Uncertainty quantification with compound density networks

Agustinus Kristiadi
Department of Computer Science
University of Tübingen
72074 Tübingen
`agustinus.kristiadi@uni-tuebingen.de`

Sina Däubener, Asja Fischer
Department of Mathematics
Ruhr University Bochum
44801 Bochum
`sina.daeubener@rub.de`
`asja.fischer@rub.de`

Abstract

Despite the huge success of deep neural networks (NNs), finding good mechanisms for quantifying their prediction uncertainty is still an open problem. Bayesian neural networks are one of the most popular approaches to uncertainty quantification. On the other hand, it was recently shown that ensembles of NNs, which belong to the class of mixture models, can also be used to quantify prediction uncertainty. In this paper, we enhance the flexibility of mixture models by replacing the fixed mixing weights by an adaptive, input-dependent distribution represented by NNs, and by considering uncountable many mixture components. The resulting class of models can be seen as the continuous counterpart to mixture density networks and is therefore referred to as *compound density networks* (CDNs). We employ likelihood maximization to train CDNs, and empirically show that they yield better uncertainty estimates on out-of-distribution data and are more robust to adversarial examples than previous approaches.

1 Compound density networks

We generalize mixture density networks (MDNs) [1] from finite mixture distributions to mixtures of an uncountable set of components. The resulting model is given by

$$p(\mathbf{y}|\mathbf{x};\boldsymbol{\psi}) = \int p(\mathbf{y}; f(\mathbf{x};\boldsymbol{\theta}))p(\boldsymbol{\theta}; g(\mathbf{x};\boldsymbol{\psi}))\,\mathrm{d}\boldsymbol{\theta} = \mathbb{E}_{p(\boldsymbol{\theta};g(\mathbf{x};\boldsymbol{\psi}))}[p(\mathbf{y}; f(\mathbf{x};\boldsymbol{\theta}))] \ , \tag{1}$$

and, in correspondence to MDNs, is referred to as *compound density network* (CDN). Here, $f(\mathbf{x};\boldsymbol{\theta})$ corresponds to a neural network (NN) with parameters $\boldsymbol{\theta}$ that outputs the parameters of the compound distribution (e.g. the mean and variance of a Gaussian). The parameters $\boldsymbol{\theta}$ are random parameters themselves with a distribution that is parametrized by another function $g(\mathbf{x};\boldsymbol{\psi})$ modelled by NNs. As MDNs, CDNs can be trained by maximizing the log-likelihood of $\boldsymbol{\psi}$ given the dataset $\mathcal{D} = (\mathbf{x}_n, \mathbf{y}_n)_{n=1}^N$. For regularization, one can subtract a Kullback-Leibler divergence (D_{KL}) term encouraging the mixing distribution to stay close to some distribution $p(\boldsymbol{\theta})$, which leads to the objective

$$\mathcal{L}_{\mathrm{ML}}(\boldsymbol{\psi}) = \sum_{n=1}^{N} \log \mathbb{E}_{p(\boldsymbol{\theta};g(\mathbf{x}_n;\boldsymbol{\psi}))}[p(\mathbf{y}_n; f(\mathbf{x}_n;\boldsymbol{\theta})] - \lambda \sum_{n=1}^{N} D_{\mathrm{KL}}[p(\boldsymbol{\theta}; g(\mathbf{x}_n;\boldsymbol{\psi}))\|p(\boldsymbol{\theta})] \ , \tag{2}$$

where λ is a hyperparameter controlling the strength of the regularization. Interestingly, the approximation of this objective gets equivalent to the approximation of the variational information bottleneck (VIB) [2] when it is based on a single sample of $\boldsymbol{\theta}$. When setting $\lambda = 1$ in addition it also gets equivalent to the single sample based approximation of the ELBO employed for performing variational inference for an amortized BNN. However, using more samples for the approximations demonstrates, that the objective in eq. (6) leads to better results than VIB or ELBO, as we show in Appendix D.1. Note, that only a variational inference based training leads to a Bayesian model.

1.1 Probabilistic hypernetworks

A hypernetwork is an NN that generates the parameters of another NN [3][1]. We follow this approach for modeling a CDN, that is, we model the mixing distribution $p(\boldsymbol{\theta}; g(\mathbf{x}; \boldsymbol{\psi}))$ over network parameters with NNs. Since now the hypernetworks map $\mathbf{x}$ to a distribution over parameters instead of a specific value $\boldsymbol{\theta}$, we refer to them in analogy to Bayesian hypernetworks [4][2] as *probabilistic hypernetworks*. In the following, we will describe this idea in more detail.

