a category of generative models, the *energy-based model*, which is a key idea of our proposed method.

2.1 Problem Definition

Let $\mathcal{X} \subset \mathbb{R}^D$ be an input space, a subspace of D-dimensional real Cartesian space, and $\mathcal{Y}$ be an output space (either discrete or continuous). A set of input-output paired data $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ assumed to be i.i.d. samples from a joint distribution $p(x, y)$ is given. For K-classes classification problem, we assume that the target $y \in \mathcal{Y}$ indicates one of K classes, $y \in \{1, \ldots, K\}$. For K-dimensional regression problem, we assume that the target is K-dimensional real-valued vector

$y \in \mathbb{R}^K$. To solve the K-classes classification and K-dimensional regression problems, we train DNN model $f_\theta : \mathcal{X} \to \mathcal{Y}$ on a dataset $\mathcal{D}$.

Although the DNN model shows remarkable predictive performance in many prediction tasks [1, 2, 3, 4, 5], it is hard to build a perfect DNN model with a limited dataset and computational budget. Therefore, if the model's prediction is inaccurate for some reason, it is necessary to quantify it and this is called predictive uncertainty quantification.

Predictive uncertainty can be divided into two types depending on the source. The first is caused by the existence of data, which is called *epistemic uncertainty*. This kind of uncertainty gradually decreases as more diverse training data are collected. The second is the uncertainty caused by the observation noise of the input x, which is also called *aleatoric uncertainty*. The aleatoric uncertainty can not be reduced by collecting more data, to reduce it, more information must be provided in the input observation. Both types of uncertainty make it difficult to fully trust the model's prediction, and by modeling these well, the DNN model's prediction can become more reliable. Also, information about what types of uncertainty dominate at test time can tell the direction of improvement of the DNN prediction system. The Figure 2.1 illustrate the types of uncertainties.

The epistemic uncertainty quantification is related to the out-of-distribution detection problem. The *out-of-distribution* data (or outlier) means that the input unlikely to be sampled from the input data distribution $p(x)$. For outliers, it is not possible to know what kind of prediction the trained model will make, and even in some cases, ground truth targets of inputs can lie on the out of the pre-defined output space $\mathcal{Y}$, so it is necessary to filter it separately. To detect outliers, we can use an OOD detector, which is defined with *OOD score function* $s : \mathbb{R}^D \to \mathbb{R}$ that outputs lower values for inliers and higher values for outliers, and classifies

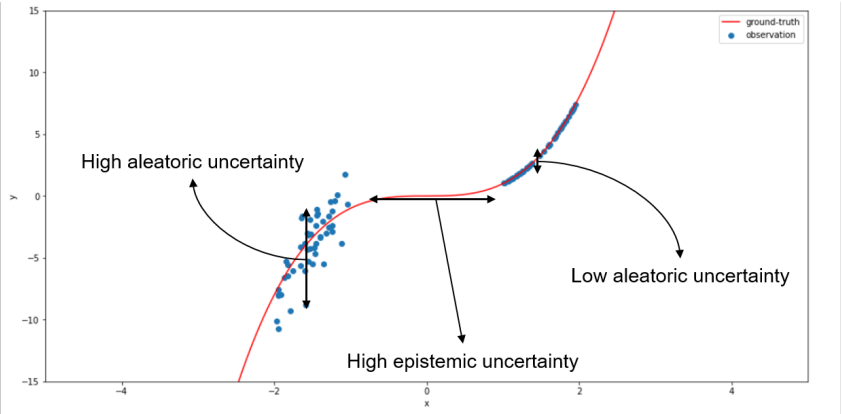

Figure 2.1: The types of uncertainty.

an input x as OOD if $s(x) > \eta$ for some threshold value η.

Among the remaining inputs after filtering outliers, there might exist falsely predicted inputs. This is because of either underfitting of the DNN model or aleatoric uncertainty in given inputs. In any case, it is a risk that trusts the model predictions unconditionally. One choice to handle this problem is modeling conditional probability density function of target $p(y|x;\theta)$ rather than the DNN model $f_\theta()$ just predict target $y \in \mathcal{Y}$.

That is to say, we target the problem of learning (i) a binary classifier that judges whether an input $x \in \mathcal{X}$ is an inlier or not, called an OOD detector, and (ii) a probabilistic prediction model, i.e., a conditional probability density function $p(y|x)$.

2.2 Uncertainty quantification using prediction models

2.2.1 Probabilistic DNNs

The deep neural networks (DNNs) consist of an input layer, a number of hidden layers, and an output layer. Normally, the output layer for the DNN prediction model is the linear layer that takes the hidden feature z and outputs a prediction $\hat{y}$. Then training of the DNN prediction model is proceeded by minimizing mean-squared-error (MSE) $\sum_{x,y\in\mathcal{D}} \|y - \hat{y}\|_2^2$.

However such point estimation $\hat{y}$ couldn't have any notion of predictive uncertainty. Therefore, the probabilistic DNN that has an additional output layer on top of DNN is widely used. In the case of classification problems, an output layer consists of a linear layer and *softmax* function in equation 2.2.1.

$$y_k = \frac{\exp(z_k)}{\sum_{i=1}^{K} \exp(z_i)} \tag{2.2.1}$$

The softmax function makes output y to sum-to-one vector, i.e., $\sum_{k=1}^{K} y_k = 1$. Then the k-th element of output vector is used as conditional probability $p(y = k|x)$ and the probabilistic DNN model with softmax funciton is trained by minimizing cross-entropy loss in equation 2.2.2. Minimizing the cross-entropy loss is equal to minimizing Kullback-Leibler divergence between ground truth probability density and predicted probability density.

$$\mathcal{L}_{ce} = -\sum_{x,y\in\mathcal{D}} \sum_{k=1}^{K} y_k \log \hat{y}_k \tag{2.2.2}$$

In the case of regression, it is not intuitive to design output probability density $p(y|x)$. One of popular choice is model output probability distribution $p(y|x)$ using exponential family (e.g., a Gaussian distribution). In other word the DNN model

outputs parameters of exponential family (e.g., mean $\mu \in \mathbb{R}^K$ and variance $\Sigma \in \mathbb{R}^{K\times K}$). In the rest of this thesis, we assume 1-dimensional regression problem and model output probability distribution as the Gaussian distribution $p(y|x;\theta) = N(y;\mu_\theta(x),\sigma^2_\theta(x))$ (where mean $\mu_\theta : \mathbb{R}^K \to \mathbb{R}$ and variance $\sigma^2_\theta : \mathbb{R}^K \to \mathbb{R}$).

2.2.2 Deep Ensemble

Deep Ensemble (DE) consists of multiple probabilistic deep neural network models, each of which is a conditional probability density function denoted by $p(y|x;\theta_i)$ for $i = 1, \cdots, M$ where θ_i is the parameter of the i-th model. We will denote the set of model parameters by $\theta_{\text{ens}} := \{\theta_1, \cdots, \theta_M\}$ and the mean prediction model by $p_E(y|x) := 1/M \sum_{i=1}^M p(y|x;\theta_i)$.

In the original DE paper [14], under the expectation that each model in the ensemble predicts diverse values y for an outlier x, the OOD detector is defined using the following score function:

$$s_{\text{ev}}(x;\theta_{\text{ens}}) := \sum_{i=1}^{M} D_{\text{KL}}(p(y|x,\theta_i)\|p_E(y|x)), \tag{2.2.3}$$

where $D_{\text{KL}}(\cdot||\cdot)$ denotes the KL divergence. This score function measures diversity of the model predictions for an input x. We call $s_{\text{ev}}(x;\theta_{\text{ens}})$ an **Ensemble Variance Score (EVS)** of x. Meanwhile, the predictive uncertainty can be quantified using the mean prediction model; one popular choice is the entropy of the mean prediction, i.e., $\mathcal{H}(p_E(y|x)) := \mathbb{E}_{y\sim p_E(y|x)}[-\log p_E(y|x)]$, which is often called the total uncertainty.

2.3 Ensemble diversification

To successfully detect outliers, DE methods should be trained in a way such that the variances of the models' prediction values (e.g., EVS) are high for outliers x and low for inliers. While the vanilla DE method has shown only limited performance for OOD detection, some training methods [18, 15, 19, 16] have been suggested to increase the diversity of the models in the ensemble, expecting the predicted values for outliers to diverge. These methods maximize the following form of joint loss function:

$$\alpha \cdot \sum_{i=1}^{M} \left(\mathbb{E}_{(x,y)\sim p(x,y)} [\log p(y|x;\theta_i)] \right) + \text{Diversity}(\{\text{NN}(x;\theta_i)\}_{i=1}^{M}), \tag{2.3.4}$$

where the second term is a measure of the diversity between neural network models $\text{NN}(x;\theta_i), i = 1, \cdots, M$ in the ensemble. In this framework, training results are heavily dependent on how the model diversity is defined, or equivalently, how differences between models are measured. The simplest way is to use the Euclidean distance in the weight space [18, 16], i.e., $\|\theta_i - \theta_j\|$, but this cannot properly capture the difference between functions.

