Dense Uncertainty Estimation

Jing Zhang, Yuchao Dai, Mochu Xiang, Deng-Ping Fan, Peyman Moghadam
Mingyi He, Christian Walder, Kaihao Zhang, Mehrtash Harandi and Nick Barnes

Abstract—Deep neural networks can be roughly divided into deterministic neural networks and stochastic neural networks. The former is usually trained to achieve a mapping from input space to output space via maximum likelihood estimation for the weights, which leads to deterministic predictions during testing. In this way, a specific weights set is estimated while ignoring any uncertainty that may occur in the proper weight space. The latter introduces randomness into the framework, either by assuming a prior distribution over model parameters (*i.e.* Bayesian Neural Networks) or including latent variables (*i.e.* generative models) to explore the contribution of latent variables for model predictions, leading to stochastic predictions during testing. Different from the former that achieves point estimation, the latter aims to estimate the prediction distribution, making it possible to estimate uncertainty, representing model ignorance about its predictions. We claim that conventional deterministic neural network based dense prediction tasks are prone to overfitting, leading to over-confident predictions, which is undesirable for decision making. In this paper, we investigate stochastic neural networks and uncertainty estimation techniques to achieve both accurate deterministic prediction and reliable uncertainty estimation. Specifically, we work on two types of uncertainty estimations solutions, namely ensemble based methods and generative model based methods, and explain their pros and cons while using them in fully/semi/weakly-supervised framework. Due to the close connection between uncertainty estimation and model calibration, we also introduce how uncertainty estimation can be used for deep model calibration to achieve well-calibrated models, namely dense model calibration. Code and data are available at https://github.com/JingZhang617/UncertaintyEstimation.

Index Terms—Uncertainty Estimation, Dense Prediction, Stochastic Predictions, Model Calibration.

1 Introduction

Deep models are prone to becoming over-confident [1], where [1] explains that a deep neural network is capable of fitting any type of noise. Many different regularizations [2] have been proposed to prevent neural networks from overfitting, such as L1 or L2 regularization [3], early stopping [4] and dropout [5]. L1 or L2 regularization [3] try to add constraints to the weights set, thus to limits weights' flexibility. The early stopping technique [4] is based on the observation that the model we get at the end is not the best we could have possibly created. To this end, we may obtain a proper model before the model starts overfitting. Dropout [5] is another popular method of regularization. The main idea behind it is to assign a probability (dropout ratio) to each unit of network of being temporarily ignored in the learning stage. Although those solutions have proven effective in various tasks in preventing the model from being over-confident, there exists no explicit output from those models to explain the degree of model calibration [1], or how much the model is regularized before we resort to the ground truth of testing sets.

Given the training dataset $D = \{x_i, y_i\}_{i=1}^N$, the objective of machine learning methods is to minimize the expected loss function, or in practice the empirical loss function (also known as empirical risk minimization [6]) as:

$$\begin{aligned}\min_\theta \mathbb{E}_{x,y}[\mathcal{L}(f(x;\theta),y)] &= \int \mathcal{L}(f(x;\theta),y)dp(x,y) \\ &\approx \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(f(x_i;\theta),y_i), \quad (x_i,y_i)\sim p(x,y),\end{aligned} \tag{1}$$

where x, y are input and output variables from the sampled training dataset D, θ is the learned model parameter set, $p(x,y)$ is the joint data distribution, (x_i, y_i) represents a sampled pair from the joint data distribution $p(x,y)$, and $\mathcal{L}(.,.)$ is the loss function. Under the maximum likelihood framework, we maximize the likelihood to achieve the minimized $\mathcal{L}(.,.)$. Given training dataset D, the predictive distribution $p(y|x)$ is defined as:

$$p(y|x) = \int p(y|x,\theta)p(\theta)d\theta, \tag{2}$$

where $p(y|x,\theta)$ is the likelihood, $p(\theta)$ is the approximate posterior for model parameters θ. For a regression task, $p(y|x,\theta)$ is usually defined as Gaussian likelihood, which is $p(y|x,\theta) = \mathcal{N}(f(x,\theta),\Sigma))$, where $f(x,\theta)$ is the model output, Σ is the diagonal covariance matrix, indicating the inherent noise within the label. For a classification task, $p(y|x,\theta)$ is defined as Softmax likelihood, leading to $p(y|x,\theta) = Softmax(f(x,\theta)/exp(\sigma^2))$ [7], where σ^2 is the noise level, and $exp(\sigma^2)$ serves as the temperature as in model calibration [1], where we have $exp(\sigma^2) = 1$ for confident predictions and $exp(\sigma^2) > 1$ for uncertain predictions.

Eq. 2 shows that the randomness or uncertainty of y comes from both inherent noise Σ (or σ^2 for classification) and $p(\theta)$, which is an approximation of the true posterior distribution of model parameters $p(\theta|D)$. [7], [8] define the former as aleatoric uncertainty (or data uncertainty) and the latter as epistemic uncertainty (or model uncertainty), where "uncertainty" is a term

- *Jing Zhang, Kaihao Zhang and Nick Barnes are with School of Computing, the Australian National University. (Email: zjnwpu@gmail.com, kaihao.zhang@anu.edu.au, nick.barnes@anu.edu.au)*
- *Yuchao Dai, Mochu Xiang and Mingyi He are with School of Electronics and Information, Northwestern Polytechnical University, China. (Email: daiyuchao@gmail.com, xiangmochu@mail.nwpu.edu.cn, myhe@nwpu.edu.cn)*
- *Deng-Ping Fan is with the CS, Nankai University, China. (Email: dengpfan@gmail.com)*
- *Peyman Moghadam and Christian Walder are with CSIRO, Australia. (Email: peyman.moghadam@data61.csiro.au, christian.walder@data61.csiro.au)*
- *Mehrtash Harandi is with Monash University, Australia. (Email: mehrtash.harandi@monash.edu)*

to explain the randomness of model predictions. Aleatoric uncertainty comes from the intrinsic randomness of the task [8], which cannot not be reduced, and is usually independent of the choice of the learner or the learned model. Epistemic uncertainty, on the other hand, comes from the limited knowledge about which model generated the provided dataset. In this case, the aleatoric uncertainty cannot be reduced with a larger training dataset, while the epistemic uncertainty can be explained away with a larger and more diverse training dataset, leading to a comprehensive understanding about the model that generated our dataset. The goal of uncertainty estimation [9]–[16] is then to produce a measure of confidence for model predictions, either by learning from the data directly [7], [17], or by acquiring from a diverse set of predictions [17], [18].

A systematic way to deal with uncertainty is via Bayesian statistics [7], [19]–[25]. Bayesian Neural Networks aim to learn a distribution over each of the network parameters by placing a prior probability distribution over network weights. Calculating the exact Bayesian posterior is computational intractable. More effort has been put into developing approximations of Bayesian Neural Network that can work in practice. In the following, we will introduce the each type of uncertainty and induce their computation details within both Bayesian Neural Networks and non-Bayesian Neural Networks.

1.1 Aleatoric Uncertainty

As discussed above, the aleatoric uncertainty [7], [25], [26] captures the observation noise Σ inherent in the task, which can be categorised as image independent uncertainty (homoscedastic uncertainty that does not vary with the input) and image conditional uncertainty hetroscedastic uncertainty that various with the input) [7]. For the former, the noise is totally random, and for the latter, it is assumed that the noise is related to the context of the input image x. As we want to perform variational inference over the noise level Σ, non-Bayesian Neural Networks can be used with a deterministic model parameter set θ. In this following, we introduce the aleatoric uncertainty for both regression tasks and classification tasks. For all the presentation below, we define x and y as the input and output pair, and θ the model parameters.

Aleatoric uncertainty for regression tasks: For a regression task, the observed label y is assumed to be corrupted with the observational noise ϵ:

$$y = f(x, \theta) + \epsilon, \tag{3}$$

where the observational noise ϵ can be either image-independent or image-dependent. In the following, we only discuss the image dependent noise, or more specifically, the pixel-level noise $\epsilon = \epsilon(x) \sim \mathcal{N}(0, \Sigma)$, where Σ is the diagonal covariance matrix representing the inherent noise.

With the above noise corruption process, we define the Gaussian likelihood for the regression task as $p(y|x, \theta) = \mathcal{N}(f(x, \theta), \Sigma)$, where $\Sigma = diag(\sigma^2(x))$. Given the Gaussian likelihood:

$$p(y|x, \theta) = \frac{1}{\sqrt{2\pi\sigma^2(x)}} exp(\frac{-[y - f(x, \theta)]^2}{2\sigma^2(x)}), \tag{4}$$

the log-likelihood can be obtained as:

$$\log(p(y|x, \theta)) = -\frac{[y - f(x, \theta)]^2}{2\sigma^2(x)} - \frac{1}{2}\log(\sigma^2(x)) - \frac{1}{2}\log(2\pi). \tag{5}$$

Then the loss function with noise corruption model as in Eq. 3 is achieved as [7], [27]:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\frac{1}{2\sigma^2(x_i)}\mathcal{L}_2 + \frac{1}{2}log(\sigma^2(x_i))), \tag{6}$$

where the constant $\frac{1}{2}\log(2\pi)$ is removed. $\mathcal{L}_2$ is L2 loss, and $\sigma(x)$ is a noise estimation module, representing the pixel-level aleatoric uncertainty. According to Eq. 3, we need to estimate both mean prediction $f(x, \theta)$ and variance $\sigma^2(x)$ indicating the noise level, representing the aleatoric uncertainty. [7], [27] achieve this with a multi-head network, where the head of the network is split to produce both task related prediction $f(x, \theta)$ and the aleatoric uncertainty $\sigma^2(x)$.

To train with the loss function in Eq. 6, we only need supervision y for the $\mathcal{L}_2$ term, and the noise estimation module does not need any supervision. In practice, usually the log variance is predicted as $s = \log\sigma^2(x)$ to achieve stable training [7], leading to:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\frac{1}{2}exp(-s_i)\mathcal{L}_{ce} + \frac{1}{2}s_i), \tag{7}$$

where the uncertainty representation s serves as both weight for the original loss function $\mathcal{L}_2$ and regularizer to achieve accurate prediction and reliable uncertainty estimation.

Aleatoric uncertainty for classification tasks: For a classification task with Softmax likelihood, noise-corruption is achieved by squashing the model prediction with uncertainty related variable [28]:

$$p(y|x, \theta) = Softmax(\frac{f(x, \theta)}{\exp(\sigma^2(x))}), \tag{8}$$

where $\exp(\sigma^2(x))$ can also be explained as temperature as in [1].

Let's define $T = \exp(\sigma^2(x))$, the log-likelihood can then be obtained as:

$$\log(p(y|x, \theta)) = \frac{1}{T}f_\theta(x) - \log(\exp(\frac{1}{T}f(x, \theta)) + 1). \tag{9}$$

We also have:

$$(\exp(f(x, \theta)) + 1)^{\frac{1}{T}} \approx \frac{1}{T}(\exp(\frac{1}{T}f(x, \theta)) + 1), \tag{10}$$

which becomes equal when $T = \exp(\sigma^2) \to 1$, indicating the uncertainty is not very high, which is consistent with our basic understanding about aleatoric uncertainty, which should be not too large and not too small.

