A Comparative Study of Uncertainty Estimation Methods in 2D Object Detection: BayesOD and Deep Ensembles

Andreas Kaag Thomsen*, Asger Song Høøck Poulsen* and Niels Viggo Stark Madsen*

***Abstract*—The deployment of deep neural network-based object detection models in safety-critical domains such as autonomous driving is hindered by their lack of uncertainty awareness; estimating uncertainty reliably is thus essential to assess prediction reliability. This report compares two methods for uncertainty estimation in deep neural network-based object detectors: BayesOD and Deep Ensembles. While both methods modify the detection head for probabilistic outputs, they differ in estimating epistemic uncertainty and post-processing. The study aims to clearly illustrate their shared principles as well as key differences and evaluate their performance on the Berkeley DeepDrive dataset. The experiments indicate that while both methods maintain comparable standard detection performance to a baseline RetinaNet model, BayesOD demonstrates superior performance in regression uncertainty metrics. We hypothesize that this advantage stems from its probabilistic modeling and Bayesian fusion approach—an interpretation explored further in the study. Classification uncertainty metrics did not show a clear advantage for either method.**

***Index Terms*—Object Detection, Uncertainty Estimation, Bayesian Neural Networks, Deep Ensembles, Monte Carlo Dropout, Bayesian Inference.**

I. Introduction

While detection models based on deep neural networks (DNNs) have achieved remarkable performance across various applications, their deployment in safety-critical domains like autonomous driving is hindered by a lack of awareness of uncertainty: standard models often provide predictions without indicating how reliable these predictions are. For example, a pedestrian partially occluded by a parked car might be confidently misclassified or missed entirely, with no accompanying signal of low confidence posing a serious safety risk. Estimating uncertainty is thus essential to assess prediction reliability, support downstream decision-making, enhance safety, and improve interpretability [1], [2] (see Figure 1).

This study[1] compares two methods for incorporating uncertainty estimation into DNN-based object detectors: **BayesOD**, introduced by Harakeh et al. [3] in 2019, and **Deep Ensembles**, proposed by Lakshminarayanan et al. [4] in 2017. These methods were selected based on their strong performance in a review and comparative study by Feng et al. [2], whose implementation framework we also adopt in this study. Deep Ensembles have gained widespread popularity in the literature [2], [5], [6], evidenced by their frequent use and high citation count. They are regarded as a strong baseline due to their simplicity, effectiveness, and compatibility with a variety of model architectures. BayesOD was chosen as a complementary comparison point due to its strong Bayesian reformulation of object detection.

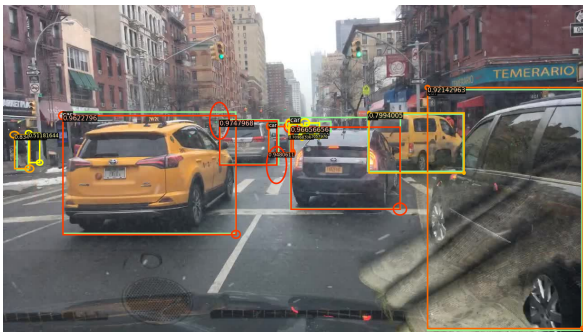


Fig. 1: The output of a 2D object detector with uncertainty estimation for both category (scalar) and bounding boxes (Gaussian distributions). The method from this sample is BayesOD on a test image from the BDD test set.

While we considered exploring more recent methods, this was not achievable. Newer approaches such as MC DropBlock [7] and EvCenterNet [8] show promise but lack publicly available code. Consequently, we focused on established methods with accessible and reproducible implementations to ensure a robust evaluation.

Both methods share a common foundation in how they adapt the detection head to produce probabilistic outputs for bounding boxes and categories. However, they diverge significantly in their strategies for estimating uncertainty components (particularly epistemic uncertainty) and in their post-processing techniques for handling redundant per-anchor predictions. BayesOD utilizes Monte Carlo (MC) Dropout and fuses anchor predictions via Bayesian Inference, while Deep Ensembles leverages multiple independently trained models and relies on Non-Maximum Suppression (NMS).

The objective of this work is to restructure the presentation of these methods to clearly delineate their shared principles and key differences, following the comparative structure outlined initially. The rest of the report is structured as follows: In Section II, we first discuss the general concepts of uncertainty in object detection, then detail the common probabilistic

*Department of Electrical & Computer Engineering, Aarhus University. {au691667, au704738, au704610}uni.au.dk

[1]Our code is publicly available at: https://github.com/akthomsen/CV_Uncertainty_Estimation

modeling approach used in the detection head by both methods in Section III. Subsequently, in Section IV, we elaborate on the specific mechanisms of BayesOD and Deep Ensembles for uncertainty estimation and post-processing. We aim to provide a clear comparative analysis to facilitate the understanding and evaluation of these techniques. Finally, in Section V, both methods are evaluated on the same dataset to clearly show differences in performance.

II. Background: Uncertainty in Object Detection

Probabilistic object detection seeks not only to classify and localize objects but also to quantify the uncertainty associated with these predictions. This uncertainty provides information about the reliability of the detector's output.

A. 2D Object Detection

2D object detection is a fundamental task in computer vision that involves identifying and localizing objects within an image by predicting bounding boxes and associated class labels. Modern detectors typically follow one of two main architectures: one-stage detectors (e.g., YOLO [9], RetinaNet [10]), which perform classification and localization in a single pass; and two-stage detectors (e.g., Faster R-CNN [11]), which first generate region proposals and then refine them. These models have demonstrated strong performance across a range of benchmarks, but standard implementations yield deterministic outputs and provide no information about prediction confidence or uncertainty.

B. Types of Uncertainty

Predictive uncertainty in deep learning models can generally be decomposed into two primary types [1], [12]:

- **Aleatoric Uncertainty:** Also known as data uncertainty, this captures the inherent noise or randomness in the observations themselves. Examples include uncertainty due to sensor noise, occlusions, or inherent ambiguity in the input data (e.g., a distant object with few pixels). This type of uncertainty cannot be reduced by collecting more data, as it is a property of the data distribution itself.
- **Epistemic Uncertainty:** Also known as model uncertainty, this reflects the uncertainty in the model parameters due to limited training data. It captures the model's lack of knowledge about regions of the input space that were under-represented in the training set. Epistemic uncertainty can, in principle, be reduced by increasing the size and diversity of the training dataset.