A better way is to use function space distance metrics [15, 19, 16], which in general take the form $\mathbb{E}_{x\sim q(x)}[D(\{\text{NN}(x;\theta_i)\}_{i=1}^{M})]$, where the sampling distribution q and diversity measure D that computes the degree of differences between network output values at given x are engineering choices. Considering that only the prediction values for outliers should diverge, while those for inliers should be consistent, outliers should be more likely to be sampled from the sampling distribution $q(x)$. However, there is no straightforward method nor principle for designing such a q. Indeed, existing works [15, 19, 16] use mostly ad hoc choices, e.g., the kernel density estimator (KDE) fitted to the training dataset or a uniform distribution,

resulting in much less than desirable OOD detection performance.

The EBDE, by learning $p(x)$, naturally achieves the desired property of maximizing the diversity of the model's predictions for outliers only. Interestingly, during EBDE training, outliers that are thought to be inliers are sampled without needing to define the sampling distribution $q(x)$ as in existing approaches, and their prediction values are also diversified.

2.4 Energy-based models

An energy-based model (EBM) represents a probability density function $p(x;\theta)$ with an unnormalized scalar-valued energy function $E(\cdot;\theta) : \mathbb{R}^D \to \mathbb{R}$ via the Gibbs distribution, i.e.,

$$p(x;\theta) := \frac{1}{\Omega_\theta} \exp(-E(x;\theta)/T), \tag{2.4.5}$$

where θ is a parameter, $T \in (0, \infty)$ is called the temperature, and Ω_θ is the normalization constant.

Although it is expensive to compute Ω_θ and hence difficult to calculate an exact likelihood $p(x;\theta)$, one can still perform maximum likelihood training of θ with the following gradient computation method [20]:

$$\begin{aligned} \mathbb{E}_{x\sim p(x)}[-\nabla_\theta \log p(x;\theta)] &= \mathbb{E}_{x\sim p(x)}[\nabla_\theta E(x;\theta)]/T + \nabla_\theta \log \Omega_\theta \\ &= \mathbb{E}_{x\sim p(x)}[\nabla_\theta E(x;\theta)]/T - \mathbb{E}_{x\sim p(x;\theta)}[\nabla_\theta E(x;\theta)]/T. \end{aligned} \tag{2.4.6}$$

The gradient of the log normalization constant $\nabla_\theta \log \Omega_\theta$ is evaluated from the expected energy gradients of the samples generated from the model $x \sim p(x;\theta)$, which are often called the "negative" samples. The detailed derivation of equation 2.4.6 is in Section A.1.

Likelihood maximization is achieved by minimizing the expected energy gradients of the "positive" samples generated from the data distribution $p(x)$ – we often use the empirical distribution – and maximizing the expected energy gradients of the "negative" samples generated from $p(x;\theta)$. When $p(x;\theta)$ becomes identical to $p(x)$, the two gradient terms in the equation cancel each other, and training of EBM converges. In practice, the two expectations are approximated with a mini-batch of samples during each training iteration, where negative samples are often sampled using MCMC, e.g., Langevin Monte Carlo (LMC). For more details on EBMs and their training methods, we refer the reader to [21].

3

Energy-Based Deep Ensembles

In this chapter, we propose the **Energy-Based Deep Ensemble (EBDE)**, a new family of DE that improves OOD detection performance. The EBDE consists of not only a prediction model $p(y|x)$, but also a density estimator $p(x)$. Given an ensemble of multiple prediction models $\{p(y|x;\theta_i)\}_{i=1}^{M}$, the conditional probability density function $p(y|x)$ is defined in the usual way as the mean prediction model $p_E(y|x)$. In this thesis, we introduce an energy-based model for the input data probability density function $p(x;\theta_{\text{ens}})$, which is then jointly trained with prediction models.

3.1 Ensemble Disagreement as an Energy Function

A data point x is an outlier if it is unlikely to be sampled from the input data distribution $p(x)$. In this sense, it is natural to connect the OOD detection criteria of DE, the degree of disagreement between each model's prediction, to the data density $p(x)$. Let $E_{\text{ed}}(x;\theta_{\text{ens}})$ be some measure of the **Ensemble Disagreement**,

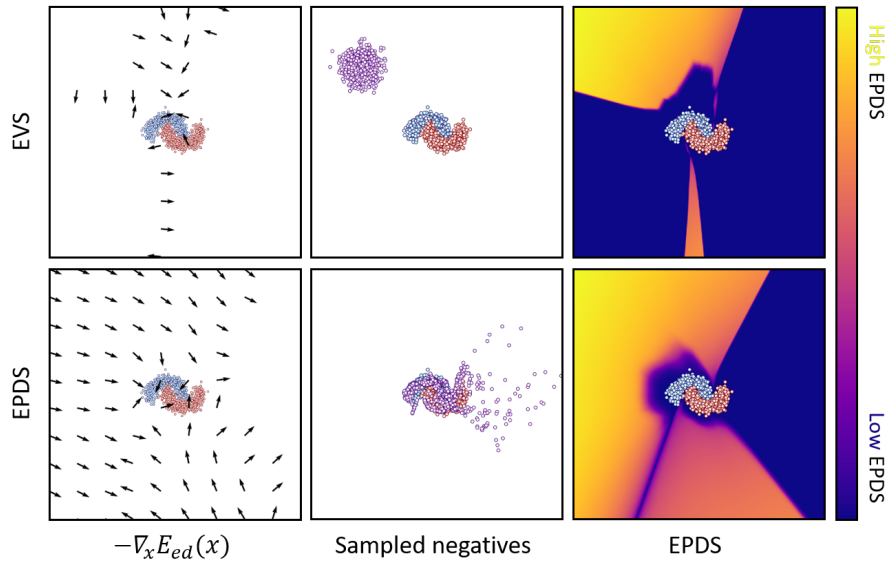

Figure 3.1: Comparison of EVS and EPDS as energy functions for EBDE.

i.e., how different the $p(y|x;\theta_i)$ are for $i = 1, \cdots, M$. We then model $p(x;\theta_{\text{ens}})$ by treating E_{ed} as an energy function:

$$p(x;\theta_{\text{ens}}) = \frac{1}{\Omega_{\theta_{\text{ens}}}} \exp(-E_{\text{ed}}(x;\theta_{\text{ens}})/T). \tag{3.1.1}$$

If θ_{ens} is optimized not only to learn $\{p(y|x;\theta_i)\}_{i=1}^{M}$ but also $p(x;\theta_{\text{ens}})$, then for data x that have a lower likelihood, the trained models in the ensemble will disagree more than for data that have a higher likelihood.

The Ensemble Variance Score (EVS) of equation (2.2.3) is one straightforward choice for the ensemble disagreement energy function $E_{\text{ed}}(x;\theta_{\text{ens}})$. However, for such a choice, we empirically discover that training the energy-based model is numerically quite challenging. Specifically, when sampling from the model $p(x;\theta_{\text{ens}})$,

which is a necessary step for training an energy-based model as explained in Section **??**, we experience the gradient vanishing problem, strongly diminishing the performance of the sampling algorithm. As shown in Figure 3.1, when using EVS as the energy function, $-\nabla_x E_{\text{ed}}(x)$ vanishes at both high and low energy regions. Thus, the sampled points are often confined within the high energy region while it is desired to be sampled from the low energy region. On the other hand, using EPDS (defined below) as the energy function does not suffer from the gradient vanishing problem. A more in-depth analysis of this phenomenon is provided in the Appendix.

In this thesis, we propose to use another measure of the ensemble disagreement as an energy function, an *Ensemble Pairwise Divergence Score (EPDS)*: [Ensemble Pairwise Divergence Score (EPDS)] Given multiple probabilistic prediction models $\{p(y|x;\theta_i)\}_{i=1}^{M}$, an ensemble pairwise divergence score for an input $x \in \mathcal{X} \subset \mathbb{R}^D$ is

$$s_{\text{epd}}(x;\theta_{\text{ens}}) = \frac{1}{M(M-1)} \sum_{i \neq j} D(p(y|x;\theta_i) \| p(y|x;\theta_j)), \tag{3.1.2}$$

where D is some divergence measure (e.g., K-L divergence). Any divergence D can be selected; throughout this thesis, we always use the K-L divergence unless otherwise specified. This score function is a valid measure of the ensemble disagreement, in the sense that the score is zero if and only if the predictions are the same for all $i = 1, \cdots, M$. Using the EPDS as the energy function has multiple practical advantages over EVS. First, when sampling from the energy-based model, it does not suffer from the gradient vanishing problem as illustrated in Figure 3.1. Second, while EVS often does not have close-form expressions (even when the K-L divergence D and Gaussian prediction model $p(y|x;\theta_i)$ are used), EPDS has closed-form expressions if the terms $\sum_{i \neq j} D(p(y|x;\theta_i) \| p(y|x;\theta_j))$ in the summation have analytic expressions (which for most cases is true).

