Measuring Uncertainty in 3D Object Detection

Luoyue Xu[*], Nan Jiang[*], Yirui Liu[*], Zicheng Xie[*†]

Abstract

3D object detection is a critical task in applications such as autonomous driving and robot navigation, as it provides spatial positions, geometric shapes, and orientations of objects. In this work, we emphasize the importance of measuring uncertainty in 3D object detection models, particularly epistemic uncertainty, which arises from model limitations, and aleatoric uncertainty, which reflects inherent noise in data.

To characterize and quantify these uncertainties, we leverage Mixture Density Networks (MDNs) to model aleatoric uncertainty and Monte Carlo (MC) Dropout to quantify epistemic uncertainty. Additionally, we introduce an uncertainty-aware sample weighting strategy that prioritizes samples with higher predictive uncertainty during training, thereby improving model robustness by dynamically focusing on more challenging samples. Using the KITTI dataset, we evaluated several PointRCNN variants and demonstrated that combining MDNs with MC Dropout, along with the proposed weighting strategy, significantly enhances detection robustness and performance. Our analysis also reveals a strong correlation between model confidence and epistemic uncertainty, highlighting its critical role in decision-making.

Our Contributions:[1] (i) We provide a theoretical analysis of epistemic and aleatoric uncertainties. (ii) We implemented MDNs to quantify aleatoric uncertainty and **MC Dropout** to quantify epistemic uncertainty. (iii) We demonstrated that **combining MDNs with MC Dropout** enhances detection robustness and performance. (iv) We proposed an **uncertainty-aware sample weighting strategy** to improve model training. (v) We **established the relationship** between model uncertainty and confidence scores. (vi) We **visualized** the point cloud data along with the object detection results.

1 Background and Significance

3D object detection is crucial for applications like autonomous driving and robot navigation, where precision and reliability are vital. However, challenges such as sparse data, occlusion, and complex environments can lead to high uncertainty

[*]These authors contributed equally to this work.

[1]The contributions highlighted in **bold** represent methodologies developed entirely by ourselves.

in predictions. Introducing uncertainty quantification allows models to identify high-uncertainty targets, enabling manual inspection to prevent critical errors. Uncertainty-aware models can also focus on challenging samples during training, improving robustness and reducing the need for extensive labeled data. For instance, in autonomous driving, uncertainty estimation could have helped avoid a fatal accident caused by misidentifying the white side of a trailer as the sky, emphasizing its importance in safety-critical scenarios. [1]

2 Current Research Status on Uncertainty Measurement

2.1 Introduction to Uncertainty

In machine learning, uncertainty in models can be broadly categorized into two types: **Epistemic Uncertainty (model uncertainty, also called EPS)** and **Aleatoric Uncertainty (data uncertainty, also called ALS)**.

2.1.1 Epistemic Uncertainty

Epistemic uncertainty arises from a lack of knowledge or understanding about the model or the problem. It occurs when model parameters are poorly determined due to insufficient or incomplete data. This type of uncertainty can be reduced by collecting more data, as it reflects the model's ignorance of certain regions in the input space. Mathematically, epistemic uncertainty can be expressed as:

$$\mathcal{U}(x) = \mathrm{Var}(P(y \mid x, \theta))$$

where x is the input, y is the predicted output, θ represents the model parameters, and $P(y \mid x, \theta)$ is the predictive distribution of y given x and θ.

2.1.2 Aleatoric Uncertainty

Aleatoric uncertainty comes from inherent noise or stochasticity in the data. It reflects the randomness inherent in the process being modeled, such as measurement noise or natural variability. Unlike epistemic uncertainty, aleatoric uncertainty cannot be reduced by collecting more data, as it is intrinsic to the data itself. Aleatoric uncertainty is commonly measured using the entropy of the predictive distribution, defined as:

$$H[P(y \mid x)] = -\sum_{y \in Y} P(y \mid x) \log P(y \mid x)$$

where $P(y \mid x)$ represents the probability distribution over the possible outcomes y for a given input x, and Y is the set of all possible outcomes.

2.2 Methods for Uncertainty Quantification

Uncertainty quantification methods are mainly categorized into three approaches: Epistemic Uncertainty Quantification, Aleatoric Uncertainty Quantification, and Combined Methods, with supplementary information-theoretic measures. These methods are crucial for minimizing uncertainty impacts during optimization and decision-making, with applications across computer vision, medical imaging, and natural language processing.

2.2.1 Epistemic Uncertainty Quantification

Epistemic uncertainty quantification focuses on capturing the model's ignorance about the data due to insufficient knowledge.

- **Bayesian Methods**: Methods such as Monte Carlo (MC) Dropout approximate Bayesian inference by generating multiple samples during inference, providing efficient uncertainty estimation for tasks like autonomous driving [5]. Markov Chain Monte Carlo (MCMC) and Variational Inference (VI) are also widely used to approximate posterior distributions [6,8].
- **Ensemble Techniques**: Deep Ensembles train multiple independent models and aggregate their predictions to reduce model variance and improve robustness [9].

