Appendix of "Laplace Approximation Based Epistemic Uncertainty Estimation in 3D Object Detection"

Peng Yun *
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
pyun@cse.ust.hk

Ming Liu †
Hong Kong University of Science and Technology (Guangzhou)
eelium@ust.hk

Abstract: This appendix contains the derivation of Laplace approximation, implementation details, and more experiment results of "Laplace Approximation Based Epistemic Uncertainty Estimation in 3D Object Detection".

1 Derivation of Laplace approximation

Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)|i = 1, ..., N\} = \{\mathbf{X}, \mathbf{Y}\}$, we have a neural network $f_\theta(\mathbf{x})$ with its prior weight distribution $p(\theta) \sim \mathcal{N}(\theta; \mathbf{0}, \Sigma_0)$. We want to derive the posterior weight distribution $p(\theta|\mathbf{X}, \mathbf{Y})$, which explains the dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$, so that we can conduct inference and get the predictive distribution with

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \theta)p(\theta|\mathbf{X}, \mathbf{Y})d\theta, \tag{1}$$

where the $p(\mathbf{y}^*|\mathbf{x}^*, \theta)$ depicts the predictive model. For classification tasks, it is commonly defined as a categorical distribution with

$$p(\mathbf{y}^c = 1|\mathbf{x}, \theta) = \frac{\exp(f_\theta^c(\mathbf{x}))}{\sum_{c'} \exp(f_\theta^{c'}(\mathbf{x}))} = \text{softmax}^c(f_\theta(\mathbf{x})). \tag{2}$$

For regression tasks, it can be defined as

$$p(\mathbf{y}|\mathbf{x}, \theta) = \mathcal{N}(\mathbf{y}; f_\theta(\mathbf{x}), \tau^{-1}\mathbf{I}). \tag{3}$$

The τ can be either a hyper-parameter or an estimation from data. We can estimate $p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})$ with the Monte-Carlo estimator by calculating the population moments as [1]. In our manuscript, we denote this step as moment estimation. In classification,

$$p(\mathbf{y}^{*,c} = 1|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) \approx \frac{1}{T} \sum_{\theta \in P(\theta|\mathbf{X}, \mathbf{Y})} \text{softmax}^c(f_\theta(\mathbf{x}^*)). \tag{4}$$

The mode can be estimated by $c^* = \arg\max_c p(\mathbf{y}^{*,c} = 1|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})$, and the uncertainty can thus be quantified in terms of $p(\mathbf{y}^{*,c^*} = 1|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})$.

In regression, we first consider the predictive distribution mean $\mathbb{E}_{P(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})}[\mathbf{y}^*]$:

$$\begin{aligned}
\mathbb{E}_{P(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})}[\mathbf{y}^*] &= \int \mathbf{y}^* p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})d\mathbf{y}^* && (5)\\
&= \int \int \mathbf{y}^* p(\mathbf{y}^*|\mathbf{x}^*, \theta)p(\theta|\mathbf{X}, \mathbf{Y})d\theta d\mathbf{y}^* && (6)\\
&= \int \int \mathbf{y}^* \mathcal{N}(\mathbf{y}^*; f_\theta(\mathbf{x}^*), \tau^{-1}\mathbf{I})d\mathbf{y}^* p(\theta|\mathbf{X}, \mathbf{Y})d\theta && (7)\\
&= \int f_\theta(\mathbf{x}^*)p(\theta|\mathbf{X}, \mathbf{Y})d\theta && (8)
\end{aligned}$$

*Peng Yun is also with Clear Water Bay Institute of Autonomous Driving, Nanshan, Shenzhen.

†Ming Liu (corresponding author) is also with Hong Kong University of Science and Technology, Hong Kong SAR, China and HKUST Shenzhen-HongKong Collaborative Innovation Research Institute, Futian, Shenzhen.

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

Even though the integration over the posterior weight distribution is complicated to compute, we can estimate it with a Monte-Carlo estimator and have

$$\mathbb{E}_{P(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y})}[\mathbf{y}^*] = \frac{1}{T}\sum_{\theta\in p(\theta|\mathbf{X},\mathbf{Y})} f_\theta(\mathbf{x}^*). \tag{9}$$

Then before computing the covariance, we derive the second-order moment $\mathbb{E}_{P(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y})}[\mathbf{y}^{*T}\mathbf{y}^*]$:

$$\mathbb{E}_{P(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y})}[\mathbf{y}^{*T}\mathbf{y}^*] = \int \mathbf{y}^{*T}\mathbf{y}^* p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y})d\mathbf{y}^* \tag{10}$$

$$= \int\int \mathbf{y}^{*T}\mathbf{y}^* p(\mathbf{y}^*|\mathbf{x}^*,\theta)p(\theta|\mathbf{X},\mathbf{Y})d\theta d\mathbf{y}^* \tag{11}$$

$$= \int [\tau^{-1} + f_\theta^T(\mathbf{x}^*)f_\theta(\mathbf{x})]p(\theta|\mathbf{X},\mathbf{Y})d\theta \tag{12}$$

$$\tag{13}$$

To evaluate (12), we can seek help from Monte-Carlo estimator as before and have

$$\mathbb{E}_{P(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y})}[\mathbf{y}^{*T}\mathbf{y}^*] = \frac{1}{T}\sum_{\theta\in p(\theta|\mathbf{X},\mathbf{Y})} f_\theta^T(\mathbf{x}^*)f_\theta(\mathbf{x}) + \tau^{-1}\mathbf{I} \tag{14}$$

