Introspective Robot Perception using Smoothed Predictions from Bayesian Neural Networks

Jianxiang Feng*[1], Maximilian Durner*[1],
Zoltán-Csaba Márton[1], Ferenc Bálint-Benczédi[2], and Rudolph Triebel[1,3]

* Equal contribution; [1] Institute of Robotics and Mechatronics, German Aerospace Center (DLR), `first.last@dlr.de`; [2] Institute for Artificial Intelligence, University of Bremen, `balintbe@cs.uni-bremen.de`; [3] Department of Computer Science, Technical University of Munich, `rudolph.triebel@in.tum.de`

Abstract. This work focuses on improving uncertainty estimation in the field of object classification from RGB images and demonstrates its benefits in two robotic applications. We employ a Bayesian Neural Network (BNN), and evaluate two practical inference techniques to obtain better uncertainty estimates, namely Concrete Dropout (CDP) and Kronecker-factored Laplace Approximation (LAP). We show a performance increase using more reliable uncertainty estimates as unary potentials within a Conditional Random Field (CRF), which is able to incorporate contextual information as well. Furthermore, the obtained uncertainties are exploited to achieve domain adaptation in a semi-supervised manner, which requires less manual efforts of annotating data. We evaluate our approach on two public benchmark datasets that are relevant for robot perception tasks.

Keywords: BNN, CRF, introspective classification

1 Introduction

Visual scene understanding plays an important role in the field of robotic perception. In recent years, deep learning showed promising results within this context (e.g. object classification, detection or segmentation). Yet, although the applied deep neural networks outperform most traditional methods, they lack a significant property for robots in real world: a reliable uncertainty estimation. Advanced robotics highly rely on perceptual systems in order to be able to understand and adapt to its environment. Providing also the confidence of predictions based on the perceived information enhances the ability of robotic systems even further. It equips robots with the ability to know when it does and when it does not know. Besides the safety issue – for the robot itself and its surroundings – introspection about the predictions also has a positive impact on decision making, failure recovery and human-robot interaction. Furthermore, reliable uncertainty estimation is beneficial for active learning [1], reinforcement learning [2–4], detection of the unknown classes and adversarial attacks [5–7]. Recent research on improving the uncertainty estimation of deep neural networks includes BNN[2, 8–15], bootstrapping [3], ensemble methods [16] and so on. Among them, BNN is more theoretically sound and able to provide promising performances. By taking into account the practicality in real-world applications, we evaluate BNN with two inference techniques which are CDP [11] and LAP [14] in term of comprehensive metrics. However, we are more curious about the question, to which extent the improved uncertainty estimates can boost the performances on uncertainty-relevant tasks, rather than satisfied on demonstrating the improvements of uncertainty estimation solely on different metrics. Therefore, in this work we focus on the improvements by exploiting uncertainty estimates from BNN which are demonstrated by applying them to support another probabilistic model, CRF, to improve the performance with additional contextual information and achieve better performance with less manual efforts in domain adaptation tasks.

In the line of combining deep learning and Probabilistic Graphical Model (PGM) [17], previous works [18–21] mainly focus on joint training of these two kinds of model in order to share the advantages of both, which are abilities of expressive representation learning and structured learning, respectively. None of them emphasize the role of uncertainty estimation when combining them as sub-modules, which can improve the robustness of the system in practical applications such as real world robotics. In this work, we propose to use improved uncertainty estimates to improve classification by combining CRF (see Fig. 1).

On the other hand, robotics deployed in a new situation are often confronted with environmental changes and novel objects. Nevertheless, in most of the time a base classifier trained on an easily obtainable dataset (e.g. public large-scale or synthetic) is available beforehand. The classifier needs to be adapted to the test environment, while the manual efforts of collecting and annotating the adaptation data should be kept as low as possible. This requirement can be cast into the field of domain adaptation in a self/semi-supervised manner. Self-supervised learning refers to learning with self-provided supervisions such as geometrical cues within images [22] instead of strong but laborious human-supervisions and these self-supervisions can be extended to self-generated pseudo labels by the model itself, which can be used for domain adaptation naturally [23–25]. This task can also be framed into a semi-supervised manner, when a small amount of manual annotations are allowed to be taken into the procedure [26, 27]. Among these prior works, none of them highlights the importance of uncertainty estimates which can help distinguishing true positives (served for automatic-annotation) and false positives in both self-supervised and semi-supervised manner. As far as we know, we are the first to make use of uncertainty estimates from BNN in this kind of tasks.

The remaining of the paper is organized as follows: we review prior works in the related areas in section 2. While section 3 recaps the theoretical concept of BNN and CRF, section 4 explains our proposed approaches in this work. Then we show eperimental results demonstrating their effectiveness in section 5 and conclude in section 6.

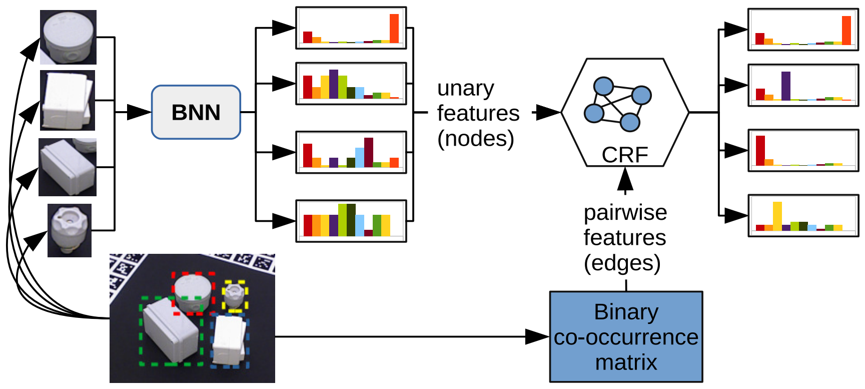

Fig. 1. Combination of BNN and CRF: the predictive distributions of objects in the scene from BNN serve as unary features in CRF.