Method Type	Method Name	Description
Bayesian Techniques	**1. Monte Carlo (MC) Dropout**	Uses dropout during inference to approximate Bayesian inference and estimate uncertainty in predictions.
	2. Markov Chain Monte Carlo (MCMC)	Uses a Markov chain to generate samples, gradually approaching the posterior distribution, suitable for complex inference but computationally expensive.
	3. Variational Inference (VI)	Transforms Bayesian inference into an optimization problem, approximating the posterior distribution by optimizing the variational lower bound.
	4. Bayesian Active Learning (BAL)	Combines Bayesian methods with active learning, selecting the most informative samples based on uncertainty measures to improve learning efficiency.
	5. Bayes By Backprop (BBB)	Learns the distribution of Bayesian neural network weights through backpropagation, optimizing the network weights to quantify uncertainty in predictions.
Ensemble Techniques	**1. Deep Ensemble**	Trains multiple independent neural networks and combines their predictions to improve robustness and accuracy.
	2. Deep Ensemble Bayesian/Bayesian Deep Ensemble	Combines Bayesian inference with ensemble learning, integrating multiple Bayesian neural networks to improve uncertainty quantification.
	3. Uncertainty in Dirichlet Deep Networks	Models uncertainty in the model's output using a Dirichlet distribution, commonly used for multi-class classification tasks to quantify uncertainty in category predictions.

Figure 1: Uncertainty Quantification Methods.

2.2.2 Aleatoric Uncertainty Quantification

Aleatoric uncertainty quantification captures the inherent randomness in data.

- **Mixture Density Networks (MDNs)**: MDNs model the conditional probability distribution of outputs (e.g., Gaussian mixtures), making them ideal for tasks with noisy or multimodal data [10].
- **Heteroscedastic Noise Models**: These models predict both the mean and variance of outputs to explicitly capture data noise.

2.2.3 Combined Methods

Combined methods capture both epistemic and aleatoric uncertainties to provide a comprehensive quantification.

- **Bayesian Neural Networks (BNNs)**: BNNs integrate Bayesian principles into neural networks, modeling both parameter and output uncertainties.
- **Deep Ensembles with Uncertainty Estimation**: Ensemble techniques, when combined with output variance estimation, can quantify both types of uncertainty.

3 Theoretical Analysis

3.1 Measuring Data Uncertainty: Mixture Density Networks (MDNs)

3.1.1 Introduction to MDNs

To deal with the uncertainty of data, we introduce MDN to our model.

The idea of MDN is trying to build a model to let the output of neutral network be a distribution rather than some number. In this way, the uncertainty of data can be measured by the property of the output distribution.

$$P(t|x) = \sum_{k=1}^{K} \pi_k(x) D_k(t|\theta_k(x)) \tag{1}$$

where π_k is a weighting factor and D_k are probability density functions.

In our model, we choose the output distribution the Gaussian distribution. Then,

$$P(t|x) = \sum_{k=1}^{K} \pi_k(x) N_k(t|\mu_k(x), \sigma_k^2(x) I) \tag{2}$$

The structure of MDN in our model is like the picture below.

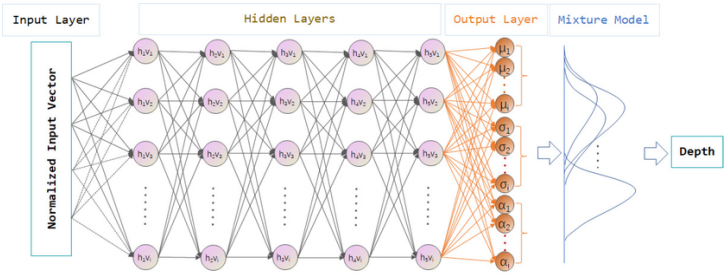


Figure 2: Net Structure of MDN.

For the weighting vector π_k, we can let it follow the form of softmax:

$$\pi_i(x) = \frac{\exp(a_{\pi_i})}{\sum_{k=1}^{K} \exp(a_{\pi_k})} \tag{3}$$

With the output distributions we have, we can finally give the output by the maximal likelihood estimation. The likelihood function is defined by:

$$L(t|\mu, \sigma^2) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k(x_n) N_{nk}(t_n|\mu_k(x_n), \sigma_k^2(x_n) I) \tag{4}$$

and the loss function is defined by:

$$E(w) = -\log L(w) = -\sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k N_{nk}(t_n|\mu_k(x_n, w), \sigma_k^2(x_n, w) I) \right), \tag{5}$$