Capturing both types is crucial for robust perception systems. Aleatoric uncertainty informs about the quality of the current observation, while epistemic uncertainty indicates whether the model is operating within its domain of expertise.

C. Probabilistic Modeling in Object Detectors

Modern object detectors, whether one-stage or two-stage, often rely on a dense grid of predefined anchor boxes. For each anchor, the detector predicts corrections to the anchor's bounding box and probabilities for object categories.

To enable uncertainty estimation, the deterministic outputs are replaced by probabilistic ones [2]. This typically involves modifying the network's output layer (the "detection head") and the loss function used for training. The subsequent sections detail the common approach adopted by both BayesOD and Deep Ensembles (as implemented by Feng et al. [2]) at the detection head level.

III. Core Probabilistic Modeling at the Detection Head

A key similarity between the BayesOD [3] and Deep Ensemble [4] implementations compared here lies in how they model the raw per-anchor outputs probabilistically. Both methods treat bounding box regression and category classification as tasks requiring probability distributions rather than point estimates.

A. Modeling Bounding Boxes as Gaussian Distributions

Instead of directly regressing bounding box coordinates (e.g., top-left and bottom-right corners), the detection head is modified to output parameters of a probability distribution for the bounding box.

- **Distribution Choice:** The bounding box coordinates $\mathcal{B}_i$ for anchor i are modeled as a multivariate Gaussian distribution: $\mathcal{B}_i \sim \mathcal{N}(\mu(x_i), \Sigma(x_i))$ [3], [4]. $\mu(x_i)$ represents the predicted mean coordinates, and $\Sigma(x_i)$ represents the predicted covariance matrix, capturing the uncertainty.
- **Network Output:** The network's detection head outputs the mean vector $\mu(x_i)$ and parameters representing the covariance matrix $\Sigma(x_i)$. Often, a diagonal covariance matrix is predicted initially, representing only the variances of each coordinate [4].
- **Loss Function Modification:** To train the network to predict these distributional parameters, the standard regression loss is replaced with a loss derived from the negative log-likelihood (NLL) of the chosen distribution [3], [4]. For a Gaussian distribution, this results in a loss function that encourages the mean prediction $\mu(x_i)$ to be close to the ground truth y_i, while also learning the variance $\Sigma(x_i)$. A common form for univariate variance σ^2 prediction is given by Lakshminarayanan et al. [4]:

$$L_{reg}(x_i, \theta) = \frac{(y_i - \mu(x_i, \theta))^2}{2\sigma^2(x_i, \theta)} + \frac{1}{2}\log\sigma^2(x_i, \theta) \quad (1)$$

BayesOD utilizes a multivariate version to learn the full covariance matrix [3], incorporating terms for the inverse covariance and its determinant:

$$L_{mv}(x_i, \theta) = \frac{1}{2}(f(x_i, \theta) - y_i)^T \Sigma_a(x_i, \theta)^{-1}(f(x_i, \theta) - y_i) + \frac{1}{2}\log\det\Sigma_a(x_i, \theta) \quad (2)$$

The first term in Equation (2)[2] is a Mahalanobis distance, where bounding box errors are weighted by the inverse

[2]The authors in [3] also introduce a *stable surrogate equation* to ensure numerical stability. The interpretation is the same, and is omitted here.

aleatoric uncertainty, as outputted by the network. This means that if the network predicts a large uncertainty (large values in Σ_a, thus small values in Σ_a^{-1}) for a given input x_i, the penalty for a large error $(f(x_i, \theta) - y_i))$ is reduced. Conversely, if the network predicts small uncertainty, large errors are heavily penalised. The second term penalizes overly confident predictions, i.e., those with very small predicted uncertainty Σ_a. If the model predicts a small error and a tiny variance, it might appear very confident, even if it is wrong. Without this regularizer, the model could "cheat" by making the variance approach zero to minimize the main error term. The interpretation for Equation (1) is similar.

B. Modeling Categories as Categorical Distributions

For object classification, the standard approach of using the softmax function output is maintained, as it already provides a probabilistic interpretation [3], [4].

- **Distribution Choice:** The category label $\mathcal{S}_i$ for anchor i is modeled using a Categorical distribution $\mathcal{S}_i \sim Cat([p_1, \ldots, p_K])$, where K is the number of classes and p_k is the probability of the k-th class.
- **Network Output:** The network outputs logits $g(x_i, \theta)$, which are then passed through a SoftMax function to obtain the category probabilities $[p_1, \ldots, p_K]$.

$$p_k = \text{SoftMax}(g(x_i, \theta))_k$$

Although both methods share this foundational probabilistic modeling at the detection head, they differ substantially in how they estimate epistemic uncertainty and how they aggregate these per-anchor distributions into final object detections.

IV. Uncertainty Estimation Approaches and Post-Processing

This section outlines the distinct methodologies of BayesOD and Deep Ensembles for estimating uncertainty and performing post-processing to obtain final detections.

A. BayesOD

BayesOD [3] integrates Bayesian principles throughout the detection pipeline, notably using MC-Dropout for epistemic uncertainty and Bayesian fusion for post-processing. The necessary prerequisite knowledge on Bayesian Neural Networks (BNNs) is provided in Appendix A.

1) Uncertainty Estimation via Monte-Carlo Dropout: BayesOD uses MC-Dropout [13] as an approximation for Bayesian inference to estimate epistemic uncertainty. During inference, the input image is passed through the network T times with dropout layers kept active. Each pass yields slightly different outputs due to the stochasticity of dropout, effectively sampling from an approximate posterior distribution over the network weights.

