Deep Gaussian Mixture Ensembles

Abstract

This work introduces a novel probabilistic deep learning technique called deep Gaussian mixture ensembles (DGMEs), which enables accurate quantification of both epistemic and aleatoric uncertainty. By assuming the data generating process follows that of a Gaussian mixture, DGMEs are capable of approximating complex probability distributions, such as heavy tailed or multimodal distributions. Our contributions include the derivation of an expectation-maximization (EM) algorithm used for learning the model parameters, which results in an upper-bound on the log-likelihood of training data over that of standard deep ensembles. Additionally, the proposed EM training procedure allows for learning of mixture weights, which is not commonly done in ensembles. Our experimental results demonstrate that DGMEs outperform state-of-the-art uncertainty quantifying deep learning models in handling complex predictive densities.

1 INTRODUCTION

Uncertainty quantification plays a key role in the development and deployment of machine learning systems, especially in applications where user safety and risk assessment are of paramount importance [Abdar et al., 2021]. While deep learning (DL) has cemented its superiority in terms of raw predictive performance for a variety of applications, the principled incorporation of uncertainty quantification in DL models remains an open challenge. Since standard DL models are unable to properly quantify predictive uncertainty, one common challenge for deep learning models is detecting out-of-distribution (OOD) inputs. It is often the case that OOD inputs lead a DL model to making erroneous predictions [Ovadia et al., 2019]. Without uncertainty quantification, one cannot reason about whether an input is OOD and this can be catastrophic in applications such as machine-assisted medical decision making [Begoli et al., 2019] or self-driving vehicles [Michelmore et al., 2018]. Moreover, uncertainty quantification can also be used as a means to select samples to label in active learning and for enabling exploration in reinforcement learning algorithms [Clements et al., 2019, Charpentier et al., 2022].

Uncertainty in machine learning models is derived from two different sources: aleatoric uncertainty and epistemic uncertainty [Kendall and Gal, 2017, Hüllermeier and Waegeman, 2019, Valdenegro-Toro and Mori, 2022]. Aleatoric uncertainty derives from measurement process of the data, while epistemic uncertainty derives from the uncertainty in the parameters of the machine learning model. A variety of approaches have been proposed to quantify both types of uncertainty in DL models from both a Bayesian and frequentist perspective; we refer the reader to Gawlikowski et al. [2021] for a comprehensive review. Under the Bayesian paradigm, the goal is to infer the posterior predictive density of the target variable given the input features and the training data, which can encode both types of uncertainty. Unfortunately, exact Bayesian inference algorithms (e.g., Neal 2012) cannot scale to the parameter space of modern DL architectures and one often has to resort to mini-batching [Chen et al., 2014] or forming a rough parametric approximation of the posterior distribution of the parameters, such as the Laplace approximation [Daxberger et al., 2021] or stochastic variational inference [Graves, 2011, Hoffman et al., 2013]. The drawback of the parametric approach is the inability to express more complex (e.g., heavy-tailed or multimodal) predictive distributions. As an example, approximations such as mean-field variational inference form a Gaussian predictive distribution that tends to underestimate the true uncertainty of more complex models.

In recent years, there have been developments in probabilistic DL which exploit the inherent stochasticity in learning to quantify predictive uncertainty. Examples include techniques such as probabilistic backpropagation [Hernández-Lobato and Adams, 2015], Monte Carlo dropout (MCD,

Submitted to the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023). **To be used for reviewing only**.

Gal and Ghahramani 2016), Monte Carlo batch normalization [Teye et al., 2018], deep ensembles (DEs, Lakshminarayanan et al. 2017) among others. MCD and DEs have emerged as state-of-the-art solutions for quantifying uncertainty in DL models due to their simplicity and effectiveness. MCD utilizes the inherent stochasticity of dropout (i.e., random masking of neural network weights) to form an ensemble-based approximation of the predictive distribution through multiple stochastic forward passes of the model to account for epistemic uncertainty. Aleatoric uncertainty is handled as a post-processing step under the assumption that the underlying data noise is homoscedastic. DEs, on the other hand, independently train an small ensemble of dual-output neural networks, where the outputs characterize the mean and variance of the predictive distribution. Each network in the ensemble is independently trained to maximize the likelihood of the data under the heteroscedastic Gaussian assumption. At test time, the networks are linearly combined into a single Gaussian approximation of the predictive distribution. Unfortunately, both MCD and DEs are not adequate solutions for modeling more complex data distributions (e.g., distributions that have heavy tails or multiple modes).

Contributions. Our contribution are as follows:

1. We propose a novel probabilistic DL technique called *deep Gaussian mixture ensembles* (DGMEs) for jointly quantifying epistemic and aleatoric uncertainty. DGMEs trains a weighted deep ensemble using the expectation maximization algorithm;
2. We show DGMEs optimizes the joint data likelihood directly, unlike deep ensembling that targets a lower bound of the data likelihood. As a consequence, DGMEs achieving superior loss to deep ensembling, which is corroborated by our experimental results;
3. We empirically show that our model is more expressive than standard probabilistic DL approaches and can capture both heavy-tailedness and multimodality.

2 RELATED WORK

Mixture Density Networks. Mixture density networks (MDNs, Bishop 1994) use a deep neural network to simultaneously learn the means, variances and mixture weights of a Gaussian mixture model. MDNs have been successfully used in many machine learning applications, such as computer vision [Bazzani et al., 2016], speech synthesis [Zen and Senior, 2014], probabilistic forecasting [Zhang et al., 2020], astronomy [D'Isanto and Polsterer, 2018], chemistry [Unni et al., 2020] and epidemiology [Davis et al., 2020], among others. While MDNs are closely related to DGMEs in terms of uncertainty quantification, they are not an ensemble technique per se, as the epistemic and aleatoric uncertainty cannot be disentangled in MDNs. Moreover, without the ensemble structure, MDNs cannot easily be trained in a distributed settings, whereas DGMEs are naturally trained in a distributed manner.