Backward Propagation By simply calculating gradients we need, we can update the parameters by gradients. The detailed formulas are as follows:
Partial derivative of error function with respect to π_j:

$$\frac{\partial E_n}{\partial a_{\pi_j}} = \frac{-1}{\sum_{k=1}^{K} \pi_k N_{nk}} \sum_{k=1}^{K} \left(N_{nk} \frac{\partial \pi_k}{\partial a_{\pi_j}} \right) \tag{6}$$

$$\frac{\partial \pi_k}{\partial a_{\pi_j}} = \begin{cases} \pi_j - \pi_k \pi_j & \text{for } j = k, \\ -\pi_k \pi_j & \text{for } j \neq k. \end{cases} \tag{7}$$

Partial derivative of error function with respect to σ_j:

$$\frac{\partial E_n}{\partial a_{\sigma_j}} = \frac{-1}{\sum_{k=1}^{K} \pi_k N_{nk}} \sum_{k=1}^{K} \left(\pi_k \frac{\partial N_{nj}}{\partial a_{\sigma_j}} \right) \tag{8}$$

$$\frac{\partial N_{nj}}{\partial a_{\sigma_j}} = N_{nj} \left(\frac{-L}{\sigma_j} + \frac{\|t_n - \mu_j\|^2}{\sigma_j^3} \right) \tag{9}$$

Partial derivative of error function with respect to μ_j:

$$\frac{\partial E_n}{\partial a_{\mu_j}} = \frac{-1}{\sum_{k=1}^{K} \pi_k N_{nk}} \sum_{k=1}^{K} \left(\pi_j N_{nj} \frac{\partial N_{nj}}{\partial a_{\mu_j}} \right) \tag{10}$$

$$\frac{\partial N_{nj}}{\partial a_{\mu_j}} = \frac{N_{nj}}{\sigma^2} \|x_n - \mu_j\|. \tag{11}$$

given the formulas of gradients,we can easily update the parameters by gradient descent with a given learning rate. In this way, the model can be trained for better performance after enough iterations.

3.1.2 ALS for Measuring data uncertainty

To measure the uncertainty comes from noise of the data and the possibility of data coming from multi-models. We introduce aleatoric uncertainty(ALS). ALS can be measured by predictive entropy which is of the following formula:

$$H[P(y|x)] = -\sum_{y \in Y} P(y|x) \log P(y|x) \tag{12}$$

In MDNs case, the formula is particularly written as:

$$\text{ALS} = \sum_{k=1}^{K} \pi_k(x) \sigma_k^2(x) \tag{13}$$

So, given an input vector x, we can compute ALS to measure the corresponding data uncertainty.

3.2 Measuring Model Uncertainty: Bayesian Neural Networks (BNNs)

Traditional neural networks use point estimation methods like Maximum Likelihood Estimation (MLE) or Maximum A Posteriori (MAP) to optimize parameters.

Maximum Likelihood Estimation (MLE): MLE optimizes the likelihood function:

$$\mathcal{L}(\theta; D) = \prod_{i=1}^{N} p(y_i|x_i, \theta),$$

and the solution is:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \sum_{i=1}^{N} \log p(y_i|x_i, \theta).$$

Maximum A Posteriori (MAP): MAP incorporates a prior distribution $p(\theta)$ and maximizes the posterior distribution:

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} \left[\sum_{i=1}^{N} \log p(y_i|x_i, \theta) + \log p(\theta) \right].$$

Both methods yield deterministic point estimates, which fail to capture prediction uncertainty.

Instead of a point estimate, BNNs approximate the posterior $p(\theta|D)$, capturing parameter uncertainty.

Predictive Distribution: The predictive distribution for a new input x^* is:

$$p(y^*|x^*, D) = \int p(y^*|x^*, \theta) p(\theta|D)\, d\theta,$$

which incorporates uncertainty in θ when making predictions.

3.2.1 Variational Inference for Bayesian Neural Networks

Due to the intractability of the true posterior $p(\theta|D)$, variational inference approximates it with a simpler distribution $q(\theta)$, such as a Gaussian. The goal is to minimize the Kullback-Leibler (KL) divergence between $q(\theta)$ and $p(\theta|D)$.

Optimization Objective: The KL divergence is:

$$D_{\text{KL}}(q(\theta) \| p(\theta|D)) = \int q(\theta) \log \frac{q(\theta)}{p(\theta|D)} d\theta.$$

Expanding using Bayes' theorem:

$$D_{\text{KL}}(q(\theta) \| p(\theta|D)) = \int q(\theta) \log \frac{q(\theta)}{p(D|\theta)p(\theta)} d\theta + \log p(D).$$

The variational objective becomes:

$$\mathcal{F}(D, q(\theta)) = D_{\text{KL}}(q(\theta) \| p(\theta)) - \mathbb{E}_{q(\theta)}[\log p(D|\theta)],$$

where: - $D_{\text{KL}}(q(\theta) \| p(\theta))$ regularizes $q(\theta)$ to stay close to the prior $p(\theta)$. - $\mathbb{E}_{q(\theta)}[\log p(D|\theta)]$ ensures the model fits the data.