With the approximation of $\exp(\frac{1}{T}f(x, \theta)) + 1$ as in Eq. 10, we take it back to Eq. 9 and obtain the approximated likelihood as:

$$\log(p(y|x, \theta)) = -\log T - \frac{1}{T}\mathcal{L}_{ce}, \tag{11}$$

where $\mathcal{L}_{ce}$ is the cross-entropy loss. The corresponding loss function is then:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\frac{1}{T_i}\mathcal{L}_{ce} + \log T_i), \tag{12}$$

Aleatoric uncertainty within BNN: The aleatoric uncertainty $\sigma^2(x)$ is induced from a non-Bayesian Neural Network (BNN) with fixed model parameters. [25] defines the aleatoric uncertainty within the BNN as mean entropy or the Bayesian Active Learning by Disagreement Objective (BALD) [29] of model prediction:

$$U_a = \mathbb{E}_{p(\theta|D)}[H(y|x, \theta)], \tag{13}$$

where $H(.)$ is the entropy. Eq. 13 integrates out the model parameters θ, which only captures the prediction randomness caused by the inherent noise. Note that the aleatoric uncertainty in Eq. 7 or Eq. 12 is obtained via a dual-head network, which can produce model prediction and uncertainty simultaneously, while aleatoric uncertainty in Eq. 13 is within the BNN, which can be computed directly from the prediction itself by integrating out the contribution of model parameters θ. Alternatively, [22] define aleatoric uncertainty as the uncertainty based on the best model given the training dataset D. In this way, the aleatoric uncertainty with the best model can be obtained via:

$$U_a = H(y|x, \theta^*), \tag{14}$$

where θ^* is the model parameter that yields the best performance (or the smallest loss).

1.2 Epistemic Uncertainty

Aleatoric uncertainty can be directly estimated through the auxiliary head $\sigma^2(x)$ or by the mean conditional prediction entropy as in Eq. 13. Epistemic uncertainty, on the contrary, is usually estimated as the residual of predictive uncertainty and aleatoric uncertainty. The predictive uncertainty is defined as the total uncertainty representing prediction randomness from both the inherent noise and the model weights space $\theta \sim p(\theta|D)$. Specifically, the predictive uncertainty is usually defined as: $U_p = H(y|x)$, which is obtained as the entropy of mean model prediction with model parameters sampled from its posterior distribution. With both predictive uncertainty U_p and aleatoric uncertainty U_a, the epistemic uncertainty is then defined as:

$$\begin{aligned} U_e &= U_p - U_a \\ &= H(y|x) - \mathbb{E}_{p(\theta|D)}[H(y|x, \theta)], \end{aligned} \tag{15}$$

where $H(y|x) - \mathbb{E}_{p(\theta|D)}[H(y|x, \theta)]$ is the mutual information of model prediction and parameters. [25] then defines the epistemic uncertainty as the mutual information of model prediction and model parameters conditioned on the input image x, which is: $U_e = I(y, \theta|x)$.

Eq. 15 can be used in both regression tasks and classification tasks [25]. [7] defines an alternative solution for epistemic uncertainty estimation within the BNN for regression tasks, which uses variance instead of entropy for uncertainty measure[1] as:

$$U_e = \mathbb{E}_{p(\theta|D)}[p(y|x, \theta)^2] - (\mathbb{E}_{p(\theta|D)}[p(y|x, \theta)])^2. \tag{16}$$

With epistemic uncertainty U_e in Eq. 16, the predictive uncertainty can then be defined as:

$$U_p = \mathbb{E}_{p(\theta|D)}[p(y|x, \theta)^2] - (\mathbb{E}_{p(\theta|D)}[p(y|x, \theta)])^2 + \mathbb{E}_{p(\theta|D)}[\sigma^2(x)], \tag{17}$$

where $\sigma^2(x)$ is obtained via the dual-head structure [7], [27].

2 Uncertainty Approximation

Before the deep learning era, the dual-head based aleatoric uncertainty [27], or disagreement based epistemic uncertainty via Gaussian processes [30] could be used directly for uncertainty modeling. Most of the recent techniques for epistemic uncertainty estimation [7], [22]–[25] are based on Bayesian Neural Networks (BNN), where a pre-defined prior distribution $p(\theta|D)$ is set as a constraint to regularize the distribution of the model parameters and achieve stochastic prediction. In this way, the epistemic uncertainty is obtained via the posterior distribution of network weights, which captures the set of plausible model parameters given the training dataset D. According to the Bayesian rule, the posterior over model parameters $p(\theta|x, y)$ (or $p(\theta|D)$) can be achieved by $p(\theta|x, y) = p(y|x, \theta)p(\theta)/p(y|x)$. As the marginal likelihood $p(y|x)$ cannot be evaluated analytically, $p(\theta|x, y)$ is then not available in closed-form, making it difficult to estimate aleatoric uncertainty, epistemic uncertainty or predictive uncertainty as in Eq. 13, Eq. 15, Eq. 16 and Eq. 17. In this way, approximations are required to achieve Bayesian posterior inference, including Variational Inference (VI) [31], [32] and Markov Chain Monte Carlo (MCMC) [33]. The former approximate the intractable posterior distribution $p(\theta|D)$ with an auxiliary tractable distribution $p_\phi(\theta)$ by minimizing the Kullback–Leibler divergence between $p_\phi(\theta)$ and $p(\theta|D)$, where ϕ is the variational parameters set. The latter is a sampling based posterior approximation, which draws a correlated sequence of θ_t from $p(\theta|D)$ (where t indicates the iteration of sampling), and uses it to approximate the posterior predictive distribution as a Monte Carlo average.

In this paper, we divide the uncertainty estimation techniques by the uncertainty they focus on modeling, either on aleatoric uncertainty or epistemic uncertainty. We then discuss two types of methods, namely ensemble based methods, which construct model ensemble for effective posterior approximation (mainly for epistemic uncertainty estimation), and generative model based methods for inherent noise modeling (specifically for aleatoric uncertainty estimation). We also introduce the Bayesian latent variable model for both types of uncertainty modeling.

2.1 Ensemble based Posterior Approximation

Ensemble methods [17], [18], [34]–[39] are machine learning techniques that combine several base models in order to produce one optimal predictive model.

MC-dropout: Monte Carlo Dropout (MC Dropout) [18] has been shown to be an effective variational inference, where dropout is used before every weight layer during both training and testing stages. [18] showed that networks with dropout following every weight layer are equivalent to deep Gaussian Process [40] marginalized over its covariance functions. Later, [7] found that the dropout following each layer strategy is too strong a regulariser in practice, which may lead to very slow learning. Instead, they proposed to drop out the deepest half of the encoder and decoder units [41]. With the relaxed MC Dropout, the predictive distribution in Eq. 2 can be approximated using MC integration as:

$$p(y|x) \approx \frac{1}{T}\sum_{t=1}^{T} p(y_t|x, \theta_t), \tag{18}$$

where T indicates the T iterations of sampling from the approximate posterior distribution $q(\theta|\gamma)$ (γ represents our ignorance about the model based on the training dataset D.), and θ_t is the model parameter set of the t^{th} iteration of sampling. With MC-dropout as posterior approximation, the aleatoric uncertainty in Eq. 13 can be achieved as:

$$U_a = \frac{1}{T}\sum_{t=1}^{T} H(y|x, \theta_t). \tag{19}$$

1. Entropy measures the average amount of information contained in a distribution, and is reduced to a function of variance when the distribution is Gaussian

Similarly, the predictive uncertainty and epistemic uncertainty in Eq. 15 can be obtained as:

$$U_p = H(\frac{1}{T}\sum_{t=1}^{T} p(y|x,\theta_t)), \tag{20}$$

and epistemic uncertainty as $U_e = u_p - U_a$. For regression task, the epistemic uncertainty and predictive uncertainty in Eq. 16 and Eq. 17 can then be re-wrote as:

$$U_e = \frac{1}{T}\sum_{t=1}^{T} p(y|x,\theta_t)^2 - (\frac{1}{T}\sum_{t=1}^{T} p(y|x,\theta_t))^2, \tag{21}$$

and:

$$U_p = \frac{1}{T}\sum_{t=1}^{T} p(y|x,\theta_t)^2 - (\frac{1}{T}\sum_{t=1}^{T} p(y|x,\theta_t))^2 + \frac{1}{T}\sum_{t=1}^{T} \sigma^2(x). \tag{22}$$

The main advantage of MC dropout [18], [41] is that it imposes Bernoulli distribution over the network weights, without requiring any additional model parameters.

Deep Ensemble: Different from MC-dropout [18] that is based on the same base network parameters. Deep ensemble solutions [17], [42], [43] intend to generate multiple models to approximate the posterior distribution of the true model parameters. [17] found that random initialization of the network parameters, along with random shuffling of the data points, was sufficient to obtain good performance in practice. Specially, they train M different networks with random initialization and get their final prediction as a mean of above M ensembles. Differently, [44] proposed to train different networks with the same training dataset to account for ensembles. Both above methods need re-train the same [17] or different [44] networks several times to obtain ensembles. [45] proposed an ensemble generation structure with shared encoder and several different decoders (the "Multi-head solution"). In this case, prediction from each decoder represents one ensemble. Compared with training the entire network several times, this strategy is much more convenient. Similar to above idea, pseudo multi-task learning [46] was proposed to achieve one task in different ways, thus forming a pseudo multi-task learning framework, and prediction of each pseudo task can be treated as one ensemble.

Snapshots Ensemble: The success of deep ensemble solutions [17] based on multiple copies of network from multiple random initializations of the same network, which is computationally expensive in practice. [42] introduce snapshot ensembles, where the multiple predictions are obtained from models of various epochs, leading to a cheap uncertainty estimation solution. However, although [42] is easy to implement with good uncertainty estimation, [43] argued that such techniques tend to introduce biased estimates for instances whose predictions are supposed to be highly confident. Thus they develop an uncertainty estimation algorithm that selectively estimates the uncertainty of highly confident points using earlier snapshots of the trained model inspired by [42]. They use one single forward pass in the testing stage to estimate uncertainty without any other hyper-parameters.

2.2 Generative Model for Aleatory Modeling

With an extra latent variable z, the generative models [47]–[50] are capable of generating stochastic predictions, making it possible to evaluate the uncertainty of model prediction via a generative model based framework [51]–[54]. Given the latent variable z, the predictive distribution $p(y|x)$ is defined as:

$$p(y|x) = \int p(y|x,z,\theta)p(\theta)p(z)d\theta dz, \tag{23}$$

where $p(y|x,z,\theta)$ is the likelihood, which is $p(y|x,z,\theta) = \mathcal{N}(f(x,z,\theta),\Sigma)$, similar to the deterministic neural network. For a non-Bayesian Neural Network, the model parameter θ is estimated via point estimation by minimizing the regularized negative log-likelihood. In this way, the randomness of model prediction has its origin in both z and Σ, which captures the aleatoric uncertainty of model prediction. In this paper, we discuss four generative models [47], [49], [50] and apply them for uncertainty estimation.

Conditional Variational Autoencoders (CVAE): A CVAE [49] is a conditional directed graph model, which includes three variables, the input x or conditional variable that modulates the prior on the Gaussian latent variable z and generates the output prediction y. Two main modules are included in a conventional CVAE based framework: a generator model $p(y|x,z,\theta)$, which generates prediction y with input x and z as input, and an inference model $p(z|x,y,\theta)$, which infers the latent variable z with input x and output y as input. Learning a CVAEs framework involves approximation of the true posterior distribution of z with an inference model $q(z|x,y,\phi)$, where ϕ is network parameter set of the inference model.

The parameter sets of CVAEs (including both generator model parameters θ and inference model parameters ϕ) can be estimated in Stochastic Variational Bayes (SGVB) [48] framework by maximizing the expected variational lower bound (ELBO) as:

$$\begin{aligned} L(\theta,\phi;x) = {} & \mathbb{E}_{q(z|x,y,\phi)}[\log(p(y|x,z,\theta))] \\ & - D_{KL}(q(z|x,y,\phi)||p(z|x,\theta)), \end{aligned} \tag{24}$$

where the first term is the expected log-likelihood and the second term measures the information lost using $q(z|x,y),\phi$ to approximate the true posterior distribution of latent variable z.

Generative Adversarial Networks (GAN): Similar to the CVAE based model, two different modules (a generator and a discriminator) play the minimax game in GAN based framework as shown below:

$$\begin{aligned} \min_G \max_D V(D,G) = {} & E_{(x,y)\in p_{data}(x,y)}[\log D(y|x)] \\ & + E_{z\in q(z)}[\log(1-D(G(x,z))], \end{aligned} \tag{25}$$

where G and D are the generator model and discriminator model respectively, $p_{data}(x,y)$ is the joint distribution of training data, $q(z)$ is the prior distribution of the latent variable z, which is usually defined as $q(z) = \mathcal{N}(0,\mathbf{I})$. Specifically, the discriminator is designed with a cross-entropy loss to distinguish ground truth y (with supervision as 1) and model prediction $f(x,z,\theta)$ (with supervision as 0).

