Implicit Weight Uncertainty in Neural Networks

Nick Pawlowski, Andrew Brock, Matthew C.H. Lee, Martin Rajchl, and Ben Glocker

presented by Alexander Donets

Novosibirsk State University, 2020

Contents

Introduction

Real-world decision making processes that wish to leverage neural networks are frequently faced with:

- lack of data
- need for reliable uncertainty estimates

and these two problems are often entangled. Unsolved, they lead model to overconfident behaviour and in some situations (e.g. medical applications, self-driving cars, etc.) it can be dangerous.

Introduction

To address the issue of overconfident predictions, recent works have proposed approaches like *calibration methods*, frequentist interpretations of *ensembles*, and approximate *Bayesian inference*.

The current research in BDL is primarily divided into *variational inference methods* and *Monte Carlo methods*.

Introduction

Hypernetworks are known as able to model a wide range of distributions and can therefore provide rich variational approximations.

We propose to combine prior work on implicit variational inference with the concept of hypernetworks. This builds **Bayes by Hypernet** as we reinterpret hypernet-works as implicit distributions similar to generators in generative adversarial networks and use them to approximate the posterior distribution of the weights of a neural network.

Next figure shows example of uncertainty difference for various machine learning methods.

Introduction

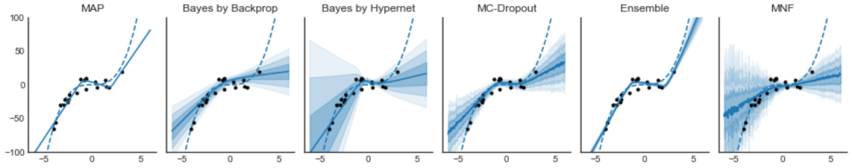


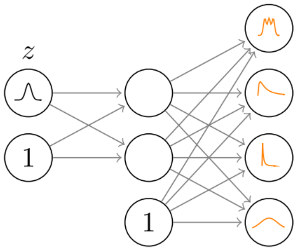
Рис.: Toy example: Real function (dashed line) with sampled data points (black dots). The proposed BbH exhibits the best trade-off between predictive uncertainty and regression fit. MC-Dropout, deep ensembles and the MAP produce a good fit but underestimate the predictive uncertainty. Multiplicative Normalizing Flows (MNF) and Bayes by Backprop (BbB) achieve a slightly worse fit and predictive uncertainty than BbH.

Introduction

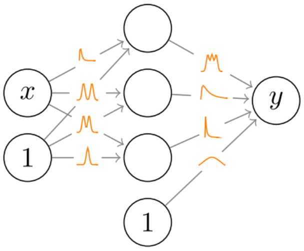
We introduce **Bayes by Hypernet** (**BbH**) which avoids hand-crafted strategies of building variational approximations and instead exploits the inherent capabilities of learned approximations to model rich, varied distributions.

We show that compared to other Bayesian methods, BbH achieves competitive performance on small networks and demonstrates comparable predictive accuracy without compromising predictive uncertainty, while being the least vulnerable against *adversarial attacks*.

Bayes by Hypernet



(a) *Hypernetwork G*



(b) Main Network

Рис.: Illustration of the components of Bayes by Hypernet: The hypernetwork G takes a sample z of distribution $p(z)$ and converts it into a sample of the weights w of the main network. The hypernetwork in Fig. 2 (a) generates samples of the weights of the second layer of the main network. The main network takes a data sample x and generates an output y using the weight samples generated by the hypernetworks.

Variational Bayesian Neural Networks

Variational Bayesian Neural Networks:
Given a dataset D with data points $(x_1, y_1), ..., (x_n, y_n)$ variational inference for Bayesian neural networks aims to approximate the posterior distribution $p(w|\mathbb{D})$ of the weights w of a neural network.

Given this distribution we can estimate the posterior prediction $\hat{y}$ of a new data point $\hat{x}$ as

$$P(\hat{y}|\hat{x}, \mathbb{D}) = \mathbb{E}_w[P(\hat{y}|\hat{x}, w)].$$

Because exact Bayesian inference is usually technically difficult in neural networks we find a variational approximation $Q(w|\mathbb{R})$ with parameters $\mathbb{R}$ that maximises the evidence lower bound ($ELBO$):

$$\mathbb{R}^* = \arg\min_{\mathbb{R}} \mathbb{KL}[Q(w|\mathbb{R})||P(w|\mathbb{D})]$$

Hypernetworks

Hypernetworks as Implicit Distributions:
Implicit distributions are distributions that may have probability densities hard to calculate but allow for easy sampling.

They enable simple calculation of approximate expectations and their corresponding gradients. Probably the most well-known group of implicit distributions are generative adversarial networks that can transform a sample from a simple noise distribution into high-fidelity images.

Hypernetworks

Using an implicit distribution to model the weights of a neural network requires a generator that is able to capture inherent complexity of neural networks weights.

Hypernetworks are shown to be able to generate weights of networks like ResNets or RNNs while still achieving competitive state-of-the-art performances.