3.2.2 MC Dropout as Variational Inference

Monte Carlo (MC) Dropout provides a practical method for approximate Bayesian inference by using Dropout to define a Bernoulli variational distribution over network parameters.

Bernoulli Variational Distribution: In MC Dropout, the variational distribution $q(\theta)$ is implicitly defined by Dropout:

$$q(\theta) = \prod_{i,j} \text{Bernoulli}(z_{i,j}; p),$$

where: - $z_{i,j} \sim \text{Bernoulli}(p)$ is a binary mask applied to the weights or activations. - The sampled weights are:

$$\tilde{\theta}_{i,j} = z_{i,j} \cdot \theta_{i,j}.$$

This defines a discrete two-point distribution over the parameters.

Predictive Distribution: The predictive distribution for a new input x^* is approximated using T stochastic forward passes:

$$p(y^*|x^*, D) \approx \frac{1}{T} \sum_{t=1}^{T} p(y^*|x^*, \tilde{\theta}_t),$$

where $\tilde{\theta}_t$ is sampled by applying Dropout during each forward pass.

Predictive Mean (Point Prediction): The mean prediction is estimated as the average of T stochastic forward passes:

$$\hat{y}^* = \frac{1}{T} \sum_{t=1}^{T} p(y^*|x^*, \tilde{\theta}_t).$$

Predictive Variance (Uncertainty): The uncertainty of the prediction is captured by the variance of the predictions across T forward passes:

$$\text{Var}(y^*) = \frac{1}{T} \sum_{t=1}^{T} \left(p(y^*|x^*, \tilde{\theta}_t) - \hat{y}^* \right)^2.$$

4 Experiments

4.1 Experimental Setup

We evaluated four variations of the PointRCNN 3D object detection model on the KITTI validation set, which includes 3,769 annotated samples across three object classes: *Car*, *Pedestrian*, and *Cyclist*. Each sample was preprocessed to include a maximum of 16,384 points, with points outside the detection range of $[0, -40, -3]$ to $[70.4, 40, 1]$ being removed. Point shuffling was applied during training to enhance generalization but disabled during testing.

The data augmentation pipeline incorporated techniques such as ground truth sampling, random world flips along the x-axis, random rotations within

$[-45°, 45°]$, and scaling within the range $[0.95, 1.05]$. The PointRCNN model utilized the PointNet++ architecture as the 3D backbone with a multi-scale grouping (MSG) strategy to extract features at varying spatial resolutions. Our evaluated models are as follows:

- **Vanilla PointRCNN:** The baseline PointRCNN model without any modifications.
- **PointRCNN-MDN:** This version incorporates a Mixture Density Network (MDN) into the ROI head, using three Gaussian distributions to explicitly model output uncertainty.
- **PointRCNN with Monte Carlo (MC) Dropout (Ours):** In this variant, a dropout rate of 0.1 is applied to the fully connected layers of the ROI head's regression and classification branches during training to enhance generalization. During evaluation, dropout remains enabled, and the model is evaluated 30 times to quantify predictive uncertainty.
- **PointRCNN-MDN with Monte Carlo (MC) Dropout (Ours):** This model combines the MDN architecture with MC Dropout, enabling both Gaussian-based uncertainty modeling and stochastic sampling during evaluation.

Loss functions were weighted to balance regression and classification tasks; for the MDN model, a specialized MDN loss handled box regression uncertainty. Post-processing for all models included Non-Maximum Suppression (NMS) during training and testing to eliminate redundant detections.

All models were trained for 50 epochs on an NVIDIA L40 GPU with a batch size of 2 using the Adam optimizer with an initial learning rate of 0.01. The learning rate decayed at epochs 35 and 45. Weight decay of 0.01 and gradient clipping with a maximum norm of 10 were applied to stabilize training.

Each model was evaluated based on 3D Average Precision (AP) and recall metrics (*recall_roi* and *recall_rcnn*) at IoU thresholds of 0.3, 0.5, and 0.7 for the three object classes.

4.2 Analysis of Training Loss Curves

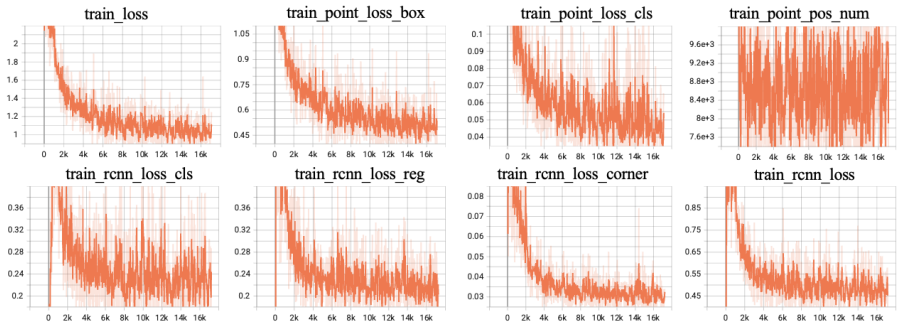


Figure 3: Loss Curves of Vanilla Pointrcnn Model.