For EBDE training, we jointly learn the ensemble of prediction models $p(y|x;\theta_i)$, classifiers or regressors, and the energy-based density model $p(x;\theta_{\text{ens}}) \propto \exp(-s_{\text{epd}}(x;\theta_{\text{ens}})/T)$ by maximizing the following joint objective function:

$$\alpha \cdot \sum_{i=1}^{M} \left(\mathbb{E}_{(x,y)\sim p(x,y)}[\log p(y|x;\theta_i)]\right) + \mathbb{E}_{x\sim p(x)}[\log p(x;\theta_{\text{ens}})], \tag{3.1.3}$$

where $\alpha > 0$. The gradient of the objective function (needed to apply stochastic gradient descent methods) can be computed as in usual prediction model and energy-based model training methods; the resulting implementation is given in Algorithm 1.

3.2 OOD Score Function

In this section, we propose two types of OOD score function for EBDE by exploiting the learned input data density function $p(x;\theta_{\text{ens}})$: (i) negative log-likelihood and (ii) gradient norm of the log-likelihood. Given a learned data density, existing works on OOD detection have suggested roughly two directions for defining the OOD score function. The first direction is to use the log-likelihood, i.e., $s(x;\theta_{\text{ens}}) := -\log p(x;\theta_{\text{ens}})$, based on the idea that data that have a lower likelihood are more likely to be outliers. In our case, the OOD score function is proportional to the ensemble pairwise divergence score, EPDS, a measure of the ensemble disagreement. We may use other measures of the ensemble disagreement as the OOD score function, e.g., the ensemble variance score, EVS, as in usual DE approaches, although they are not strictly equivalent to the negative log-likelihood. Somewhat counter-intuitively, we empirically find that using EVS results in much better OOD detection performance than using EPDS. The comparison results are included in Appendix.

The other direction is to consider the probability mass rather than the likelihood. While it is reasonable to regard data with a low likelihood to be outliers, it is possible for data to have high likelihood yet be nearly impossible to be sampled. To be specific, a single data point may have high density but if its neighborhood points have very low densities, then the local probability mass around that data point is small and that point is less likely to be sampled. This phenomenon has actually been reported in [22, 23] across various types of density models [24, 25, 26]. For this class of high-likelihood outliers, the probability density is expected to change rapidly nearby, and hence the gradient norm of the log-likelihood is expected to be large. Based on this expectation, [27] proposed following OOD score function:

$$s(x;\theta_{\text{ens}}) = \left\| \left(\frac{\partial \log p(x;\theta_{\text{ens}})}{\partial x} \right) \right\|_2, \tag{3.2.4}$$

referred to as an Approximate Mass Score (AMS). Following [27], we also use AMS in our work.

Algorithm 1 EBDE

Input : Training data $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$, probabilistic prediction models $\{p(y|x;\theta_i)\}_{i=1}^{M}$, weight α, batch size B

Output: Parameters of ensemble model $\theta_{\text{ens}} = \{\theta_i\}_{i=1}^{M}$

while *not converged* **do**

Sample positive mini-batch $\mathcal{B}^+ = \{(x_b, y_b)\}_{b=1}^{B} \subset \mathcal{D}$

Sample negative mini-batch $\mathcal{B}^- = \{x_b\}_{b=1}^{B} \sim p(x;\theta_{\text{ens}})$ (Section **??**)

Compute energy based model objective $\mathcal{L}_{ebm}$:

$\mathcal{L}_{ebm} \leftarrow \frac{1}{B}\sum_{x\in\mathcal{B}^+} E_{ed}(x;\theta_{\text{ens}}) - \frac{1}{B}\sum_{x\in\mathcal{B}^-} E_{ed}(x;\theta_{\text{ens}})$

while $i = 1, \ldots, M$ **do**

Sample positive mini-batch $\mathcal{B}_i^+ = \{(x_b, y_b)\}_{b=1}^{B} \subset \mathcal{D}$

Compute i-th model's negative log-likelihood $\mathcal{L}_{nll}^{(i)}$:

$\mathcal{L}_{nll}^{(i)} \leftarrow -\frac{1}{B}\sum_{(x,y)\in\mathcal{B}_i^+} \log p(y|x;\theta_i)$

Compute gradients $\Delta\theta_i \leftarrow \alpha\nabla_{\theta_i}\mathcal{L}_{nll}^{(i)} + \nabla_{\theta_i}\mathcal{L}_{ebm}$

end

while $i = 1, \ldots, M$ **do**

Update parameters θ_i based on $\Delta\theta_i$ using optimizer

end

end

4

Results and Discussion

In this section, we compare the proposed EBDE with the vanilla DE and its variants, the ensemble diversification methods introduced in Section **??**, for classification and regression tasks. In classification, we use a neural network model that first maps an input x to the logit space, and then a softmax layer takes the logit and outputs a probability vector. In regression, we model the output distribution as the Gaussian $\mathcal{N}(\mu(x;\theta), \sigma^2(x;\theta))$ unless otherwise specified. The baseline methods that are subject to comparisons are:

Vanilla DE consists of multiple probabilistic deep neural network models, and they are trained individually with differently initialized network weights.

w-WGD [16] defines the ensemble diversity measure between models $\{\text{NN}(x;\theta_i)\}_{i=1}^{M}$ in the weight space (the second term in equation (2.3.4).

f-WGD [16] defines the ensemble diversity measure in the function space. As discussed in Section **??**, to measure the difference between neural network models in the function space, we need to determine the sampling distribution q in the input space and the diversity measure D in the neural network output space. For

regression tasks, we use the empirical data distribution with added noise for q, and the diversity measure D is defined by using the predicted mean values, i.e., we compute the diversity of $\{\mu(x;\theta_i)\}_{i=1}^{M}$. For classification tasks, we use the empirical data distribution both with and without added noise for q, and the diversity measure D is either defined in logit space, i.e., intermediate feature space right before the output space, or the output space after the softmax layer. Therefore, we test four combinations: (i) without noise, logit space (f-WGD-l), (ii) with noise, logit space (f-WGD-l-n), (iii) without noise, output space (f-WGD-sm), and (iv) with noise, output space (f-WGD-sm-n).

In our implementations, for w-WGD and f-WGD, we use the RBF kernels when computing the diversity measure with the median heuristics to choose the kernel bandwidths following [18, 16].

Evaluation Metrics. For quantitative comparisons, we use the following three evaluation metrics: (i) Mean Squared Error (MSE) and Accuracy for regression and classification, respectively, used to compute the prediction performances, (ii) Negative Log Likelihood (NLL) of the mean prediction model used to measure the performance of the predictive uncertainty quantification, and (iii) Area Under the Receiver Operating Characteristic (AUROC), a threshold-free measure for evaluating OOD detector, computed with two score functions EVS and AMS in Section 3.2.

4.1 Synthetic Experiments

1D regression. We qualitatively evaluate the performance of the proposed method on a one-dimensional toy regression task. For each method, we train 5 MLP model to fit a constant-valued function, $y = 0 \cdot x$. For training, we use 40 points, where

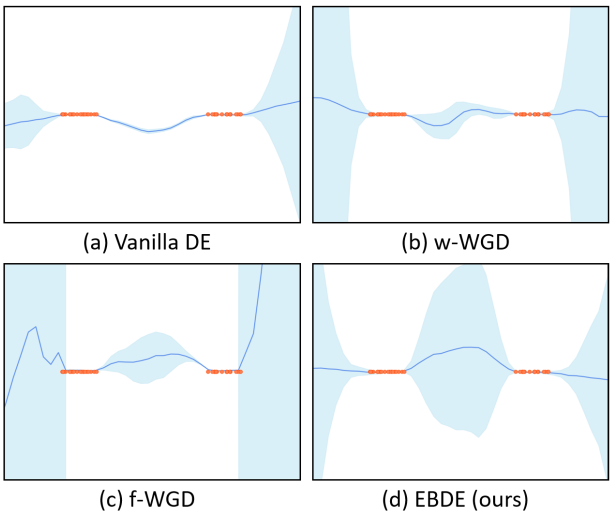

Figure 4.1: The 1D regression on $y = 0 \cdot x$.