Hypernetworks

Let $\mathbb{G}$ be a hypernetwork with parameters $\mathbb{R}$. Further, let z be an input vector to the hypernetwork $\mathbb{G}$ that contains information about the weight w to generate. Then weights w of the main network are generated as $w = \mathbb{G}(z|\mathbb{R})$.

When z is a sample from a simple auxiliary random variable the hypernetwork resembles a generator within the *GAN* framework. Rather than generating high-fidelity image samples, *our generator predicts samples of the weight distribution of the main network*.

Experiments

We aim to assess the predictive accuracy of a method, and also its ability to estimate the predictive uncertainty. We test:

- accuracy
- the entropy of the softmax outputs (as a measure of predictive uncertainty and the method's robustness against adversarial examples, aimed to "fool"the network)

The high uncertainty predictions on unseen data can be important in real-life decision processes as they can be used to trigger a request for human support.

Comparison or setups

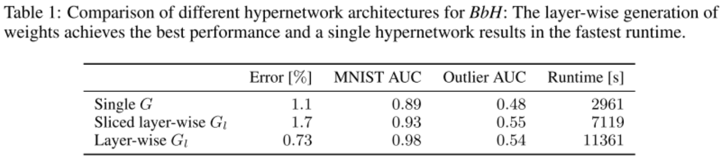


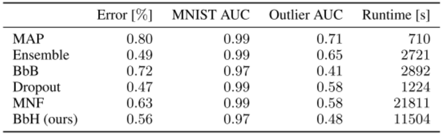
Table 1: Comparison of different hypernetwork architectures for *BbH*: The layer-wise generation of weights achieves the best performance and a single hypernetwork results in the fastest runtime.

	Error [%]	MNIST AUC	Outlier AUC	Runtime [s]
Single G	1.1	0.89	0.48	2961
Sliced layer-wise G_l	1.7	0.93	0.55	7119
Layer-wise G_l	0.73	0.98	0.54	11361

We employ 3-layer fully-connected networks with [64, 256, 512] units as hypernetworks for all experiments, as we did not find a general improvement by adding more layers or units. We train the deep ensembles without predictive uncertainty as we found it to sometimes result in numerically unstable training.

Comparison of predictions

Table 3: Compared methods on the MNIST classification task: The Error [%] is the methods classification error on the MNIST test set. The MNIST AUC and Outlier AUC refers to the AUC under the CDF of the predictive entropy on the corresponding data set. Compared methods: maximum a posteriori training (MAP), deep ensembles [21] (Ensembles), Bayes by Backprop [3] (BbB), MC-Dropout [6] (Dropout), Multiplicative Normalizing Flows [24] (MNF) and Bayes by Hypernet (*BbH*).



	Error [%]	MNIST AUC	Outlier AUC	Runtime [s]
MAP	0.80	0.99	0.71	710
Ensemble	0.49	0.99	0.65	2721
BbB	0.72	0.97	0.41	2892
Dropout	0.47	0.99	0.58	1224
MNF	0.63	0.99	0.58	21811
BbH (ours)	0.56	0.97	0.48	11504

All methods achieve comparable accuracy, with BbH only being outperformed by deep ensembles and MC-Dropout. However, BbH exhibits a higher predictive uncertainty, only outperformed by BbB on this metric. The runtime of BbH and MNF are significantly increased over other approaches, because of a relative high overhead to generate the weights compared to the actual network architecture.

Comparison of predictions

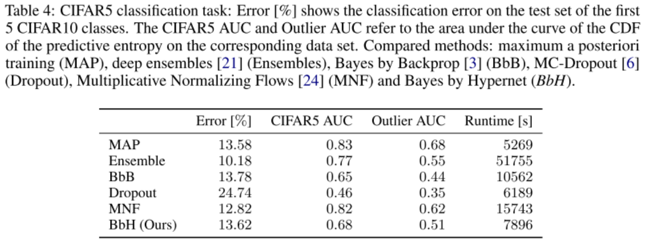


Table 4: CIFAR5 classification task: Error [%] shows the classification error on the test set of the first 5 CIFAR10 classes. The CIFAR5 AUC and Outlier AUC refer to the area under the curve of the CDF of the predictive entropy on the corresponding data set. Compared methods: maximum a posteriori training (MAP), deep ensembles [21] (Ensembles), Bayes by Backprop [3] (BbB), MC-Dropout [6] (Dropout), Multiplicative Normalizing Flows [24] (MNF) and Bayes by Hypernet (*BbH*).

	Error [%]	CIFAR5 AUC	Outlier AUC	Runtime [s]
MAP	13.58	0.83	0.68	5269
Ensemble	10.18	0.77	0.55	51755
BbB	13.78	0.65	0.44	10562
Dropout	24.74	0.46	0.35	6189
MNF	12.82	0.82	0.62	15743
BbH (Ours)	13.62	0.68	0.51	7896

ЙОХОХО

Conclusion

Thanks for your attention!