The training loss curves for the Vanilla PointRCNN are demonstrated in Figure 3. The overall training loss and sub-losses (e.g., `train_loss_box`, `train_loss_cls`, `train_rcnn_loss`) steadily decrease, indicating smooth convergence during training.

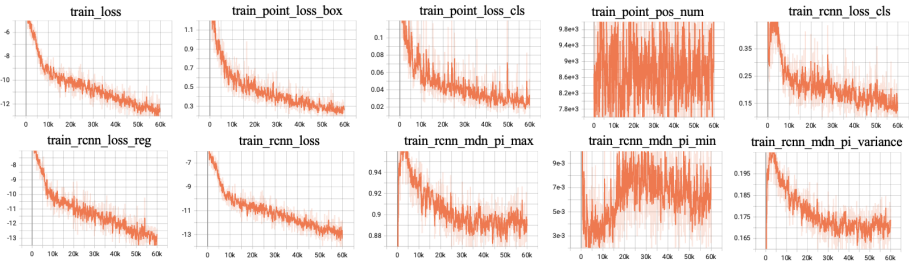


Figure 4: Loss Curves of Pointrcnn-MDN Model.

The training loss curve for the PointRCNN-MDN shows a steeper initial decrease compared to the Vanilla PointRCNN, likely due to the Mixture Density Network (MDN) components, which introduce additional probabilistic modeling of outputs.

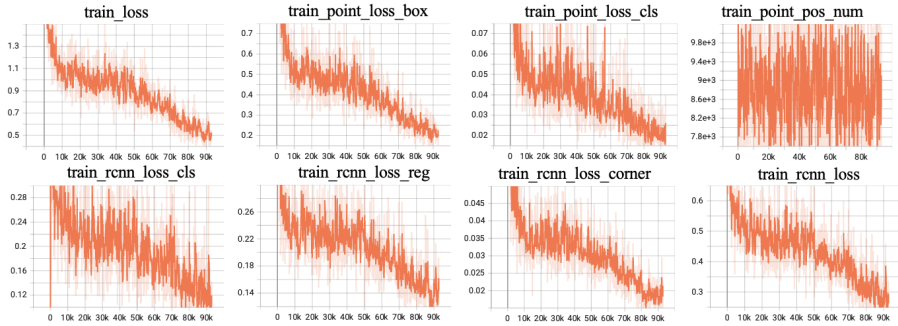


Figure 5: Loss Curves of Pointrcnn with MC-Dropout.

PointRCNN with MC-Dropout exhibits a similar overall training loss trend to the Vanilla PointRCNN, but with slightly higher fluctuations in intermediate losses such as `train_rcnn_loss_cls` and `train_rcnn_loss_reg`. These fluctuations are likely caused by the stochastic nature of the dropout mechanism during training.

The lower variance observed in `train_loss_box` and `train_loss_cls` compared to other loss components suggests that the dropout mechanism primarily influences the deeper layers of the network, where more complex features are learned.

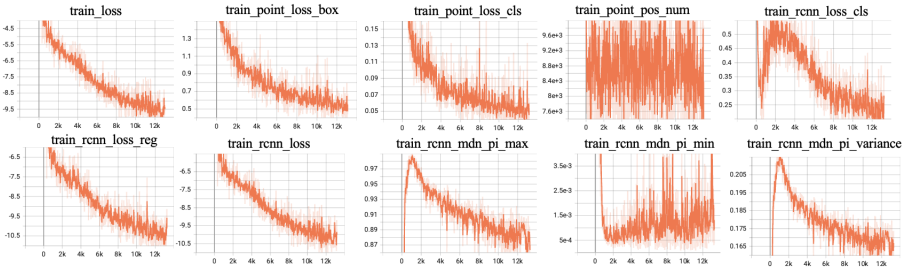
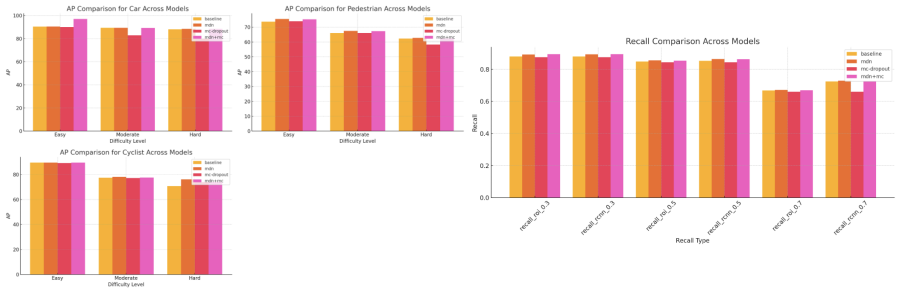


Figure 6: Loss Curves of PointRCNN-MDN with MC-Dropout.