Monte Carlo Dropout. MCD exploits the stochasticity of dropout training to quantify epistemic uncertainty in DL models. At test time, stochastic forward passes through a DL model with dropout produce "approximate" samples from the underlying posterior predictive distribution, which are typically summarized using first- and second-order moments (e.g, mean and variance of the samples). Aleatoric uncertainty is accounted for in a post-processing step, whereby the optimal homescedastic variance that maximizes the evidence lower-bound is obtained via cross-validation. MCD's popularity can be attributed to its simple implementation, as no changes to the standard DL training procedure are required. While vanilla MCD can yield favorable results in additive Gaussian settings, the method is less effective when dealing with more complex data generating processes (e.g., heavy-tailed or multimodal predictive densities). In this work, we incorporate MCD in our training procedure to account for epistemic uncertainty (see Section 4.4).

Deep Ensembles. DEs quantify both aleatoric and epistemic uncertainty by building an ensemble of independently trained models under different neural network weight initializations. Combined with adversarial training [Goodfellow et al., 2014], DEs achieve competitive or better performance than MCD in most settings in terms of calibration of predictive uncertainty and in terms of reasoning about OOD inputs. It has been argued in several works that DEs can be interpreted as a Bayesian approach, where the learned weights of each ensemble member correspond to a sample from the posterior distribution of the network weights [Wilson and Izmailov, 2020]. In recent years, new variations of deep ensembles have been proposed, such as anchored DEs [Pearce et al., 2020], deep-split ensembles [Sarawgi et al., 2020], and hybrid training approaches that combine DEs with the Laplace approximation [Hoffmann and Elster, 2021]. We emphasize that a key distinction between DEs and DGMEs is that each sample in DEs is treated as an i.i.d. sample from a Gaussian distribution, whereas DGMEs assume that data are distributed according to a Gaussian mixture. This gives DGMEs the key advantage of being able to learn more complex data generating processes.

3 PROBLEM FORMULATION

Consider a set of training data $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$, where $x_n \in \mathbb{R}^{d_x}$ is the feature vector and y_n is the output, which can be real-valued if we are dealing with a regression task or integer-valued if we are dealing with a classification task. We would like to train a model that allows us to predict an output y given its corresponding input vector x.

From a probabilistic perspective, the goal is to determine

the posterior predictive distribution $p(y|x, \mathcal{D})$. We assume a statistical model $p_\theta(y|x) \triangleq p(y|x, \theta)$ that relates each output to its corresponding feature vector through a set of parameters $\theta \in \Theta$. Then, the predictive distribution can be determined as

$$p(y|x, \mathcal{D}) = \int_\Theta p_\theta(y|x) p(\theta|\mathcal{D}) d\theta. \tag{1}$$

While this integral is generally intractable, it can be approximated using a Monte Carlo average, where samples are taken from the posterior $p(\theta|\mathcal{D})$. Let $Y = \{y_1, \ldots, y_N\}$ and let $X = \{x_1, \ldots, x_N\}$. According to Bayes' theorem, the posterior distribution $p(\theta|\mathcal{D})$ is

$$p(\theta|\mathcal{D}) = \frac{p(Y|X, \theta) p(\theta)}{p(Y|X)}, \tag{2}$$

where $p(Y|X, \theta) = \prod_{n=1}^N p_\theta(y_n|x_n)$ is called the *data likelihood* under the i.i.d. assumption, $p(\theta)$ is the *prior distribution* of θ, and $p(Y|X) = \int_\Theta p(Y|X, \theta) p(\theta) d\theta$ is called the *marginal likelihood*. The posterior can only be computed analytically when $p(\theta)$ is a conjugate prior for the likelihood function $p(Y|X, \theta)$. For deep learning models, an analytical solution to the posterior cannot be determined and one must resort to an approximation.

Goal. The goal is to acquire an approximation of the posterior predictive distribution. Ideally, samples from the approximation form consistent estimators of key moments of the predictive distribution that allow one to (i) formulate predictions, (ii) identify the underlying stochastic risk associated with the prediction (e.g., aleatoric uncertainty), and (iii) reason about the model's uncertainty in the presence of the OOD data (i.e., epistemic uncertainty).

4 DEEP GAUSSIAN MIXTURE ENSEMBLES

In this work, we propose DGMEs to effectively learn a mixture distribution that accurately represents the true conditional density of the labels given the features. Since Gaussian mixtures are universal approximators for smooth probability density functions [Bacharoglou, 2010, Goodfellow et al., 2016], modeling the conditional density $p_\theta(y|x)$ as a Gaussian mixture allows for learning more complex distributions, such as skewed, heavy-tailed, and multimodal distributions. Under the assumption that our data follows a mixture distribution with K mixture components, the conditional density of a particular example (x, y) is given by:

$$p_\theta(y|x) = \sum_{k=1}^K \pi_k p_k(y|x, \theta_k), \tag{3}$$

where $\theta_k \in \Theta_k \subset \mathbb{R}^{d_\theta}$ denotes the underlying parameters of the k-th mixtures and π_k denotes weight of the k-th mixture and represents the probability that the example (x, y) is distributed according to $p_k(y|x, \theta_k)$. Throughout the rest of the text, we refer to all unknown parameters in the mixture as $\theta = \{\pi_1, \theta_1, ..., \pi_K, \theta_K\}$. Hereafter, we consider the problem of learning the parameters of the mixture in (3) in the context of a regression task.[1]

To effectively model this mixture, we make the following assumptions:

Assumption 4.1. The mixture weights $(\pi_1, \ldots, \pi_K) \in \mathcal{S}_K$ do not depend on the input features, where $\mathcal{S}_K$ denotes the K-dimensional probability simplex.