Alternating Back-propagation: With the CVAE [49] based framework, an extra inference model $q(z|x,y,\phi)$ is used to approximate the true posterior distribution of the latent variable. For the GAN [47] based framework, the discrinimator is designed to estimate the quality of the model prediction. Alternating back-propagation [50] is another type of generative model, which samples the latent variable directly from it's true posterior distribution via Langevin Dynamic based MCMC [55].

Alternating Back-Propagation [50] was introduced for learning the generator network model. It updates the latent variable and network parameters in an EM-manner. Firstly, given the network

prediction with the current parameter set, it infers the latent variable by Langevin dynamics based MCMC, which they call "Inferential back-propagation" [50]. Secondly, given the updated latent variable, the network parameter set is updated with gradient descent. They call it "Learning back-propagation" [50]. Following the previous variable definitions, given the training example (x, y), we intend to infer z and learn the network parameter θ to minimize the reconstruction error as well as a regularization term that corresponds to the prior on z.

As a non-linear generalization of factor analysis, the conditional generative model aims to generalize the mapping from the continuous latent variable z to the prediction y conditioned on the input image x. Similar to traditional factor analysis, we define our generative model as:

$$z \sim q(z) = \mathcal{N}(0, \mathbf{I}), \tag{26}$$
$$y = f(x, z, \theta) + \epsilon, \epsilon \sim \mathcal{N}(0, \Sigma), \tag{27}$$

where $q(z)$ is the prior distribution of z, $\Sigma = \text{diag}(\sigma^2)$ is the inherent label noise. The conditional distribution of y given x is $p(y|x, \theta) = \int q(z)p(y|x, z, \theta)dz$ with the latent variable z integrated out. We define the observed-data log-likelihood as $L(\theta) = \sum_{i=1}^{N} \log p(y_i|x_i, \theta)$, where the gradient of $p(y|x, \theta)$ is defined as:

$$\begin{aligned} \frac{\partial}{\partial \theta} \log p(y|x, \theta) &= \frac{1}{p(y|x, \theta)} \frac{\partial}{\partial \theta} p(y|x, \theta) \\ &= \mathrm{E}_{p(z|x,y,\theta)} \left[\frac{\partial}{\partial \theta} \log p(y, z|x, \theta) \right]. \end{aligned} \tag{28}$$

The expectation term $\mathrm{E}_{p(z|x,y,\theta)}$ can be approximated by drawing samples from $p(z|x, y, \theta)$, and then computing the latent variable z. Following ABP [50], we use Langevin Dynamics based MCMC (a gradient-based Monte Carlo method) to sample z, which iterates:

$$z_{t+1} = z_t + \frac{s^2}{2} \left[\frac{\partial}{\partial z} \log p(y, z_t|x, \theta) \right] + s\mathcal{N}(0, \mathbf{I}), \tag{29}$$

with

$$\frac{\partial}{\partial z} \log p(y, z|x, \theta) = \frac{1}{\sigma^2}(y - f(x, z, \theta)) \frac{\partial}{\partial z} f(x, z, \theta) - z, \tag{30}$$

where t is the time step for Langevin sampling, and s is the step size. Eq. 29 provides an efficient way of updating the latent variable z by directly sampling it from the true posterior distribution $p(z|x, y, \theta)$.

Energy-based Model: Energy-based models (EBM) [56]–[62], which are usually represented as a deep neural networks, aim to learn an energy function, which assigns low energy to in-distribution samples and vice versa. In this way, the energy function is designed to estimate the compatibility of the input variable x and the output variable y.

Let's define the energy-based model as $p(y|x, \theta)$, which represents a distribution of prediction y given an image x by:

$$p(y|x, \theta) = \frac{p(y, x, \theta)}{\int p(y, x, \theta)dy} = \frac{1}{Z(x; \theta)} \exp[-U(y, x, \theta)], \tag{31}$$

where the energy function $U(y, x, \theta)$, parameterized by a bottom-up neural network with parameters θ, maps the input/output pair to a scalar. $Z(x; \theta) = \int \exp[-U(y, x, \theta)]dy$ is the normalizing constant. When the energy function U is learned and an image x is given, the model prediction y can be achieved by Langevin sampling [55] $y \sim p(y|x, \theta)$, which makes use of the gradient of the energy function and iterates the following step:

$$y_{\tau+1} = y_\tau - \frac{\delta^2}{2} \frac{\partial U(y_\tau, x, \theta)}{\partial y} + \delta\Delta_\tau, \Delta_\tau \sim N(0, \Sigma), \tag{32}$$

where τ indexes the Langevin time steps, and δ is the step size. Langevin dynamics [55] is equivalent to a stochastic gradient descent algorithm that seeks to find the minimum of the objective function defined by $U(y, x, \theta)$. The Gaussian noise term Δ_τ is a Brownian motion that prevents gradient descent from being trapped by local minima of $U(y, x, \theta)$.

Two main issues exist in the above energy based model. Firstly, we need to set a start point y_0 for the Langevin sampling process as in Eq. 32. Secondly, we should decide whether to fix the start point (cold start) or iteratively update it (warm start) while training the energy function $U(y, x, \theta)$. Following the cold start solution, we can train a deterministic network and fixed it while training the energy function $U(y, x, \theta)$, or we can define y_0 as random noise and learn $U(y, x, \theta)$ to gradually update it. Although random initialization is easy to define, its quite slow during training [59]. For the warm start, the initial prediction (start point) generator can be a deterministic network or latent variable model, which produces initial prediction, and then the energy function $U(y, x, \theta)$ updates the initial prediction by running several steps of Langevin sampling via Eq. 32. The latent variable model as warm start solution has been discussed in [59].

For both cold start and warm start, we need to update the task related generator (y_0 generator) and the energy function $U(y, x, \theta)$. The y_0 generator can be learned following the deterministic learning pipeline or latent variable learning pipeline depending on its framework. The energy function $U(y, x, \theta)$ can be updated via MLE:

$$\Delta\theta \approx \frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial \theta} U(\tilde{y}_i, x_i, \theta) - \frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial \theta} U(y_i, x_i, \theta), \tag{33}$$

where $\tilde{y}$ is the updated prediction from the start point y_0 with several steps of Langevin sampling.

Uncertainty computation: As we have no distribution assumption for model parameters within the generative models, we cannot estimate epistemic uncertainty within generative models. We can only estimate the aleatoric uncertainty (which is the total uncertainty or predictive uncertainty in this circumstance) by sampling from the distribution of the latent variable, and define the mean entropy of the predictions as aleatoric uncertainty.

2.3 Bayesian Latent Variable Model

As discussed, the Bayesian Neural Network is effective in modeling the epistemic uncertainty, and the generative model makes it easier to estimate the aleatoric uncertainty. To achieve both types uncertainty estimation, a Bayesian latent variable model (BLVM) can be designed following [25], [63]. Within the Bayesian latent variable model, the predictive distribution is defined as:

$$p(y|x) = \int p(y|x, z, \theta)p(\theta)p(z)d\theta dz, \tag{34}$$

where $p(y|x, z, \theta) = \mathcal{N}(f(x, z, \theta), \Sigma))$ is the likelihood as in Eq. 23. Different from Eq. 23 where $p(\theta)$ is constant, indicating a non-Bayesian Neural Network. Within the BLVM, $p(\theta)$ is the approximate posterior distribution of $p(\theta|D)$. In this way, the randomness of y comes from three parts: 1) the approximated

posterior distribution $p(\theta)$; 2) the latent variable z and 3) the inherent label noise Σ, where $p(\theta|D)$ contributes to epistemic uncertainty and the other two contribute to aleatoric uncertainty.

Given the Bayesian latent variable model in Eq. 34, the total uncertainty (predictive uncertainty) is quantified as $U_p = H(y|x)$. [25] defines the other two types of uncertainty as:

$$U_a = \mathbf{E}_{p(\theta)}[H(y|x,\theta)], \tag{35}$$

and

$$U_e = U_p - U_a = H(y|x) - \mathbf{E}_{p(\theta)}[H(y|x,\theta)] = I(y,\theta|x). \tag{36}$$

$H(y|x,\theta)$ is the entropy of $p(y|x,\theta)$, which is defined as: $p(y|x,\theta) = \int p(y|x,\theta,z)p(z)dz$. For a specific model parameter θ, we obtain model prediction $p(y|x,\theta)$, and compute its entropy. Then, the expectation of $p(y|x,\theta)$ under $p(\theta)$ is used to quantify uncertainty come from z and inherent noise Σ, which is the aleatoric uncertainty. Epistemic uncertainty is then the residual of predictive uncertainty and aleatoric uncertainty, which is the mutual information of y and θ conditioned on the input x.

[25] introduces variance as another type of uncertainty measure, which defines the total uncertainty as variance of the predictive distribution, which is $U_p = \sigma^2(y|x)$, where $\sigma^2(.)$ computes the variances of a probability distribution. Based on the law of total variance:

$$\sigma^2(y|x) = \sigma^2_{p(\theta)}(\mathbf{E}[y|x,\theta]) + \mathbf{E}_{p(\theta)}[\sigma^2(y|x,\theta)], \tag{37}$$

where $\mathbf{E}[y|x,\theta]$ and $\sigma^2(y|x,\theta)$ are the mean and variances of prediction according to $p(y|x,\theta)$. Specifically, we fixed θ, and sample z to obtain the mean and variance of $p(y|x,\theta)$. $\sigma^2_{p(\theta)}(\mathbf{E}[y|x,\theta])$ is then the variance of the mean when $\theta \sim p(\theta)$, which ignores the contribution of z and inherent noise Σ, making it the epistemic uncertainty, which is:

$$U_e = \sigma^2_{p(\theta)}(\mathbf{E}[y|x,\theta]). \tag{38}$$

Further, $\mathbf{E}_{p(\theta)}[\sigma^2(y|x,\theta)]$ is the mean variance when $\theta \sim p(\theta)$, which represents uncertainty from z and Σ, and is defined as aleatoric uncertainty, which is:

$$U_a = \mathbf{E}_{p(\theta)}[\sigma^2(y|x,\theta)]. \tag{39}$$

In practice, to compute aleatoric uncertainty U_a, with fixed θ, we sample from the latent space multiple times and obtain the mean prediction $\mathbf{E}[y|x,\theta]$ and the variance $\sigma^2(y|x,\theta)$ of the multiple predictions. Then, with $\theta \sim p(\theta)$, we define the mean of above variance ($\mathbf{E}_{p(\theta)}[\sigma^2(y|x,\theta)]$) as aleatoric uncertainty, and define variance of multiple $\mathbf{E}[y|x,\theta]$ with $\theta \sim p(\theta)$ as the epistemic uncertainty.

3 Bayesian Latent Variable Model for Efficient Dense Uncertainty Estimation

We analyse existing uncertainty estimation solutions and introduce an efficient uncertainty estimation methods.

3.1 Trivial Solution for Aleatoric Uncertainty

As discussed in Section 1.1, the loss function with noise corruption model for regression task is achieved as [7], [27]:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\frac{1}{2\sigma^2(x_i)}\mathcal{L}_2 + \frac{1}{2}log(\sigma^2(x_i))), \tag{40}$$

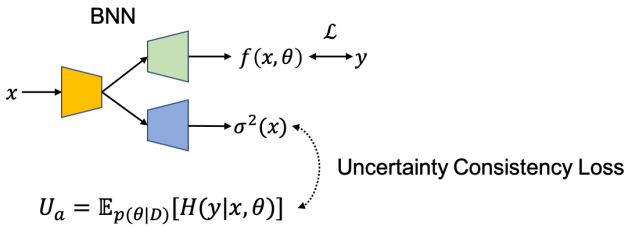

Fig. 1. Uncertainty consistency loss within BNN to avoid trivial solution of aleatoric uncertainty from the dual-head structure.