2 Related Work

A **BNN** [28, 29] provides a principal way to obtain model uncertainty by considering the distribution on model parameters. However, it has difficulty scaling to complex network architectures and large training sets nowadays. Besides sampling based methods [8, 15], Variational Inference (VI) [30] suits practical applications due to its ability of fast inference. In the era of deep learning, there is a bunch of research works in this direction [2, 10, 12–14]. CDP [11] is an extension of Monte Carlo Dropout (MCD) [9] which can learn dropout rates from the data without efforts of manual tuning. More than that, CDP can be inserted into existing network architectures very easily and LAP does not require re-training and thus suits most of the already-trained networks.

Combination of deep learning and PGM: Liu et al. [19] trained a Convolutional Neural Network (CNN) and CRF jointly for depth estimation, while Tompson et al. [18] integrated Markov random fields with CNN for pose estimation. Wang et al. [20] combined deep learning with Bayesian networks for recommendation systems and topic models. Johnson et al. [21] proposed Structured variational autoencoder (SVAE) to learn a structured and thus more interpretable latent representation. Our work differ from them in the way of training. Since we want to evaluate the effects of uncertainty estimates, it's better to analyze them separately. Similar to us, Liu et al. [31] combined features learned from deep neural nets and CRF for segmentation tasks. But they trained another classifier with these features for the unary potentials without evaluating the uncertainty estimates.

Semi/Self-supervised domain adaptation: Some works [22, 25] aim to learn a more generalized feature distribution via designing specific *pretext* tasks without explicit human supervisions such as class labels. Others [27, 26, 24] focus on employing true positive predictions as self-supervisions during adaptation.

Zou et al. [24] mentioned the class imbalance problem and proposed to mitigate it by normalizing the class-wise confidence. To note that this problem is obvious in this kind of task, which was verified and mitigated by class-balanced augmentations in our experiments.

3 Bayesian Neural Networks

In general, a neural network can be modelled as a function $f^{\boldsymbol{\omega}}(\mathbf{x}) = \mathbf{y}$ that maps from an input space $\mathcal{X}$ to an output space $\mathcal{Y}$, where $\boldsymbol{\omega} = \{W_{1:L}, \mathbf{b}_{1:L}\}$ are the weights of the network consisting of matrices W_i and biases $\mathbf{b}_i$ for each of its L layers. In the training phase, the weights $\boldsymbol{\omega}$ are determined by optimizing a loss function $E(f^{\boldsymbol{\omega}}(\mathbf{x}_i), \mathbf{y}_i)$ for a given training data set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N\}$. In contrast, a Bayesian Neural Network (BNN) not only aims to find an optimal $\boldsymbol{\omega}$, but also defines a *posterior distribution* $p(\boldsymbol{\omega} \mid X, Y)$, where X and Y are the matrices consisting of all training data samples. Given this posterior, inference on a new test sample $(\mathbf{x}^*, \mathbf{y}^*)$ can be done using the *predictive distribution*

$$p(\mathbf{y}^* \mid \mathbf{x}^*, \mathcal{D}) = \int p(\mathbf{y}^* \mid \mathbf{x}^*, \boldsymbol{\omega}) p(\boldsymbol{\omega} \mid \mathcal{D}) d\boldsymbol{\omega}, \tag{1}$$

where for classification tasks the likelihood $p(\mathbf{y}^* \mid \mathbf{x}^*, \boldsymbol{\omega})$ is usually obtained from the *softmax* of the prediction $f^{\boldsymbol{\omega}}(\mathbf{x}^*)$. The benefit of using (1) for predictions instead of only using the likelihood is that the model also incorporates the *epistemic* uncertainty, i.e. the one that stems from incorrect model parameters, thereby providing better (less overconfident) uncertainty estimates.

Unfortunately, obtaining the parameter posterior $p(\boldsymbol{\omega} \mid X, Y)$ is not tractable in all but the simplest cases due to the high dimensionality of the parameter space $\mathcal{W}$. Therefore, approximations need to be used, and we investigate two common ones: concrete dropout and the Laplace approximation.

3.1 Concrete dropout

Dropout [32] was originally proposed to regularize the training process of Deterministic Neural Network (DNN) to improve their generalization performance, although yet without a formal interpretation. Then, Gal [33] showed that using dropout can be interpreted as sampling from a distribution $q_\theta(\boldsymbol{\omega})$ that approximates the posterior $p(\boldsymbol{\omega} \mid X, Y)$ in terms of the KL-divergence

$$KL(q_\theta(\boldsymbol{\omega}) \| p(\omega \mid X, Y)) = -\int q_\theta(\boldsymbol{\omega}) \log \frac{p(\boldsymbol{\omega} \mid X, Y)}{q_\theta(\boldsymbol{\omega})}. \tag{2}$$

where $\theta = \{\boldsymbol{\omega}, \mathbf{p}\}$, $\mathbf{p}$ is the vector of dropout rates of layers in which dropout is inserted. Minimizing this is equivalent to minimizing the negative lower bound

$$\begin{aligned} \mathcal{L}(\theta) &= -\sum_{i=1}^{N} \int q_\theta(\boldsymbol{\omega}) \log p(\mathbf{y}_i \mid f^{\boldsymbol{\omega}}(\mathbf{x}_i)) d\boldsymbol{\omega} + \mathrm{KL}(q_\theta(\boldsymbol{\omega}) || p(\boldsymbol{\omega})) && (3) \\ &\approx -\sum_{i \in \mathcal{S}} \frac{N}{K} \int q_\theta(\boldsymbol{\omega}) \log p(\mathbf{y}_i \mid f^{\boldsymbol{\omega}}(\mathbf{x}_i)) d\boldsymbol{\omega} + \mathrm{KL}(q_\theta(\boldsymbol{\omega}) || p(\boldsymbol{\omega})), && (4) \end{aligned}$$