Assumption 4.2. The conditional density $p_k(y|x, \theta_k)$ is a Gaussian distribution whose parameters are modeled via parameterized functions dependent on x:

$$p_k(y|x, \theta_k) = \mathcal{N}(y; \mu_{\theta_k}(x), \sigma^2_{\theta_k}(x)), \tag{4}$$

where θ_k denote the parameters of functions $\mu_{\theta_k}(\cdot)$ and $\sigma^2_{\theta_k}(\cdot)$ that output the mean and variance of the kth mixture, respectively. Importantly, these function are assumed to share parameters, just as in the original work on DEs [Lakshminarayanan et al., 2017].

Under the above assumptions, learning the mixture representation of $p_\theta(y|x)$ is equivalent to learning the parameters θ to maximize the data likelihood of the training examples $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$.

4.1 LEARNING THE MIXTURE PARAMETERS

We obtain the maximum likelihood (ML) estimate or maximum a posteriori (MAP) estimate of the unknown parameters θ using the EM algorithm. Let $Y = \{y_1, \ldots, y_N\}$ and $X = \{x_1, \ldots, x_N\}$. Furthermore, let $Z = \{z_1, \ldots, z_N\}$, where each $z_n \in \{1, \ldots, K\}$ is a latent variable that denotes membership assignment of the training example (x_n, y_n) to a particular mixture component, where $\pi_k \triangleq P_\theta(z_n = k)$ is the probability that the example (x_n, y_n) belongs to the k-th component. Assuming that the training examples are independent and identically distributed, we can write the joint likelihood as

$$p_\theta(Y, Z|X) = \prod_{n=1}^N \prod_{k=1}^K \pi_k^{\mathbb{I}(z_n=k)} \mathcal{N}(y_n; \mu_{\theta_k}(x_n), \sigma^2_{\theta_k}(x_n))^{\mathbb{I}(z_n=k)},$$

with corresponding log-likelihood of

$$\log p_\theta(Y, Z|X) = \sum_{n=1}^N \sum_{k=1}^K \mathbb{I}(z_n = k)(\log \pi_k + \ell_{\theta_k}(x_n, y_n))$$

[1] For classification tasks, we instead can consider a mixture of categorical distributions, whereby $p_k(y|x, \theta_k) = \prod_{i=1}^{d_y} p^i_{\theta_k}(x)^{\mathbb{I}(y=i)}$, where $p^i_{\theta_k}(x)$ denotes the probability that y belongs to the i-th class according to the k-th mixture.

where $\ell_{\theta_k}(x, y) = \log\left(\mathcal{N}(y; \mu_{\theta_k}(x), \sigma^2_{\theta_k}(x))\right)$. Our goal is to solve the following optimization problem:

$$\theta^\star = \arg\max_\theta \log p_\theta(Y|X) \tag{5}$$

$$= \arg\max_\theta \log\left(\mathbb{E}_{Z|X,Y,\theta}\left[p_\theta(Y, Z|X)\right]\right), \tag{6}$$

which we numerically solve using the EM algorithm. In the following, we describe both the expectation step (E-Step) and maximization step (M-Step) as it relates to our model. As a note, all results presented hereafter also apply to the more general problem of obtaining the MAP estimate of the parameters θ.[2]

E-Step: We update the posterior probabilities of each z_n given the parameters θ and the example (x_n, y_n) for each n, denoted by $\gamma_{k,n} \triangleq P_\theta(z_n = k|x_n, y_n)$. This can be done directly using Bayes' theorem:

$$\gamma_{k,n} = \frac{p_k(y_n|x_n, \theta_k)P_\theta(z_n = k)}{\sum_{j=1}^K p_j(y_n|x_n, \theta_j)P_\theta(z_n = j)} \tag{7}$$

$$= \frac{\pi_k \mathcal{N}(y_n; \mu_{\theta_k}(x_n), \sigma^2_{\theta_k}(x_n))}{\sum_{j=1}^K \pi_j \mathcal{N}(y_n; \mu_{\theta_j}(x_n), \sigma^2_{\theta_j}(x_n))} \tag{8}$$

M-Step: The parameters θ are updated in the maximization step by maximizing the expected joint log-likelihood $Q(\theta, \theta') \triangleq \mathbb{E}_{Z|X,Y,\theta'}\left[\log p_\theta(Y, Z|X)\right]$ given the previous parameter values θ', which is equivalent to doing lower-bound maximization on the true log-likelihood [Minka, 1998]. The function $Q(\theta, \theta')$ can be readily determined as:

$$Q(\theta, \theta') = \sum_{n=1}^N \sum_{k=1}^K \gamma_{k,n}(\log(\pi_k) + \ell_{\theta_k}(x_n, y_n)). \tag{9}$$

The optimization of the mixture weights $(\pi_1, \ldots, \pi_K)$ can be carried out analytically and done independently of optimizing the mixture parameters $\{\theta_1, \ldots, \theta_K\}$:

$$(\pi_1^\star, \ldots, \pi_K^\star) = \arg\max_{(\pi_1,\ldots,\pi_K)\in\mathcal{S}_K} Q(\theta, \theta'), \tag{10}$$

where for each k

$$\pi_k^\star = \frac{1}{N}\sum_{n=1}^N \gamma_{k,n}. \tag{11}$$

Since the mixture parameters are assumed to be parameterized by neural networks, their optimization must be carried out using stochastic optimization. It is easy to see that the optimization of each θ_k can be done independently as:

$$\theta_k^\star = \arg\max_{\theta_k\in\Theta_k} \sum_{n=1}^N \gamma_{k,n}\ell_{\theta_k}(x_n, y_n) \tag{12}$$

$$= \arg\min_{\theta_k\in\Theta_k} \sum_{n=1}^N \gamma_{k,n}\left(\log\sigma^2_{\theta_k}(x_n) + \frac{(y_n - \mu_{\theta_k}(x_n))^2}{\sigma^2_{\theta_k}(x_n)}\right) \tag{13}$$

This optimization step can be thought as training a deep ensemble, where each sample (x_n, y_n) is weighted by $\gamma_{k,n}$ in its negative log-likelihood contribution.