By substituting the 1-st and 2-nd moments, we have the covariance of the predictive distribution as We can get the covariance matrix of the posterior distribution by substituting (9) and (14) into

$$\text{Cov}[\mathbf{y}^*] = \mathbb{E}_{P(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y})}[\mathbf{y}^{*T}\mathbf{y}^*] - \mathbb{E}_{P(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y})}[\mathbf{y}^*]^T\mathbb{E}_{P(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y})}[\mathbf{y}^*]. \tag{15}$$

As a result, if we derive the posterior weight distribution $p(\theta|\mathbf{X},\mathbf{Y})$, we can conduct Bayesian inference and get the predictive distribution with Monte-Carlo estimators. In the following paragraphs, we will detail the Laplace approximation method to evaluate this posterior distribution.

Laplace Approximation in neural networks The posterior weight distribution can be unfolded with the Bayes formula:

$$\log p(\theta|\mathbf{X},\mathbf{Y}) = \log p(\mathbf{Y}|\mathbf{X},\theta) + \log p(\theta) - \log p(\mathbf{Y}|\mathbf{X}), \tag{16}$$

where the last term $-\log p(\mathbf{Y}|\mathbf{X})$ is independent to θ. We assume that the loss is given by the negative log probability associated with a predictive distribution $P_{\mathbf{Y}|f(\mathbf{X},\theta)}$, as well as a regularizer, i.e.

$$L(\theta,\mathbf{X},Y) = -\log p(\mathbf{Y}|f(\mathbf{X},\theta)) + \lambda||\theta||, \tag{17}$$

where p is $P(\mathbf{Y}|f(\mathbf{X},\theta))$'s density function, and $||\cdot||$ can be a L1 or L2 norm[3]. This is the case for standard least-squared and cross-entropy objective functions, where the predictive distributions are multivariate normal and multinomial, respectively. Minimizing this loss function $L(\theta,\mathbf{X},\mathbf{Y})$ can be seen as maximizing the posterior $\log(\theta|\mathbf{X},\mathbf{Y})$. We denote the optimal weight point as θ^*.

Laplace approximation unfolds $\log p(\theta|\mathbf{X},\mathbf{Y})$ with the second-order Taylor expansion at θ^*, which is the optimal weight point maximizing $\log p(\theta|\mathbf{X},\mathbf{Y})$. It satisfies $\frac{\partial}{\partial\theta}\log p(\theta^*|\mathbf{X},\mathbf{Y}) = 0$.

$$\log p(\theta|\mathbf{X},\mathbf{Y}) \approx \log p(\theta^*|\mathbf{X},\mathbf{Y}) + \frac{\partial}{\partial\theta}\log p(\theta^*|\mathbf{X},\mathbf{Y})(\theta-\theta^*) \tag{18}$$

$$+ \frac{1}{2}(\theta-\theta^*)^T\frac{\partial^2}{\partial^2\theta}\log p(\theta^*|\mathbf{X},\mathbf{Y})(\theta-\theta^*) \tag{19}$$

$$= \frac{1}{2}(\theta-\theta^*)^T\mathbf{H}_{\log p(\theta^*|\mathbf{X},\mathbf{Y})}(\theta-\theta^*) + \text{const} \tag{20}$$

We denote $\mathbf{H}_{\log p(\theta^*|\mathbf{X},\mathbf{Y})} = \frac{\partial^2}{\partial^2\theta}\log p(\theta^*|\mathbf{X},\mathbf{Y})$ as the Hessian Matrix of $\log p(\theta^*|\mathbf{X},\mathbf{Y})$. By substituting (16) into this definition, we have

$$\mathbf{H}_{\log p(\theta^*|\mathbf{X},\mathbf{Y})} = \frac{\partial^2}{\partial^2\theta}[\log p(\mathbf{Y}|\mathbf{X},\theta) + \log p(\theta) - \log p(\mathbf{Y}|\mathbf{X})] \tag{21}$$

$$= \frac{\partial^2}{\partial^2\theta}[\log p(\mathbf{Y}|\mathbf{X},\theta) + \log p(\theta)] \tag{22}$$

[3] If $||\cdot||$ is a L2-norm, the corresponding $p(\theta)$ is a Normal distribution $\mathcal{N}(\theta;0,\lambda^{-1}I)$. It will be a Laplace distribution, if $||\cdot||$ is a L1-norm.