Figure 6 presents the training loss curves of the PointRCNN-MDN model with MC-Dropout across various loss components.

Over time, the metrics exhibit stabilization, with initial oscillations gradually smoothing out. This behavior suggests that the MDN components, in combination with the dropout mechanism, effectively adapt to the training data.

4.3 Performance Evaluation



(a) (b)

Figure 7: (a)Average Precision (AP) for Different Classes. (b)Recall of Objects of Different Models.

Figure 7 demonstrate the Average Precision (AP) and Recall for different models. PointRCNN-MDN+MC demonstrates the highest recall across all thresholds, indicating enhanced detection robustness. For instance, `recall_rcnn_0.7` is highest for PointRCNN-MDN+MC at 0.729, compared to 0.725 for PointRCNN-MDN, 0.659 for PointRCNN-Dropout, and 0.724 for PointRCNN-Baseline.

For cars, PointRCNN-MDN+MC consistently achieves better AP in the *easy* and *moderate* categories compared to PointRCNN with Dropout and Baseline. However, for *hard* cases, the improvement is marginal over PointRCNN-MDN. For pedestrians, the trend is similar, with PointRCNN-MDN+MC slightly outperforming others in the *easy* and *moderate* categories, but the differences narrow for *hard* cases. For cyclists, the performance is slightly more varied; PointRCNN-MDN+MC leads in the *easy* and *moderate* scenarios, while PointRCNN-MDN shows competitive results for the *hard* category. Meanwhile, PointRCNN-Dropout lags across most cases, particularly for *moderate* and *hard* difficulties.

Overall, PointRCNN-MDN+MC demonstrates robust performance across recall and AP metrics, particularly excelling in scenarios with *easier* and *moderate* difficulties.

We evaluate the performance of PointRCNN-MDN with MC-Dropout under different dropout ratios. The analysis reveals that as the dropout ratio increases, the recall metrics for both ROI and RCNN stages exhibit slight improvements, particularly at stricter IoU thresholds like 0.7.

4.4 Enhancing Model Training with Uncertainty-Based Sample Weighting

To improve the training process of our model, we incorporated an **uncertainty-aware sample weighting** mechanism. Specifically, we utilized the `rcnn_al`

metric, which represents the aleatoric uncertainty of each sample, to dynamically adjust the training loss. The primary goal of this modification is to **encourage the model to focus more on samples with higher uncertainty**, which are often more challenging and informative.

During training, we modified the `forward` function to calculate the `rcnn_al` metric for each sample and store it in `forward_ret_dict`. In the `get_box_reg_layer_loss` function, we applied **sample-specific weighting** by multiplying the regression loss with a factor proportional to the sample's uncertainty:

$$\text{weighted\_loss} = \text{loss} \times (1 + \alpha \times \text{rcnn\_al})$$

where α is a hyperparameter controlling the influence of the uncertainty. We also applied clamping to prevent numerical instability due to overly large uncertainty values.

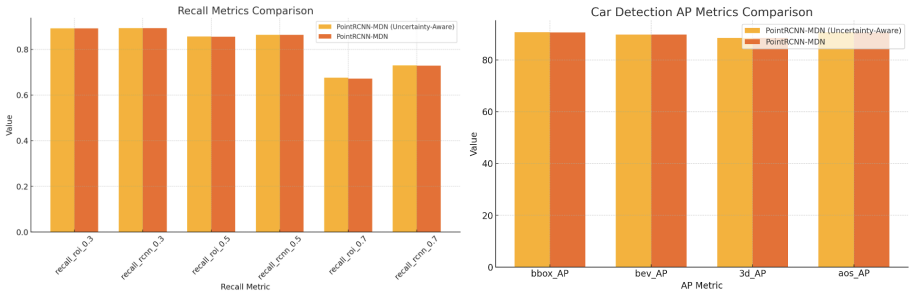


Figure 8: Performance of PointRCNN-MDN with and without uncertainty-aware sample weighting.

Figure 8 compares the performance of PointRCNN-MDN with uncertainty-aware sample weighting and PointRCNN-MDN without uncertainty-aware sample weighting. The inclusion of uncertainty-aware sample weighting provides a slight improvement in performance, particularly for high IoU recall and car detection in 3D AP metrics. However, the performance gap between the two models is not substantial.