where $\mathcal{S}$ is a mini-batch of size K. To estimate the expected log likelihood in the first term, Monte Carlo integration is used, i.e. samples are generated from $q_\theta(\boldsymbol{\omega})$, and the integral is approximated by summing likelihood terms over the samples. The problem here is that using this standard method, this first term can not be derived with respect to θ, which is necessary to minimize $\mathcal{L}(\theta)$. Therefore, the *re-parameterization trick* is used, i.e. a bivariate transformation $g(\theta, \epsilon)$ is used to separate the parameters θ from samples $\epsilon \sim p(\cdot)$ that are generated from a distribution with fixed parameters. Originally, this could be done only for a Gaussian dropout distribution, later Gal *et al.* [11] showed that for Bernoulli dropout, a *con*tinuous relaxation of this dis*crete* distribution can be found, i.e. a concrete distribution [34], which can then be derived wrt. θ for optimization. This is denoted *concrete dropout.* In our experiments, we use the implementation provided by Gal *et al.* [11].

3.2 Laplace approximation

The idea within the so-called Laplace approximation is to employ a second-order Taylor expansion at the maximum of the log posterior:

$$\log p(\omega \mid X, Y) \approx \log p(\omega^* \mid X, Y) - \frac{1}{2}(\boldsymbol{\omega} - \boldsymbol{\omega}^*)^T H(\boldsymbol{\omega} - \boldsymbol{\omega}^*), \tag{5}$$

where $\boldsymbol{\omega}^*$ is the parameter vector that maximizes the log posterior and H is the Hessian of the negative log posterior. Note that the first derivative vanishes at $\boldsymbol{\omega}^*$ and H is p.s.d. because $\boldsymbol{\omega}^*$ is assumed to be a local maximum. After taking the exponential and normalizing we obtain

$$p(\omega \mid X, Y) \approx \mathcal{N}(\boldsymbol{\omega}^\star, H^{-1}). \tag{6}$$

Unfortunately, the dimensionality of this multi-variate normal distribution is in most cases too high to be practical. Also, H needs to be computed on the entire data set, which is also infeasible. Instead, it is approximated by the expected Hessian $\mathbb{E}_{p(X,Y)}[H]$, computed on mini-batches. To reduce the dimensionality, a first step is to assume independence across the layers of the DNN, i.e. H is block-diagonal with L blocks H_i, one for each layer.

Under certain conditions, the Fisher information matrix F, which is the outer product of the first derivatives, is an approximation to the expected Hessian. Furthermore, in each layer i the block F_i can be approximated by a Kronecker product of two much smaller matrices G_i and A_i, where $G_i = \mathbf{g}_i\mathbf{g}_i^T$ is the outer product of gradients of pre-activation of i-th layer and $A_i = \mathbf{a}_{i-1}\mathbf{a}_{i-1}^T$ is the outer product of activation from the previous layer. This is known as the Kronecker-factored approximate curvature (K-FAC) [35]. If a Gaussian prior is used and F is scaled by the set of the training set N, then the resulting posterior can be written as matrix normal distribution [36]:

$$\mathbf{W}_i \sim \mathcal{MN}(\mathbf{W}_i^\star, (\sqrt{N}\mathbb{E}[\mathbf{A}_i] + \sqrt{\tau}\mathbf{I})^{-1}, (\sqrt{N}\mathbb{E}[\mathbf{G}_i] + \sqrt{\tau}\mathbf{I})^{-1}) \tag{7}$$

where τ is the standard deviation of the Gaussian prior. The two parameters N and τ can also be treated as hyper-parameters and tuned based on the performance on a validation set.

4 Improvements based on uncertainty estimates

In this section, we describe how the better uncertainty estimates can be utilized with contextual information within CRF for further improvements. Then, these techniques are employed in a practical use-case of semi-supervised domain adaptation.

4.1 Utilizing uncertainty estimates with CRF

While the BNN approach is very useful to provide reliable uncertainty estimates for single object instances, it does not incorporate any *context information* specific for a scene, such that, e.g. more likely object constellations can be accounted for. In order to exploit such contextual information within the classification, we combine the output of the BNN and the relationships between objects within a scene via a CRF(see Fig. 1).

In details, we define a scene as a set of n object instances $\mathbf{x} = \{\mathbf{x_1}, \dots, \mathbf{x_n}\}$ with corresponding class labels $\mathbf{y} = \{\mathbf{y_1}, \dots, \mathbf{y_n}\}$ represented as one-hot enconcodings, i.e. $\mathbf{y_i} \in \{0,1\}^C$ and $|\mathbf{y_i}| = 1$, where C is the number of potential object classes. The CRF models the joint probability $p(\mathbf{y} \mid \mathbf{x})$ as an undirected graph consisting of cliques of random variables. Here a *pairwise* CRF is used, consisting of nodes $\mathcal{V}$ and edges $\mathcal{E}$, where the node potentials are modeled as $\phi_u(\mathbf{x}_i, \mathbf{y}_i)$ for individual object instances and the edge potentials $\phi_p(\mathbf{x}_i, \mathbf{x}_j, \mathbf{y}_i, \mathbf{y}_j)$ for pairs of objects $(\mathbf{x}_i, \mathbf{x}_j)$ which are in the scene. Concretely, we define ϕ_u as the predictive probability of each instance (see Eq. (1)) and ϕ_p as the co-occurrence probability of two objects. Co-occurrence probabilities can be obtained from an independent source (as in our household use-case, discussed shortly in Section 6). In case the list of expected objects in the scene is known (as in our industrial use-case, evaluated in subsection 5.2), the pairwise feature is binary and provided automatically per scene. Thus, the CRF has the following form:

$$p(\mathbf{y}|\mathbf{x};\theta) = \frac{1}{Z(\mathbf{x},\theta)} \exp\left(\theta_u \sum_{i \in \mathcal{V}} p(\mathbf{y}_i|\mathbf{x}_i) + \theta_p \sum_{(i,j)\in\mathcal{E}} M(\mathbf{y}_i, \mathbf{y}_j)\right), \tag{8}$$

where $\theta = \{\theta_u, \theta_p\}$ are the node respectively the edge weights, Z is the partition function, and M is a $C \times C$ binary matrix modelling the co-occurrence of two object classes $\mathbf{y}_i$ and $\mathbf{y}_j$. The training process of the CRF involves minimizing the negative log likelihood, i.e. finding optimal model parameters θ^* such that $\theta^* = \arg\min_\theta\{-\log p(\mathbf{y} \mid \mathbf{x};\theta)\}$. To do this, we employ Stochastic Gradient Descent (SGD) with momentum, which requires the calculation of gradients and thus an inference step for the likelihood shown in Eq. (8). We use a fully connected CRF, i.e. an exact inference of the likelihood is intractable. Therefore, we apply Loopy Belief Propagation (LBP) for approximate inference. In our implementation, we use the C++ library UPGM++ [37] for this purpose.

4.2 Semi-supervised Domain Adaptation

The domain gap between training data and test data distribution can decrease the performance of most of classifiers. This problem becomes more obvious when the classifier is trained on easily obtainable dataset such as a public large-scale or synthetic dataset.

Another use case to present the improvements with better uncertainty estimates is to adapt the classifier to the test environment with as little manual efforts as possible. The proposed pipeline for adaptive learning is visualized in Fig. 2. In this use-case, the classifier should be introspective and express reliable confidences about its predictions.

At first, the initialization phase includes training model on an easily obtainable or accessible dataset, which can be a public or synthetic dataset.

Next there is an adaptation phase for the classifier with help of the aforementioned techniques before deployment in the test environment. In this phase, the classifier should be able to adapt itself to the test environment by fine-tuning on a collected and labeled adaptation dataset. The labels for the adaptation dataset are collected in a semi-supervised manner (including both automatic and manual manner). Regarding the automatic way, the predictions with high confidence are used for pseudo labels, which requires the classifier to provide reliable uncertainty estimation. In the manual way, the classifier would ask people to label a small and random portion of data interactively.

In the end, the adapted classifier is deployed in the test environment (arrow B in Fig. 2). If the relationships between objects in the test environment are complementary to the BNN classifier and can be encoded well with pairwise feature, the CRF can be applied to capture them for further improvement (arrow C). The unadapted classifier can also be tested as a baseline (arrow A).

5 Experiments

In this section, we firstly compared performance on uncertainty estimates of two approximate inference techniques for BNN, which are CDP and LAP on a household objects dataset in terms of comprehensive metrics. Then the one with better performance was applied in the following experiments, which are the evaluations of the combination with CRF and the use-case in domain adaptation.

Two types of datasets were employed in our experiments. The first type is the household objects including the RGB-D Dataset from Washington University (WRGB-D) [38] and the UniHB dataset recorded by ourselves trying to mimic the WRGB-D dataset but with only one instance in each category. They contain multi-view images of household objects from 51 categories, with a 15° step in elevation (from 30°to 60°) and 2° step in azimuth (from 0° to 360°). Besides

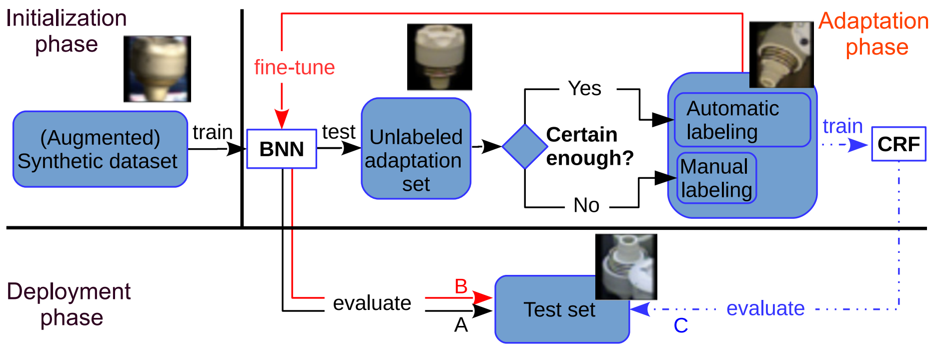

Fig. 2. Pipeline for semi-supervised domain adaptation which includes initialization, adaptation and deployment phase. Better uncertainty estimates can assist in automatic labeling in adaptation phase (illustrated on the T-LESS dataset and best viewed in color).

them, we have recorded some household objects of novel categories which serve as Out-of-distribution (OOD) dataset, as well as tabletop scenes captured by our robot. The second one is an industrial dataset, T-LESS dataset [39], which has little texture and similar appearance. This dataset contains multi-view images of objects from 30 kinds of industrial components. The training images depict objects in isolation with a black background, while the test images are from 20 table-top scenes with arbitrarily arranged objects placed on a table (as in a kitting or sorting task). Besides the original T-LESS dataset, we have also generated a synthetic dataset trying to mimic the original T-LESS training set. Since there are lots of occlusions in the test scenes, we employed data augmentations both to the original and synthetic T-LESS training set.