Recall that the prior weight distribution $p(\theta) \sim \mathcal{N}(\theta; \mathbf{0}, \Sigma_0)$, we have

$$\mathbf{H}_{\log p(\theta^*|\mathbf{X},\mathbf{Y})} = \frac{\partial^2}{\partial^2 \theta}[\log p(\mathbf{Y}|\mathbf{X}, \theta)] - \Sigma_0^{-1} \tag{23}$$

$$= \mathbf{H}_{\log p(\mathbf{Y}|\mathbf{X},\theta)} - \Sigma_0^{-1} \tag{24}$$

According to the Fisher information matrix definition, the Fisher information matrix of $p(\mathbf{X}, \mathbf{Y}|\theta)$ is given by

$$\mathbf{F} = \mathbb{E}_{P(\mathbf{X},\mathbf{Y}|\theta)}[\nabla \log p(\mathbf{X}, \mathbf{Y}|\theta)^T \nabla \log p(\mathbf{X}, \mathbf{Y}|\theta)] = -\mathbb{E}_{P(\mathbf{X},\mathbf{Y}|\theta)}[\mathbf{H}_{\log p(\mathbf{X},\mathbf{Y}|\theta)}] \tag{25}$$

Since the joint distribution $p(\mathbf{X}, \mathbf{Y}|\theta) = p(\mathbf{Y}|\mathbf{X}, \theta)p(\mathbf{X})$, where $p(\mathbf{X})$ is the data distribution and does not dependent on θ, we have

$$\nabla \log p(\mathbf{X}, \mathbf{Y}|\theta) = \nabla \log p(\mathbf{Y}|\mathbf{X}, \theta) + \nabla \log p(\mathbf{X}) = \nabla \log p(\mathbf{Y}|\mathbf{X}, \theta). \tag{26}$$

As a result, $\mathbf{F}$ can be written as the expectation (w.r.t. $P_{\mathbf{X}}$) of the Fisher information matrix of $P_{\mathbf{Y}|\mathbf{X},\theta}$, so that

$$\mathbf{F} = \mathbb{E}_{P_{\mathbf{X}}}[\mathbb{E}_{P_{\mathbf{Y}|\mathbf{X},\theta}}[\nabla \log p(\mathbf{Y}|\mathbf{X}, \theta)^T \nabla \log p(\mathbf{Y}|\mathbf{X}, \theta)]] \tag{27}$$

$$= -\mathbb{E}_{P_{\mathbf{X}}}[\mathbb{E}_{P_{\mathbf{Y}|\mathbf{X},\theta}}[\mathbf{H}_{\log p(\mathbf{Y}|\mathbf{X},\theta)}]] \tag{28}$$

Empirical Fisher The empirical fisher was adopted as a low-cost surrogate of the standard Fisher or Hessian in [2–4]. It is defined as

$$\tilde{\mathbf{F}} \doteq \frac{1}{N} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} [\nabla \log p(\mathbf{y}|\mathbf{x}, \theta)^T \nabla \log p(\mathbf{y}|\mathbf{x}, \theta)]. \tag{29}$$

Diagonal Fisher information matrix In neural networks, the number of parameters could be in the order of a million. It is hard to compute the exact Fisher and its inversion directly. Researchers simplify the Fisher information matrix approximation with structure information to make the computation feasible. The most commonly used approach is diagonal Fisher information matrix approximation [5, 6], which considers each single weight parameter as independent:

$$\text{diagonal standard Fisher: } \text{diag}(\mathbf{F}) = \mathbb{E}_{P_{\mathbf{X}}}[\mathbb{E}_{P_{\mathbf{Y}|\mathbf{X},\theta}}[\text{diag}(\nabla \log p(\mathbf{y}|\mathbf{x}, \theta))^2]], \tag{30}$$

$$\text{diagonal empirical Fisher: } \text{diag}(\tilde{\mathbf{F}}) = \frac{1}{N} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} [\text{diag}(\nabla \log p(\mathbf{y}|\mathbf{x}, \theta))^2] \tag{31}$$

where $\text{diag}(\cdot)$ converts a matrix or a vector to its corresponding diagonal matrix.

2 Monte-Carlo based Fisher evaluation

Calculating the standard diagonal Fisher with Monte-Carlo sampling requires sampling from the predictive distribution. For the post-hoc Laplace approximation method in this paper, the predictive distribution here might be ill-defined. It is because we only have a deterministic model before calculating the Fisher. Thus, a prior predictive distribution must be introduced so that we can sample from it and compute the Fisher. Algorithm A1 [4] illustrates this process. The result of the Monte-Carlo-based approach converges to the exact Fisher calculation [7] when the number of samples increases, as shown in Figure A1.

3 A heuristic approach to sampling from weight distributions

An important step in moment estimation is sampling from the approximate weight distribution $p(\theta|\mathbf{X}, \mathbf{Y}) = \mathcal{N}(\theta; \theta^*, \Sigma)$, where $\Sigma = [\mathbf{F} + \Sigma_0^{-1}]^{-1}$. In practice, the corresponding elements of some parameters in the prior precision matrix Σ_0^{-1} must remain large to compensate for the intrinsic inaccuracy of Fisher approximation, but it has the side effect of washing out some small but meaningful values [8].

Through trial and error, we find a relatively simple and effective way to heuristically determine the priors and get the final covariance matrix for sampling. It contains three steps: (1) remove

[4] The output $\mathbf{y}_0 = \{\mathbf{y}_0^{cls}, \mathbf{y}_0^{reg}\}$ contains classification predictions $\mathbf{y}_0^{cls}$ and regression predictions $\mathbf{y}_0^{reg}$. The diag-to-matrix(v) construct a diagonal matrix with the input vector v.