Algorithm 1 Deep Gaussian Mixture Ensembles (DGMEs)

1: **Inputs:**
- Training dataset $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$
- Number of mixture components K
- Number of EM steps J

2: **Initialize mixture parameters:**
- Sample $\theta_k^{(0)} \sim p(\theta)$ for all k.
- Set $\pi_k^{(0)} = \frac{1}{K}$ for all k.

3: **for** $j = 1, \ldots, J$ **do**

4: **E-Step:** Update posterior probabilities $\gamma_{k,n}^{(j)}$ according to (8) with mixture weights $\pi_k^{(j-1)}$ and mixture parameters $\theta_k^{(j-1)}$ for all k and n.

5: **M-Step:** Update mixture weights $\pi_k^{(j)}$ and parameters $\theta_k^{(j)}$ for all k as

$$\pi_k^{(j)} = \frac{1}{N}\sum_{n=1}^N \gamma_{k,n}^{(j)}$$

and

$$\theta_k^{(j)} = \arg\max_{\theta_k\in\Theta_k} \sum_{n=1}^N \gamma_{k,n}^{(j)}\ell_{\theta_k}(x_n, y_n)$$

6: **end for**

7: **Return:** $\pi_k^\star = \pi_k^{(J)}$ and $\theta_k^\star = \theta_k^{(J)}$ for all k.

4.2 IMPLEMENTATION DETAILS

Our implementation of DGMEs trained via the EM algorithm is summarized in Algorithm 1. To initialize the ensemble, the parameters of each network in the ensemble are randomly initialized, while the mixture weights are assumed to be equal. The algorithm is run for J steps or alternatively until some stopping criterion is met. The E-Step for updating the posterior probabilities is computed directly for each sample in the training set. In the M-Step, the updates for the mixture weights are also carried out analytically, but for mixture component parameters θ_k we use the stochastic optimization to numerically solve for the updates, as an analytical solution is not available. At round j, we initialize

[2] That is, the maximizer of $\log p(Y, \theta|X) = \log p_\theta(Y|X) + \log p(\theta)$, where $p(\theta)$ is the prior distribution of the mixture parameters

each network to $\theta_k^{(j-1)}$ and then run the Adam optimizer for E epochs to minimize the weighted negative log-likelihood in (13), where the weights are given by $\gamma_{k,n}^{(j)}$ for all n. Finally, we note that the computational complexity of each EM step is equivalent to that of DEs and the overall time complexity scales linearly with the number of EM steps.

4.3 THEORETICAL INSIGHTS

In this section we provide insights into the connection between DGMEs and DEs, along with general results on convergence of our training procedure using DGMEs. See Supplementary Materials, Section A in the Supplementary Material for details on the proofs of each propositions.

Proposition 4.3 shows that maximizing the data likelihood directly as in DGMEs achieves an equal or better likelihood than maximizing each ensemble member's likelihood separately as in DEs.

Proposition 4.3. *Under the assumption that $\pi_i = 1/K$ for $i = 1, .., K$, maximizing the Gaussian mixture data likelihood directly achieves better or equal joint likelihood than maximizing each ensemble member's likelihood separately.*

Proof Sketch. The result can be obtained by using Jensen's inequality on the joint log-likelihood of equation (5) along with the assumption.

Proposition 4.4 connects DGMEs and DEs, showing DEs is equivalent to a single-EM-step of DGMEs under specific neural network weights initialization.

Proposition 4.4. *If the weights of each ensemble members are initialized to 0 with fixed bias terms, a single EM step for DGMEs is equivalent to perform DEs.*

Proof Sketch. The initialization schema implies that mixture membership is equal across samples in the first expectation round of the EM. Hence, the first M-step consists in training k-separate networks with each log-likelihood contribution being weighted equally.

Finally, Proposition 4.5 combines recent results on neural network convergence in regression by Arora et al. [2019] and Farrell et al. [2021] with classical EM analysis [Wu, 1983] to give intuition on why DGMEs should converge towards the maximum of the data likelihood[3].

Proposition 4.5. *Under assumptions A.3 - A.5, let the mean and variance in each ensemble model being estimated via a separate 2-layer deep ReLu network from a common feature extraction layer. Then the DGMEs EM algorithm convergences to a non stationary point that maximizes the data likelihood with high-probability.*

Proof Sketch. The result follows if one shows that $Q(\theta; \theta^{(j)})$ is an increasing function of the EM steps j (Wu 1983), for parameter values $\theta^{(j)}$ that are not stationary points of $Q(\theta; \theta^{(j)})$. In the DGMEs case, this corresponds to proving that the weighted log-likelihood in each ensamble increases at every round j. The result follows by combining assumptions on the non-flatness of the weighted log-likelihood (A.3), the smoothness of true mean function (A.4) and the smoothness of the true variance function (A.5) with the results obtained about convergence of deep ReLU networks by Farrell et al. [2021] and Arora et al. [2019] respectively.