Let $f = f_L \circ \cdots \circ f_1$ be a multi-layer perceptron (MLP) with L-layers, parametrized by a set of layers' weight matrices[3] $\boldsymbol{\theta} = \{\mathbf{W}_l\}_{l=1}^L$, that computes the parameters of the CDN's component distribution $p(\mathbf{y}; f(\mathbf{x}; \boldsymbol{\theta}))$ in eq. (1). Let $\mathbf{h}_l = f_l(\mathbf{W}_l^T \mathbf{h}_{l-1})$, for $l = 1, ..., L$, and define $\mathbf{h}_0 = \mathbf{x}$. We now assume the weight matrices $\{\mathbf{W}_l\}_{l=1}^L$ to consist of random variables, which are independent to each other given the state of the previous hidden layer. We define a series of probabilistic hypernetworks $g = \{g_l\}_{l=1}^L$ (parametrized by $\boldsymbol{\psi} = \{\boldsymbol{\psi}_l\}_{l=1}^L$), where g_l maps $\mathbf{h}_{l-1}$ to the parameters of the distribution of $\mathbf{W}_l$, and let the joint distribution over $\boldsymbol{\theta}$ be given by

$$p(\boldsymbol{\theta}; g(\mathbf{x}; \boldsymbol{\psi})) = \prod_{l=1}^{L} p(\mathbf{W}_l; g_l(\mathbf{h}_{l-1}; \boldsymbol{\psi}_l)) \ . \tag{3}$$

An illustration of a stochastic two-layer network is given in Figure 2 in the appendix.

For the practical implementation we follow [5] by modelling $p(\mathbf{W}_l; g_l(\mathbf{h}_{l-1}; \boldsymbol{\psi}_l))$ with a matrix variate normal distribution, and employ a probabilistic hypernetwork to compute its parameters (i.e. the mean matrix and the covariance matrices of rows and columns, respectively). Moreover, we use the reparametrization trick [6] and vector scaling [3] for an improved scalability. A detailed description of our implementation can be found in Appendix A.

2 Experiments

Here we present results for two standard tasks on MNIST: classification under out-of-distribution data (Section 2.1) and detection of and defense against adversarial examples [7] (Section 2.2). Results for two toy regression problems and the Fashion-MNIST datasets can be found in Appendix D.3 and D.4. The following recent models are considered as the baselines: Variational Matrix Gaussian (VMG) [5], Multiplicative Normalizing Flow (MNF) [8], Dirichlet Prior Networks (DPN) [9], noisy Kronecker-factored Approximate Curvature (noisy K-FAC) [10], Monte Carlo Dropout (MCD) [11] and Deep Ensemble (DE) [12].[4]

We estimate the predictive distribution $p(\mathbf{y}|\mathbf{x})$ of the CDNs, based on 100 samples of $\boldsymbol{\theta} \sim p(\boldsymbol{\theta}; g(\mathbf{x}; \boldsymbol{\psi}))$. We also draw 100 samples from the approximate posterior to approximate the predictive distribution of BNN baselines. If not stated otherwise, we use a single sample to perform Monte Carlo integration during training. We pick the regularization hyperparameter λ in eq. (6) for CDN out of the set $\{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$ which maximizes the validation accuracy. We use Adam [13] with default hyperparameters for optimization in all experiments and the implementations provided by [8][5] and [10][6] for MNF and noisy K-FAC, respectively. As for the network architecture, we use for the main network a MLP of size 784-100-10 with one hidden layer along with probabilistic MLP hypernetworks with 50 hidden units for CDNs. We use mini-batches of size 200. All models

[1]Specifically, it was proposed to apply a hypernetwork to compute the weight matrix of a recurrent NN at each time-step, given the current input and the previous hidden state.