4.5 Statistical Analysis

4.5.1 Bivariate Analysis

Prediction Scores The variable `pred_scores` is the final box scores obtained after processing the classification scores through Non-Maximum Suppression (NMS) in the object detection task. It represents the classification confidence for each retained box. `Pred_scores` can be considered as an indirect validation of the actual meaning of the uncertainty we estimate.

The variables `pred_cls_var` and `pred_reg_var` represent the model uncertainty for classification and regression tasks in the Point_RCNN_mc model, re-

spectively, while `als` denotes the data uncertainty in the Point_RCNN_mdn model. Scatter plots were created to compare `pred_cls_var`, `pred_reg_var`, and `als` with `pred_scores`.

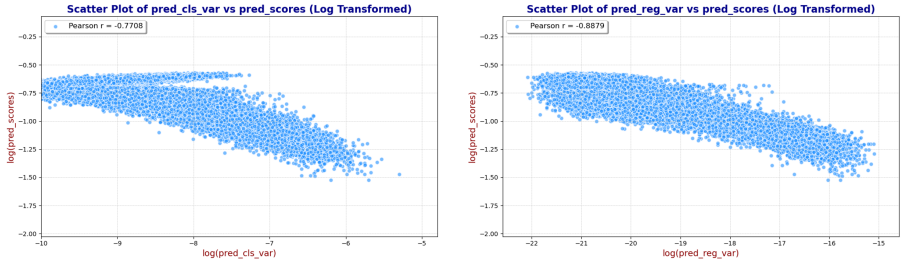


Figure 9: Correlation between model uncertainty (`pred_cls_var`, `pred_reg_var`) and `pred_scores` in the Point_RCNN_mc model.

Figures 9 show that the model uncertainty computed in the Point_RCNN_mc model has a strong negative correlation with the `pred_scores` in the baseline model, with correlation coefficients (r) of -0.77 for the classification task and -0.89 for the regression task. This indirectly validates that the computed `pred_cls_var` and `pred_reg_var` represent meaningful model uncertainty.

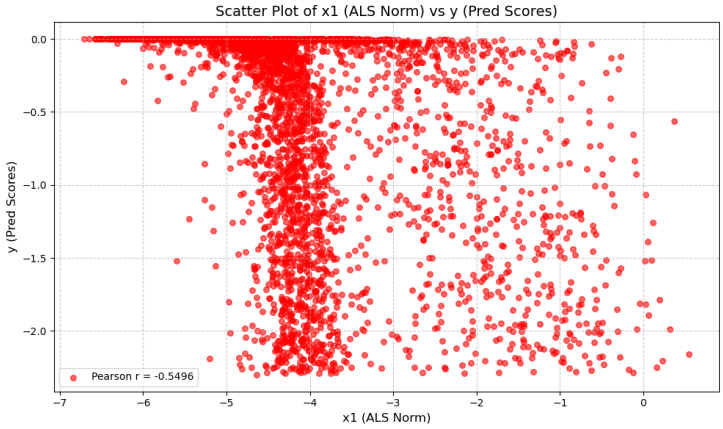


Figure 10: Correlation between `als` and `pred_scores` in the Point_RCNN_mdn model.

Figure 10 indicates that the data uncertainty (`als`) computed in the Point_RCNN_mdn model has a weak correlation with the `pred_scores` from the baseline model. This indirectly confirms that we have successfully separated the model uncertainty from the data uncertainty.

4.5.2 Visualization

Figure 11: Real-world Scenario.

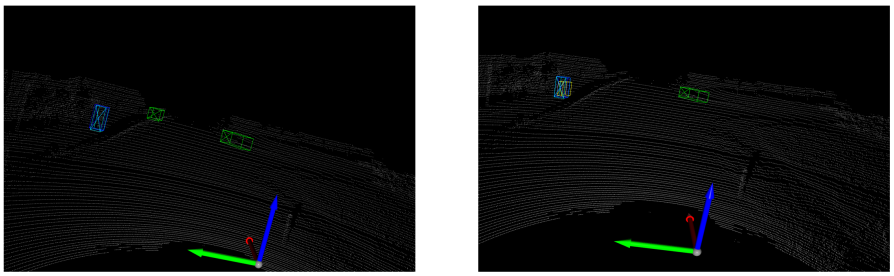


Figure 12: Comparison of prediction results between the Point_RCNN and Point_RCNN_mdn models.

Figure 11 shows a real-world scenario, while Figures 12 show the prediction results of Point_RCNN and Point_RCNN_mdn, respectively. In these images, the light blue boxes represent pedestrians, yellow boxes represent bicycles, and green boxes represent cars.

The Point_RCNN model incorrectly identifies a road sign as a bicycle and fails to detect the vehicle at the far left. In contrast, our trained MDN model makes the correct predictions in line with the actual situation. This demonstrates that **our model outperforms the baseline model in specific cases**, and an examination of the uncertainty in the predicted bounding boxes reveals that it is lower than the average uncertainty.