4.4 QUANTIFYING EPISTEMIC UNCERTAINTY

It is important to highlight that up until this point, we have not explicitly considered epistemic uncertainty in DGMEs. This is because the operation of training DGMEs according to Algorithm 1 yields a single set of parameters of the assumed Gaussian mixture model.[4] To account for model uncertainty, we need to account for the uncertainty in all parameters of the mixture (e.g., the mixture weights and the weights of the ensemble neural networks). One simple way to do this is to apply MCD to the training procedure of DGMEs — although we emphasize other techniques can be applied to account for epistemic uncertainty (e.g., Laplace approximation or a variational approximation to the posterior parameters).

Let $a_k = [a_{k,1}, \ldots, a_{k,d_\theta}]^\intercal \in \{0,1\}^{d_\theta}$ denote a random binary vector of the same size as each θ_k and let $p_d \in [0,1]$ denote a fixed dropout probability. Also, let $\theta^\star = \{\pi_1^\star, \theta_1^\star, \ldots, \pi_K^\star, \theta_K^\star\}$ denote the parameters learned by running Algorithm 1 with dropout incorporated in the training in the M-Step. We can obtain an approximate samples $\theta_k^{(m)}$ from the posterior distribution of each θ_k with the following procedure:

$$a_{k,i}^{(m)} \sim \text{Bernoulli}(p_d), \quad i = 1, \ldots, d_\theta, \tag{14}$$

$$\theta_k^{(m)} = a_k^{(m)} \odot \theta_k^\star. \tag{15}$$

Given M samples $\theta_k^{(1)}, \ldots, \theta_k^{(M)}$ for each network k, the

[3]We note Proposition 4.5 covers a specific setup, in which mean and variance function estimation is performed separately by using a shared pre-trained feature extraction layer; a more thorough investigation is outside of the scope of this paper.

[4]This point highlights the intrinsic difference in training DEs versus training DGMEs. DGMEs do not have a "Bayesian" interpretation, because the EM algorithm used to train them only outputs a single set of possible parameters for the DGMEs (i.e., the corresponding posterior distribution of the weights is a Dirac measure centered at the learned parameter values).

predictive distribution can then be approximated as

$$p(y|x, \mathcal{D}) \approx \frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} \pi_k^\star \mathcal{N}(\mu_{\theta_k^{(m)}}(x), \sigma^2_{\theta_k^{(m)}}(x)).$$

If desired, the above approximation can be further summarized as Gaussian, similar to how MCD and DEs predictive distributions can be summarized.

5 EXPERIMENTS

We evaluate the empirical performance of DGMEs via three different numerical experiments. We compare our method to the MDNs [Bishop, 1994], MCD [Gal and Ghahramani, 2016], and DEs [Lakshminarayanan et al., 2017]. MCD and DEs are widely considered to be state-of-the-art solutions for quantifying predictive uncertainty in deep learning models and have repeatedly been used as baselines for developing new techniques. Additional results and figures can be found in the Supplementary Material, Section B.

5.1 TOY REGRESSION

Consider the following model:

$$y_n = u_n x_n^3 + \epsilon_n, \tag{16}$$

where $u_n \in \{-1, 1\}$ with $p_u \triangleq P(u_n = -1)$ and $\epsilon_n \sim p(\epsilon)$ for all $n = 1, \ldots, N$. We generate $N = 800$ training samples from this model for the training set, where the input values x_n range from -4 to 4. For each considered setting, we use a learning rate of $\eta = 0.01$, a batch size of 32, and $E = 80$ epochs to resolve the stochastic optimization problem in the M-step. For each method, we utilize a dropout probability $p_d = 0.1$ to account for epistemic uncertainty. Additionally, we generate data from this toy model under three different noise settings to demonstrate the flexibility and expressive power of DGMEs as compared to other baselines. Unless otherwise stated, we assume $K = 5$ networks in each mixture model-based approaches (i.e., MDNs, DEs, and DGMEs). Experimental results are described below for each noise scenario. Additional experimental results are provided in the Supplementary Material, Section B.1 .

Case 1 - Gaussian Noise: We set $p_u = 0$ and assume that the noise is zero-mean and Gaussian distributed with variance of 9. This is analogous to the setup utilized in Hernández-Lobato and Adams [2015]. Figure 3 (Supplementary Materials, Section B.1) shows the performance of DGMEs as compared to the baselines, where we observe that it outperforms MDNs and obtains comparable results to MCD and DEs.

Case 2 - Heavy-tailed Noise: We set $p_u = 0$ and assume that the noise distributed according to a zero-mean Student-t distribution with $\nu = 3$ degrees of freedom with variance of 9. Figure 1 shows the histogram of samples from the predictive distribution of both a training and a test input with their corresponding sample (excess) kurtosis. We observe that on the training examples (i.e., purple histograms), only MDNs and DGMEs are able to learn the heavy-tailedness of the noise, as both MCD and DEs obtain a kurtosis close to 0. Unlike the baseline approaches, which are unable to learn the tail behavior in the test example, we observe that DGMEs is the best method at capturing the heavy-tailedness of the test examples, as it gives the largest corresponding kurtosis.

Case 3 - Bimodal Gaussian Noise: We set $p_u = 0.3$ and assume that the noise is zero-mean and Gaussian distributed with variance of 9. For this example, we only compare the mixture-based approaches assuming $K = 2$ components. Figure 2 shows the predictive density for the corresponding 99% credible interval for each mixture in each approach, where for DGMEs we also show the learned mixture weights of each component. We observe that only MDNs and DGMEs are able to capture the bimodality of the data, with DGMEs also accurately capturing the mixture weight proportions. DEs instead overestimates the heteroscedastic variance in each network. This is due to the fact that DEs train each ensemble member independently under the assumption of Gaussian likelihood. We also show that DGMEs can robustly estimate this bimodality, even if the assumed number of mixture components is larger than 2 (see Section B.1 in the Supplementary Material).