Each predicted bounding box $\hat{\mathcal{B}}_i$ is modelled as a Gaussian Distribution: $\hat{\mathcal{B}}_i \sim \mathcal{N}(\mu(x_i), \Sigma(x_i))$ The *mean bounding box* and *epistemic covariance matrix* are given by Equations (3) and (4), where $f(x_i, \theta_t)$ is the i-th bounding box output of the DNN for the t-th MC-Dropout run.

$$\mu(x_i) = \frac{1}{T} \sum_{t=1}^{T} f(x_i, \theta_t) \tag{3}$$

$$\Sigma_e(x_i) = \frac{1}{T} \sum_{t=1}^{T} (f(x_i, \theta_t) f(x_i, \theta_t)^\top) - \mu(x_i)\mu(x_i)^\top \tag{4}$$

Equation (3) calculates the average bounding box prediction across the T MC-Dropout runs, giving the most likely bounding box location. Equation (4) measures how much the individual predictions vary from the mean, meaning that high variance indicates high epistemic uncertainty.

BayesOD estimates **aleatoric uncertainty** by having the network directly predict an aleatoric covariance matrix $\Sigma_a(x_i, \theta_t)$ for each pass, trained via the multivariate NLL loss (Eq. 2). The average aleatoric covariance is computed:

$$\Sigma_a(x_i) = \frac{1}{T} \sum_{t=1}^{T} \Sigma_a(x_i, \theta_t) \tag{5}$$

Epistemic and averaged aleatoric covariances are summed to get the total covariance for the likelihood function:

$$\Sigma(x_i) = \Sigma_e(x_i) + \Sigma_a(x_i) \tag{6}$$

For the **object category classification**, the T sets of category logits $g(x_i, \theta_t)$ are converted to probabilities via SoftMax, and then averaged to get the final per-anchor category probabilities $[\hat{p}_1, \ldots, \hat{p}_K]$:

$$\hat{p}_k = \frac{1}{T} \sum_{t=1}^{T} \text{SoftMax}(g(x_i, \theta_t))_k \tag{7}$$

The variance across these T probability vectors indicates epistemic uncertainty in classification. The aleatoric uncertainty is considered implicit in the averaged probabilities [3].

2) Per-Anchor Posteriors with Priors: BayesOD incorporates prior beliefs for each anchor before fusion [3]. These priors represent initial assumptions about the bounding box and category for each anchor.

For the **bounding box**, the prior belief $p(\mathcal{B}|x_i)$ is modeled as a multivariate Gaussian distribution $\mathcal{N}(\mu_0, \Sigma_0)$. This prior distribution is combined with the likelihood function derived from the network's output (represented by the mean $\mu(x_i)$ and total covariance $\Sigma(x_i)$ from Equations (3) and (6)) using Bayes' Theorem to obtain the posterior distribution of the i-th anchor:

$$p(\mathcal{B}|x_i, \mathcal{D}, \hat{\mathcal{B}}_i) \propto p(\hat{\mathcal{B}}_i|x_i, \mathcal{D}, \mathcal{B}) p(\mathcal{B}|x_i, \mathcal{D}) \tag{8}$$

Since both the prior and the likelihood are Gaussian, the resulting per-anchor posterior distribution for the bounding box is also Gaussian. Its mean $\mu'(x_i)$ and covariance $\Sigma'(x_i)$ are calculated in closed form via the multivariate Gaussian conjugate update:

$$\Sigma'(x_i) = (\Sigma_0^{-1} + \Sigma(x_i)^{-1})^{-1} \tag{9}$$

$$\mu'(x_i) = \Sigma'(x_i)(\Sigma_0^{-1}\mu_0 + \Sigma(x_i)^{-1}\mu(x_i)) \tag{10}$$

In the absence of specific prior information from the 2D datasets used, BayesOD employs a non-informative prior for the bounding box.

For the **category**, instead of placing a prior directly on the category label $\mathcal{S}$, BayesOD defines a prior distribution over the *parameters* $\mathcal{P} = [p_1, \ldots, p_K]$ of the Categorical distribution $Cat(\mathcal{P})$ that is assumed to generate the label. BayesOD uses a Dirichlet distribution, $Dir(\alpha_1, \ldots, \alpha_K)$, as the prior $p(\mathcal{P}|x_i, \mathcal{D})$. The parameters α_k of the Dirichlet distribution can be thought of as 'pseudo-counts' representing the strength of our prior belief for each category k. A higher α_k value indicates a stronger prior belief favoring category k.

The Dirichlet distribution is chosen because it is the conjugate prior for the Categorical likelihood function $p(\hat{Z}_i|x_i, \mathcal{D}, \mathcal{P})$. Here, $\hat{Z}_i = [\hat{z}_1, \ldots, \hat{z}_H]$ represents H category samples drawn from the network's output distribution for anchor i (obtained via MC-Dropout sampling). The Bayesian update follows the standard rule:

$$p(\mathcal{P}|x_i, \mathcal{D}, \hat{Z}_i) \propto p(\hat{Z}_i|x_i, \mathcal{D}, \mathcal{P})p(\mathcal{P}|x_i, \mathcal{D}) \tag{11}$$

Due to conjugacy, the posterior distribution $p(\mathcal{P}|x_i, \mathcal{D}, \hat{Z}_i)$ is also a Dirichlet distribution, $Dir(\alpha'_1, \ldots, \alpha'_K)$. The updated parameters α'_k are calculated by simply adding the observed counts to the prior pseudo-counts:

$$\alpha'_k = \alpha_k + \sum_{h=1}^{H} \hat{z}_{hk}$$

Here $\sum_{h=1}^{H} \hat{z}_{hk}$ is the total number of times category k was predicted among the H samples $\hat{z}_h$ drawn from the network's output for anchor i. Intuitively, this update rule combines our prior belief (strength α_k) with the evidence from the network's predictions (number of times category k was seen). The posterior parameter α'_k reflects this combined knowledge.

Finally, the resulting per-anchor **posterior category distribution** $p(\mathcal{S}|x_i, \mathcal{D}, Z_i)$ (the probability distribution for the *actual* category label $\mathcal{S}$ for anchor i) is a Categorical distribution $Cat(p'_1, \ldots, p'_K)$. The parameters p'_k of this distribution are obtained by taking the *mean* of the posterior Dirichlet distribution $Dir(\alpha'_1, \ldots, \alpha'_K)$:

$$p'_k = \frac{\alpha'_k}{\sum_{j=1}^{K} \alpha'_j} \tag{12}$$

This calculation gives the expected probability for each category k based on our updated beliefs about the parameters $\mathcal{P}$ (represented by the posterior Dirichlet parameters α'_k). Dividing by the sum of all α'_j ensures that the resulting probabilities p'_k sum to 1. Similarly to the bounding box, a non-informative Dirichlet prior is used.