Algorithm A1: Monte-Carlo-based approach to calculating the standard diagonal Fisher

Input : $\mathbf{X}$: inputs; f_θ: deterministic neural network;
T_f: number of Monte-Carlo samples;
Σ_0^{reg}: prior covariance matrix
Output : $\mathbf{F}$: the Fisher

initialize $\mathbf{F}$; $N \leftarrow len(\mathbf{X})$;
foreach $\boldsymbol{x} \in \boldsymbol{X}$ **do**
  $\mathbf{y}_0 \leftarrow f_\theta(\mathbf{x})$;
  for $i \leftarrow 1$ **to** T_f **do**
    $\tilde{\mathbf{y}}^{cls} \leftarrow$ sample from categorical$(\mathbf{y}_0^{cls})$;
    $\tilde{\mathbf{y}}^{reg} \leftarrow$ sample from $\mathcal{N}(\mathbf{y}_0^{reg}, \Sigma_0^{reg})$;
    $\mathbf{g} \leftarrow \nabla_\theta \log p(\tilde{\mathbf{y}}^{cls}, \tilde{\mathbf{y}}^{reg} | f_\theta(\mathbf{x}))$;
    $\mathbf{F} += \frac{1}{NM}$diag-to-matrix$(\mathbf{g}^2)$;
  end
end
return $\mathbf{F}$;

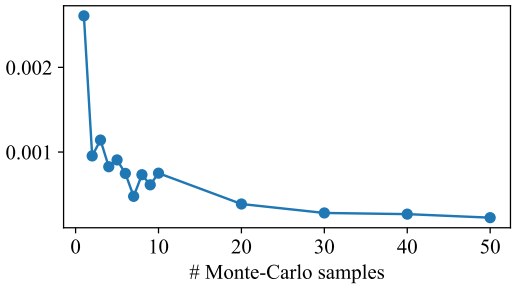

Figure A1: Mean absolute difference between the Fishers, computed with exact and Monte-Carlo approaches (Algorithm A1). It is based on a Linear model on the MNIST dataset.

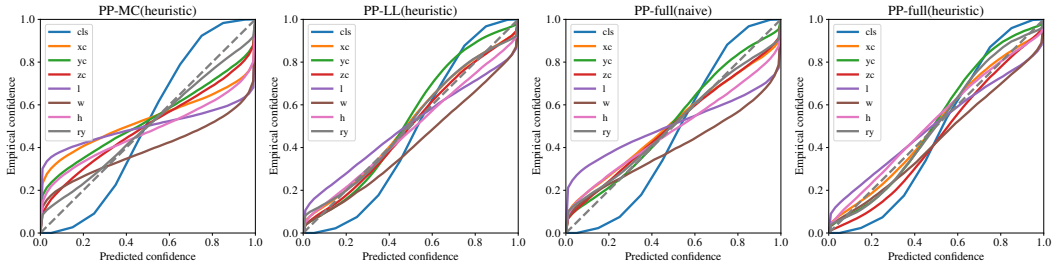

Figure A2: Calibration plots of PointPillar. The curves closer to the gray line (upper bound) represent better results. The curve of 'cls' denotes the calibration plot of the classification estimation of the detector. The other curves denote the calibration plots of the regression estimations of the detector. The parameterization of a 3D bounding box used in this paper is $[xc, yc, zc, l, w, h, ry]$ representing the coordinate of the box center, size of the box and the rotation along the z-axis.

the effects of inaccurate Fisher elements, (2) set a proper prior, and (3) set a proper scaler. As illustrated in Algorithm A2, we compute a threshold t for each layer. All the parameters in each layer, the corresponding values of which in the Fisher diagonal are smaller than t, are considered as the parameters with inaccurate Fisher values and will be given an infinitely large value in Σ_0^{-1}. For the other parameters, we consider their values in the Fisher are accurate, and give them a prior with the value of t in the Σ_0^{-1}. In our experiment, we empirically set t as the 30-quantile positive element in each layer. A parameter α is introduced to scaling the covariance matrix, and we found that setting α proportional to t works well in practice. We compare the uncertainty estimation quality of our proposed heuristic approach to the naive method with a homogenous Σ_0 for all parameters in our experiments. Superior performance of the heuristic method can be seen by comparing the last two plots in Figure A2.

4 Implementation details

We adopt well-trained 3D object detectors [5] as deterministic models and convert them with the Laplace-approximation-based approaches as in Section 3 in our paper. The adopted 3D detectors contain one-stage detectors PointPillar[9] and Second[10] , as well as a two-stage detector PointRCNN[11]. In particular, we compute the Fisher on the KITTI training set and inference with it on the validation set.