5.2 REGRESSION ON REAL DATASETS

We evaluate the performance of DGMEs in regression against MDNs, MCD and DEs on a set of UCI regression benchmark datasets [Dua and Graff, 2017]; see Supplementary Materials, Section B, for further details on the datasets. We use the experimental setup used in Hernández-Lobato and Adams [2015], with each dataset split into 20 train-test folds. We use the same network architecture across each dataset: a single layer MLP with ReLU activations, containing 50 hidden units. For each dataset we train for $E = 40$ total epochs with a batch size of 32 and a learning rate of $\eta = 0.001$. To be consistent with previous evaluations, we used $K = 5$ networks in our ensemble and provide results for DGMEs for different numbers of EM steps $J \in \{1, 2, 5, 10\}$. Our results are shown in Tables 1 and 2, where we evaluate the root-mean-squared error (RMSE) and the negative log-likelihood (NLL) on the test set averaged over the different folds, respectively. In the same table, we also report the results for MDNs, MCD and DEs. Experimental results for MCD and DEs can also be found in their respective papers [Gal and Ghahramani, 2016, Lakshminarayanan et al., 2017]. Note that in this experiment, we do not apply dropout to MDNs and DGMEs and

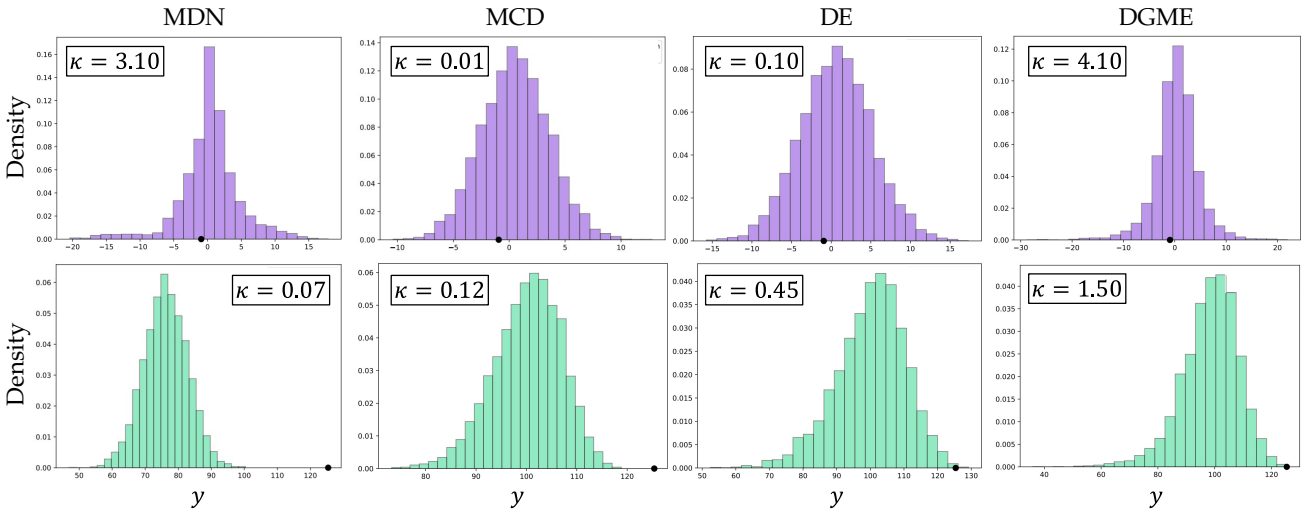

Figure 1: Histogram of samples from the predictive distributions for a single training example (top panel) and for a single test example (bottom panel) from the heavy-tailed toy regression example, shown with corresponding sample kurtosis value κ. DGMEs generally estimate heavier tailed predictions for both training and test samples, while baseline approaches samples are closer to following a Gaussian distribution.

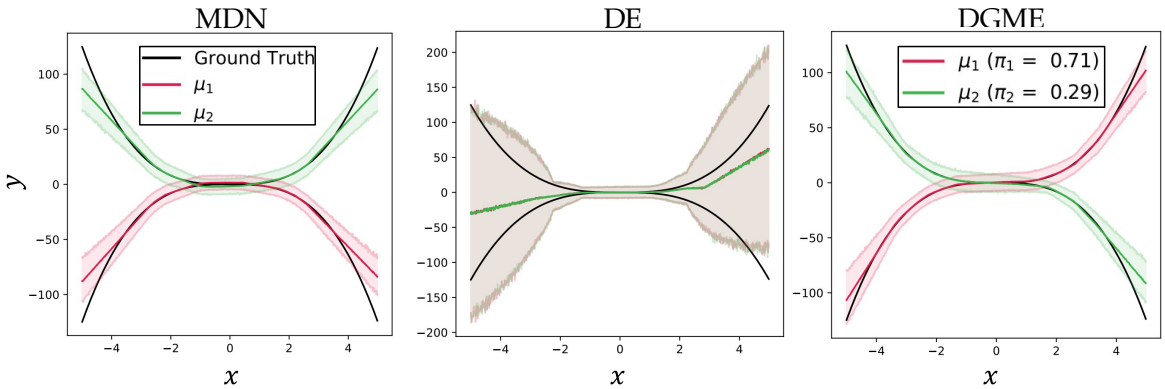

Figure 2: Predictive distribution plots for the bimodal Gaussian toy regression example. DEs cannot capture the multimodality of the noise, while MDNs and DGMEs can. Furthermore, DGMEs approximate the mixture weights of the noise accurately (ground truth: $\pi_1 = 0.7$ and $\pi_2 = 0.3$).

only account for the uncertainty obtained from training the models to maximize the NLL of the samples according to the Gaussian mixture assumption in (3).