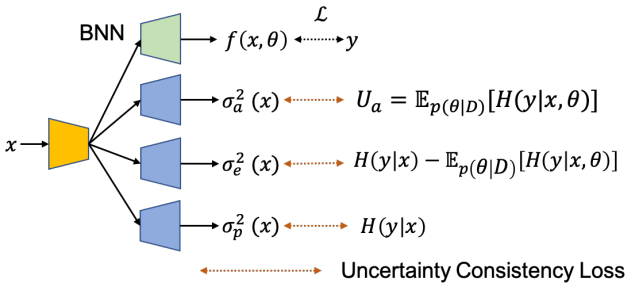

Fig. 2. Approximating uncertainty with multi-head uncertainty consistency loss within BNN.

and for classification task is achieved as:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\frac{1}{T_i}\mathcal{L}_{ce} + \log T_i), \tag{41}$$

where $T = \exp(\sigma^2(x))$, representing the temperature as in model calibration [1]. As there is no extra supervision or constraints for the aleatoric uncertainty from the dual-head framework, we observe one trivial solution of aleatoric uncertainty for both the regression tasks and classification tasks. For the former, the $\sigma^2(x) = 1$ will lead to the original loss function, $\mathcal{L}_2$ in particular, resulting in image-independent uncertainty. For the latter, $\sigma^2(x) = 0$ will also lead to the original cross-entropy loss $\mathcal{L}_{ce}$. To avoid the trivial solution, we can relate the two definitions of aleatoric uncertainty from both the dual-head structure [7], [27] and mean entropy of prediction within the Bayesian Neural Network (BNN) as shown in Fig. 1. Specifically, we define uncertainty consistency loss within a BNN, which is a cross-entropy loss or a symmetric loss function, *i.e.* SSIM [64], [65] to regularize the aleatoric uncertainty from dual head to be similar to the sampling based uncertainty as in Eq. 13. The main advantage of this strategy is that, the trained dual-head based aleatoric uncertainty estimation module can achieve sampling-free uncertainty estimation during testing.

3.2 Efficient Uncertainty Estimation within BLVM

[25] define three origins of uncertainty within BNN: 1) inherent noise Σ; 2) latent variable z and 3) the approximated posterior distribution $p(\theta)$, where the first two are claimed as the origins of aleatoric uncertainty and the last one is related to epistemic uncertainty. For both types of uncertainty estimation and the total uncertainty, the expectation term indicates that the sampling process is usually required to estimate the three types of uncertainties, which is less efficient during testing.

Inspired by the dual-head structure for aleatoric uncertainty estimation, we introduce uncertainty approximation with two extra heads for epistemic uncertainty estimation and predictive uncertainty estimation as shown in Fig. 2. In this way, we avoid the tedious sampling process, leading to efficient uncertainty estimation during testing.

3.3 Uncertainty Computation

4 Experimental Results

4.1 Setup

Dataset: We perform experiments on two binary segmentation tasks, namely salient object detection [66]–[70] and camouflaged object detection [71]–[76], and two regression tasks, namely monocular depth estimation [77]–[79] and image deblurring [80]–[82], to verify the effectiveness of each uncertainty estimation method. Salient object detection aims to localize the region that attract human attention. Camouflaged object detection models, on the contrary, are designed to identify the camouflaged objects hiding in the environment, which usually share similar pattern as their surroundings. Monocular depth estimation is a well-studied task, aiming to estimate depth from a single view RGB image. Image deblurring techniques try to recover a sharp image from its blurred version.

For salient object detection, we used the DUTS dataset [83] for training. Testing images include 1) DUTS testing dataset, 2) DUT [84], 3) HKU-IS [85] and 4) PASCAL-S [86]. For camouflaged object detection, the benchmark training dataset is the combination of 3,040 images from COD10K training dataset [76] and 1,000 images from CAMO training dataset [87]. We then test our model on four benchmark testing datasets, namely CAMO testing dataset [87], COD10K testing dataset [76], CHAMELEMON dataset [88] and NC4K dataset [89].

For monocular depth estimation, the typical training and testing strategy is that the training and testing images should come from the same dataset as in [78]. [77] presents a mixed training dataset to achieve zero-shot cross-dataset transfer. Following the same training dataset setting, [79] also train with the mixed large training dataset, and fine-tune on the other dataset for performance evaluation. In this paper, we adopt both settings, 1) following [78], we train and test on NYUv2 [90] and KITTI dataset [91], with the backbone network initialized with parameters for image classification on ImageNet; 2) following [77], the backbone is initialized with parameters trained for monocular depth estimation with the mixed large training dataset, and then fine-tune further on NYUv2 [90] and KITTI dataset [91] for performance evaluation.

For image deblurring, the training samples are 2103 pairs of blurry images and the corresponding ground truth sharp images from the GoPro training dataset [80]. The testing images include 1) 1111 pairs from the GoPro testing dataset. 2) 2025 pairs from the HIDE testing dataset [81]. 3) 4937 pairs from the RealBlur-J dataset [82]. The blurry images from GoPro and HIDE datasets are synthesized via averaging continuous frames. The blurry images from RealBlur-J testing dataset are captured in the real world.

Metrics: As an uncertainty estimation network, we aim to produce both task related prediction and the corresponding uncertainty map representing model awareness about it's prediction. For the former, the task related metric can be used, *i.e.*we use mean F-measure and mean absolute error (MAE) for both the salient object detection and camouflaged object detection task. For monocular depth estimation, following the conventional solution [77], [79], we use the percentage of the pixel with $\delta = max(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i} > 1.25)$ to evaluate model performance. For image deblurring, the candidate metrics are PSNR, SSIM [64] and LPIPS [92]. In this paper, we use the two widely used metrics, which are PSNR and SSIM [64].

For uncertainty estimation, we use the metric from [93]. In [93], they define two conditional probabilities: 1) $p(accurate|certainty)$: the probability that the model is accurate given that it is confident; and 2) $p(uncertainty|inaccurate)$: the probability that the model is uncertainty given that it is inaccurate. The basic assumption is that for certain pixels (or regions), the model should have accurate predictions and for inaccurate predictions, the corresponding certain level should be low.

To implement above metric, [93] introduce three scores, namely $p(accurate|certainty)$, $p(uncertainty|inaccurate)$ and Patch accuracy vs patch uncertainty (PAvPU), which has been used in [94] for uncertainty evaluation with the instance segmentation task. For each patch s_k, [93] computes the patch accuracy a_k (pixel accuracy metric defined in [95]). Then, [93] decides whether the patch is accurate based on certain threshold h_a, *e.g.*prediction of s_k is accurate if $a_k > h_a$. Similarly, the uncertainty of patch s_k is achieved if its uncertainty $u_k > h_u$, where h_u is a confidence related threshold. Further, they define four metric related variables: 1) n_{ac} is the number of patches which are accurate and certain; 2) n_{au} is the number of patches which are accurate and uncertain; 3) n_{ic} is the number of patches which are inaccurate and certain; and 4) n_{iu} is the number of patches which are inaccurate and uncertain. Finally, three uncertainty evaluation scores are generated as:

$$p(accurate|certainty) = \frac{n_{ac}}{(n_{ac} + n_{ic})}, \tag{42}$$

$$p(uncertain|inaccurate) = \frac{n_{iu}}{(n_{ic} + n_{iu})}, \tag{43}$$

and

$$PAvPU = \frac{(n_{ac} + n_{iu})}{(n_{ac} + n_{au} + n_{ic} + n_{iu})}. \tag{44}$$

A model with higher above scores is a better performer.

Instead of defining a uniform patch as in [93], we over-segment image to super-pixels to achieve uniform accuracy/uncertainty with in a same super-pixel considering the semantic consistency attribute. Further, the three above scores are based on hand-crafted hyper-parameters, namely accuracy related threshold h_a and uncertainty related threshold h_u. Inspired by the expected calibration error [1], we use binning instead of hard-thresholding. Specifically, we use threshold in the range of $[0, 1]$ with 10 bins, and the final n_{ac}, n_{au}, n_{ic}, n_{iu} are all 10-dimensional vectors, as well as the $p(accurate|certainty)$, $p(uncertain|inaccurate)$ and $PAvPU$. We then define mean $p(accurate|certainty)$, $p(uncertain|inaccurate)$ and $PAvPU$ for uncertainty performance evaluation.

Note that, the pixel accuracy [95] a_k is defined as follows. Given n_{ij} as number of pixels of class i predicted as class j, and $t_i = \sum_j n_{ij}$ is the total number of pixels of class i, the pixel accuracy is defined as:

$$pixel\_acc = \sum_i n_{ii} / \sum_i t_i \tag{45}$$

For binary segmentation:

$$pixel\_acc = \frac{(n_{ff} + n_{bb})}{H * W} \tag{46}$$

where H and W are image height and width.

4.2 Implementation Details

We design both ensemble based models, generative model based models and Bayesian latent variable model based models for the four classification and regression tasks.

Task related generator: We trained the salient object detection, camouflaged object detection and monocular depth estimation using PyTorch with a maximum of 30 epochs. ResNet50 [96] is chosen as backbone, and we choose the decoder from [77] as our decoders for the three tasks, which gradually aggregates higher level features with lower level features with residual connections. For both salient object detection and camouflaged object detection, we adopt the structure-aware loss function [68]. For monocular depth estimation, following [97], we apply the loss function as the weighted sum of point-wise L1 loss, gradient loss and SSIM loss on inverse-depth predictions. Different from above tasks, which are usually built upon backbone networks [96], [98], image deblurring methods do not rely on existing backbone networks. Among these methods, the multi-scale networks DeepDeblur [80] is one of the most popular deblurring methods, and the multi-stage network MPRNet [99] is the state-of-the-art method. Therefore, we conduct uncertainty estimation experiments based on them. Models are trained based on the PyTorch framework with a maximum of 200 epochs. The loss functions are the same as the original papers.

Uncertainty estimation models: Given the backbone features $\{s_i\}_{i=1}^4$ for the chosen ResNet50 [96] backbone[2], to achieve MC-dropout [18] ("MD"), we add a dropout of rate 0.3 before $\{s_i\}_{i=1}^4$, and then feed the features after dropout to the decoder. For the deep ensemble [17] models ("DE"), we attach five decoders of the same structure after the backbone feature to generate five different predictions. For the snapshots ensembles [42] ("SE"), we save the model after five epochs, and generate prediction with each snapshot model.

For the CVAE [49] based framework ("CVAE"), we use two extra encoders to generate the prior and posterior distribution of the latent variable, which share the same network structures as in [100]. For the GAN [47] based framework ("GAN"), we design a fully convolutional discriminator as in [101]. For the ABP [50] based framework ("ABP"), we update the latent variable via Langevin Dynamics [55] as shown in Eq. 29. For the EBM [60] based framework ("EBM"), we design the same energy-based model as in [102], and we then generate multiple predictions via Eq. 32.

For the Bayesian latent variable model, we simply apply MC-dropout [18] to the CVAE, GAN and ABP based framework, leading to a Bayesian model "BCVAE", "BGAN", "BABP" respectively. As an extension, we extend the designed energy-based model to the latent variable based EBM, where the prediction from the three latent variable model serve as the start point for the energy function based Langevin Dynamics as in Eq. 32, leading to "ECVAE", "EGAN", "EABP" respectively

For all the generative models, empirically, we set the dimension of the latent space as $K = 8$, which is introduced to the network by concatenating with the highest level backbone feature, *i.e.*s_4 in this paper.

2. We use backbone network for salient object detection, camouflaged object detection and monocular depth estimation.

TABLE 1
Ensemble based solutions for **camouflaged object detection**. ↑ indicates the higher the score the better, and vice versa for ↓.