4.1 Hyperparameter tuning

The important hyperparameters in this paper include: α of our heuristic sampling approach, prior of predictive distribution, number of Monte-Carlo samples, and dropout rate (Monte-Carlo Dropout baseline). We use 10 Monte-Carlo samples to approximate the Bayesian inference and one Monte-

[5] https://github.com/open-mmlab/OpenPCDet

Algorithm A2: Heuristic method to calculate covariances of posterior weight distribution

Input : $\mathbf{F}^l, l = 1, ..., L$: Fisher matrix of the l-th layer; α: scaler of covariance matrices.
Output : $C^l, l = 1, ..., L$: covariance matrix of the parameters in the l-th layer.
foreach $\bar{\boldsymbol{F}}^l$ **do**
  initialize $diag_P^l$with all zeros ;
  $diag\_F^l = \text{diag}(\mathbf{F}^l)$;
  $t \leftarrow$ the 30-quantile positive element in $diag\_F^l$;
  $idx \leftarrow$ indices of $diag\_F^l > t$;
  $diag\_P^l[idx] \leftarrow t$;
  $P^l \leftarrow$ diag-to-matrix$(diag_P^l)$;
  $C^l \leftarrow \alpha t(\mathbf{F}^l + P^l)^{-1}$;
end
return $C^l, l = 1, ..., M$;

Table A1: Hyperparameter tuning space. The $*$ superscript in each line denotes the selected optimal hyperparameters.

Hypaperameters	Tuning space
α (standard Fisher)	$\{0.0003, 0.001, 0.003^*, 0.01, 0.03\}$
α (empirical Fisher)	$\{0.0003, 0.001, 0.003, 0.01^*, 0.03\}$
prior of predictive distribution (cls.)	$\{1e-6, 1e-5, 1e-4, 1e-3, 1e-2^*\}$
prior of predictive distribution (reg.)	$\{1e-6, 1e-5, 1e-4, 1e-3^*, 1e-2\}$
dropout rate (PP)	$\{0.01, 0.05^*, 0.1, 0.2, 0.3\}$
dropout rate (SC)	$\{0.01, 0.05, 0.1, 0.2^*, 0.3\}$
dropout rate (PR)	$\{0.1, 0.2, 0.3, 0.4^*, 0.5, 0.6, 0.7\}$
number of Monte-Carlo samples in Bayesian inference	$\{2, 4, 10^*, 20\}$

Carlo sample in the Fisher calculation. The hyperparameters are tuned on a hand-hold set with 500 data points. We tune each hyperparameter independently, according to the $\mathbb{E}_{\text{class, level}}[\text{mAP(JIoU)}]$ performance. The hyperparameter tuning space is shown in the Table A1. When tuning α, $\mathbb{E}_{\text{class, level}}[\text{mAP(JIoU)}]$ performance is monotonically decreasing with α. But if α is too small, it will degrade to a deterministic model and does not output meaningful uncertainty. Therefore, we select a relatively big value of α which performs good $\mathbb{E}_{\text{class, level}}[\text{mAP(JIoU)}]$, as well as outputs meaningful epistemic uncertainty. We set $\alpha = 0.003$ for the standard Fisher, and $\alpha = 0.01$ for the empirical Fisher. The prior of the predictive distribution is set as $0.01 \times \mathbf{I}$ for classification and $0.001 \times \mathbf{I}$ for regression, where $\mathbf{I}$ denotes the identity matrix. The dropout rates for the Monte-Carlo baselines of PointPillar, SECOND, and PointRCNN are 0.05, 0.2, 0.4, respectively.

4.2 Metrics

We evaluate the quality of estimated uncertainty with the expected calibration error (ECE) for classification [12] and regression [13]. It is noted that the ECE metric, which was proposed for univariate regression in [13], cannot be directly used in multivariate regression, like 3D object detection. Therefore we treat each dimension independently, following [14].

To evaluate the mAP(JIoU), we generate the ground-truth predictive distribution from the generative model proposed in [15] with 3D bounding box annotations and LiDAR observations. The mAP(JIoU) is evaluated between the estimated and ground-truth predictive distributions. We evaluate mAP(JIoU) in both bird's-eye-view (BEV) and 3D space with the threshold of 0.5 in evaluating the "Car" class, 0.25 in evaluating the "Pedestrian" and "Cyclist" classes. For more details on mAP(JIoU) justification and the generative model, readers can refer to [15].

method	class	bev			3d		
		easy	mod.	hard	easy	mod.	hard
PP	car	80.21 (1.64)	78.90 (1.05)	75.54 (1.45)	43.71 (2.24)	27.95 (1.59)	23.51 (1.19)
	ped.	19.20 (0.38)	20.80 (0.43)	20.69 (0.34)	29.87 (1.16)	29.75 (1.12)	27.62 (0.94)
	cyclist	48.29 (2.02)	42.60 (1.36)	40.34 (1.10)	46.26 (1.96)	32.96 (1.80)	31.25 (1.76)
SC	car	87.00 (0.50)	84.83 (0.61)	81.25 (0.30)	58.20 (0.44)	37.56 (0.53)	31.30 (0.56)
	ped.	23.55 (0.73)	26.19 (0.80)	25.71 (0.64)	30.86 (1.01)	30.90 (0.92)	28.20 (0.81)
	cyclist	50.63 (1.70)	48.33 (1.37)	46.39 (1.32)	45.46 (1.31)	34.72 (1.21)	33.10 (1.09)
PR	car	81.10 (1.14)	78.81 (1.33)	75.10 (1.30)	41.40 (0.64)	29.95 (0.70)	26.07 (0.70)
	ped.	25.99 (0.86)	25.92 (1.09)	23.99 (0.76)	39.43 (1.29)	35.75 (1.21)	31.33 (0.76)
	cyclist	62.73 (2.74)	51.95 (1.93)	49.36 (2.14)	61.26 (2.68)	43.06 (1.27)	40.71 (1.23)

Table A2: mAP(JIoU) (standard Fisher, full-net) results on the KITTI val set. The results are mean and std. (in the brackets) values computed from 5 random seeds.