As mentioned in subsection 4.2, an easily obtainable dataset is used for training in initialization phase. This can be a large-scale public dataset like WRGB-D dataset or synthetic one like the synthetic T-LESS training set we generated. The (independent) adaptation and testing datasets simulate the data that the classifier encounters in the test environment.

5.1 Uncertainty estimates evaluation

In this part, we performed extensive experiments to evaluate uncertainty estimates on a household objects dataset. We trained models on the entire WRGB-D dataset and tested them on objects of 30° and 60° in the UniHB dataset.

Different metrics were used for the evaluation. To evaluate calibration performance we used Expected Calibration Error (ECE) and Maximal Calibration Error (MCE) [40]. For summary of both accuracy and calibration we used predictive Negative Log Likelihood (NLL) and brier score, which belong to proper scoring rules [41]. Additionally, we also employed metrics such as area under Receiver Operating Characteristic (ROC) curve and area under Precision Recall

(PR) curve to measure the separability between correct predictions and miss-classifications as well as OOD predictions. Apart from quantitative metrics, a qualitative (visual) metrics, the histogram (see Fig. 3 & Fig. 4) of uncertainty estimates was employed. For a better visualization, we set the normalizer in the histogram as the amount of the corresponding type of prediction. Regarding the uncertainty measure, we evaluated three different ones including confidence (maximum predictive likelihood), predictive entropy and mutual information [33]. The separability metrics list in the Table 1 were chosen based on the uncertainty measure with best performance.

The DNNs and BNNs were implemented in Tensorflow and the optimization was performed using RMSprop with an initial learning rate of $1e^{-5}$ and L2 regularization with coefficient of $3.5e^{-6}$ as well as the dropout regularization with coefficient of $1.0e^{-5}$. Early stopping was applied for model selection, based on the performance on a validation set. During inference, the number of samples drawn from the posterior distribution was set to 50 for both inference methods.

In order to preserve the powerful feature extraction ability of ResNet50 and incorporate the better uncertainty estimation from BNN, we slightly modify the ResNet50 by appending three fully connected layers with 1024 hidden units before the output layer. CDPs are inserted into the flatten layer and the three new fully connected layers. The weights of these layers were initialized from a Gaussian prior ($\mathcal{N}(0, 0.1)$) and the rest from model pre-trained on ImageNet [42]. This avoids destroying the pre-trained features and enables the model to possess large enough model capacity which is reduced by the insertions of dropout [32]. Furthermore, the computation complexity during inference can be reduced by only executing the forward pass of the additional layers instead of the whole network. In the following, we show both qualitative and quantitative results in Fig. 3 and Table 1, in which we denote original version of ResNet50 by **ORI** (without additional fully-connected layers), concrete dropout by **CDP**, Laplace approximation by **LAP**. The point estimate model for LAP was obtained by training networks with CDP. We set the hyper-parameter N as 1 and τ as 15 in LAP.

As can be seen, BNN can achieve better performance of uncertainty estimates in terms of all metrics when compared with original version of ResNet50. At the same time, CDP has a better performance than LAP in terms of proper scoring rules and calibration metrics. When OOD predictions were considered along with miss-classifications, ECE and MCE decreased significantly. This is because prediction of OOD data is always incorrect and not all OOD predictions provided high uncertainty. If their predictions are highly uncertain, the calibration metrics would have similar values with the situation without OOD data. Both inference methods yield similar results on separability metrics. Based on these experimental results, we used CDP in the following experiments.

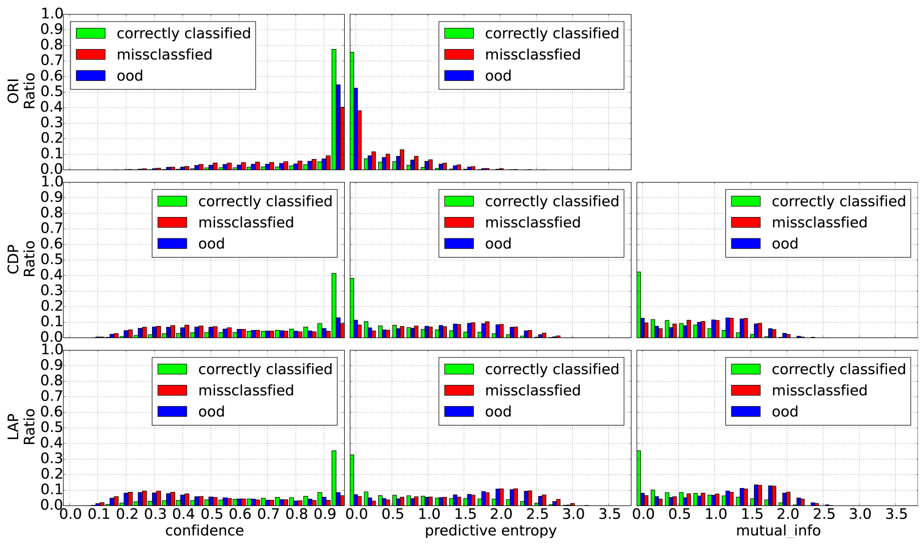

Fig. 3. Histograms of different uncertainty measures including confidence, predictive entropy and mutual information of three approaches (best viewed in color).