20 points are sampled from Uniform(-1.25, 0.75) and the other 20 points are sampled from Uniform(0.75, 1.25), visualized as orange dots in Figure 4.1. The width of blue shaded area represents degree of the ensemble disagreement (measured by EPDS), the wider the higher. For all methods, five DNN models are used.

As shown in Figure 4.1, the mean predictions of all methods (blue solid lines) fit well on training inputs. The width of blue shaded area represents the degree of the ensemble disagreement (the wider, the higher). The w-WGD and f-WGD produce slightly better results than the vanilla DE, yet the EBDE only successfully produces high ensemble disagreements for all three OOD regions (left, middle, and right parts). In addition, we note that f-WGD results in high ensemble disagreement even for inliers (both ends of training data), as prediction values for samples near training data are diversified in the f-WGD training.

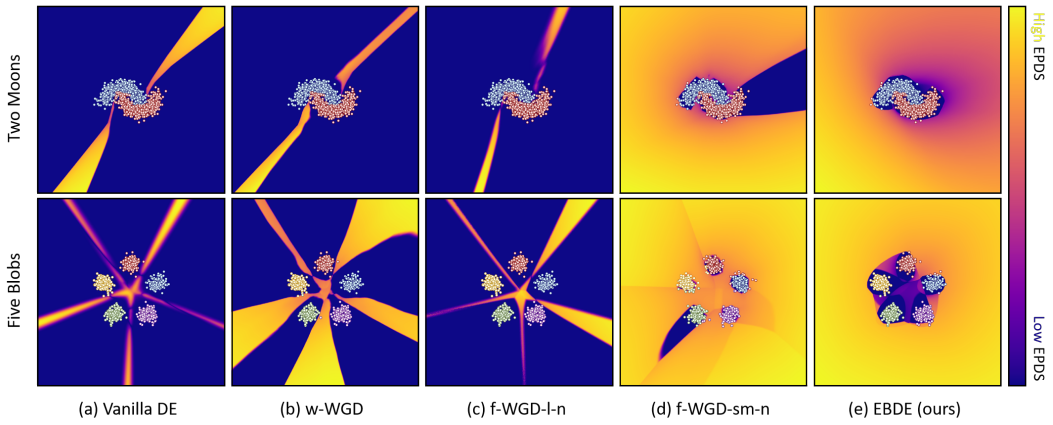

Figure 4.2: The qualitative results on 2D classification benchmarks, *two moons* (top row) and *five blobs* (bottom row) benchmarks.

2D classification. We investigate the EBDE and the other baseline methods on two-dimensional toy classification tasks, *two moons* and *five blobs*. First, we compare qualitatively as shown in the Figure 4.2. In the figures, the colors of dots represent predicted classes by the mean prediction model. The background color represents the ensemble pairwise divergence score (EPDS), the brighter the higher. For all methods, two DNN models are used. The vanilla DE, w-WGD, and f-WGD-l-n fail to produce high EPDS for most of the outliers that are far from the classification boundaries. The f-WGD-sm-n produces slightly better results, still there are some outliers that have low EPDS. In addition, there are some inliers that have high EPDS as observed in the above toy regression example. The EBDE shows the best results compared to the other baselines.

Secondly, we provide quantitative comparison results of OOD detection performance using synthetic outliers. To generate the synthetic outliers, from uniformly sampled candidate points in a square region surrounding the training data, we

only accept points of which $l2$ norm are at least ϵ (we fix $\epsilon = 0.3$) away from all training data. We investigate the OOD detection performance of EBDE and the other baseline methods by changing the number of models in the ensemble. For all experiments, we report AUROC using EPDS as an OOD score, run each experiment 5 times at different seeds, and report averages and standard errors. The best one is bolded. As shown in Table 4.1, EBDE outperforms the other baseline methods and shows robust OOD detection performances even when using a small number of models. Details of experiments are in Appendix.

Table 4.1: 2D synthetic OOD detection results.

Dataset	# of models	Deep Ensembles	w-WGD	f-WGD-l-n	f-WGD-sm-n	EBDE (ours)
Two moons	2	19.75(±1.30)	23.29(±3.21)	17.99(±1.74)	93.41(±5.79)	**98.46**(±1.35)
	3	21.55(±1.79)	25.94(±1.33)	27.22(±2.64)	99.61(±0.06)	**99.78**(±0.06)
	5	23.43(±4.37)	38.75(±12.87)	26.49(±2.69)	99.87(±0.13)	**99.83**(±0.11)
	10	26.33(±1.60)	44.42(±3.41)	37.58(±2.26)	99.90(±0.04)	**99.95**(±0.02)
Five blobs	2	47.00(±1.19)	60.07(±3.08)	55.66(±2.09)	97.27(±1.16)	**99.52**(±0.15)
	3	47.94(±0.99)	71.87(±1.34)	55.12(±2.76)	99.35(±0.06)	**99.49**(±0.18)
	5	49.72(±1.55)	70.24(±3.26)	51.89(±4.60)	99.62(±0.15)	**99.72**(±0.13)
	10	50.18(±0.74)	77.62(±1.15)	57.59(±2.16)	99.33(±0.06)	**99.71**(±0.04)

4.2 Image Classification

Moving on to real-world data, we compare performances of EBDE and the baseline methods in the prediction, uncertainty quantification, and OOD detection tasks on the FashionMNIST [28] dataset. For OOD detection, we use MNIST [29], KMNIST [30], and Omniglot [31] as OOD datasets. We compare the prediction performance of our methods and the others by classification accuracy and compare uncertainty quantification by negative log-likelihood (NLL) for the test dataset. Also,

we provide OOD detection performance against three OOD datasets (MNIST, KMNIST, Omniglot). As the OOD score $s(\cdot)$, we use both ensemble variance score (EVS) and approximate mass score (AMS). We run each experiment 5 times at different seeds and report a mean and standard error. The best one is bolded. Details of experiments are in Appendix.

Table 4.3 shows that the EBDE outperforms the other baselines for most cases in OOD detection tasks. Furthermore, we empirically observe that the EBDE does not degrade the prediction and uncertainty quantification performances compared to the vanilla DE as shown in the Table 4.2 (the Accuracy is even higher, and the NLL is even lower). In addition, we provide the density plot in Figure 4.3, which is a continuous version of the histogram of the ensemble variance score (EVS). The horizontal axis is log-scale ensemble variance scores (EVS) normalized with minimum and maximum values, and the vertical axis is the density. Similarly, for EBDE, the density graphs between the inlier dataset and OOD datasets are more separated than the other baselines.

Table 4.2: The predictive performance on the FashionMNIST dataset.

Method	Accuracy	NLL
Vanilla DE	89.98(±0.11)	0.2945(±0.0013)
w-WGD	89.78(±0.10)	0.3027(±0.0013)
f-WGD-l	89.78(±0.08)	0.3006(±0.0022)
f-WGD-l-n	89.77(±0.07)	0.3071(±0.0043)
f-WGD-sm	89.65(±0.21)	0.3816(±0.0041)
f-WGD-sm-n	89.85(±0.13)	0.3042(±0.0024)
EBDE (ours)	**90.17**(±0.09)	**0.2939**(±0.0042)

Table 4.3: The OOD detection performance on the FashionMNIST dataset.

Method	AUROC against OOD (EVS/AMS)		
	MNIST	KMNIST	Omniglot
Vanilla DE	87.85(±0.58) / 89.02(±0.58)	85.57(±0.39) / 86.54(±0.42)	90.80(±0.53) / 92.51(±0.31)
w-WGD	91.19(±0.40) / 92.05(±0.49)	88.39(±0.08) / 89.84(±0.24)	90.73(±0.92) / 92.18(±0.82)
f-WGD-l	92.43(±0.18) / 92.29(±0.78)	89.20(±0.35) / 89.19(±0.69)	91.35(±0.66) / 91.99(±0.86)
f-WGD-l-n	92.45(±0.38) / 92.64(±0.47)	88.17(±0.65) / 88.13(±1.15)	90.95(±0.70) / 91.51(±1.03)
f-WGD-sm	77.18(±0.63) / 84.81(±0.91)	71.25(±0.45) / 79.15(±0.39)	78.03(±1.92) / 86.95(±0.86)
f-WGD-sm-n	93.96(±0.93) / 93.73(±0.62)	**96.66**(±0.31) / **96.71**(±0.20)	92.16(±0.79) / 91.61(±0.53)
EBDE (ours)	**96.36**(±0.65) / **97.93**(±0.23)	93.06(±0.70) / 95.17(±0.44)	**92.76**(±0.36) / **95.02**(±0.51)

4.3 Generative results

Figure 4.4 shows images sampled from vanilla DE, w-WGD, f-WGD-sm-n, and EBDE where liners were FMNIST. For all methods, we updated images to have low energy starting from random noise using Langevin dynamics until sampled images had a similar or lower energy to the inliers energy ((a), (b), (c), and (d) in Figure 4.4). For EBDE, we provide the other images obtained from the persistence chain during training ((e) in Figure 4.4).