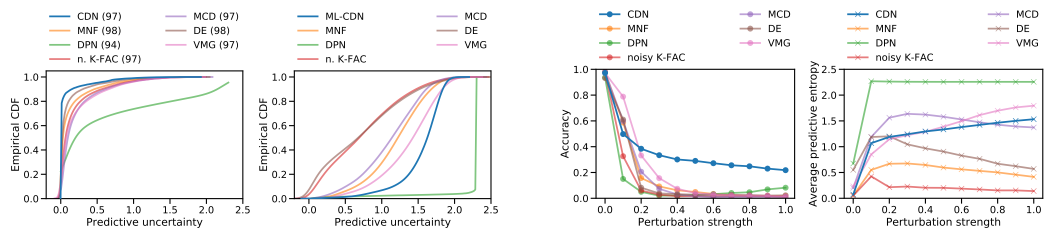
[2]Bayesian hypernetworks are hypernetworks that map Gaussian random noise to an approximate posterior distribution $q(\xi) = q(h(\epsilon))$ over the parameters ξ of a BNN.

[3]We assume that the bias parameters are absorbed into the weight matrix.

[4]Additional information on these and other related algorithms plus additional experiments are presented in D in the Appendix.

[5]`https://github.com/AMLab-Amsterdam/MNF_VBNN`

[6]`https://github.com/gd-zhang/noisy-K-FAC`



(a) MNIST (b) notMNIST (c) Accuracy (d) Entropy

Figure 1: The first two figures show the CDFs of the empirical entropy of the predictive distribution of the models, trained on MNIST. The caption of each figure indicates the test set used, the y-axis denotes the fraction of predictions having entropy less than the corresponding value on the x-axis. (c) and (d) show the prediction accuracy and average entropy of models trained on MNIST when attacked by FGSM-based adversarial examples. Values in the parenthesis denote the test accuracy.

are optimized over 20000 iterations ($\approx$67 epochs). We chose ReLU as the non-linearity of the CDNs' hypernetworks. For the details on the selection of the model-specific hyperparameters of the baselines, we refer the reader to Appendix C.1.

For a quantitative comparison, we calculated the mean maximal confidence for in-distribution (MMC-in) and OOD data (MMC-out) as well as the area under receiver operating characteristic (AUROC). The results can be found in Table 1. Our model clearly has the highest MMC value for in-distribution

Table 1: Mean maximal confidence (MMC) for in distribution (MNIST) and OOD data (notMNIST) and area under receiver operating characteristic (AUROC).

Algorithm	**MMC-in**	**MMC-out**	**AUROC**
CDN	**0.978**	**0.430**	**0.993**
VMG	0.938	0.507	0.964
MNF	0.959	0.504	0.977
MCD	0.950	0.665	0.928
DE	0.970	0.740	0.862
noisy-KFAC	0.949	0.744	0.848

data and the highest AUROC, while having the lowest MMC value for OOD data.

2.1 Out-of-distribution classification

Following [12], we train all models on the MNIST training set[7] and investigate their performance on the MNIST test set and the notMNIST dataset[8], which contains images (of the same size and format as MNIST) of letters from the alphabet instead of handwritten digits. On such an out-of-distribution (OOD) test set, the predictive distribution of an ideal model should have maximum entropy, i.e. it should have a value of $\ln 10 \approx 2.303$ which would be achieved if all ten classes are equally probable. We present the results in Figure 1 (a) and (b), where we plotted the cumulative distribution function (CDF) of the empirical entropy of the predictive distribution, following [8]. A CDF curve close to the top-left corner of the figure implies that the model yields mostly low entropy predictions, indicating that the model is very confident. While one wishes to observe high confidence on data points similar to those seen during training, the model should express uncertainty when exposed to OOD data. That is, we prefer a model to have a CDF curve closer to the bottom-right corner on notMNIST, as this implies it makes mostly uncertain (high entropy) predictions, and a curve closer to the upper-left corner for MNIST. The results show, that the CDN is more confident than all other models on within-distribution data, while showing higher uncertainty than all other models, except

[7]We use Fashion-MNIST as OOD data for training the DPN.

[8]`http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html`.

from the DPN, on OOD data. Note however, that training DPNs requires additional OOD data (which makes the comparison unfair) and that the DPN's prediction accuracy and confidence on the MNIST test set are low compared to all other models.

2.2 Adversarial examples

To investigate the robustness and detection performance of CDNs w.r.t. adversarial examples [7], we apply the Fast Gradient Sign Method (FGSM) [14] to a 10% fraction (i.e. 1000 samples) of the MNIST test set.[9] We do so, by making use of the implementation provided by Cleverhans [15]. Note, that we do not use adversarial training when training the Deep Ensemble in this experiment to allow for a fair comparison. Figure 1 (c) and (d) present the accuracy and the average empirical entropy of the predictive distribution w.r.t. adversarial examples for MNIST with varying levels of perturbation strength (between 0 and 1). We observe that the CDN is significantly more robust in terms of accuracy to adversarial examples than all other models, while showing a competitive and nicely increasing entropy.