4.3 Predictive distribution visualization

We visualize predictive distributions by transferring them to spatial distribution, following [15]. It extended the idea of Probabilistic Detection Quality [16] from 2D axis-aligned bounding boxes to 3D rotated bounding boxes, and defined the spatial distribution for 3D object detection $p_G(\mathbf{u})$ as

$$p_G(\mathbf{u}) \doteq \int_{\mathbf{v}_0 \in B(\mathbf{y})} p_{V(\mathbf{v}_0,\mathbf{y})}(\mathbf{u}) dv_0 \tag{32}$$

$$= \int_{\{\mathbf{y}|\mathbf{u}\in B(\mathbf{y})\}} p_{\mathbf{y}}(\mathbf{y}|\mathbf{x}) d\mathbf{y}, \tag{33}$$

where the equation (33) is an extension of PDQ [16]. It calculates the probability of a point $\mathbf{u} \in \mathbb{R}^3$ belonging to the object enclosed by a rotated bounding box $B(\mathbf{y})$.

We sample 3D points $\mathbf{v}_0 \in [-0.5, 0.5]^3$ and transform it in terms of $\mathbf{y} = [c_1, c_2, c_3, l, w, h, ry]^T$ with

$$v(\mathbf{y}) = R_y \begin{bmatrix} l & 0 & 0 \\ 0 & w & 0 \\ 0 & 0 & h \end{bmatrix} \mathbf{v}_0 + \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}; R_y = \begin{bmatrix} cos(ry) & -sin(ry) & 0 \\ sin(ry) & cos(ry) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{34}$$

It denotes the $V(\mathbf{y}, \mathbf{v}_0) \in \mathbb{R}^3$, and we can use it to compute the $P_G(u)$ with (32). Since we assume the density function of $\mathbf{y}$ is $\mathcal{N}(\mathbf{y}; \mathbf{y}_0, \Sigma_{\mathbf{y}_0})$, we can get the density function of $\mathbf{v}$ with first-order error propagation [17], and get $\mathbf{v} \sim \mathcal{N}(\mathbf{v}; V(\mathbf{y}_0, \mathbf{v}_0), J^T \Sigma_{\mathbf{y}_0} J)$, where J denotes the Jacobian matrix in terms of (34).

5 Table of mAP(JIoU) with standard deviation values

In Table A2, we report both mean and standard deviation values of mAP(JIoU) performance.

6 Evaluation on the NuScenes dataset

We conduct experiments on the NuScenes dataset [18] with the same protocol as Section4.1 in our paper. It is a real-world dataset in an autonomous driving context for object detection and tracking. We conduct experiments on the v1.0-mini split, containing ten scenes, each corresponding to a 20-second snippet. We adopt SECOND [19] as the deterministic detector, and compare the deterministic mode, MCDropout [1], and Laplace approximation methods in Table A3. The Laplace approximation method shows superior uncertainty quality over the deterministic model and the MC-Dropout baselines.

7 Calibration plots

Figure A3 and A4 shows all calibration plots in our experiments.