As shown in Figure 4.4, the images which are sampled from vanilla DE, w-WGD, and f-WGD-sm-n, didn't look like inliers even though they had energy as low as inliers' energy ((a), (b), and (c) in Figure 4.4). On the other hand, EBDE generated the images that look like inliers ((d) and (e) in Figure 4.4). This qualitative results support that density model of EBDE $p(x;\theta_{\text{ens}})$ learns data distribution $p(x)$ and then inliers are likely to be drawn from learned probability density $p(x;\theta_{\text{ens}})$. However, the other methods (vanilla DE, w-WGD, and f-WGD) aren't learning data distribution $p(x)$, so even images that look like noise are assigned to low EPDS (ensemble disagreement).

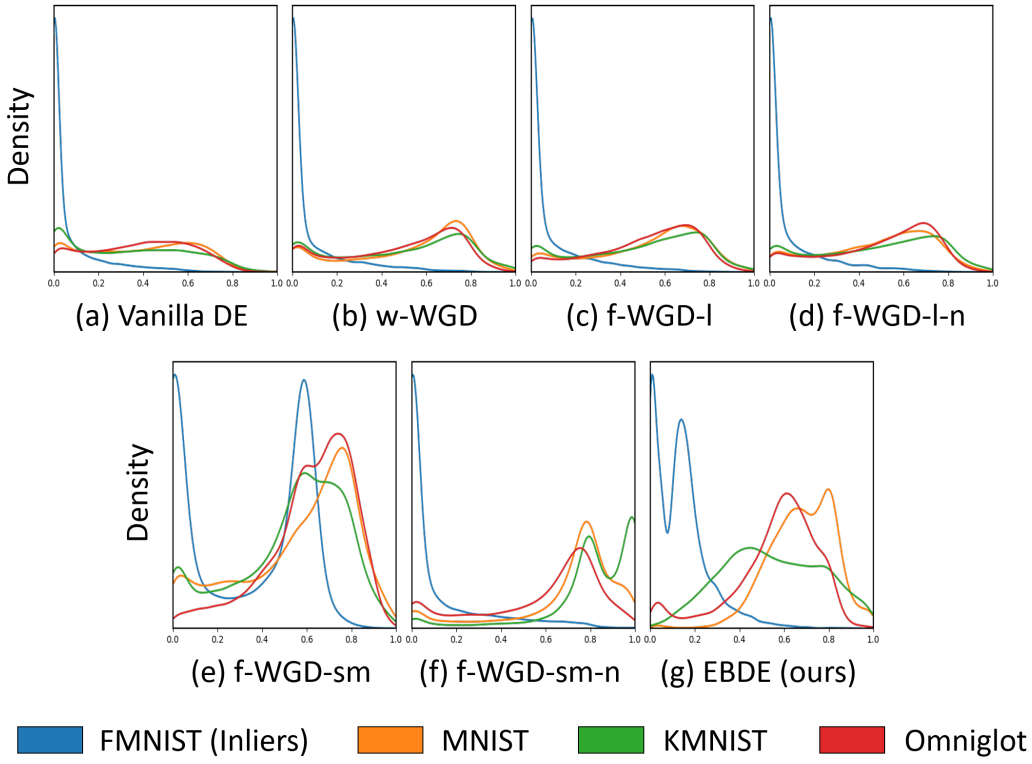

Figure 4.3: The FMNIST vs. OOD datasets density plots (MNIST, KMNIST, and Omniglot) using ensemble variance score (EVS).

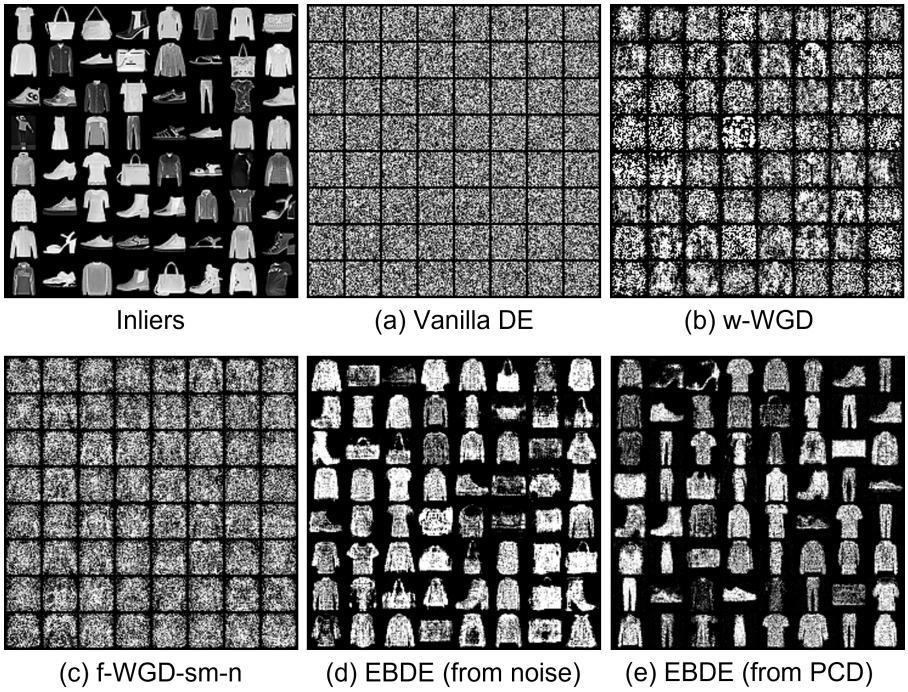

Figure 4.4: The sampled images where inliers are FMNIST.

4.4 Regression on UCI Datasets

In this section, we use six UCI regression datasets following [14, 15, 11, 32]. To measure the OOD detection performances, we first generate synthetic outliers using the normalizing flow model, the RealNVP [33], which is a model for $p(x)$ where we can sample data from and that can be used to measure the likelihood of given data. Specifically, (i) we train the RealNVP model $p_{\text{realNVP}}(x)$ on UCI regression data, (ii) we sample data points from Gaussian distribution (large enough to encapsulate the training data), and (iii) we accept the sampled points as outliers if the likelihood of them estimated by the trained RealNVP are lower than those of all inliers with some margin.

In the Table 4.4 and the Table 4.5, we report the mean squared error (MSE), negative log-likelihood (NLL), and AUROC for synthetic OOD detection. We run each experiment 10 times at different train/valid/test splits and report mean values. The best and the second best are bolded, and the worst is colored red. The OOD detection performances using the other scores (EPDS and AMS) and the other experimental details are in Appendix. As shown in Table 4.5, both f-WGD and EBDE outperform vanilla DE and w-WGD in terms of OOD detection performance. Meanwhile, EBDE shows comparable prediction and uncertainty quantification performance with DE and w-WGD, whereas f-WGD does not.

Table 4.4: Regression results on the 6 UCI benchmarks datasets.

Datasets	MSE				NLL			
	DE	w-WGD	f-WGD	EBDE	DE	w-WGD	f-WGD	EBDE
Concrete	**0.1615**	**0.1608**	0.3981	0.1623	**0.3654**	**0.3565**	0.5953	0.3766
Energy	**0.0491**	**0.0499**	0.0711	0.0512	**-0.5135**	**-0.4995**	-0.3200	-0.4276
Housing	**0.1505**	**0.1505**	0.3654	**0.1482**	0.2898	**0.2799**	0.7448	**0.2836**
Power	**0.0543**	**0.0536**	0.3891	**0.0543**	**-0.0584**	**-0.0703**	-0.0314	-0.0567
Wine	**0.5479**	0.5526	2.1321	**0.5428**	1.0868	**1.0845**	1.1323	**1.0807**
Yacht	**0.0549**	0.0835	0.1840	**0.0526**	**-1.3939**	-1.0980	-1.0114	**-1.1087**

We plot the trade-off curve (red curve in Figure 4.5) of NLL and (1-AUROC) to investigate the effect of α in equation 3.1.3 in EBDE (The lower and more left, the better.). The trade-off curve for EBDE and f-WGD are obtained by changing α (in equation 3.1.3) and the magnitude of noise added to input, respectively. Averages (dots) and standard errors (ellipses) are computed with 10 times experiments. As α increases, the AUROC decreases (in the graph, it goes right) and the NLL decreases (in the graph, it goes down). Compared to the other methods, the trade-off curve of EBDE is located lower and more left, meaning that the