Method	CAMO [87]		CHAMELEON [88]		COD10K [76]		NC4K [89]	
	$F_\beta \uparrow$	$\mathcal{M} \downarrow$	$F_\beta \uparrow$	$\mathcal{M} \downarrow$	$F_\beta \uparrow$	$\mathcal{M} \downarrow$	$F_\beta \uparrow$	$\mathcal{M} \downarrow$
Base	.757	.079	.848	.029	.731	.035	.803	.048
MD	.767	.080	.842	.028	.731	.035	.803	.048
DE	.729	.088	.846	.030	.718	.037	.796	.051

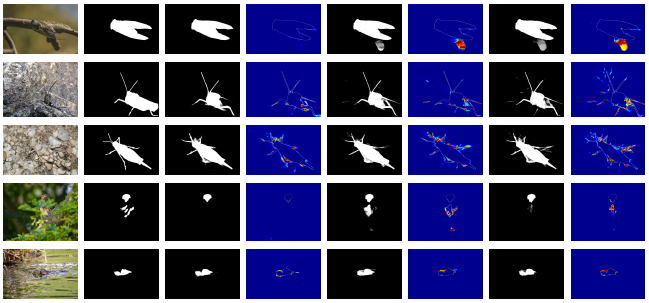

Fig. 3. Predictive uncertainty of ensemble based solutions for **camouflaged object detection**.

4.3 Task Related Uncertainty Analysis

Uncertainty for Camouflaged Object Detection: Camouflaged object detection [76], [87] aims to localize the camouflaged objects within an given input RGB image. Due to the complex attributes that makes the object camouflaged, *i.e.* color, texture, and *etc.* , and also the similar pattern of camouflaged object compared with its surrounding, there exists both task related ambiguity and labeling level ambiguity for camouflaged object detection [103], where the former can be modeled as epistemic uncertainty and the latter can be represented as aleatoric uncertainty.

Uncertainty for Salient Object Detection: Saliency is defined as the attribute that make the object distinct from its surrounding [104], [105]. Many factors can lead something to be "salient", including the stimulus itself that makes the item distinct, *i.e.* color, texture, direction of movement and *etc.* , and the internal cognitive state of the observer, leading to his/her understanding of saliency. The "subjective nature" [100], [102] of saliency leads to the ambiguity in both task understanding and labeling, which can be explained as epistemic uncertainty and aleatoric uncertainty respectively.

Uncertainty for Monocular Depth Estimation: Monocular depth estimation aims to estimate depth of a scene from a single RGB image [77], [106]. As a distance perception strategy, the accuracy and model's confidence in its prediction is important for real-life applications, *i.e.* it is essential for an autonomous driving system to understand when it makes a mistake [28], [107]–[111]. In this way, uncertainty estimation techniques that can produce both accurate prediction and reliable uncertainty map are necessary for monocular depth estimation to be used in real-life scenarios.

Uncertainty for Image Deblurring: Image deblurring [112] aims to recover a sharp image from its blurred version. Due to the difficulty in recovering the detail information of objects of different categories, there exists inherent uncertainty for image deblurring [113]–[115]. Further, as there exists ambiguity in estimating the true blur kernel space that leads to the degraded

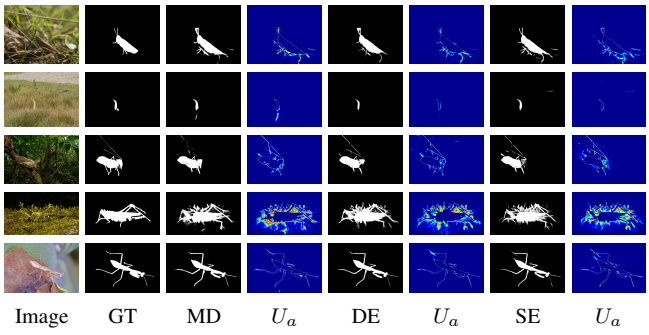

Fig. 4. Aleatoric uncertainty of ensemble based solutions for **camouflaged object detection**.

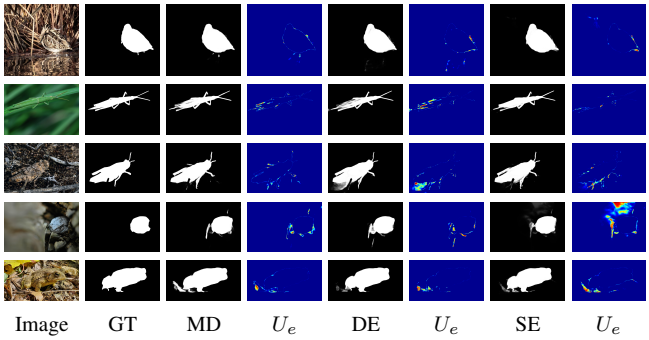

Fig. 5. Epistemic uncertainty of ensemble based solutions for **camouflaged object detection**.

TABLE 2
Ensemble based solutions for **salient object detection**. ↑ indicates the higher the score the better, and vice versa for ↓.

Method	DUTS [83]		DUT [84]		HKU-IS [85]		PASCAL [86]	
	$F_\beta \uparrow$	$\mathcal{M} \downarrow$	$F_\beta \uparrow$	$\mathcal{M} \downarrow$	$F_\beta \uparrow$	$\mathcal{M} \downarrow$	$F_\beta \uparrow$	$\mathcal{M} \downarrow$
Base	.842	.037	.760	.055	.904	.030	.828	.064
MD	.854	.036	.763	.056	.911	.028	.840	.061
DE	.828	.040	.738	.061	.897	.031	.825	.065

blurring image, it is necessary to estimate the uncertainty within the image deblurring task.

4.4 Uncertainty Estimation

We estimate uncertainty for each task, and show experimental results for each task. Specifically, we analyse the ensemble based uncertainty estimation techniques, the generative model based techniques and the Bayesian latent variable model based solutions.

4.4.1 Ensemble based uncertainty estimation

We show the base model performance as "Base" in Table 1, which is a deterministic neural network with no ensemble strategies or generative model for uncertainty estimation. Then, we introduce MC-dropout [18] and deep ensemble [17] to the base model respectively, and show the model performance as "MD" and "DE". The relatively stable deterministic performance of "MD" compared with the base model illustrate the MC-dropout can keep the original deterministic prediction accuracy. However, we observe slightly worse performance of the deep ensemble solution. For the MC-dropout based model, although we perform random weights erasing, due to the huge capacity of the deep

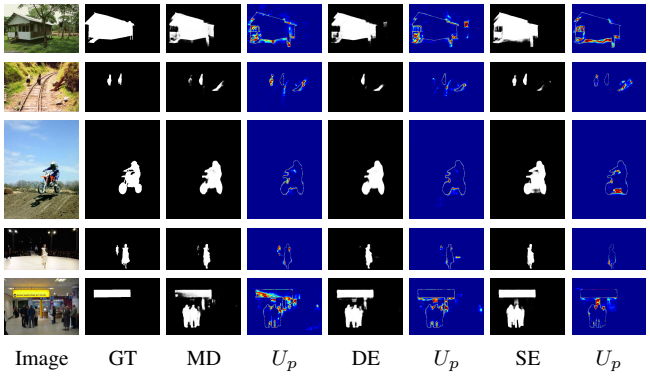

Fig. 6. Predictive uncertainty of ensemble based solutions for **salient object detection**.

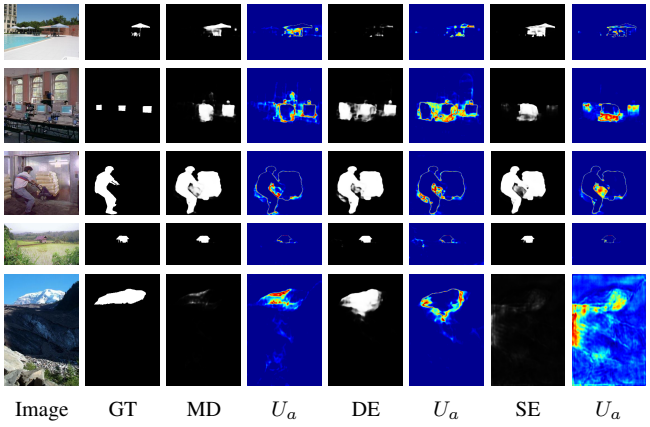

Fig. 7. Aleatoric uncertainty of ensemble based solutions for **salient object detection**.

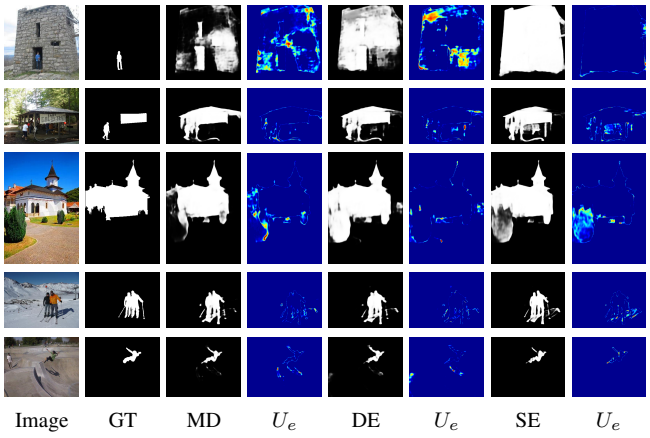

Fig. 8. Epistemic uncertainty of ensemble based solutions for **salient object detection**.

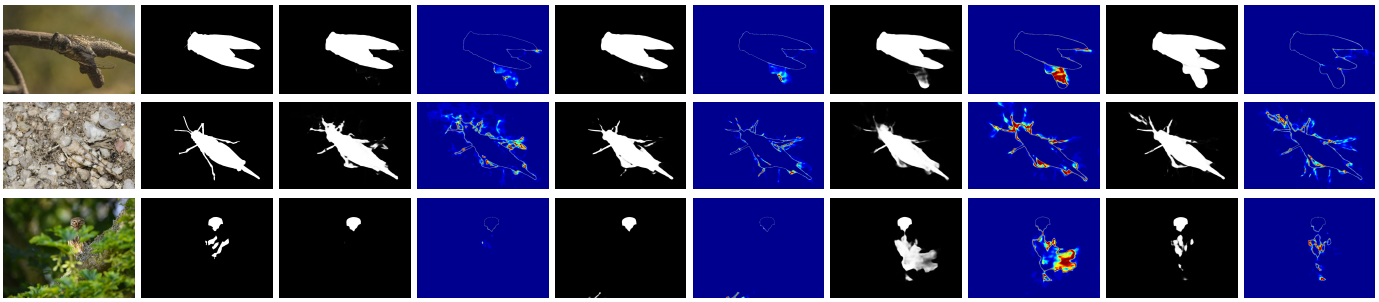

Fig. 9. Predictive uncertainty of generative model based solutions for **camouflaged object detection**.

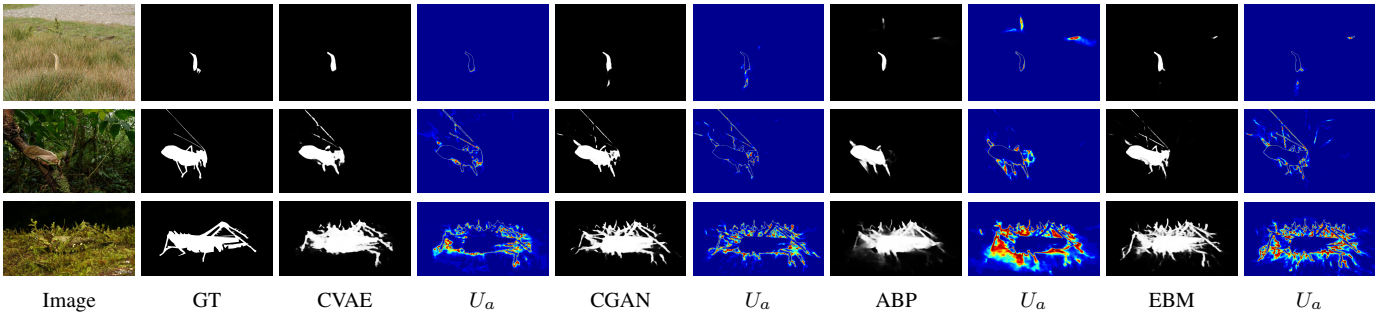

Fig. 10. Aleatoric uncertainty of generative model based solutions for **camouflaged object detection**.