We observe that in this experiment DGMEs are able to obtain competitive (or better) performance with respect to the baseline methods. For certain datasets, we observe that increasing the number of EM steps greatly improves the performance (e.g., Concrete, Energy, Power Plant, and Yacht). We theorize that this can be due to the distribution of the test set in each of those datasets being similar to that of the training set, implying that increasing the number of EM steps obtains a better parameter values for maximizing the log-likelihood of the test set. We can see that this is not generally true for all datasets: for example, for the Boston housing dataset, increasing the number of EM steps begins to degrade the performance of the model in terms of NLL.

5.3 UNCERTAINTY EVALUATION

In the final experiment, we compare DGMEs ability to reason about the underlying uncertainty of new samples with the baseline approaches. Specifically, for each method, we train a MLP network with 3 hidden layers and 200 hidden units per layer with ReLU activations on the MNIST dataset, including only digits 0-3 and 5-9. After the models are trained, we evaluate the average predictive entropy over three different datasets: the training dataset (known classes), a dataset containing only the digit 4 (unknown classes), and the fashion MNIST dataset (unrelated data). We use $M = 100$ samples from the predictive distribution to form an estimate of the predictive entropy for each method.We describe in more detail how the average predictive entropy is computed for each method in the Supplementary Material, Section B.3. The results for this experiment are shown in

Table 1: Average RMSE of test examples for regression experiments on real datasets. DGMEs obtain competitive or better performance in terms of RMSE on the majortity of datasets as compared to the baselines.

TEST RMSE							
Dataset	MDNs	MCD	DEs	DGMEs (J=1)	DGMEs (J=2)	DGMEs (J=5)	DGMEs (J=10)
Boston housing	**2.79 ± 0.84**	2.97 ± 0.85	3.28 ± 1.00	3.11 ± 0.94	3.00 ± 0.90	2.87 ± 0.86	2.83 ± 0.91
Concrete	5.21 ± 0.56	5.23 ± 0.53	6.03 ± 0.58	5.67 ± 0.57	5.36 ± 0.51	5.20 ± 0.59	**5.14 ± 0.58**
Energy	**0.71 ± 0.14**	1.66 ± 0.19	2.09 ± 0.29	2.01 ± 0.29	1.79 ± 0.24	1.22 ± 0.25	1.07 ± 0.41
Kin8nm	0.08 ± 0.00	0.10 ± 0.00	0.09 ± 0.00	0.08 ± 0.00	0.08 ± 0.00	**0.07 ± 0.00**	**0.07 ± 0.00**
Power plant	4.12 ± 0.17	**4.02 ± 0.18**	4.11 ± 0.17	4.12 ± 0.16	4.10 ± 0.15	4.07 ± 0.15	4.05 ± 0.13
Wine	0.66 ± 0.04	**0.62 ± 0.04**	0.64 ± 0.04	0.63 ± 0.04	0.64 ± 0.04	0.64 ± 0.04	0.66 ± 0.05
Yacht	0.96 ± 0.36	1.11 ± 0.38	1.58 ± 0.48	0.98 ± 0.38	0.85 ± 0.36	0.83 ± 0.40	**0.70 ± 0.26**

Table 2: Average NLL of test examples for regression experiments on real datasets. DGMEs obtain competitive or better performance in terms of NLL on the majority of datasets as compared to the baselines.

TEST NLL							
Dataset	MDNs	MCD	DEs	DGMEs (J=1)	DGMEs (J=2)	DGMEs (J=5)	DGMEs (J=10)
Boston housing	2.62 ± 0.43	2.46 ± 0.25	2.41 ± 0.25	2.34 ± 0.19	**2.33 ± 0.22**	2.41 ± 0.25	2.46 ± 0.31
Concrete	3.11 ± 0.26	3.04 ± 0.09	3.06 ± 0.18	3.04 ± 0.11	3.00 ± 0.12	2.95 ± 0.13	**2.94 ± 0.14**
Energy	**1.18 ± 0.30**	1.99 ± 0.09	1.38 ± 0.22	1.71 ± 0.19	1.48 ± 0.15	1.20 ± 0.23	1.20 ± 0.40
Kin8nm	-1.18 ± 0.04	-0.95 ± 0.03	-1.20 ± 0.02	-1.20 ± 0.02	-1.23 ± 0.03	-1.24 ± 0.02	**-1.25 ± 0.02**
Power plant	2.81 ± 0.04	2.80 ± 0.05	**2.79 ± 0.04**	2.82 ± 0.03	2.81 ± 0.03	2.81 ± 0.03	2.79 ± 0.03
Wine	1.01 ± 0.10	**0.93 ± 0.06**	0.94 ± 0.12	0.95 ± 0.11	0.96 ± 0.11	0.96 ± 0.12	1.10 ± 0.09
Yacht	1.18 ± 0.17	1.55 ± 0.12	1.18 ± 0.21	1.07 ± 0.22	0.75 ± 0.22	0.60 ± 0.29	**0.49 ± 0.29**

Table 3: Average predictive entropy for classification datasets. DGMEs are able to appropriately reason about the underlying uncertainty of OOD samples (MNIST with unknown classes and fashion MNIST) and is competitive with respect to state-of-the-art approaches.