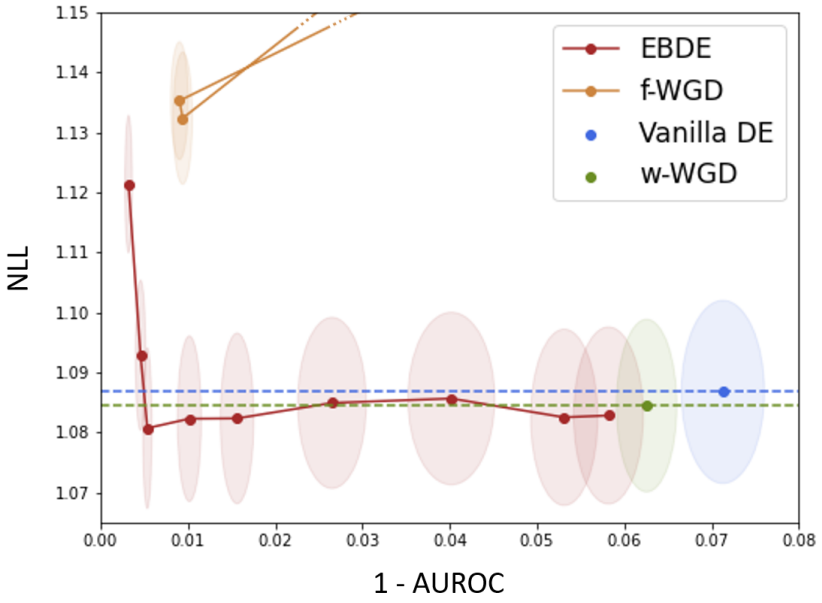

Figure 4.5: The trade-off curve between NLL and "1-AUROC".

Table 4.5: Synthetic OOD detection results on the 6 UCI benchmarks datasets.

Datasets	AUROC (EVS)			
	DE	w-WGD	f-WGD	EBDE
Concrete	98.58	98.58	**99.88**	**99.69**
Energy	**99.92**	99.87	**100.0**	**99.92**
Housing	96.82	96.94	**99.27**	**98.46**
Power	99.90	99.77	**99.98**	**99.96**
Wine	92.87	93.74	**99.06**	**99.46**
Yacht	90.10	89.81	**99.69**	**93.33**

EBDE outperforms in both aspects, uncertainty quantification and OOD detection. Additionally, for f-WGD, we test how NLL and AUROC vary by changing the magnitude of noise added to input data. Regardless of the noise level, the f-WGD produces high NLL, and the yellow curve located upper and more right than the red curve.

5

Experimental Methods

5.1 Experimental Details

Synthetic 1D regression. We used 40 points as a training dataset, where 20 points were sampled from $\mathcal{U}(-1.25, 0.75)$ and the other 20 from $\mathcal{U}(0.75, 1.25)$. Target y of each point is all 0, i.e., ground truth was constant-valued function, $y = 0 \cdot x$.

In each experiment, we train five differently initialized MLP models of size (1-128-128-128-2) with a leaky ReLU activation function. The first output was used for the mean of Gaussian $\mu(x; \theta)$, and the second output was used for the variance of Gaussian $\sigma^2(x; \theta)$. Because the variance of Gaussian should be positive, we passed the second dimension of outputs to softplus function $softplus(z) = \log(1 + \exp(z))$. We trained the models for 1000 epochs using an Adam [34] optimizer with a learning rate of 1e-4 for all methods. For w-WGD, f-WGD and EBDE, we fixed α as 1 in equation (8). Additive noise on inputs in f-WGD did follow $\mathcal{N}(0, 1)$.

Synthetic 2D classification. For both *two moons* and *five blobs* dataset, we used 1000 points for training and different 1000 points for testing. To generate synthetic outliers, (i) we sample 10000 candidate points from $\mathcal{U}(-3, 3)$, (ii) and only accept points of which $l2$ norm is at least 0.3 away from all training data. The 0.3 is a number selected so that the generated outliers and training set are visually disjoint.

In each experiment, we trained differently initialized MLP models of size (2-256-256-256-K) with a leaky ReLU activation function where K was the number of classes. We trained the models for 5000 epochs using an Adam [34] optimizer with a learning rate of 1e-4 for all methods. For w-WGD and f-WGDs(f-WGD-l, f-WGD-l-n, f-WGD-sm, and f-WGD-sm-n), we fixed α as 1 in equation (8) following [16]. Additive noise on inputs in f-WGD-l-n and f-WGD-sm-n did follow $\mathcal{N}(0, 0.0625I)$. For EBDE, the weight of EBM loss was 0.001, and we selected this number by reducing the weight of EBM loss from 1.0 until the losses of prediction models $\{p(y|x; \theta_i)\}_{i=1}^{M}$ stably reduced.

FMNIST classification. All datasets (FMNIST, MNIST, KMNIST, and Omniglot) used in experiments are downloaded through each dataset class in the PyTorchVision library. For all datasets, we normalized the images to [-1, 1]. For FMNIST, we split it as 58000/2000/10000 for training/validation/testing sets respectively. During the training, we applied random horizontal flipping with 0.5 probability and random cropping with size 4 padding. For MNIST, KMNIST and Omniglot, we use their test split (10000/10000/13180 images respectively) as outliers for the OOD detection task. Because the Omniglot has a higher pixel value on the background of images, we invert the pixel values of Omniglot in our experiments.

In each experiment, we trained three differently initialized MLP models of size (784-500-500-100-10) with a ReLU activation function. For all methods, we trained

the models for 100000 iterations using an Adam [34] optimizer. The batch size was 250. The results in Table 2 and Figure 5 were reported using the model with the lowest negative log-likelihood (NLL) on the validation set. For each method, the learning rate was selected among [1e-4, 5e-4, 1e-3, 5e-3] based on the lowest NLL on the validation set. The selected learning rates were 5e-4, 1e-3, 1e-3, and 1e-3 for vanilla DE, w-WGD, f-WGDs, and EBDE respectively. Also, for EBDE, we use betas of the Adam optimizer as [0., 0.9] following [27]. For w-WGD and f-WGDs, we fixed α as 1 in equation (8) following [16]. The standard deviation of additive noise for f-WGD-l-n and f-WGD-sm-n was selected among [0.05, 0.1, 0.25, 0.5, 1.0], and 0.5 was selected based on the lowest NLL on the validation set. For EBDE, α was selected as 0.2 among [0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0] based on the lowest NLL on the validation set. For each method, we repeated experiments five times on different random seeds (337, 1337, 2337, 3337, 4337)

UCI regression. All datasets (concrete, energy, housing, power, wine, and yacht) were randomly divided for training/validation/testing at ratios of 0.8/0.1/0.1 respectively (this split was changed during repeating each experiment).

In each experiment, we trained five differently initialized MLP models of size (D-50-50-50-2) with a ReLU activation function (where D denotes the dimension of input space). As in synthetic 1D regression, the first and second dimensions of outputs were used as the mean and variance of Gaussian respectively. For in Table 3, we trained the models for 100 epochs using Adam [34] optimizer with a learning rate of 1e-4 and reported performance using the model with the lowest NLL on the validation set. The batch size was 50. For w-WGD and f-WGDs, we fixed α as 1 in equation (8) following [16]. The standard deviation of additive noise for f-WGD-l-n and f-WGD-sm-n was selected among [0.0, 1.0, 5.0, 10.0] based on NLL on the validation set. For EBDE, we fixed α as 1 excepting wine dataset. In the

cases of wine, α was selected as 0.1 among [0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0] based on NLL on the validation set.

5.2 Persistent Contrastive Divergence for EBM training

For all EBDE experiments, we trained an energy-based model (EBM) using PCD [35]. In PCD, negative samples are generated from persistence chains, i.e., Langevin Monte Carlo (LMC) sampling starts from past negative samples which are stored in a replay buffer. The sampling process using PCD is given in the following algorithm:

Algorithm 2 PCD sampling

Input : The number of sample steps η, step size β, standard deviation of noise γ, replay buffer B, replay ratio ρ, gradient clipping magnitude a, initial random noise distribution $\mathcal{R}$, energy funciton $E(\cdot;\theta_{\text{ens}})$

Output: Negative sample $\hat{x}_\eta$

Initialize Sample $\hat{x}_0 \sim B$ with probability $1-\rho$, else $\hat{x}_0 \sim \mathcal{R}$.

while $t = 1, \ldots, \eta$ **do**

  Compute gradient of energy at current input $\hat{x}_{t-1}$, $g \leftarrow \frac{\partial E_{\text{ed}}(\hat{x}_{t-1};\theta_{\text{ens}})}{\partial \hat{x}_{t-1}}$

  Clip gradient, $g \leftarrow \max(\min(g, |a|), -|a|)$

  Sample noise of Langevin dynamics, $\epsilon \sim \mathcal{N}(0, I)$

  Update sample, $\hat{x}_t = \hat{x}_{t-1} - \beta \cdot g + \gamma \cdot \epsilon$

end

Push negative sample $\hat{x}_\eta$ to replay buffer B

For all EBDE experiments, the parameters of PCD sampling were chosen so that both prediction models $\{p(y|x;\theta_i)\}_{i=1}^M$ and density model $p(x;\theta_{\text{ens}})$ are trained stably (i.e., training losses of each prediction model were reduced stably and EBM

loss oscillated near zero). Empirically, appropriate gradient clipping was effective for the stability of the sampling procedure, and the number of sample steps η was quite significant for the quality of sampling which is crucial for successful EBM training. The longer η is better, but it increases training costs.