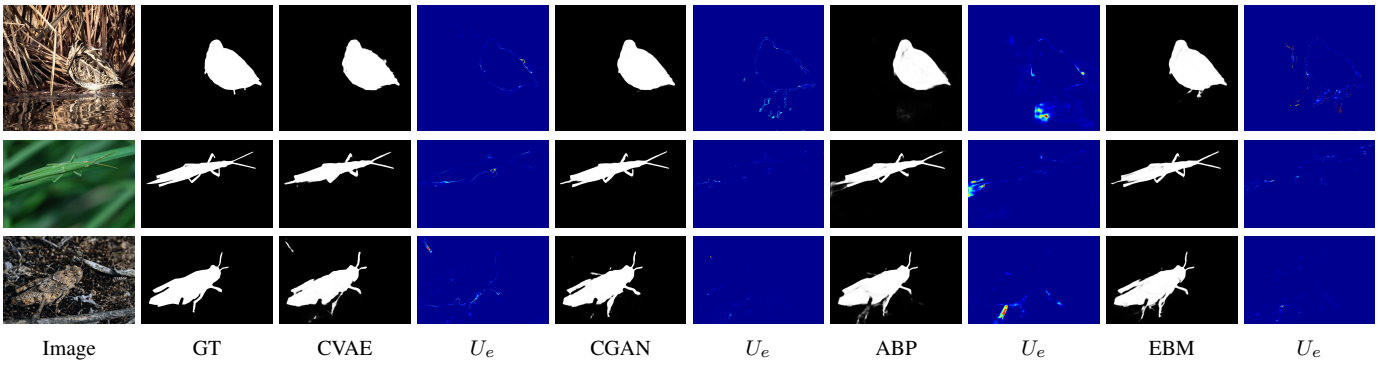

Fig. 11. Epistemic uncertainty of generative model based solutions for **camouflaged object detection**.

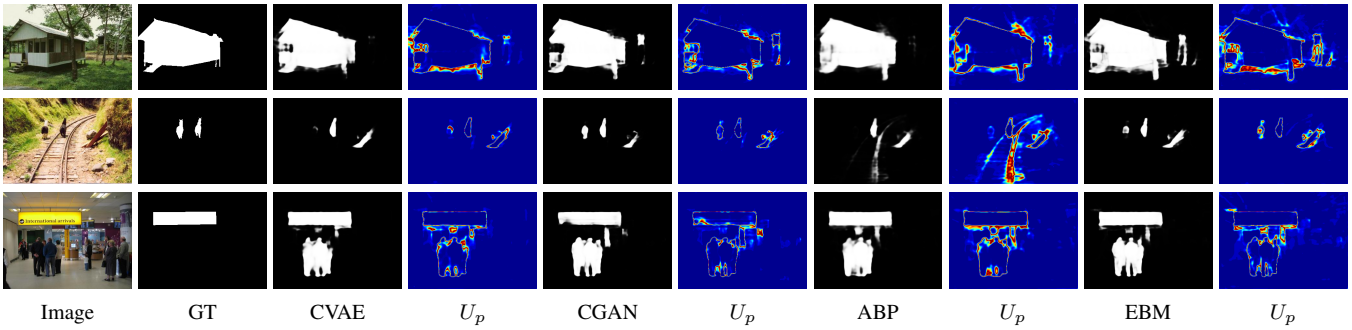

Fig. 12. Predictive uncertainty of generative model based solutions for **salient object detection**.

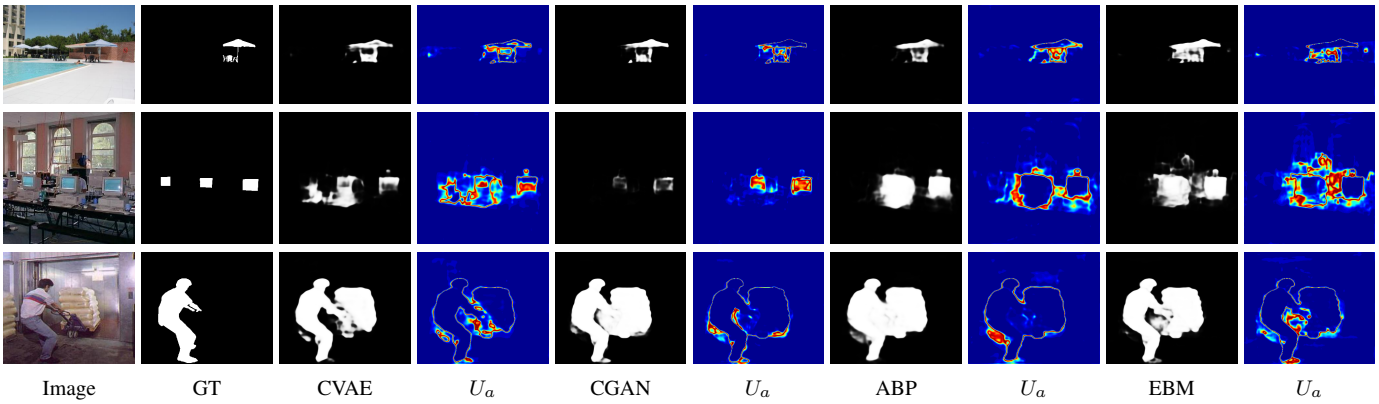

Fig. 13. Aleatoric uncertainty of generative model based solutions for **salient object detection**.

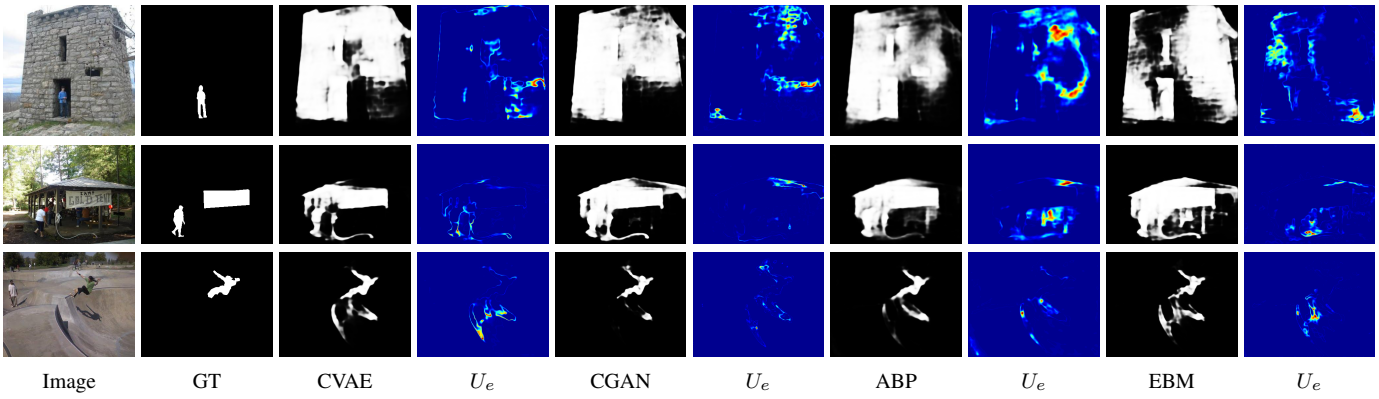

Fig. 14. Epistemic uncertainty of generative model based solutions for **salient object detection**.

network, the final model may not deviate too much from the base model with the same initialization [96]. Within deep ensemble model, we attached five decoders [77] of the same structure but different initialization to generate five predictions. The different initialization of deep ensemble model compared with the base model explains their different predictions. As discussed in [116], model with more random initialization may need longer training time. As we fixed the maximum epoch for the ensemble based models, the deep ensemble based model may not converge to the best performance, which explians the slightly worse performance of the deep ensemble solution.

We further show the uncertainty maps of the ensemble based solutions for camouflaged object detection in Fig. 3, Fig. 4 and Fig. 5, representing the predictive uncertainty, aleatoric uncertainty and epistemic uncertainty respectively. The uncertainty maps indicate that aleatoric uncertainty usually focus on object boundary, where the labeling noise usually occurs. The epistemic uncertainty has less activation region than the aleatoric uncertainty, which usually highlight the hard regions with camouflage attributes that are not learned with the current training dataset.

Similarly, we train ensemble based salient object detection model and show model performance in Table 2, and uncertainty maps in Fig. 6, Fig. 7 and Fig. 8 respectively, representing the predictive uncertainty, aleatoric uncertainty and epistemic uncertainty. The conclusion is similar to the camouflaged object detection task.

4.4.2 *Generative model based uncertainty estimation*

We introduce four generative model based uncertainty estimation techniques, including three latent variable models, namely CVAE based framework [49], GAN based framework [47] ABP based framework [50], and one energy-based model [59], [60] for prediction distribution estimation. We apply those generative model based solutions to camouflaged object detection and salient object detection, and show performance in Table 3 and Table 4 respectively. We observe similar performance of those generative model based methods compared with the base models. In general, the GAN [47] based framework achieves slightly better performance than the other generative models. The main reason is the adversarial loss term within the generator that serves as higher-order similarity measure.

Further, we show uncertainty maps of the generative model based methods for both camouflaged object detection and salient object detection in Fig. 9, Fig. 10, Fig. 11, Fig. 12, Fig. 13 and Fig. 14. Although the ABP [50] and EBM [59], [60] based solutions performs worse than the GAN [47] based model for deterministic performance evaluation as shown in Table 3 and Table 4, the ABP [50] and EBM [59], [60] based solutions can generate better uncertainty map. The main reason is ABP samples from the true latent variable posterior distribution, and EBM samples from the true predictive distribution, making them suffer less from the posterior collapse issue [117] as VAE [48] or GAN [47] based framework.

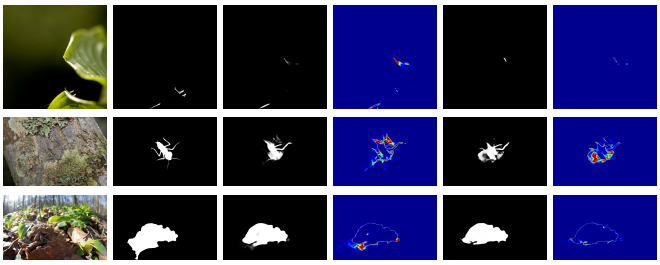

Fig. 15. Predictive uncertainty maps comparison between the MC-dropout based solution and CGAN based solution for **camouflaged object detection**.

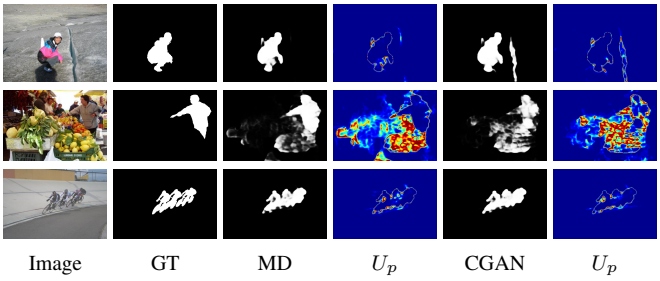

Fig. 16. Predictive uncertainty maps comparison between the MC-dropout based solution and CGAN based solution for **salient object detection**.

4.4.3 *Ensemble vs Generative Model*

We further compare the predictive uncertainty of the ensemble based uncertainty estimation method, MC-dropout [18] in particular, with the generative model based uncertainty estimation method (CGAN), and show uncertainty maps for the camouflaged object detection and salient object detection tasks in Fig. 15 and Fig. 16 respectively. As the two types of uncertainty estimation techniques focus on different origins of uncertainties, we only show the predictive uncertainty to explain how the model can be aware about it's prediction when it makes mistakes.