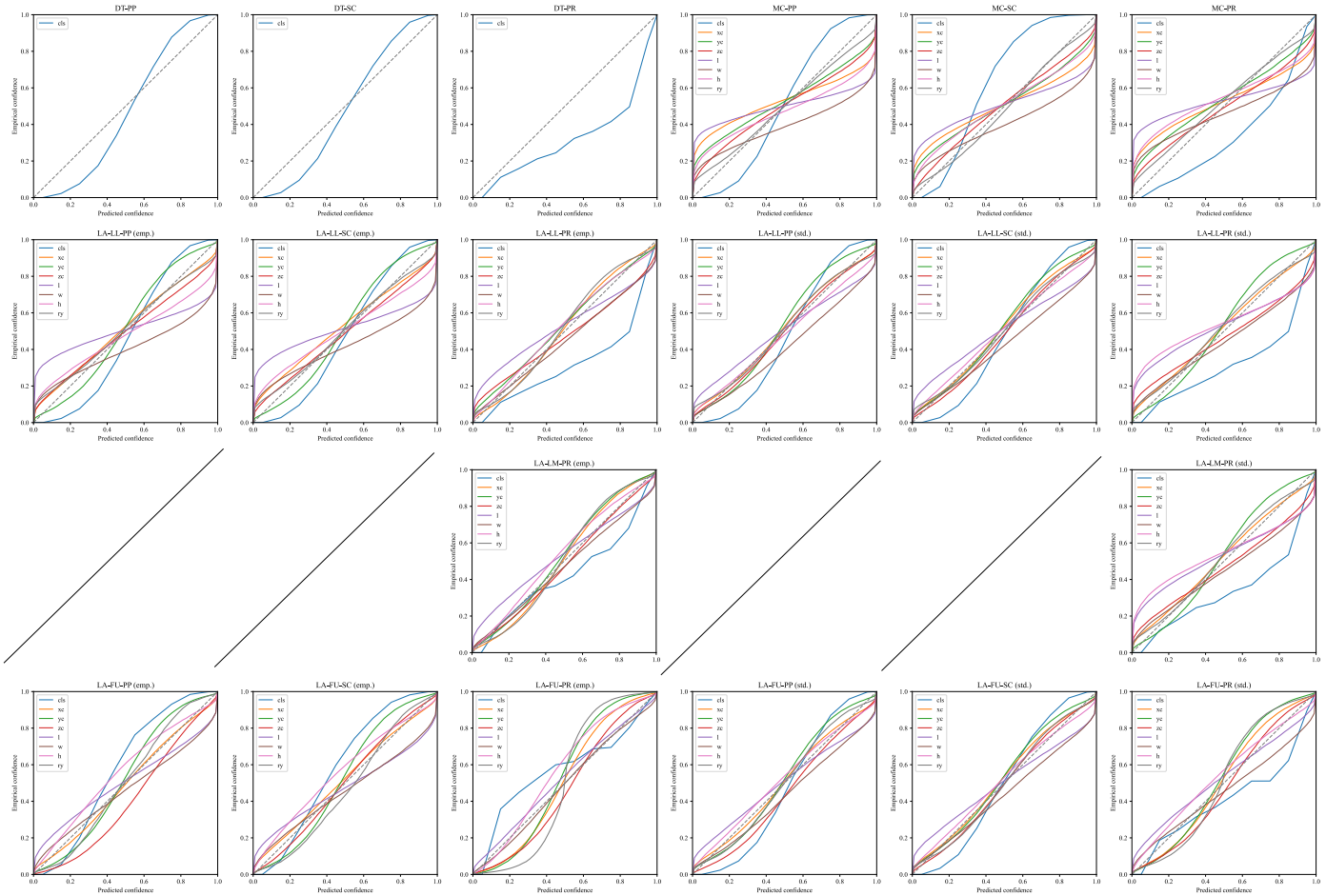

Figure A3: Calibration plots of deterministic (DT), Monte-Carlo dropout(MC), Last-layer Laplace approximation (LL), Last-module Laplace approximation (LM), and Full-net Laplace approximation (FU). The detectors are Pointpillar (one-stage detector, PP), SECOND (one-stage detector, SC), and PointRCNN (two-stage detector, PR). Both empirical- and standard-Fisher-based Laplace approximation (emp./std.) results are reported.

method	SC classification	SC regression
det.	0.123	-
mcropout	0.123	2.075
LL(emp.)	0.120	1.666
LM(emp.)	0.123	1.336
full(emp.)	0.118	**1.278**
LL(stand.)	0.121	1.657
LM(stand.)	**0.116**	1.393
full(stand.)	0.121	1.342

Table A3: Expected calibration error (↓) on the NuScenes dataset (v1.0mini split). The "det.", "LL", "LM", and "full" are short for "deterministic", "last-layer", "last-module", and "full-net".

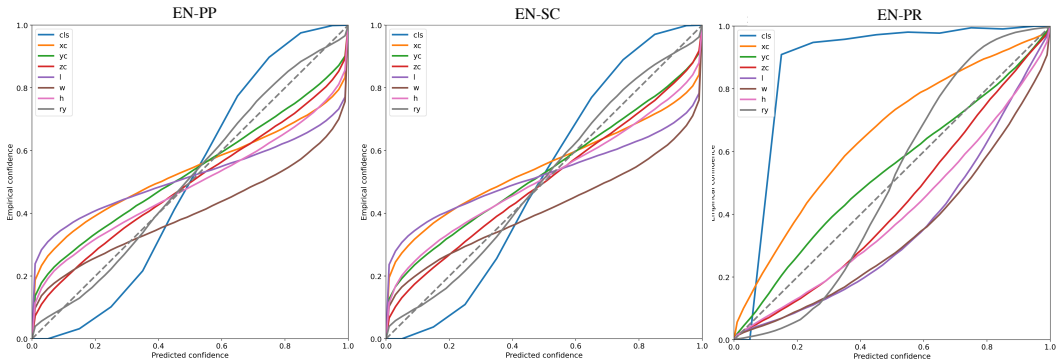

Figure A4: Calibration plots of DeepEnsembles.

8 Discussion on the number of Monte-Carlo samples

Figure A5 shows the effect of the number of Monte-Carlo samples $T = \{2, 4, 8, 10, 20\}$ on NLL. It demonstrates that the NLL converges with an increasing number of Monte-Carlo samples. In various Monte-Carlo samples, Laplace-approximation-based methods perform better than the MCDropout method.

9 Discussion on metrics

The metrics mAP(IoU) and mAP(JIoU) can evaluate the mode accuracy and general performance of predictive distribution so that we can tune hyperparameters and analyze their performance in different classes. However, they are not good metrics for comparing different types of detectors because a fair comparison requires aligning the weight perturbation at the same level. It is impractical to implement if their network structures are different. One possible solution is to compare the performance of downstream tasks with the estimated epistemic uncertainties as inputs, like active learning.