3) Bayesian Fusion: One key innovation of BayesOD is replacing the standard NMS step with Bayesian Fusion. Instead of discarding redundant anchor predictions, BayesOD fuses the information from overlapping anchors using Bayesian inference. In the following, it is described how the per-anchor posteriors for bounding box and category are fused into two final posterior distributions.

The process begins with **clustering** the per-anchor posterior outputs based on their spatial proximity. Anchors whose predicted bounding boxes significantly overlap (high spatial affinity) are grouped into the same cluster. BayesOD uses a greedy clustering approach for efficiency.

Within each cluster containing N anchors, Bayesian inference is performed to combine their information. The general approach is that one anchor, often the one with the highest initial category score, is chosen as the cluster center (indexed $i = 1$). This anchor is described by its posterior distributions $p(\mathcal{B}|x_1, \mathcal{D}, \hat{\mathcal{B}}_1)$ and $p(\mathcal{P}|x_1, \mathcal{D}, \hat{Z}_1)$ for bounding box $\mathcal{B}$ and category parameters $\mathcal{P}$, respectively. The predictions from the other $N-1$ anchors in the cluster are treated as *measurements* to update the center anchor's posterior.

For the **bounding box**, the final fused posterior distribution $p(\mathcal{B}|\mathcal{X}, \mathcal{D}, [\hat{\mathcal{B}}_1, \ldots, \hat{\mathcal{B}}_N])$, where $\mathcal{X}$ represents the inputs for all N anchors, is obtained by multiplying the cluster center's posterior with the likelihoods from the other anchors' predictions:

$$\overbrace{p(\mathcal{B}|\mathcal{X}, \mathcal{D}, [\hat{\mathcal{B}}_1, ..., \hat{\mathcal{B}}_N])}^{\text{Fused bounding box posterior}} \propto \overbrace{p(\mathcal{B}|x_1, \mathcal{D}, \hat{\mathcal{B}}_1)}^{\text{Center posterior}} \times \underbrace{\prod_{i=2}^{N} p(\hat{\mathcal{B}}_i|x_i, \mathcal{D}, \mathcal{B})}_{\text{Likelihoods of measurements}} \tag{13}$$

The resulting fused distribution is also a Gaussian: $\mathcal{N}(\mu''(\mathcal{X}), \Sigma''(\mathcal{X}))$. Its statistics are computed in closed form by combining the per-anchor posterior statistics $(\mu'(x_i), \Sigma(x_i))$ from Equations (9) and (10)) of all N anchors in the cluster:

$$\Sigma''(\mathcal{X}) = \left(\sum_{i=1}^{N} \Sigma'(x_i)^{-1} \right)^{-1} \tag{14}$$

$$\mu''(\mathcal{X}) = \Sigma''(\mathcal{X}) \left(\sum_{i=1}^{N} \Sigma'(x_i)^{-1} \mu'(x_i) \right) \tag{15}$$

For the **category**, a similar fusion is performed on the Dirichlet posterior distributions of the parameters $\mathcal{P}$. The posterior parameter distribution for the cluster center $p(\mathcal{P}|x_1, \mathcal{D}, \hat{Z}_1)$ (from Equation (11)) is updated using the category predictions $\hat{Z}_i$ from the other $N-1$ anchors:

$$\overbrace{p(\mathcal{P}|\mathcal{X}, \mathcal{D}, [\hat{Z}_1, ..., \hat{Z}_N])}^{\text{Fused category posterior}} \propto \overbrace{p(\mathcal{P}|x_1, \mathcal{D}, \hat{Z}_1)}^{\text{Center posterior}} \times \underbrace{\prod_{i=2}^{N} p(\hat{Z}_i|x_i, \mathcal{D}, \mathcal{P})}_{\text{Likelihoods of measurements}} \tag{16}$$

The resulting fused posterior distribution over the parameters is also a Dirichlet distribution, $Dir(\alpha''_1, \ldots, \alpha''_K)$. The parameters α''_k are computed by summing the contributions from all anchors in the cluster:

$$\alpha''_k = \alpha'_k + \sum_{i=2}^{N} \sum_{h=1}^{H} \hat{z}_{ihk} \tag{17}$$

The final fused category distribution $p(\mathcal{S}|\mathcal{X}, \mathcal{D}, [\hat{Z}_1, \ldots, \hat{Z}_N])$ is then a Categorical distribution $Cat(p''_1, \ldots, p''_K)$, with parameters calculated as the mean of the fused Dirichlet posterior:

$$p''_k = \frac{\alpha''_k}{\sum_{j=1}^{K} \alpha''_j} \tag{18}$$

Overall, every anchor within a cluster contributes information to the final fused estimate for both the bounding box and category. This fusion process avoids the information loss associated with NMS and allows for a more robust estimation of both the prediction and its uncertainty.

B. Deep Ensembles

Deep Ensembles [4] offer a simple, effective alternative to BNNs for uncertainty estimation. The core idea involves training multiple independent models and combining their predictions. This approach generally yields good results, is straightforward to implement, easily parallelized, and requires minimal additional hyperparameter tuning [4]. By averaging the outputs of several similar models, the ensemble method inherently captures epistemic uncertainty arising from variations in model parameters found during different training runs.

Compared to MC-Dropout [13], which can also be viewed as a form of ensembling where multiple predictions are averaged, Deep Ensembles differ significantly in that each model has independent parameters. MC-Dropout relies on shared parameters across its stochastic forward passes. This independence in Deep Ensembles might prevent potential information loss associated with parameter sharing or dropout layers [4].

In the context of object detection as implemented by Feng et al. [2], this ensemble approach uses NMS for post-processing, distinguishing it from the original ensemble paper's focus on classification/regression benchmarks, which did not involve object detection-specific post-processing like NMS.

1) Uncertainty Estimation via Multiple Models: The core idea is to train M identical network architectures independently and average their predictions. This randomization-based approach ensures models can be trained in parallel and helps de-correlate their outputs [4].