Table 1. Different quantitative results averaged over 3 different random seeds

	ACC ↑	predictive NLL ↓	Brier score↓	ECE (w/o. OOD/ w. OOD)↓	MCE (w/o. OOD/ w. OOD)↓	AUROC (vs. Miss-classified/ vs. OOD)↑	AUPR (vs. Miss-classified/ vs. OOD)↑
ORI	0.568 ±0.008	3.342 ±0.340	0.722 ±0.019	0.304±0.016/ 0.633±0.065	0.461±0.027/ 0.362±0.025	0.750±0.007/ 0.664±0.011	0.802±0.008/ 0.751±0.018
CDP	**0.577** ±0.008	**2.088** ±0.181	**0.594** ±0.013	**0.124**±0.023/ **0.288**±0.048	**0.206**±0.015/ **0.374**±0.018	0.775±0.008/ **0.783**±0.022	0.825±0.007/ **0.850**±0.022
LAP	0.576 ±0.009	2.322 ±0.350	0.602 ±0.011	0.129±0.058/ 0.341±0.157	0.235±0.073/ 0.406±0.070	**0.779**±0.004/ 0.782±0.017	**0.826**±0.007/ 0.849±0.016

5.2 Context-based improvement

In this experiment, we want to evaluate the idea introduced in subsection 4.1. The test set of T-LESS was employed in this part. We split the scenes 2, 3, 5, 8 off for training our CRF and the scenes 1, 4, 6, 7 for testing. These splits were chosen in this way, in order to keep as many categories as possible occurring in both training and testing (an evaluation on the whole T-LESS test set is shown in the next experiment). The maximum number of iterations during training is 30K, the initial learning rate is $1e^{-4}$, and the size of mini-batch is 16.

In order to see the influence of reliable uncertainty estimates we firstly trained DNNs and BNNs which provide the unary potential in the next step. Preliminary experiments, which are not displayed here, show a significant lower performance of the DNN trained without dropout compared to the BNN. On the other hand, the DNN trained with dropout but turned off MCD during inference (denoted as **NOMCD** in the following) resulted in worse uncertainty estimates but a better accuracy. Hence, since we want to investigate the effect of uncertainty estimates on the CRF, we compared the proposed BNN with the **NOMCD** approach.

Comparing the weights obtained by training the CRF with the uncertainty estimates of NOMCD (θ_u= 4.875; θ_p= 6.073) and BNN (θ_u= 8.122; θ_p= 6.59) a different rating of the provided information can be observed (θ_u vs. θ_p). While in the BNN case the CRF relies more on the classifier, in the NOMCD case the co-occurrence statistics are given a higher importance, reflecting the added usefulness of the correct uncertainty estimates (since the NOMCD and BNN accuracies without smoothing are similar, as seen in Table 2).

Table 2. Results of CRF trained and tested with different unary features

	type of unary features in testing	ACC with unary potentials	ACC with unary and pairwise potentials
CRF trained with unary features from NOMCD	NOMCD	58.48%	68.6%
	BNN	60.36%	76.19%
CRF trained with unary features from BNN	NOMCD	58.48%	68.62%
	BNN	60.36%	**76.36**%

Table 2 shows that there is a much larger performance gain when using the CRF with accurate uncertainty estimates, and this is irrespective of the CRF weights used. Besides the performance gain, the CRF is also improving (or at least maintaining) the uncertainty estimates. Fig. 4 shows the histogram of confidence of the predictions made by NOMCD and BNN before and after applying LBP inference within CRF. We can see that the uncertainty estimates' quality of NOMCD has been improved and that of BNN has been maintained, which can be helpful for further improvement in the down-stream tasks.

5.3 Domain adaptation improvement

In this part, a proof-of-concept experiment is performed to evaluate the idea illustrated in Section 4.2. To this end, we employed both datasets from two different scenarios, the household objects and industrial components datasets for evaluation.

Following the pipeline in Fig. 2, in the initialization phase, we used WRGB-D dataset and the augmented, synthetic T-LESS dataset generated by ourselves for initial training, because they can be obtained more easily. During adaptation phase, objects of 30° and 60° in UniHB dataset (~17.1K) and the original training set of T-LESS (~30K) were used as adaptation dataset. In order to adapt to the test environment, the classifier should be able to collect a dataset for fine-tuning with as little manual efforts as possible. Therefore, this collected dataset can be annotated in two different manners, automatically and manually.

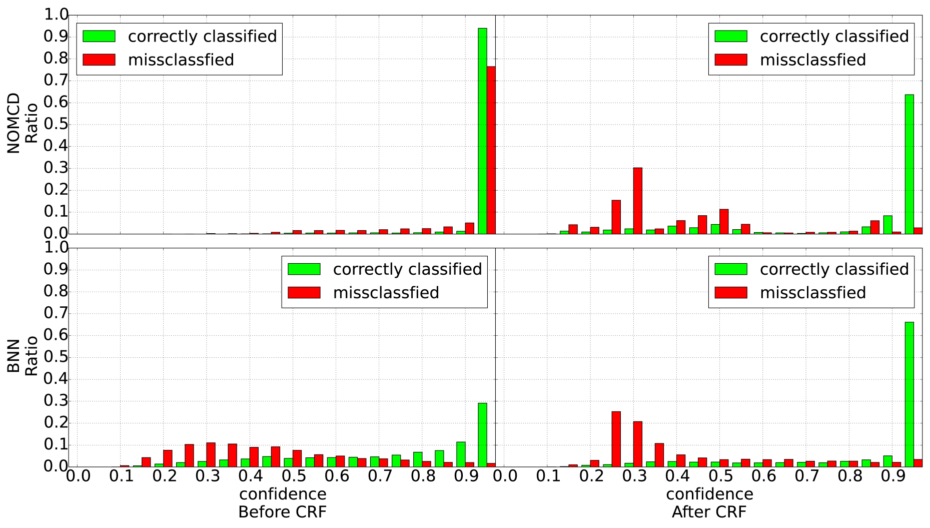

Fig. 4. Histograms of confidence of NOMCD (top row) and BNN (bottom row) before (left column) and after (right column) applying LBP in CRF (best viewed in color).