Specifically, for synthetic experiments, we used the number of sample steps 10, step size 5, a standard deviation of noise 0.005, size of replay buffer 1000, replay ratio 0.9, and gradient clipping magnitude 0.01. Initial random noise distributions $\mathcal{R}$ were $\mathcal{N}(0, 0.25I)$ and $\mathcal{N}(0, 25I)$ for 1D regression and 2D classification respectively.

For FMNIST experiments, we used the number of samples steps 50, step size 1, a standard deviation of noise 0.01, size of replay buffer 10000, replay ratio 0.95, and initial random noise distribution $\mathcal{U}(-1, 1)$.

For UCI regression experiments, we used the number of sample steps 50, step size 1, a standard deviation of noise 0.01, size of replay buffer 10000, replay ratio 0.95, gradient clipping magnitude 0.1, and initial random noise distribution $\mathcal{U}(-5, 5)$.

5.3 GPU Usage

All experiments were performed on RTX 3090 GPU with 24GB RAM while each experiment use $2 \sim 6$GB memory. Because of the sampling procedure in EBDE training, the training time for EBDE increased in proportion to the number of sample steps η, e.g., for each sample step, one time of both energy function evaluation and backpropagation is required.

6

Conclusion

In this thesis, we have proposed Energy-Based Deep Ensemble (EBDE) methods, in which the prediction model $p(y|x)$ and input data distribution $p(x)$ are learned simultaneously, with the ensemble disagreement used for the energy-based modelling of $p(x)$. For stable EBDE training, we have defined an Ensemble Pairwise Divergence Score (EPDS) as a new ensemble disagreement measure, and after the training, the learned EBDE assigns high EPDS for outliers, i.e., the learned density model produces low likelihood for outliers. We use two different types of OOD score functions for EBDE: (i) the negative log-likelihood and (ii) the gradient norm of the log-likelihood. Through extensive synthetic and real-world regression and classification experiments and comparisons with other state-of-the-art deep ensemble methods, we haven show performance advantages of the proposed EBDE especially from the perspective of OOD detection. We have also verified that the EBDE does not degrade the prediction accuracy and uncertainty quantification performance compared to the vanilla DE.

A

Appendix

A.1 (Approximate) computation of log normalization constant

In maximum-likelihood training of the energy-based models, computing the gradient of the log normalization constant can be approximately calculated as follows:

$$\nabla_\theta \log \Omega_\theta = \nabla_\theta \log \int \exp(-E_\theta(x))dx \tag{A.1.1}$$

$$= (\int \exp(-E_\theta(x))dx)^{-1} \nabla_\theta \int \exp(-E_\theta(x))dx \tag{A.1.2}$$

$$= (\int \exp(-E_\theta(x))dx)^{-1} \int \exp(-E_\theta(x))(-\nabla_\theta E_\theta(x))dx \tag{A.1.3}$$

$$= \int \frac{\exp(-E_\theta(x))}{\Omega_\theta}(-\nabla_\theta E_\theta(x))dx \tag{A.1.4}$$

$$= \int p_\theta(x)(-\nabla_\theta E_\theta(x))dx \tag{A.1.5}$$

$$= \mathbb{E}_{x\sim p_\theta(x)}[-\nabla_\theta E_\theta(x)] \tag{A.1.6}$$

$$\approx \frac{1}{|\mathcal{B}_{neg}|} \sum_{x_{neg}\in\mathcal{B}_{neg}} -\nabla_\theta E_\theta(x_{neg}) \tag{A.1.7}$$

The Monte-Carlo approximation is used in equation A.1.7 and negative inputs $x_{neg} \in \mathcal{B}_{neg}$ are sampled from the probability density model $p_\theta(x)$.

A.2 Ensemble Disagreements

As shown in Figure 3.1 in the main script, ensemble variance score (EVS) which is widely employed as an OOD detection criterion in ensemble methods, suffers a gradient vanishing problem when generating negative samples for training EBM. We confirm that this phenomenon occurs when using a softmax layer for a classification task. When predictions of models in an ensemble are clearly disagreed, as the gradient at the softmax layer becomes 0, the gradient of energy w.r.t input $\nabla_x E_{ed}(x)$ also becomes 0. In a regression task, this problem may not exist because the each prediction model $p(y|x, \theta)$ does not employ a softmax layer, but it is not straightforward for computing divergence between each prediction $p(y|x; \theta_i)$ and mean model prediction $p_E(y|x; \theta_{\text{ens}}) = 1/M \sum_{i=1}^{M} p(y|x; \theta_i)$.

On the other hand, EPDS in equation 3.1 does not suffer the gradient vanishing problem as illustrated in Figure 3.1 and is computed easily in the regression task (the summation of divergences between two unimodal Gaussian).

A.3 Relation to Joint Probability Training

The training objective of the proposed EBDE consists of vanilla DE objective and density model $p(x; \theta_{\text{ens}})$ learning objective. Meanwhile, If we train the ensemble by minimizing a negative log-likelihood of the mean prediction model $p(y|x; \theta_{\text{ens}}) = 1/M \sum_{i=1}^{M} p(y|x; \theta_i)$, we can try another objective for training the EBDE:

$$\alpha \cdot \mathbb{E}_{(x,y)\sim p(x,y)}[\log p(y|x;\theta_{\text{ens}})] + \mathbb{E}_{x\sim p(x)}[\log p(x;\theta_{\text{ens}})], \tag{A.3.8}$$

If α is 1, then the inside of objective in equation A.3.8 becomes to log joint probability $\log p(x, y; \theta_{\text{ens}})$. While it is still ambiguous which objective is better in theoretically and empirically, we expect that this joint probability model interpretation of the EBDE might improve or extend the EBDE e.g., an EBDE trained with equation A.3.8 can provide unnormalized conditional probability density $\tilde{p}(x|y;\theta_{\text{ens}}) \propto p(y|x;\theta_{\text{ens}})\tilde{p}(x;\theta_{\text{ens}})$.

A.4 Limitations

Maximum likelihood training of energy-based models is required negative samples as discussed in section 2.2 in the main script. Therefore, EBDE also requires a negative sampling procedure that η times both energy function evaluation (forward-propagation) and back-propagation is required for each training iteration. Furthermore, MCMC-based negative sampling causes instability during training EBMs [27, 36, 37]. These challenges in the MCMC-based EBM training make it hard to scale up EBDE to larger-scale benchmarks such as CIFAR10 [38], ImageNet [39], etc. Nevertheless, we expect that the training speed and stability of EBDE can be improved by applying recent works [37, 40, 21, 41] which improve both the training speed and stability of EBM training.

A.5 Additional Experimental Results

In this section, we report OOD detection performance using the other OOD score functions which are introduced in section 3.2 but not reported in Table 4.3 and

Table 4.5.

A.5.1 FMNIST classification

Table A.1, A.2 and Table A.3 extends the Table 4.3 in the main script. We report the OOD detection performance of each method using various OOD detection scores, i.e., EVS, EPDS, and AMS. As reported in previous works [27, 23, 22], OOD detection performances of EBDE are degraded when we directly use EPDS as the OOD score function. While EBDE still outperforms most of the baseline methods when using EVS and AMS as the OOD score function. In the tables, we run each experiment 5 times at different seeds and report a mean and standard error. The best one is bolded.

Table A.1: The FashionMNIST vs. MNIST OOD detection benchmark results.