M ensemble members (typically $M = 5$ is sufficient and recommended [4]) are trained on the same dataset. Unlike some ensemble methods that use bagging/bootstrapping, Deep Ensembles often achieve better results by training each model on the whole dataset, relying on random weight initializations and random shuffling of training data for each epoch to ensure diversity [4]. Each ensemble member m is trained to predict probabilistic outputs $(\mu_{\theta_m}(x_i), \Sigma_{\theta_m}(x_i))$ for bounding boxes and $p_{\theta_m}(S|x_i)$ for categories, using a proper scoring rule like the NLL loss (Eq. 1) as the training criterion.

At inference time, an input image is passed through all M models, and the predictions are combined to form the final ensemble prediction. The final **category** probabilities are the average of the probabilities from individual models:

$$p(S = k|x_i) = \frac{1}{M} \sum_{m=1}^{M} p_{\theta_m}(S = k|x_i)$$

The final **bounding box** distribution $p(\mathcal{B}|x_i) \sim \mathcal{N}(\mu_*(x_i), \Sigma_*(x_i))$ is modelled as an equally weighted mixture of the M Gaussian outputs $\mathcal{N}(\mu_{\theta_m}(x_i), \Sigma_{\theta_m}(x_i))$:

$$p(\mathcal{B}|x_i) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{N}(\mathcal{B}; \mu_{\theta_m}(x_i), \Sigma_{\theta_m}(x_i))$$

The mean $\mu_*(x_i)$ and covariance $\Sigma_*(x_i)$ of this mixture distribution represent the final prediction and its total uncertainty:

$$\mu_*(x_i) = \frac{1}{M} \sum_{m=1}^{M} \mu_{\theta_m}(x_i) \tag{19}$$

$$\Sigma_*(x_i) = \frac{1}{M} \sum_{m=1}^{M} (\Sigma_{\theta_m}(x_i) + \mu_{\theta_m}(x_i)\mu_{\theta_m}(x_i)^\top) - \mu_*(x_i)\mu_*(x_i)^\top \tag{20}$$

The total variance $\Sigma_*(x_i)$ implicitly contains both aleatoric and epistemic components. The average of the variances predicted by individual models ($\frac{1}{M}\sum \Sigma_{\theta_m}(x_i)$) corresponds to the aleatoric uncertainty, while the variance of the means across the models ($\frac{1}{M}\sum \mu_{\theta_m}\mu_{\theta_m}^\top - \mu_*\mu_*^\top$) corresponds to the epistemic uncertainty [4], [12].

2) Post-Processing: Non-Maximum Suppression: Unlike BayesOD's Bayesian Fusion, the implementation of Deep Ensembles for object detection by Feng et al. [2] relies on standard NMS for post-processing. Standard NMS is typically applied after obtaining the combined per-anchor predictions from the ensemble (averaged probabilities for categories, mixture mean, and covariance for boxes). NMS greedily selects the detection with the highest category probability and suppresses overlapping detections based on an IoU threshold. Feng et al. [2] explore variations based on *when* the ensemble fusion (i.e., creating the Gaussian mixture from the M models' outputs) occurs relative to NMS. In *Pre-NMS* fusion, the ensemble outputs are fused first, and then NMS is applied to these fused predictions. In *Post-NMS* fusion, NMS is applied individually to the outputs of each of the M models, and only the surviving detections are subsequently fused together.

NMS discards information from the suppressed detections [3], contrasting with the fusion approach of BayesOD. While simpler and faster, this can lead to less robust uncertainty estimates, particularly for the bounding box, as potentially valuable information from redundant anchors is ignored.

C. Comparison Summary

The core difference lies in the approach to epistemic uncertainty and post-processing:

- **Epistemic Uncertainty:** BayesOD [3] uses MC-Dropout [13] within a single model framework, while Deep Ensembles [4] use multiple independently trained models.
- **Aleatoric Uncertainty:** Both rely on modifying the loss function (NLL) to have the network(s) predict variance/covariance alongside the mean for bounding boxes [3], [4]. For categories, both use softmax outputs.

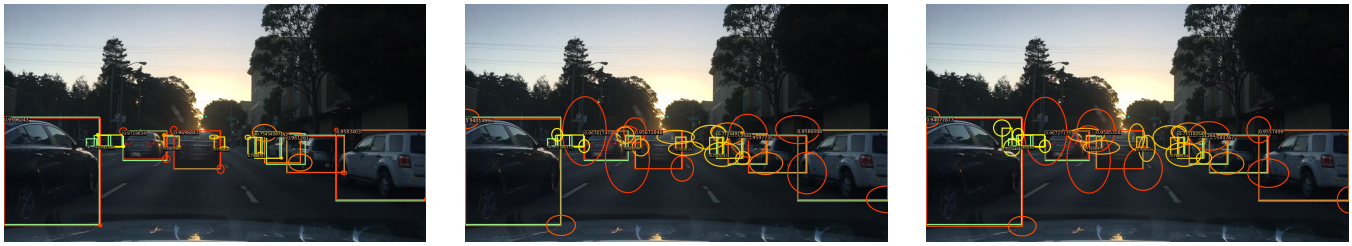
TABLE I: An evaluation of the different methods across chosen metrics

Method	mAP ↑	Cls MUE ↓	Reg MUE ↓	Cls NLL ↓	Reg NLL ↓	Cls MCE ↓	Reg ECE ↓
Deterministic RetinaNet (Baseline)	0.288	0.1949	-	0.6421	-	0.0586	-
BayesOD	0.274	**0.1877**	**0.3198**	0.6395	**12.3188**	0.0610	**0.0098**
Deep Ensembles (pre-NMS)	**0.291**	0.1964	0.4701	0.6407	14.7956	0.0644	0.0424
Deep Ensembles (post-NMS)	0.290	0.1901	0.4753	**0.6353**	14.7990	**0.0580**	0.0425

Fig. 2: Predictions from all methods on samples from the BDD test set

(a) BayesOD (day) (b) Deep Ensembles w. pre-NMS (day) (c) Deep Ensembles w. post-NMS (day)

(d) BayesOD (night) (e) Deep Ensembles w. pre-NMS (night) (f) Deep Ensembles w. post-NMS (night)



- **Post-Processing:** BayesOD employs Bayesian Fusion [3] to combine information from overlapping anchors, whereas the Deep Ensemble implementation compared here typically uses NMS [2] to suppress overlapping anchors.
- **Implementation:** Deep Ensembles are often considered simpler to implement on top of existing deterministic training pipelines, requiring training multiple models in parallel [4]. BayesOD requires modifications for MC-Dropout inference, multivariate loss calculation, and the Bayesian fusion step [3].
- **Computation:** Deep Ensembles require M times the training resources [4]. MC-Dropout requires T forward passes at inference time [13]. Bayesian fusion can be computationally more intensive than standard NMS [3].