The automatically labeled data was selected based on threshold of the uncertainty estimates. In the end, during the deployment phase, the adapted model was evaluated on the test dataset. The 45° objects in UniHB dataset and original test set of T-LESS were treated as data the robot encounters in the test environment.

Household objects dataset: We tested different versions of the proposed uncertainty based automatic labeling procedure, and found that the best results were obtained by setting the confidence threshold s.t. the accuracy of the predictions (estimated on a small manually labeled set) is 95%. The accuracy of automatically labeled data in III, IV is around 96%, matching the 95% estimate.

Our main results are shown in Table 3. As it can be seen, the manual labeling effort can be reduced based on automatic labeling. More detailed testing will be performed on the industrial dataset, based on the insights gained here.

During the experiment, we found that the balance of number of each class on the adaptation dataset plays an important role. The main reason for this should be the different visual domain gap of different objects. The initial model is more familiar with some objects instead of other and thus give lower uncertainty for these familiar ones. Since we selected predictions based on the uncertainty estimates, this would lead to an imbalanced dataset and thus bias the adapted model. Therefore it's important to mitigate this issue. We found that the manually labeled data and augmentations are useful not only to increase the diversity of the dataset, but their main contribution to improving performance is through balancing the dataset (see III and IV). Other ways of balancing the automatically labeled data (e.g. by selecting the top most confident predictions per class) decreased performance as they resulted in either too few labels or included too many incorrect ones.

Table 3. Results of fine-tuned network on household objects dataset

Dataset used for fine-tuning	Accuracy (average over 3 random seeds)
I: 0% (no fine-tuning)	66.9%
II: 3% manually labeled data, selected randomly (**balanced**)	91.7%
III: 3% automatically labeled data (**imbalanced**)	79.0%
IV: 2% automatically labeled data and 1% manually labeled data randomly, augmentation for balance (**balanced**)	89.6%

Industrial components dataset: With the same procedure of selecting automatically labeled data, the size of dataset is ∼1K with only 93% accuracy using the original ResNet50, but ∼1.6K with 96% accuracy using BNN. The summary of the results is shown in Table 4. The performance of the classifier adapted using 3% manually labeled data (VI) is matched by the use of 1% manually labeled data if automatic labeling is employed (V). Moreover, adding the automatic labeling to the 3% manually labeled data can nearly reach the the performance of classifier adapted with all available data manually labeled (III vs VII). By incorporating contextual information with CRF, the performance can be increased further (VIII).

Table 4. BNN fine-tuning with different datasets (size of dataset before augmentations is showed in the bracket).

Dataset used for fine-tuning	Accuracy
I: augmented, synthetic dataset	34.91%
II: fine-tune I with augmented, automatically labeled real dataset (∼1.6K)	53.54%
III: entire real dataset, i.e. 100% maually labeled (∼30K), augmented	72.78%
IV: fine-tune I with 1% manually labeled real dataset (∼0.3K), augmented	67.4%
V: fine-tune I with II and IV (∼1.9K), augmented	68.1%
VI: fine-tune I with 3% manually labeled real dataset (∼0.9K), augmented	68.1%
VII: fine-tune I with II and VI (∼2.5K), augmented	72.48%
VIII: Incorporating contextual information with CRF based on VII	74.64%

6 Conclusions

We presented an approach to make robots learning new objects more introspectively, by improving its awareness of possible mistakes, and leveraging this in two ways: first, for better incorporating context information (if available) through smoothing over all object predictions using a CRF, and second, for exploiting this in semi-supervised domain adaptation, where the mostly correct predictions are automatically obtained as adaptation data while asking humans for help with the more uncertain ones.

The improved uncertainty estimation from BNN plays an important role especially in the latter use-case, because it not only provides a reliable uncertainty estimation, but also increases the separability between correct predictions and false predictions, which is more useful in this task. It was found, however that it is very important to ensure that the data is balanced. For manual labeling this can be easily achieved by requesting the human operator to label a more-or-less equal number of instances of each object, e.g. repeatedly selecting random subsets and having to click all occurrences of an object (as in an image CAPTCHA), then switching to the next target object once enough samples were collected. For the automatic labeling, random selection is not a good alternative, as the accuracy penalty would be too large if the overall performance of the initial classifier is too low (as in our cases). It could be, however, incorporated if multiple rounds of adaptation are performed, and the performance is gradually increasing to acceptable levels (around 95% in our tests).

In the former use-case the importance of a clear co-occurence statistic is highlighted by the fact that the CRF failed to improve results on the household dataset (the pairwise weight was negligible) due to the difficulty of obtaining good co-occurrence statistics in this scenario (which we mined from word co-occurences in WikiHow articles) and since many household objects have similar appearances and contexts at the same time. In an industrial scenario, e.g. for kitting applications, such a list of parts is available, and the learned CRF weights generalize well over objects and scenes.

Acknowledgments: This work was partially funded by the Big Data Interdisciplinary Project of DLR e.V. under the project number 2464047.