4.4.4 *Bayesian latent variable based uncertainty estimation*

We discussed the Bayesian latent variable model in Section 2.3, and present the Bayesian latent variable model for uncertainty estimation in Section 4.2. We then propose BCVAE, BGAN, BABP and BEBM, representing the Bayesian conditional variational auto-encoder, Bayesian generative adversarial network, Bayesian alternating back-propagation and Bayesian energy-based model respectively. As an extension, we also introduce EBM as refinement and attach it to BCVAE, BAGN and BABP and achieve ECVAE, EGAN and EABP with several steps of Langevin sampling, where the prediction from BCVAE, BGAN or BABP serves as start point (warm start) for the prediction Langevin sampling as in Eq. 32.

5 Discussion

Due to the close relationship of uncertainty estimation and model calibration, we discuss how the uncertainty estimation techniques can be useful for model calibration. Further, as an effective measure of model calibration, we explain how model performance on out-of-distribution detection can be an effective measure for uncertainty estimation. We then discuss the effective strategy of applying uncertainty estimation for model calibration by measuring it's performance on out-of-distribution samples.

TABLE 3
Generative model based solutions for **camouflaged object detection**. ↑ indicates the higher the score the better, and vice versa for ↓.

Method	CAMO [87]		CHAMELEON [88]		COD10K [76]		NC4K [89]	
	$F_\beta \uparrow$	$\mathcal{M} \downarrow$	$F_\beta \uparrow$	$\mathcal{M} \downarrow$	$F_\beta \uparrow$	$\mathcal{M} \downarrow$	$F_\beta \uparrow$	$\mathcal{M} \downarrow$
Base	.757	.079	.848	.029	.731	.035	.803	.048
CVAE	.758	.081	.848	.030	.731	.034	.802	.048
CGAN	.762	.080	.852	.026	.730	.034	.807	.048
ABP	.756	.081	.846	.030	.729	.034	.801	.047
EBM	.777	.076	.844	.031	.721	.038	.796	.050

TABLE 4
Generative model based solutions for **salient object detection**, ↑ indicates the higher the score the better, and vice versa for ↓.

Method	DUTS [83]		DUT [84]		HKU-IS [85]		PASCAL [86]	
	$F_\beta \uparrow$	$\mathcal{M} \downarrow$	$F_\beta \uparrow$	$\mathcal{M} \downarrow$	$F_\beta \uparrow$	$\mathcal{M} \downarrow$	$F_\beta \uparrow$	$\mathcal{M} \downarrow$
Base	.842	.037	.760	.055	.904	.030	.828	.064
CVAE	.836	.037	.748	.055	.901	.030	.826	.063
CGAN	.846	.035	.752	.054	.905	.029	.828	.063
ABP	.829	.040	.740	.059	.889	.034	.818	.068
EBM	.834	.040	.744	.062	.900	.031	.829	.064

5.1 Uncertainty Estimation and Model Calibration

[118], [119] discovered that modern deep neural networks are prone to producing overconfident predictions. [120] further proved that a deep neural network can fit random noise with high confidence. The over-confident behavior of current deep neural network is caused by the model's ignorance of what it does not know. A well-calibrated network should produce lower confidence prediction for high uncertain samples. Model calibration (confidence calibration) [121] is then defined as the problem of predicting probability estimates representative of the true correctness likelihood [119]. There mainly exists two different directions for model calibration: 1) relaxing the supervision signal [122]; 2) smoothing the network prediction [119]. The former one smooths the ground truth with a uniform distribution, while the latter one use temperature scaling to produce soften prediction. From our perspective, both above strategies can to some extent produce a better calibrated network. The main disadvantage of those strategies is that they treat each sample equally, without considering about image-level or pixel-level uncertainty. Further, the final model calibration is usually obtained with decreased accuracy as sacrifice. We argue that a well-calibrated model should maintain high confidence prediction for confident samples/regions while lower their confidence for uncertain samples/regions. In this way, a pixel-level confidence map is desirable for model calibration to maintain the original model precision.

5.2 Model Calibration and Out-of-Distribution Detection

Out-of-distribution (OOD) detection aims to find samples whose distribution is different from the training data distribution [123]–[125], and thus can be treated as a technique of modeling epistemic uncertainty [7]. Conventional definition of out-of-distribution samples for image-level classification or semantic segmentation can be summarized as category bias, where images with instances

that do not belong to the existing categories are defined as out-of-distribution samples. As OOD highlights the distribution shift, another definition for OOD samples focus on dataset shift [126], where the joint distribution of inputs and outputs differs between the training and testing datasets. In this way, how the model perform on OOD samples can be treated as effective model calibration measure, as our basic assumption is that a well-calibrated model should has highly confident and accurate predictions for in-distribution samples, and less confident predictions for the out-of-distribution samples.

For category-aware segmentation, the OOD samples can be easily generated as samples containing objects of different categories. However, for category-agnostic segmentation, the OOD samples are different from the in-distribution samples in attribute-level. As we are not aware of the true joint data distribution $p(x, y)$, in practice, we can then generate OOD samples by finding samples with different appearance, context, or other attribute-level information. Then, how the model perform on these OOD samples can serve as effective measure for it's calibration level.

5.3 Uncertainty Estimation and Out-of-Distribution Detection

Existing deep model calibration mainly focus on image-level classification problems [119], [122], [127]–[129] without considering sample-level or pixel-level uncertainty. We work on combining uncertainty estimation with model calibration to generate reasonable prediction with reliable uncertainty map for out-of-distribution samples. The first step for our task is to estimate pixel-wise uncertainty with either the deep ensemble solutions [17], [18], [34]–[39] or generative model solutions [47]–[50], [59], [60]. The second step is to achieve model calibration with the produced uncertainty as guidance, which can be treated as a post-hoc technique or designed as online learning strategy. The final step will be combining above two steps in a unified framework to achieve dense model calibration based on uncertainty estimation and evaluate model performance on out-of-distribution samples.

6 Conclusion

In this paper, we investigate uncertainty estimation, especially ensemble based solutions for epistemic uncertainty estimation and generative model based solutions for aleatoric uncertainty estimation. We provide a detailed introduction and analysis for each solution along with implementation details. We claim that uncertainty estimation is necessary for any machine learning model to explain its prediction, thus avoid over-confident predictions.

References

[1] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proc. Int. Conf. Mach. Learn.*, pp. 1321–1330, 2017.

[2] V. N. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, 1998.

[3] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, p. 448–456, 2015.

[4] R. Gencay and M. Qi, "Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging," *IEEE Transactions on Neural Networks*, vol. 12, no. 4, pp. 726–734, 2001.

[5] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," 2015.

[6] V. Vapnik, "Principles of risk minimization for learning theory," in *Proc. Adv. Neural Inf. Process. Syst.*, 1992.

[7] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017.

[8] A. D. Kiureghian and O. Ditlevsen, "Aleatory or epistemic? does it matter?," *Structural Safety*, vol. 31, no. 2, pp. 105–112, 2009.

[9] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. M. 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," *CoRR*, vol. abs/2107.03342, 2021.

[10] L. Smith and Y. Gal, "Understanding measures of uncertainty for adversarial example detection," *ArXiv*, vol. abs/1803.08533, 2018.

[11] A. Malinin and M. Gales, "Uncertainty estimation in autoregressive structured prediction," in *Proc. Int. Conf. Learning Representations*, 2021.

[12] A. Malinin and M. Gales, "Predictive uncertainty estimation via prior networks," in *Proc. Adv. Neural Inf. Process. Syst.*, p. 7047–7058, 2018.

[13] A.-K. Kopetzki, B. Charpentier, D. Zügner, S. Giri, and S. Günnemann, "Evaluating robustness of predictive uncertainty estimation: Are dirichlet-based models reliable ?," in *Proc. Int. Conf. Mach. Learn.*, 2021.

[14] J. Van Amersfoort, L. Smith, Y. W. Teh, and Y. Gal, "Uncertainty estimation using a single deep deterministic neural network," in *Proc. Int. Conf. Mach. Learn.*, pp. 9690–9700, 2020.

[15] E. D. C. Carvalho, R. Clark, A. Nicastro, and P. H. J. Kelly, "Scalable uncertainty for computer vision with functional variational inference," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2020.

[16] J. Postels, F. Ferroni, H. Coskun, N. Navab, and F. Tombari, "Sampling-free epistemic uncertainty estimation using approximated variance propagation," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2019.

[17] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 6402–6413, 2017.

[18] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016.

[19] W. Wright, "Bayesian approach to neural-network modeling with input uncertainty," *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1261–1270, 1999.

[20] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with bernoulli approximate variational inference," *CoRR*, vol. abs/1506.02158, 2015.

[21] M. Vadera, B. Jalaian, and B. Marlin, "Generalized bayesian posterior expectation distillation for deep neural networks," in *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 719–728, 2020.

[22] M. Jain, S. Lahlou, H. Nekoei, V. Butoi, P. Bertin, J. Rector-Brooks, M. Korablyov, and Y. Bengio, "DEUP: direct epistemic uncertainty prediction," *CoRR*, vol. abs/2102.08501, 2021.

[23] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.

[24] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2015.

[25] S. Depeweg, J.-M. Hernandez-Lobato, F. Doshi-Velez, and S. Udluft, "Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning," in *Proc. Int. Conf. Mach. Learn.*, pp. 1184–1193, 2018.

[26] M. H. Shaker and E. Hüllermeier, "Aleatoric and epistemic uncertainty with random forests," in *Advances in Intelligent Data Analysis XVIII*, pp. 444–456, 2020.

[27] D. Nix and A. Weigend, "Estimating the mean and variance of the target probability distribution," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 1, pp. 55–60 vol.1, 1994.

[28] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2018.

[29] N. Houlsby, F. Huszar, Z. Ghahramani, and M. Lengyel, "Bayesian active learning for classification and preference learning," *CoRR*, vol. abs/1112.5745, 2011.

[30] C. Williams and C. Rasmussen, "Gaussian processes for regression," in *Proc. Adv. Neural Inf. Process. Syst.*, 1996.

[31] M. I. Jordan, Z. Ghahramani, and et al., "An introduction to variational methods for graphical models," in *Machine Learning*, pp. 183–233, MIT Press, 1999.

[32] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2008.

[33] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, pp. 97–109, 04 1970.

[34] T. G. Dietterich, "Ensemble methods in machine learning," in *Proceedings of the First International Workshop on Multiple Classifier Systems*, pp. 1–15, 2000.

[35] N. Durasov, T. Bagautdinov, P. Baque, and P. Fua, "Masksembles for Uncertainty Estimation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2021.

[36] J. Liu, J. Paisley, M.-A. Kioumourtzoglou, and B. Coull, "Accurate uncertainty estimation and decomposition in ensemble learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.

[37] A. Malinin and M. Gales, "Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.

[38] A. Malinin, B. Mlodozeniec, and M. Gales, "Ensemble distribution distillation," in *Proc. Int. Conf. Learning Representations*, 2020.

[39] F. Wenzel, J. Snoek, D. Tran, and R. Jenatton, "Hyperparameter ensembles for robustness and uncertainty quantification," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 6514–6527, 2020.

[40] A. Damianou and N. D. Lawrence, "Deep Gaussian processes," in *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pp. 207–215, 2013.

[41] A. Kendall, V. Badrinarayanan, , and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," in *Proc. Brit. Mach. Vis. Conf.*, 2017.

[42] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get M for free," in *Proc. Int. Conf. Learning Representations*, 2017.

[43] Y. Geifman, G. Uziel, and R. El-Yaniv, "Bias-reduced uncertainty estimation for deep neural classifiers," in *Proc. Int. Conf. Learning Representations*, 2019.

[44] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 4026–4034, 2016.

[45] K. Chitta, J. Feng, and M. Hebert, "Adaptive semantic segmentation with a strategic curriculum of proxy labels," *CoRR*, vol. abs/1811.03542, 2018.

[46] E. Meyerson and R. Miikkulainen, "Pseudo-task augmentation: From deep multitask learning to intratask sharing—and back," in *Proc. Int. Conf. Mach. Learn.*, pp. 3511–3520, 2018.