V. Experimental Results & Discussions

In this section, we present quantitative and qualitative comparisons between BayesOD and Deep Ensembles, followed by a discussion of the key factors driving these results.

A. Dataset

We use the Berkeley DeepDrive (BDD) object detection dataset [14], a large-scale collection of ∼100,000 high-resolution (1280×720) images captured from dashboard cameras in diverse driving scenarios. The dataset includes varying times of day (day/night) and weather conditions (clear, overcast, rain), making it well-suited for evaluating uncertainty in object detection.

The dataset is split into 70,000 training, 10,000 validation, and 20,000 test images, each annotated with 2D bounding boxes for 10 object categories (e.g., car, pedestrian, traffic light). In our work, we converted the original BDD annotations to COCO format using a custom script, which filters relevant object categories and reformats bounding boxes accordingly. We also limited the number of categories to the 7 most relevant objects we wanted to detect.

B. Training & Inference

All models follow the same structure as in [2] and use the original RetinaNet [10] as the backbone network. RetinaNet is a one-stage object detector, and consists of a residual neural network (ResNet-50) along with a feature neural net (FPN) forming a ResNet-50-FPN backbone network to extract features. Additionally, two fully connected subnetworks are responsible for object classification and bounding box regression, respectively.

Following [2], all models are trained for 90.000 iterations using stochastic gradient descent with 0.9 momentum with a batch size equal to 4. A base learning rate of 0.0025 is chosen and is dropped by a factor of 10 at epoch 60.000 and again at epoch 80.000. During inference, $T = 10$ stochastic dropout runs are performed for the BayesOD method. For the Deep Ensemble methods, $M = 5$ identical, but independent, ensemble members are trained, as recommended in [4].

C. Evaluation Metrics

Several metrics are employed to evaluate both the standard detection performance and the quality of the uncertainty estimates provided by the probabilistic detectors.

a) Mean Average Precision (mAP): A standard metric for object detection performance, Average Precision (AP) represents the area under the Precision-Recall (PR) curve computed for each object class. The final **mean Average Precision (mAP)** is the AP averaged across the car and pedestrian classes as well as over multiple Intersection over Union (IoU) thresholds (e.g., $[0.5, 0.55, \ldots, 0.95]$). Limiting the categories is done because they are the most important categories and the most prevalent in the dataset. While useful for assessing detection accuracy, mAP does not directly evaluate the predicted uncertainty [2]. It serves primarily to confirm that probabilistic modifications do not significantly degrade standard detection capabilities or to highlight secondary accuracy improvements.

b) Minimum Uncertainty Error (MUE): The Uncertainty Error (UE) metric quantifies how effectively an uncertainty estimate can distinguish between correct and incorrect detections [15]. For a given uncertainty threshold δ, UE measures the probability of misclassifying a detection based on its uncertainty, considering both correct detections rejected and incorrect detections accepted. UE ranges from 0 to 0.5, where 0.5 signifies performance no better than random guessing. The **Minimum Uncertainty Error (MUE)** is the lowest achievable UE across all possible thresholds δ [2], [15].

c) Negative Log Likelihood (NLL): As introduced in Section III-A, NLL evaluates the quality of the predicted probability distribution against the ground truth label [4]. It measures how well the predicted distribution fits the actual data, with lower values indicating a better fit.

d) Calibration Error (CE): Calibration assesses whether the predicted probabilities align with empirical frequencies [1], [16]. For instance, a perfectly calibrated model predicting an event with 80% probability should be correct 80% of the time. Calibration is often visualized using calibration plots, which compare predicted probabilities to observed frequencies within specific bins [2]. Common metrics derived from this include:

- **Expected Calibration Error (ECE):** Measures the weighted average difference between predicted confidence and actual accuracy across bins [2], [16].
- **Marginal Calibration Error (MCE):** Extends calibration evaluation to the multiclass setting by assessing the calibration of the predicted probability for each class individually, rather than just the top prediction [2], [16].

Lower calibration errors indicate that the model's confidence estimates are more reliable representations of the true likelihood.

D. Quantitative Evaluation

In this section, we analyze our results and highlight key findings, summarized in Table I. Additionally, we will discuss potential underlying mechanisms that may explain specific outcomes.

1) Mean Average Precision: In Table I , we see that all methods perform on par with the baseline deterministic RetinaNet model. Since the mAP is not a measure of uncertainty, but only of the model's performance, it aligns with the fact that all models are based on the same underlying network architecture. Importantly, performance has not been degraded by the added methods of calculating uncertainty estimates.

2) Class Uncertainty Results: The remaining metrics on the uncertainty of object classification (Cls MUE, Cls NLL, and Cls MCE) in Table I show no clear advantage or disadvantage to either of our uncertainty estimation methods. Any differences are small enough to be within a margin of error. This could be because using the baseline deterministic RetinaNet gives a satisfactory approximation of the uncertainties, with the output-layer through a softmax function. A factor in this could be that in our implementation, we only have 7 classes to differentiate between, and therefore, the room for uncertainty is not as great. There might be more to gain with these methods in a situation with a much larger number of objects to classify.

3) Regression Uncertainty Results: Because the baseline deterministic RetinaNet model has no output that can be interpreted in terms of uncertainty, they are not present in Table I. BayesOD is leading in all three metrics pertaining to measuring the uncertainty in the regression of the bounding boxes (Reg MUE, Reg NLL, and Reg ECE).