References

1. Gal, Y., Islam, R., Ghahramani, Z.: Deep bayesian active learning with image data. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1183–1192. JMLR. org (2017)
2. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural network. In: International Conference on Machine Learning. pp. 1613–1622 (2015)
3. Osband, I., Blundell, C., Pritzel, A., Van Roy, B.: Deep exploration via bootstrapped dqn. In: Advances in neural information processing systems. pp. 4026–4034 (2016)
4. Gal, Y., McAllister, R., Rasmussen, C.E.: Improving pilco with bayesian neural network dynamics models. In: Data-Efficient Machine Learning workshop, ICML (2016)
5. Grimmett, H., Triebel, R., Paul, R., Posner, I.: Introspective classification for robot perception. The International Journal of Robotics Research (IJRR) 35(7), 743–762 (2016)

6. Hendrycks, D., Gimpel, K.: A baseline for detecting misclassified and out-of-distribution examples in neural networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings (2017), https://openreview.net/forum?id=Hkg4TI9xl
7. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial machine learning at scale. arXiv preprint arXiv:1611.01236 (2016)
8. Balan, A.K., Rathod, V., Murphy, K.P., Welling, M.: Bayesian dark knowledge. In: Advances in Neural Information Processing Systems. pp. 3438–3446 (2015)
9. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: international conference on machine learning. pp. 1050–1059 (2016)
10. Louizos, C., Welling, M.: Structured and efficient variational deep learning with matrix gaussian posteriors. In: International Conference on Machine Learning. pp. 1708–1716 (2016)
11. Gal, Y., Hron, J., Kendall, A.: Concrete dropout. In: Advances in Neural Information Processing Systems. pp. 3581–3590 (2017)
12. Sun, S., Chen, C., Carin, L.: Learning structured weight uncertainty in bayesian neural networks. In: Artificial Intelligence and Statistics. pp. 1283–1292 (2017)
13. Louizos, C., Welling, M.: Multiplicative normalizing flows for variational bayesian neural networks. In: Proceedings of the 34th International Conference on Machine Learning. vol. 70, pp. 2218–2227. JMLR. org (2017)
14. Ritter, H., Botev, A., Barber, D.: A scalable laplace approximation for neural networks. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=Skdvd2xAZ
15. Wang, K., Vicol, P., Lucas, J., Gu, L., Grosse, R.B., Zemel, R.S.: Adversarial distillation of bayesian neural network posteriors. In: Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. pp. 5177–5186 (2018)
16. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. In: Advances in Neural Information Processing Systems. pp. 6402–6413 (2017)
17. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. MIR Press (2009)
18. Tompson, J.J., Jain, A., LeCun, Y., Bregler, C.: Joint training of a convolutional network and a graphical model for human pose estimation. In: Advances in neural information processing systems. pp. 1799–1807 (2014)
19. Liu, F., Shen, C., Lin, G.: Deep convolutional neural fields for depth estimation from a single image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5162–5170 (2015)
20. Wang, H., Yeung, D.Y.: Towards bayesian deep learning: A survey. arXiv preprint arXiv:1604.01662 (2016)
21. Johnson, M., Duvenaud, D.K., Wiltschko, A., Adams, R.P., Datta, S.R.: Composing graphical models with neural networks for structured representations and fast inference. In: Advances in neural information processing systems. pp. 2946–2954 (2016)
22. Kolesnikov, A., Zhai, X., Beyer, L.: Revisiting self-supervised visual representation learning. arXiv preprint arXiv:1901.09005 (2019)
23. Tang, K., Ramanathan, V., Fei-Fei, L., Koller, D.: Shifting weights: Adapting object detectors from image to video. In: Advances in Neural Information Processing Systems. pp. 638–646 (2012)

24. Zou, Y., Yu, Z., Vijaya Kumar, B., Wang, J.: Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 289–305 (2018)
25. Xu, J., Xiao, L., Lopez, A.M.: Self-supervised domain adaptation for computer vision tasks. arXiv preprint arXiv:1907.10915 (2019)
26. Wang, K., Zhang, D., Li, Y., Zhang, R., Lin, L.: Cost-effective active learning for deep image classification. IEEE Transactions on Circuits and Systems for Video Technology 27(12), 2591–2600 (2016)
27. Lin, L., Wang, K., Meng, D., Zuo, W., Zhang, L.: Active self-paced learning for cost-effective and progressive face identification. IEEE transactions on pattern analysis and machine intelligence 40(1), 7–19 (2017)
28. MacKay, D.J.: A practical bayesian framework for backpropagation networks. Neural computation 4(3), 448–472 (1992)
29. Neal, R.M.: Bayesian learning for neural networks, vol. 118. Springer Science & Business Media (2012)
30. Graves, A.: Practical variational inference for neural networks. In: Advances in neural information processing systems. pp. 2348–2356 (2011)
31. Liu, F., Lin, G., Shen, C.: Crf learning with cnn features for image segmentation. Pattern Recognition 48(10), 2983–2992 (2015)
32. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1), 1929–1958 (2014)
33. Gal, Y.: Uncertainty in deep learning. Ph.D. thesis, PhD thesis, University of Cambridge (2016)
34. Maddison, C.J., Mnih, A., Teh, Y.W.: The concrete distribution: A continuous relaxation of discrete random variables. arXiv preprint arXiv:1611.00712 (2016)
35. Martens, J., Grosse, R.: Optimizing neural networks with kronecker-factored approximate curvature. In: International conference on machine learning. pp. 2408–2417 (2015)
36. Gupta, A., Nagar, D.: Matrix Variate Distributions, vol. 104. CRC Press (1999)
37. Ruiz-Sarmiento, J.R., Galindo, C., González-Jiménez, J.: Upgmpp: a software library for contextual object recognition. In: 3rd. Workshop on Recognition and Action for Scene Understanding (REACTS) (2015)
38. Lai, K., Bo, L., Ren, X., Fox, D.: A large-scale hierarchical multi-view rgb-d object dataset. In: 2011 IEEE international conference on robotics and automation. pp. 1817–1824. IEEE (2011)
39. Hodaň, T., Haluza, P., Obdržálek, Š., Matas, J., Lourakis, M., Zabulis, X.: T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. IEEE Winter Conference on Applications of Computer Vision (WACV) (2017)
40. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1321–1330. JMLR. org (2017)
41. Gneiting, T., Balabdaoui, F., Raftery, A.E.: Probabilistic forecasts, calibration and sharpness. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 69(2), 243–268 (2007)
42. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision 115(3), 211–252 (2015)