AVERAGE PREDICTIVE ENTROPY				
Dataset	MDNs	MCD	DEs	DGMEs
MNIST (Known)	0.019 ± 0.005	0.012 ± 0.003	0.012 ± 0.002	0.015 ± 0.002
MNIST (Unknown)	0.192 ± 0.032	0.180 ± 0.020	0.180 ± 0.020	0.193 ± 0.016
Fashion MNIST	0.663 ± 0.110	0.714 ± 0.140	0.706 ± 0.067	0.698 ± 0.057

Table 3, which are averaged over 10 independent runs of each method. The results indicate that DGMEs are able to appropriately reason about the uncertainty in each of the datasets and is competitive with the baseline approaches in each case. DGMEs appropriately obtain that lowest entropy on the training dataset (i.e., the digits it was trained on), obtains a slightly higher entropy on the MNIST dataset containing unknown classes, and the highest entropy on the fashion MNIST dataset, which contains examples unrelated to the original classification task.

6 CONCLUSIONS

This paper proposes DGMEs, a novel probabilistic DL ensemble method for jointly quantifying epistemic and aleatoric uncertainty. Unlike deep ensembling, DGMEs optimizes the data likelihood directly and is able to capture complex behavior in the predictive distribution (e.g., heavy-tailedness and multimodality) by modeling the conditional distribution of the data as a Gaussian mixture. Our experiments show that DGMEs can capture more complex distributional properties than a variety of probabilistic DL baselines in regression settings and obtain competitive performance on detecting OOD samples in classification settings. As next steps, alternative mechanisms for handling the epistemic uncertainty can be considered. For example, one can instead form a variational approximation to the posterior of each mixture component, thereby forming a Gaussian mixture approximation to the posterior parameters of the ensemble. Additionally, a more thorough analysis on the classification setting can be considered. Rather than using a mixture of categorical distributions to model the predictive density, one can use a mixture of Dirichlet distributions to account for uncertainty in the class probabilities, similar in line to the work of [Hobbhahn et al., 2022]. Finally, DGMEs can be applied to improve the efficiency of active learning algorithms and exploration strategies in reinforcement learning.

References

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.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

Athanassia Bacharoglou. Approximation of probability distributions by convex mixtures of gaussian measures. *Proceedings of the American Mathematical Society*, 138 (7):2619–2628, 2010.

Loris Bazzani, Hugo Larochelle, and Lorenzo Torresani. Recurrent mixture density network for spatiotemporal visual attention. *arXiv preprint arXiv:1603.08199*, 2016.

Edmon Begoli, Tanmoy Bhattacharya, and Dimitri Kusnezov. The need for uncertainty quantification in machine-assisted medical decision making. *Nature Machine Intelligence*, 1(1):20–23, 2019.

Christopher M Bishop. Mixture density networks. 1994.

Bertrand Charpentier, Ransalu Senanayake, Mykel Kochenderfer, and Stephan Günnemann. Disentangling epistemic and aleatoric uncertainty in reinforcement learning. *arXiv preprint arXiv:2206.01558*, 2022.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691. PMLR, 2014.

William R Clements, Bastien Van Delft, Benoît-Marie Robaglia, Reda Bahi Slaoui, and Sébastien Toth. Estimating risk and uncertainty in deep reinforcement learning. *arXiv preprint arXiv:1905.09638*, 2019.

Christopher N Davis, T Deirdre Hollingsworth, Quentin Caudron, and Michael A Irvine. The use of mixture density networks in the emulation of complex epidemiological individual-based models. *PLoS computational biology*, 16(3):e1006869, 2020.

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: 20089–20103, 2021.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Antonio D'Isanto and Kai Lars Polsterer. Photometric redshift estimation via deep learning-generalized and pre-classification-less, image based, fully probabilistic redshifts. *Astronomy & Astrophysics*, 609:A111, 2018.

Max H Farrell, Tengyuan Liang, and Sanjog Misra. Deep neural networks for estimation and inference. *Econometrica*, 89(1):181–213, 2021.

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.

Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

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.

Marius Hobbhahn, Agustinus Kristiadi, and Philipp Hennig. Fast predictive uncertainty for classification with bayesian deep networks. In *Uncertainty in Artificial Intelligence*, pages 822–832. PMLR, 2022.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 2013.

Lara Hoffmann and Clemens Elster. Deep ensembles from a bayesian perspective. *arXiv preprint arXiv:2105.13283*, 2021.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: A tutorial introduction. 2019.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

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

Rhiannon Michelmore, Marta Kwiatkowska, and Yarin Gal. Evaluating uncertainty quantification in end-to-end autonomous driving control. *arXiv preprint arXiv:1811.06817*, 2018.

Thomas Minka. Expectation-maximization as lower bound maximization. *Tutorial published on the web at http://www-white. media. mit. edu/tpminka/papers/em. html*, 7(2), 1998.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

Tim Pearce, Felix Leibfried, and Alexandra Brintrup. Uncertainty in neural networks: Approximately bayesian ensembling. In *International conference on artificial intelligence and statistics*, pages 234–244. PMLR, 2020.

Utkarsh Sarawgi, Wazeer Zulfikar, Rishab Khincha, and Pattie Maes. Why have a unified predictive uncertainty? disentangling it using deep split ensembles. *arXiv preprint arXiv:2009.12406*, 2020.

Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. In *International Conference on Machine Learning*, pages 4907–4916. PMLR, 2018.

Rohit Unni, Kan Yao, and Yuebing Zheng. Deep convolutional mixture density network for inverse design of layered photonic structures. *ACS Photonics*, 7(10):2703–2712, 2020.

Matias Valdenegro-Toro and Daniel Saromo Mori. A deeper look into aleatoric and epistemic uncertainty disentanglement. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1508–1516. IEEE, 2022.

Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708, 2020.

CF Jeff Wu. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103, 1983.

Heiga Zen and Andrew Senior. Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 3844–3848. IEEE, 2014.

Hao Zhang, Yongqian Liu, Jie Yan, Shuang Han, Li Li, and Quan Long. Improved deep mixture density network for regional wind power probabilistic forecasting. *IEEE Transactions on Power Systems*, 35(4):2549–2560, 2020.