Reg MUE indicates that BayesOD is the only method capable of separating true positives from false positives, with a value $\sim$30% less than both ensemble methods. Both ensemble methods have a Reg MUE close to 0.5, meaning that the uncertainty estimate is no better at distinguishing true positives from false positives than a random classifier. **Reg NLL** shows BayesOD to better align with the ground truth, scoring ~ 2.5 points higher than both ensemble methods. This means that this method is better at showing high certainty with low variance when it is correct, and correspondingly a lower certainty with high variance when it is incorrect. **Reg ECE** for BayesOD is nearly 5 times smaller than Deep Ensembles. This shows how BayesOD is best at predicting the real probabilities of the bounding box placements.

These results suggest that BayesOD's probabilistic outputs more accurately reflect the underlying data probabilities. A reason for this could be BayesOD's end-to-end probabilistic treatment, where uncertainty information is consistently propagated and utilized throughout the detection pipeline. This contrasts with approaches using NMS. NMS inherently discards considerable information by selecting a single, highest-scoring detection from a group of overlapping candidates. In contrast, BayesOD's Bayesian fusion step integrates data from all relevant anchor outputs, thereby basing the final detection on a comprehensive consideration of all available evidence, which evidently leads to a more probabilistically robust outcome.

E. Qualitative Evaluation

In Figure 2, predictions generated by each method are illustrated using two different input images from the BDD test set. Uncertainty is visualized in two ways: the predicted variance

is represented by circles at the bounding box corners, while classification uncertainty is encoded in the box colour using Shannon entropy, with a gradient from red (low uncertainty) to yellow (high uncertainty).

In the BayesOD predictions in Figures 2a and 2d, we observe that bounding boxes are generally connected with lower uncertainties than the Deep Ensemble techniques in Figures 2b, 2c, 2e and 2f. They share similar proportions but differ mostly in their magnitudes. This likely occurs because the uncertainties are better calibrated to the real probabilities, as also highlighted in the results in Table I[3].

In general, we would expect the object detection models to perform better with objects closer to the camera, as there is a higher resolution and less obstruction. This is also how human perception works. This effect can, to some extent, also be seen in these results, especially as the class uncertainty gets higher, further in the background of the picture. With the bounding box uncertainty, this effect is less clear but still noticeable. The black car in the middle seems to have high bounding box uncertainty with all methods. This could be because it blends more into the shadowy background than the car behind.

F. Methodological Trade-offs and Deployment Constraints

The observed differences in uncertainty quality can be traced to how each method aggregates information. BayesOD performs Bayesian fusion over redundant detections, preserving the full posterior across overlapping anchors, whereas Deep Ensembles apply NMS, which discards much of the variance between model outputs. This structural difference likely contributes to BayesOD's superior regression uncertainty metrics (Table I), despite both methods showing comparable classification calibration [1].

The reason Deep Ensembles somewhat underperform in these metrics could be due to the relatively large size of the dataset (70.000 training samples), also noted by [2]. Therefore, the epistemic uncertainty is generally low. We expected ensembles to perform better than MC Dropout in capturing epistemic uncertainty, based on similar work [5], [6], but if the actual contribution to the final uncertainty is minimal, there is little benefit over MC Dropout. Future work should repeat the comparison under data-scarce settings, e.g. training on a stratified 10 % subset of BDD, to test the hypothesis that ensemble diversity, and thus its epistemic advantage, increases as data coverage diminishes.

In real-world applications like autonomous driving, computational constraints are critical—detection models must operate at high enough frames per second (FPS) to enable timely reactions. Both BayesOD and Deep Ensembles increase inference cost substantially, as they require multiple forward passes through the backbone model. While parallelization can mitigate latency, it comes at the cost of proportional increases in memory and hardware demand. Moreover, BayesOD incurs additional overhead from Bayesian Fusion, which is significantly more computationally intensive than the NMS step used in Deep Ensembles. In [2], BayesOD was shown to reduce FPS by an order of magnitude compared to a single-pass RetinaNet baseline, with typical performance dropping from around 20–30 FPS to 2–3 FPS, depending on the number of stochastic samples used. This performance gap highlights that neither method is a straightforward drop-in replacement for existing real-time detectors. Instead, their adoption involves a trade-off: slower inference in exchange for more principled uncertainty estimates. Whether that trade-off is acceptable depends on the safety and latency requirements of the deployment context.

[3]This trend is observed across the entire test set. The interested reader can refer to the GitHub repository for more samples.

VI. Conclusion

In conclusion, this report has presented a comparative study of two prominent uncertainty estimation methods for deep learning-based object detection: BayesOD and Deep Ensembles. Both approaches modify the standard object detection pipeline to output probabilistic predictions, thereby addressing the critical need for uncertainty awareness in safety-critical applications.

Our analysis highlighted the shared principle of adapting the detection head for probabilistic outputs, while also elucidating the key differences in their approach to estimating epistemic uncertainty and their post-processing steps. BayesOD leverages Monte-Carlo Dropout and Bayesian inference for the fusion of anchor predictions, offering an end-to-end probabilistic treatment. In contrast, Deep Ensembles rely on the aggregation of predictions from multiple independently trained models using Non-Maximum Suppression.

The experimental evaluation on the Berkeley DeepDrive dataset revealed that both BayesOD and Deep Ensembles maintain comparable standard object detection performance to a baseline RetinaNet model. Notably, BayesOD demonstrated superior performance in regression uncertainty metrics. This suggests that its inherent probabilistic nature and Bayesian fusion strategy effectively preserve information that is often lost during the Non-Maximum Suppression step employed by Deep Ensembles.

These findings underscore the potential of both BayesOD and Deep Ensembles for enhancing the reliability of object detection models through uncertainty quantification. Future research could explore hybrid approaches that combine the strengths of both methodologies, as well as investigate their performance on a wider range of datasets and in more complex real-world scenarios.