10 Time and memory analysis

Table A4 reports the time and memory consumptions of baselines and various Laplace approximation implementations in our experiments[6]. It shows that the full-net Laplace approximation requires about $20\times$ time and $1.5-2\times$ GPU memory than the deterministic counterpart in the inference stage. In contrast, the subnet Laplace approximation ("LL" and "LM" in Table A4) is a balance between the uncertainty quality and the computational efficiency, and requires about $4-7\times$ time in the infer-

[6]The test machine is equipped with a 8-core CPU (Intel Core i7-7700K) and a single NVIDIA GTX 1080Ti GPU. The number of Monte-Carlo samples is 10.

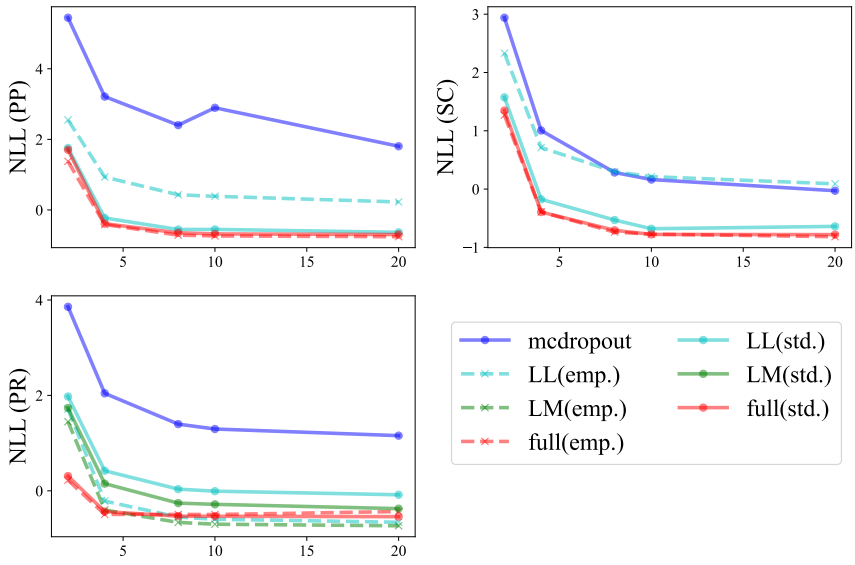

Figure A5: Relationship between the number of MC samples (x-axis) and NLL.

methods	PP		SC		PR	
	inference time (ms)	GPU memory (MB)	inference time (ms)	GPU memory (MB)	inference time (ms)	GPU memory (MB)
det.	21.7	188.4	34.8	188.9	147.8	24.9
dropout	186.8	300.5	159.3	282.2	464.7	56.3
deep-ensem.	203.8	1147.7	141.0	1077.4	556.7	190.5
LL (E.)	156.2	301.6	136.9	282.5	605.9	57.1
LM (E.)	-	-	-	-	634.8	56.3
full (E.)	530	300.4	677.5	282.4	1491.0	56.3
LL (S.)	156.1	301.6	134	282.5	617.5	57.1
LM (S.)	-	-	-	-	636.5	56.3
full (S.)	526.7	300.4	686.1	282.4	1489.8	56.3

Table A4: Inference time and GPU memory consumptions. It is computed on a 8-core CPU (Intel Core i7-7700K) and a single NVIDIA GTX 1080Ti GPU.

ence stage. The run-time performance can be improved if it parallels the Monte-Carlo computations with batched inputs [1].

References

[1] G. Yarin. Uncertainty in deep learning. *University of Cambridge, Cambridge*, 2016.

[2] J. Martens and I. Sutskever. Training deep and recurrent networks with hessian-free optimization. In *Neural networks: Tricks of the trade*, pages 479–535. Springer, 2012.

[3] N. Roux, P.-A. Manzagol, and Y. Bengio. Topmoumoute online natural gradient algorithm. *Advances in neural information processing systems*, 20, 2007.

[4] J. Martens et al. Deep learning via hessian-free optimization. In *ICML*, volume 27, pages 735–742, 2010.

[5] Y. LeCun, J. Denker, and S. Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[6] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[7] J. Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21:1–76, 2020.

[8] J. Martens and R. Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

[9] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[10] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[11] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019.

[12] M. P. Naeini, G. Cooper, and M. Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[13] V. Kuleshov, N. Fenner, and S. Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International conference on machine learning*, pages 2796–2804. PMLR, 2018.

[14] D. Feng, L. Rosenbaum, C. Glaeser, F. Timm, and K. Dietmayer. Can we trust you? on calibration of a probabilistic object detector for autonomous driving. *arXiv preprint arXiv:1909.12358*, 2019.

[15] Z. Wang, D. Feng, Y. Zhou, L. Rosenbaum, F. Timm, K. Dietmayer, M. Tomizuka, and W. Zhan. Inferring spatial uncertainty in object detection. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5792–5799, 2020. doi: 10.1109/IROS45743.2020.9340798.

[16] D. Hall, F. Dayoub, J. Skinner, H. Zhang, D. Miller, P. Corke, G. Carneiro, A. Angelova, and N. Sünderhauf. Probabilistic object detection: Definition and evaluation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1031–1040, 2020.

[17] K. O. Arras. An introduction to error propagation: derivation, meaning and examples of equation cy= fx cx fxt. Technical report, ETH Zurich, 1998.

[18] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[19] Y. Yan, Y. Mao, and B. Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10), 2018. ISSN 1424-8220. doi:10.3390/s18103337. URL https://www.mdpi.com/1424-8220/18/10/3337.