Method	AUROC against OOD		
	EVS	EPDS	AMS
Vanilla DE	87.85(±0.58)	88.89(±0.55)	89.02(±0.58)
w-WGD	91.19(±0.40)	92.25(±0.44)	92.05(±0.49)
f-WGD-l	92.43(±0.18)	93.18(±0.60)	92.29(±0.78)
f-WGD-l-n	92.45(±0.38)	93.79(±0.35)	92.64(±0.47)
f-WGD-sm	77.18(±0.63)	74.02(±1.41)	84.81(±0.91)
f-WGD-sm-n	93.96(±0.93)	**94.82**(±0.76)	93.73(±0.62)
EBDE (ours)	**96.36**(±0.65)	86.23(±3.48)	**97.93**(±0.23)

Table A.2: The FashionMNIST vs. KMNIST OOD detection benchmark results.

Method	AUROC against OOD		
	EVS	EPDS	AMS
Vanilla DE	85.57(±0.39)	86.61(±0.41)	86.54(±0.42)
w-WGD	88.39(±0.08)	89.58(±0.17)	89.84(±0.24)
f-WGD-l	89.20(±0.35)	90.05(±0.64)	89.19(±0.69)
f-WGD-l-n	88.17(±0.65)	89.09(±0.86)	88.13(±1.15)
f-WGD-sm	71.25(±0.45)	67.38(±0.86)	79.15(±0.39)
f-WGD-sm-n	**96.66**(±0.31)	**97.31**(±0.24)	**96.71**(±0.20)
EBDE (ours)	93.06(±0.70)	81.77(±2.10)	95.17(±0.44)

A.5.2 UCI regression

Table A.4 and Table A.5 extend Table 4.5 in the main script. Table A.4 and Table A.5 show the OOD detection performance of each method where OOD scores are EPDS and AMS. Similar to when EVS was used as the OOD score (Table 4.5), f-WGD and EBDE performed better than vanilla DE and w-WGD. In the tables, we run each experiment 10 times and report average and standard error. In the case of f-WGD, the summary statistics are computed using only the results of successfully learned models out of 10 times. The best and the second best are bolded, and the worst is colored red.

Table A.3: The FashionMNIST vs. Omniglot OOD detection benchmark results.

Method	AUROC against OOD		
	EVS	EPDS	AMS
Vanilla DE	90.80(±0.53)	91.59(±0.51)	92.51(±0.31)
w-WGD	90.73(±0.92)	91.52(±0.93)	92.18(±0.82)
f-WGD-l	91.35(±0.66)	91.87(±0.88)	91.99(±0.86)
f-WGD-l-n	90.95(±0.70)	91.48(±0.92)	91.51(±1.03)
f-WGD-sm	78.03(±1.92)	72.79(±1.92)	86.95(±0.86)
f-WGD-sm-n	92.16(±0.79)	**92.71**(±0.73)	91.61(±0.53)
EBDE (ours)	**92.76**(±0.36)	72.13(±5.48)	**95.02**(±0.51)

Table A.4: The synthetic OOD detection results on UCI regression dataset using (AUROC using **EPDS**).

Datasets	Vanilla DE	w-WGD	f-WGD	EBDE
Concrete	97.54(±0.33)	97.76(±0.38)	**99.91**(±0.05)	**99.39**(±0.12)
Energy	99.79(±0.06)	99.72(±0.09)	**100.0**(±0.00)	**99.87**(±0.09)
Housing	97.38(±0.37)	97.01(±0.50)	**99.08**(±0.16)	**98.88**(±0.13)
Power	99.98(±0.00)	**99.99**(±0.00)	**99.99**(±0.00)	**99.99**(±0.00)
Wine	91.88(±1.01)	93.04(±0.78)	**99.07**(±0.22)	**99.49**(±0.12)
Yacht	**99.87**(±0.06)	99.84(±0.05)	**99.94**(±0.03)	99.84(±0.08)

Table A.5: The synthetic OOD detection results on UCI regression dataset using (AUROC using **AMS**).

Datasets	Vanilla DE	w-WGD	f-WGD	EBDE
Concrete	87.87(±1.40)	90.21(±1.18)	**99.76**(±0.09)	**95.68**(±0.63)
Energy	98.16(±0.46)	97.53(±0.57)	**100.0**(±0.00)	**99.16**(±0.26)
Housing	96.73(±0.43)	96.00(±0.75)	**98.98**(±0.24)	**98.35**(±0.25)
Power	99.89(±0.03)	**99.97**(±0.01)	**99.97**(±0.00)	**99.98**(±0.01)
Wine	88.68(±1.18)	89.39(±0.88)	**98.63**(±0.36)	**99.55**(±0.11)
Yacht	**99.75**(±0.10)	99.63(±0.11)	**99.94**(±0.03)	99.73(±0.12)

Bibliography

[1] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[2] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

[5] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[6] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[7] A Philip Dawid. The well-calibrated bayesian. *Journal of the American Statistical Association*, 77(379):605–610, 1982.

[8] Morris H DeGroot and Stephen E Fienberg. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 32(1-2):12–22, 1983.

[9] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.

[10] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[11] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[12] RM Neal. Bayesian learning for neural networks [phd thesis]. *Toronto, Ontario, Canada: Department of Computer Science, University of Toronto*, 1995.

[13] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[14] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[15] Ziyu Wang, Tongzheng Ren, Jun Zhu, and Bo Zhang. Function space particle optimization for bayesian neural networks. *arXiv preprint arXiv:1902.09754*, 2019.

[16] Francesco D'Angelo and Vincent Fortuin. Repulsive deep ensembles are bayesian. *Advances in Neural Information Processing Systems*, 34, 2021.

[17] Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401, 2020.

[18] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in neural information processing systems*, 29, 2016.

[19] Siddhartha Jain, Ge Liu, Jonas Mueller, and David Gifford. Maximizing overall diversity for improved uncertainty estimates in deep ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4264–4271, 2020.

[20] Laurent Younes. On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics: An International Journal of Probability and Stochastic Processes*, 65(3-4):177–228, 1999.

[21] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

[22] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136*, 2018.

[23] Hyunsun Choi, Eric Jang, and Alexander A Alemi. Waic, but why? generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

[24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[25] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[26] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

[27] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*, 2019.

[28] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[29] Dan C Cireşan, Ueli Meier, Jonathan Masci, Luca M Gambardella, and Jürgen Schmidhuber. High-performance neural networks for visual object classification. *arXiv preprint arXiv:1102.0183*, 2011.

[30] Vinay Uday Prabhu. Kannada-mnist: A new handwritten digits dataset for the kannada language. *arXiv preprint arXiv:1908.01242*, 2019.

[31] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[32] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*, pages 1861–1869. PMLR, 2015.

[33] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[34] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[35] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.

[36] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5272–5280, 2020.

[37] David Duvenaud, Jacob Kelly, Kevin Swersky, Milad Hashemi, Mohammad Norouzi, and Will Grathwohl. No mcmc for me: Amortized samplers for fast and stable training of energy-based models. 2021.

[38] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[39] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[40] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32, 2019.

[41] Bo Dai, Zhen Liu, Hanjun Dai, Niao He, Arthur Gretton, Le Song, and Dale Schuurmans. Exponential family estimation via adversarial dynamics embedding. *Advances in Neural Information Processing Systems*, 32, 2019.

국문초록

분류 및 회귀와 같은 예측 작업에 사용되는 모델은 이상적으로 두 가지 속성을 가져야 한다. 우선, 입력 x가 분포 내(즉, 데이터 분포 $p(x)$에서 도출된)인지 분포 외(이상치 또는 OOD 데이터)인지 결정할 수 있어야 한다. 이상치가 필터링된 후에는, 모델은 예측의 불확실성의 정량적인 측도를 제공해야 한다. 최근에, 확률적 예측을 하는 딥 뉴럴넷 모델을 여러개 사용하는 딥 앙상블(DE) 방법과 DE의 몇가지 변형된 방법들은 이러한 두 가지 기능을 갖추어 널리 사용되어져 오고 있다. 그러나 OOD 감지 작업의 경우 기존 앙상블 방법의 성능은 아쉬운 점이 있는데, 이는 모델이 $p(x)$를 학습하지 않기 때문이라고 주장한다. 본 학위논문에서는 예측 모델 $p(y|x)$와 입력 데이터 분포 $p(x)$를 동시에 학습하는 *에너지 기반 딥 앙상블(EBDE)* 방법을 제안한다. 구체적으로, $p(x)$는 앙상블 불일치(즉, 각 모델에 의한 예측들의 분산)를 에너지 함수로 사용하여 에너지 기반 모델로 모델링 되어진다. 여러 종류의 데이터셋을 이용한 실험에서 제안한 EBDE가 예측 및 불확실성 정량화에서 기존의 방법들과 유사한 성능을 달성하는 동시에, OOD 탐지 작업에 대한 기존 방법보다 더 나은 성능을 보임을 확인하였다.

주요어: 딥 앙상블, Out-of-distribution 감지, 불확실도 정량화, 에너지 기반 모델, 앙상블 다양화

학번: 2020-22845