References

[1] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, and X. X. Zhu, "A survey of uncertainty in deep neural networks," *Artificial Intelligence Review*, vol. 56, no. 1, pp. 1513–1589, Oct. 2023. [Online]. Available: https://doi.org/10.1007/s10462-023-10562-9

[2] D. Feng, A. Harakeh, S. Waslander, and K. Dietmayer, "A Review and Comparative Study on Probabilistic Object Detection in Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 9961–9980, Aug. 2022, arXiv:2011.10671 [cs]. [Online]. Available: http://arxiv.org/abs/2011.10671

[3] A. Harakeh, M. Smart, and S. L. Waslander, "BayesOD: A Bayesian Approach for Uncertainty Estimation in Deep Object Detectors," Sep. 2019, arXiv:1903.03838 [cs]. [Online]. Available: http://arxiv.org/abs/1903.03838

[4] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html

[5] F. K. Gustafsson, M. Danelljan, and T. B. Schon, "Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Seattle, WA, USA: IEEE, Jun. 2020, pp. 1289–1298. [Online]. Available: https://ieeexplore.ieee.org/document/9150658/

[6] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek, "Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift," Dec. 2019, arXiv:1906.02530 [stat]. [Online]. Available: http://arxiv.org/abs/1906.02530

[7] S. H. Yelleni, D. Kumari, S. P.K., and K. M. C., "Monte Carlo DropBlock for modeling uncertainty in object detection," *Pattern Recognition*, vol. 146, p. 110003, Feb. 2024. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S003132032300701X

[8] M. R. Nallapareddy, K. Sirohi, P. L. J. Drews, W. Burgard, C.-H. Cheng, and A. Valada, "EvCenterNet: Uncertainty Estimation for Object Detection Using Evidential Learning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2023, pp. 5699–5706, iSSN: 2153-0866. [Online]. Available: https://ieeexplore.ieee.org/document/10341826

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," May 2016, arXiv:1506.02640 [cs]. [Online]. Available: http://arxiv.org/abs/1506.02640

[10] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," 2017, pp. 2980–2988. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2017/html/Lin_Focal_Loss_for_ICCV_2017_paper.html

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html

[12] A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" Oct. 2017, arXiv:1703.04977 [cs]. [Online]. Available: http://arxiv.org/abs/1703.04977

[13] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," in *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, Jun. 2016, pp. 1050–1059, iSSN: 1938-7228. [Online]. Available: https://proceedings.mlr.press/v48/gal16.html

[14] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning," Apr. 2020, arXiv:1805.04687 [cs]. [Online]. Available: http://arxiv.org/abs/1805.04687

[15] D. Miller, F. Dayoub, M. Milford, and N. Sünderhauf, "Evaluating Merging Strategies for Sampling-based Uncertainty Techniques in Object Detection," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 2348–2354, iSSN: 2577-087X. [Online]. Available: https://ieeexplore.ieee.org/document/8793821/

[16] A. Kumar, P. Liang, and T. Ma, "Verified Uncertainty Calibration," Jan. 2020, arXiv:1909.10155 [cs]. [Online]. Available: http://arxiv.org/abs/1909.10155

[17] V. Mullachery, A. Khera, and A. Husain, "Bayesian Neural Networks," Jan. 2018, arXiv:1801.07710 [cs]. [Online]. Available: http://arxiv.org/abs/1801.07710

[18] D. J. C. MacKay, "Probable networks and plausible predictions-a review of practical Bayesian methods for supervised neural networks," *Network: Computation in Neural Systems*, vol. 6, no. 3, p. 469, Aug. 1995. [Online]. Available: https://dx.doi.org/10.1088/0954-898X/6/3/011

APPENDIX

A. Bayesian Neural Networks

Bayesian Neural Networks (BNNs) represent a powerful combination of standard neural networks and probabilistic modeling principles [17]. While traditional neural networks learn fixed values for their parameters [17], BNNs treat these parameters not as single points, but as probability distributions [17], [18]. This fundamental difference allows BNNs to quantify the uncertainty associated with their predictions [3], [17].

The core of BNNs lies in **Bayes' Theorem** [17], often expressed as:

$$P(\theta|y) = \frac{P(y|\theta)P(\theta)}{P(y)} \tag{21}$$

In the following, each term in Equation (21) is elaborated:

1) **Prior Probability, $P(\theta)$:** This represents our initial beliefs about the distribution of the network's parameters (θ) before observing any data [17].
2) **Likelihood, $P(y|\theta)$:** This describes the probability of observing the data (y) given a specific set of parameters (θ) [17]. In BNNs, the neural network often defines this likelihood function; it computes how well the current parameters explain the observed data [17]. Standard neural networks typically focus only on finding parameters that maximize this likelihood [17].
3) **Posterior Probability, $P(\theta|y)$:** This is the distribution of the parameters (θ) *after* observing the data (y) [17], [18]. It represents our updated beliefs, calculated by combining the prior belief with the information from the data via the likelihood [17]. BNNs aim to infer this entire posterior distribution, not just a single best-fit parameter set [17], [18].
4) **Evidence (or Marginal Likelihood), $P(y)$:** This is the probability of observing the data, averaged over all possible parameter values: $P(y) = \int P(y|\theta)P(\theta)d\theta$ [17], [18]. It acts as a normalization constant in Bayes' Theorem [18].

In essence, BNNs perform inference on the network's parameters [18]. Instead of just finding the single set of parameters that best fits the data, they determine the entire probability distribution of plausible parameters given the data ($P(\theta|y)$) [17], [18].

A key advantage of BNNs is their ability to explicitly model and quantify uncertainty. When making predictions for new data points (x_i), BNNs do not rely on a single set of parameters. Instead, they average the predictions over the entire posterior distribution of parameters. This process, called marginalization [3], results in a **predictive distribution** [3], [18]:

$$P(\hat{y}_i|x_i, \mathcal{D}) = \int P(\hat{y}_i|x_i, \theta)P(\theta|\mathcal{D})d\theta \tag{22}$$

Here, $\mathcal{D}$ represents the training data (x, y). This predictive distribution provides not just a single output value ($\hat{y}_i$) but a probability distribution for the prediction [17]. This inherently incorporates the epistemic uncertainty learned about the parameters ($P(\theta|\mathcal{D})$) into the final prediction [18]. Consequently, BNNs can provide probabilistic guarantees or confidence levels for their predictions [17], unlike standard neural networks, which typically only offer point estimates [17].