[47] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 2672–2680, 2014.

[48] D. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. Int. Conf. Learning Representations*, 2014.

[49] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 3483–3491, 2015.

[50] T. Han, Y. Lu, S. Zhu, and Y. Wu, "Alternating back-propagation for generator network," in *Proc. AAAI Conf. Artificial Intelligence*, pp. 1976–1984, 2017.

[51] K.-C. Wang, P. Vicol, J. Lucas, L. Gu, R. Grosse, and R. Zemel, "Adversarial distillation of Bayesian neural network posteriors," in *Proc. Int. Conf. Mach. Learn.*, pp. 5190–5199, 2018.

[52] C. Henning, J. Oswald, J. Sacramento, S. Surace, J.-P. Pfister, and B. Grewe, "Approximating the predictive distribution via adversarially-trained hypernetworks," in *Proc. Adv. Neural Inf. Process. Syst. Workshop*, 2018.

[53] N. Ratzlaff and L. Fuxin, "HyperGAN: A generative model for diverse, performant neural networks," in *Proc. Int. Conf. Mach. Learn.*, pp. 5361–5369, 2019.

[54] V. Böhm, F. Lanusse, and U. Seljak, "Uncertainty quantification with generative models," in *Proc. Adv. Neural Inf. Process. Syst. Workshop*, 2019.

[55] R. Neal, "Mcmc using hamiltonian dynamics," *Handbook of Markov Chain Monte Carlo*, 06 2012.

[56] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive Science*, vol. 9, no. 1, pp. 147 – 169, 1985.

[57] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, p. 1771–1800, 2002.

[58] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," *Proceedings of AISTATS 2009*, vol. 5, pp. 448–455, 01 2009.

[59] J. Xie, Y. Lu, R. Gao, S.-C. Zhu, and Y. N. Wu, "Cooperative training of descriptor and generator networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018.

[60] Y. LeCun, S. Chopra, R. Hadsell, F. J. Huang, and et al., "A tutorial on energy-based learning," in *PREDICTING STRUCTURED DATA*, MIT Press, 2006.

[61] Y. Du and I. Mordatch, "Implicit generation and modeling with energy based models," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 3608–3618, 2019.

[62] J. Xie, Y. Lu, S.-C. Zhu, and Y. Wu, "A theory of generative convnet," in *Proc. Int. Conf. Mach. Learn.*, vol. 48, pp. 2635–2644, 2016.

[63] Y. Yacoby, W. Pan, and F. Doshi-Velez, "Mitigating model non-identifiability in bnn with latent variables," in *proceedings at the International Conference on Machine Learning: Workshop on Uncertainty & Robustness in Deep Learning (ICML)*, 2019.

[64] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Proc.*, 2004.

[65] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017.

[66] Z. Wu, L. Su, and Q. Huang, "Cascaded partial decoder for fast and accurate salient object detection," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2019.

[67] Z. Wu, L. Su, and Q. Huang, "Stacked cross refinement network for edge-aware salient object detection," in *Proc. IEEE Int. Conf. Comp. Vis.*, pp. 7264–7273, 2019.

[68] J. Wei, S. Wang, and Q. Huang, "F^3net: Fusion, feedback and focus for salient object detection," in *Proc. AAAI Conf. Artificial Intelligence*, pp. 12321–12328, 2020.

[69] B. Wang, Q. Chen, M. Zhou, Z. Zhang, X. Jin, and K. Gai, "Progressive feature polishing network for salient object detection.," in *Proc. AAAI Conf. Artificial Intelligence*, pp. 12128–12135, 2020.

[70] J. Wei, S. Wang, Z. Wu, C. Su, Q. Huang, and Q. Tian, "Label decoupling framework for salient object detection," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 13025–13034, 2020.

[71] J. Troscianko, J. Skelhorn, and M. Stevens, "Quantifying camouflage: how to predict detectability from appearance," *BMC evolutionary biology*, vol. 17, no. 1, pp. 1–13, 2017.

[72] T. W. Pike, "Quantifying camouflage and conspicuousness using visual salience," *Methods in Ecology and Evolution*, vol. 9, no. 8, pp. 1883–1895, 2018.

[73] A. Tankus and Y. Yeshurun, "Convexity-based visual camouflage breaking," *Comp. Vis. Image Understanding*, vol. 82, no. 3, pp. 208–237, 2001.

[74] F. Xue, C. Yong, S. Xu, H. Dong, Y. Luo, and W. Jia, "Camouflage performance analysis and evaluation framework based on features fusion," *Multimedia Tools and Applications*, vol. 75, no. 7, pp. 4065–4082, 2016.

[75] S. Li, D. Florencio, Y. Zhao, C. Cook, and W. Li, "Foreground detection in camouflaged scenes," in *Proc. IEEE Int. Conf. Image Process.*, pp. 4247–4251, IEEE, 2017.

[76] D.-P. Fan, G.-P. Ji, G. Sun, M.-M. Cheng, J. Shen, and L. Shao, "Camouflaged object detection," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 2777–2787, 2020.

[77] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[78] S. F. Bhat, I. Alhashim, and P. Wonka, "Adabins: Depth estimation using adaptive bins," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 4009–4018, 2021.

[79] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," *ArXiv preprint*, 2021.

[80] S. Nah, T. Hyun Kim, and K. Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 3883–3891, 2017.

[81] Z. Shen, W. Wang, X. Lu, J. Shen, H. Ling, T. Xu, and L. Shao, "Human-aware motion deblurring," in *Proc. IEEE Int. Conf. Comp. Vis.*, pp. 5572–5581, 2019.

[82] J. Rim, H. Lee, J. Won, and S. Cho, "Real-world blur dataset for learning and benchmarking deblurring algorithms," in *Proc. Eur. Conf. Comp. Vis.*, pp. 184–201, 2020.

[83] L. Wang, H. Lu, Y. Wang, M. Feng, D. Wang, B. Yin, and X. Ruan, "Learning to detect salient objects with image-level supervision," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 136–145, July 2017.

[84] C. Yang, L. Zhang, H. Lu, X. Ruan, and M. Yang, "Saliency detection via graph-based manifold ranking," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 3166–3173, 2013.

[85] G. Li and Y. Yu, "Visual saliency based on multiscale deep features," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 5455–5463, June 2015.

[86] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille, "The secrets of salient object segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 280–287, 2014.

[87] T.-N. Le, T. V. Nguyen, Z. Nie, M.-T. Tran, and A. Sugimoto, "Anabranch network for camouflaged object segmentation," *Comp. Vis. Image Understanding*, vol. 184, pp. 45–56, 2019.

[88] P. Skurowski, H. Abdulameer, J. Baszczyk, T. Depta, A. Kornacki, and P. Kozie, "Animal camouflage analysis: Chameleon database.," in *Unpublished Manuscript*, 2018.

[89] Y. Lv, J. Zhang, Y. Dai, A. Li, B. Liu, N. Barnes, and D.-P. Fan, "Simultaneously localize, segment and rank the camouflaged objects," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 11591–11601, 2021.

[90] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Proc. Eur. Conf. Comp. Vis.*, pp. 746–760, 2012.

[91] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 3354–3361, 2012.

[92] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 586–595, 2018.

[93] J. Mukhoti and Y. Gal, "Evaluating bayesian deep learning methods for semantic segmentation," *CoRR*, vol. abs/1811.12709, 2018.

[94] L. Rumberger, L. Mais, and D. Kainmueller, "Probabilistic deep learning for instance segmentation," in *ECCV Workshop*, pp. 445–457, 2020.

[95] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 3431–3440, June 2015.

[96] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 770–778, 2016.

[97] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," *arXiv e-prints*, vol. abs/1812.11941, 2018.

[98] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[99] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao, "Multi-stage progressive image restoration," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 14821–14831, 2021.

[100] J. Zhang, D.-P. Fan, Y. Dai, S. Anwar, F. S. Saleh, T. Zhang, and N. Barnes, "Uc-net: Uncertainty inspired rgb-d saliency detection via conditional variational autoencoders," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2020.

[101] A. Li, J. Zhang, Y. Lyu, B. Liu, T. Zhang, and Y. Dai, "Uncertainty-aware joint salient object and camouflaged object detection," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2021.

[102] J. Zhang, J. Xie, N. Barnes, and P. Li, "Learning generative vision transformer with energy-based latent space for saliency prediction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021.

[103] F. Yang, Q. Zhai, X. Li, R. Huang, H. Cheng, and D.-P. Fan, "Uncertainty-guided transformer reasoning for camouflaged object detection," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2021.

[104] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, pp. 1254–1259, Nov 1998.

[105] C. Koch and S. Ullman, "Shifts in selective visual attention: towards the underlying neural circuitry," *Human neurobiology*, vol. 44, pp. 219–27, 1985.

[106] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 824–840, 2009.

[107] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia, "On the uncertainty of self-supervised monocular depth estimation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2020.

[108] C. Qu, W. Liu, and C. J. Taylor, "Bayesian deep basis fitting for depth completion with uncertainty," *CoRR*, vol. abs/2103.15254, 2021.

[109] J. R. Puigvert, R. Martinez-Cantin, and J. Civera, "Bayesian deep networks for supervised single-view depth learning," *CoRR*, vol. abs/2104.14202, 2021.

[110] T. Ke, T. Do, K. Vuong, K. Sartipi, and S. I. Roumeliotis, "Deep multi-view depth estimation with predicted uncertainty," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[111] H. Choi, H. Lee, S. Kim, S. Kim, S. Kim, and D. Min, "Adaptive confidence thresholding for semi-supervised monocular depth estimation," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2021.

[112] P. Tran, A. T. Tran, Q. Phung, and M. Hoai, "Explore image deblurring via encoded blur kernel space," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 11956–11965, 2021.

[113] R. Yasarla, F. Perazzi, and V. M. Patel, "Deblurring face images using uncertainty guided multi-stream semantic networks," *IEEE Trans. Image Proc.*, vol. 29, pp. 6251–6263, 2020.

[114] N. G. Nair and V. M. Patel, "Confidence guided network for atmospheric turbulence mitigation," in *Proc. IEEE Int. Conf. Image Process.*, pp. 1359–1363, 2021.

[115] R. Yasarla and V. M. Patel, "Learning to restore images degraded by atmospheric turbulence using uncertainty," in *Proc. IEEE Int. Conf. Image Process.*, pp. 1694–1698, 2021.

[116] K. He, R. Girshick, and P. Dollar, "Rethinking imagenet pre-training," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2019.

[117] J. He, D. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick, "Lagging inference networks and posterior collapse in variational autoencoders," in *Proc. Int. Conf. Learning Representations*, 2019.

[118] R. Krishnan and O. Tickoo, "Improving model calibration with accuracy versus uncertainty optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020.

[119] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2017.

[120] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proc. Int. Conf. Learning Representations*, 2017.

[121] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 3, pp. 425–464, 2001.

[122] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 2818–2826, June 2016.

[123] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *Proc. Int. Conf. Learning Representations*, 2017.

[124] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," in *Proc. Int. Conf. Learning Representations*, 2018.

[125] S. Liu, R. Garrepalli, T. Dietterich, A. Fern, and D. Hendrycks, "Open category detection with pac guarantees," in *Proc. Int. Conf. Mach. Learn.*, 2018.

[126] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. The MIT Press, 2008.

[127] L. Neumann, A. Zisserman, and A. Vedaldi, "Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection," in *Proc. Adv. Neural Inf. Process. Syst. Workshop*, 2018.

[128] A. Kumar, S. Sarawagi, and U. Jain, "Trainable calibration measures for neural networks from kernel mean embeddings," in *Proc. Int. Conf. Mach. Learn.*, vol. 80, pp. 2805–2814, 10–15 Jul 2018.

[129] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian, "Disturblabel: Regularizing cnn on the loss layer," *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 4753–4762, 2016.