5 Conclusion and Future Work

Conclusion In this study, we modeled both aleatoric (data) and epistemic (model) uncertainties by introducing the MDN and MC-Dropout mechanisms. This approach significantly improved detection performance through the integration of the PointRCNN-MDN+MC fusion model. Additionally, we introduced an uncertainty-aware sample weighting strategy to prioritize samples with higher predictive uncertainty during training. We also observed a notable

linear relationship between model uncertainty and model confidence. Overall, the combination of uncertainty modeling and sample weighting proves to be a promising approach to improving 3D object detection performance.

Future Work Building upon these results, future work can focus on identifying decision boundaries by leveraging both model and data uncertainty for real-world applications. Additionally, we aim to further enhance model performance by optimizing the uncertainty-aware sample weighting strategy introduced in this study. By more effectively balancing the trade-off between challenging and easy samples during training, we hope to further improve model robustness and generalization, particularly in complex and diverse environments.

Teamwork and Contributions

The team members come from three distinct academic disciplines, each bringing unique expertise to the project. Based on their specializations, the division of labor is detailed in Table 1.

Member	**Major**	**Primary Responsibilities**	**Contribution Ratio**
Luoyue Xu	Statistics	Conducted theoretical analysis of MC-dropout and report writing.	25%
Nan Jiang	Mathematics	Focused on theoretical analysis of Mixture Density Networks (MDNs) and report writing.	25%
Yirui Liu	Statistics	Performed statistical analysis of experimental results and report writing.	25%
Zicheng Xie[†]	Data Science	Served as project lead, managed model training and optimization, and contributed to report writing.	25%

Table 1: Division of labor and contribution ratios among team members.

References

[1] Kendall, A., Gal, Y. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? Retrieved from https://arxiv.org/abs/1703.04977

[2] Shi, S., Wang, X., et al. (2019). PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud *Computer Vision and Pattern Recognition (cs.CV)* https://doi.org/10.48550/arXiv.1812.04244

[3] Geiger, Andreas, Philipp Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2012.

[4] Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V., & Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications, and challenges. *Information Fusion*, 76, 243-297. https://doi.org/10.1016/j.inffus.2021.05.008

[5] Brach, K., Sick, B., & Dürr, O. (2020). Single shot MC dropout approximation. *arXiv preprint arXiv:2007.03293.*

[6] Kupinski, M.A., Hoppin, J.W., Clarkson, E., & Barrett, H.H. (2003). Ideal-observer computation in medical imaging with use of Markov-chain Monte Carlo techniques. *J. Opt. Soc. Amer. A*, 20(3), 430-438.

[7] Gal, Y., Islam, R., & Ghahramani, Z. (2017). Deep Bayesian active learning with image data. Proceedings of the 34th International Conference on Machine Learning-Vol. 70, JMLR.org, 1183-1192.

[8] Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424.*

[9] Zhang, J., Kailkhura, B., & Han, T.Y.-J. (2020). Mix-n-match: Ensemble and compositional methods for uncertainty calibration in deep learning. *International Conference on Machine Learning*, PMLR, 11117-11128.

[10] Tsiligkaridis, T. (2021). Information aware max-norm Dirichlet networks for predictive uncertainty estimation. *Neural Networks*, 135, 105-114.

[11] Zeng, F., Zhang, W., Li, J., Zhang, T., & Yan, C. (2022). Adaptive Model Refinement Approach for Bayesian Uncertainty Quantification in Turbulence Model. *AIAA Journal*. https://doi.org/10.2514/1.J060889

[12] Fiedler, F., & Lucia, S. (2023). Improved Uncertainty Quantification for Neural Networks With Bayesian Last Layer. *IEEE Access*, 11, 123149-123160. https://doi.org/10.1109/ACCESS.2023.3329685

[13] Hernández, S., & Lopez, J. (2020). Uncertainty quantification for plant disease detection using Bayesian deep learning. *Applied Soft Computing*, 96, 106597. https://doi.org/10.1016/j.asoc.2020.106597

[14] Caldeira, J., & Nord, B. (2020). Deeply uncertain: comparing methods of uncertainty quantification in deep learning algorithms. *Machine Learning: Science and Technology*, 2. https://doi.org/10.1088/2632-2153/aba6f3

[15] Janjić, T., Lukácová-Medvidová, M., Ruckstuhl, Y., & Wiebe, B. (2023). Comparison of uncertainty quantification methods for cloud simulation. *Quarterly Journal of the Royal Meteorological Society*, 149, 2895-2910. https://doi.org/10.1002/qj.4537

[16] Yang, S., & Yee, K. (2023). Towards reliable uncertainty quantification via deep ensemble in multi-output regression task. *Engineering Applications of Artificial Intelligence*, 132, 107871. https://doi.org/10.1016/j.engappai.2024.107871