3 Conclusion

We introduce compound density networks (CDNs), a new class of models that corresponds to compound distributions (i.e. a mixture with uncountable components) in which both the component distribution and the mixing distribution are parametrized by input dependent NNs. An experimental analyses demonstrated that CDNs yield better uncertainty estimates on out-of-distribution data and are more robust to adversarial examples than several baseline models. These promising results indicate the benefits of applying an infinite ensemble model in conjunction with input dependencies for NN based uncertainty quantification.

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

References

[1] Christopher M Bishop. Mixture density networks. 1994.

[2] Alex Alemi, Ian Fischer, Josh Dillon, and Kevin Murphy. Deep variational information bottleneck. In *ICLR*, 2017.

[3] David Ha, Andrew Dai, and Quoc V. Le. HyperNetworks. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2017)*, 2017.

[4] David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian Hypernetworks. *arXiv:1710.04759 [cs, stat]*, October 2017. arXiv: 1710.04759.

[5] Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In *International Conference on Machine Learning*, pages 1708–1716, 2016.

[6] Diederik P Kingma, Tim Salimans, and Max Welling. Variational Dropout and the Local Reparameterization Trick. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2575–2583. Curran Associates, Inc., 2015.

[7] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. 2014.

[9]We generate the adversarial examples based on a single forward-backward pass.

[8] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational Bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2218–2227, 2017.

[9] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. *arXiv preprint arXiv:1802.10501*, 2018.

[10] Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5852–5861, 2018.

[11] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[12] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations*, 2015.

[14] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[15] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

[16] Arjun K Gupta and Daya K Nagar. *Matrix variate distributions*. Chapman and Hall/CRC, 1999.

[17] Shengyang Sun, Changyou Chen, and Lawrence Carin. Learning structured weight uncertainty in bayesian neural networks. In *Artificial Intelligence and Statistics*, pages 1283–1292, 2017.

[18] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

[19] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, April 2014.

[20] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[21] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. *Proceedings of the 37th Allerton Conference on Communication, Control and Computation*, 49, 07 2001.

[22] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1321–1330, 06–11 Aug 2017.

[23] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proceedings of International Conference on Learning Representations*, 2017.

[24] Alex Graves. Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems 24*, pages 2348–2356. 2011.

[25] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

A Implementation details

As mentioned in the main part in Section 1.1 we use matrix variate normal (MVN) distributions [16] for modelling the mixing distribution inspired by previous work [5, 17, 10, 18]. A MVN is parametrized by three matrices: a mean matrix $\mathbf{M}$ and two covariance matrices $\mathbf{A}$ and $\mathbf{B}$. It is connected to the multivariate Gaussian by the following equivalence

$$\mathbf{X} \sim \mathcal{MN}(\mathbf{X}; \mathbf{M}, \mathbf{A}, \mathbf{B}) \iff \mathrm{vec}(\mathbf{X}) \sim \mathcal{N}(\mathrm{vec}(\mathbf{X}); \mathrm{vec}(\mathbf{M}), \mathbf{B} \otimes \mathbf{A}), \tag{4}$$

where $\mathrm{vec}(\mathbf{X})$ denotes the vectorization of matrix $\mathbf{X}$. Due to the Kronecker factorization of the covariance, a MVN requires fewer parameters compared to a multivariate Gaussian, which motivates us to use it as the distribution over weight matrices in this work. Furthermore, we assume that the covariance factor matrices are diagonal matrices, following [5]. That is, we choose the mixture distribution of the CDN to be

$$p(\boldsymbol{\theta}; g(\mathbf{x}; \boldsymbol{\psi})) = \prod_{l=1}^{L} \mathcal{MN}(\mathbf{W}_l; g_l(\mathbf{h}_{l-1}; \boldsymbol{\psi}_l)) = \prod_{l=1}^{L} \mathcal{MN}(\mathbf{W}_l; \mathbf{M}_l, \mathrm{diag}(\mathbf{a}_l), \mathrm{diag}(\mathbf{b}_l)) \;, \tag{5}$$

where g_l maps the state $\mathbf{h}_{l-1}$ of the previous hidden layer onto the l-th MVN's parameters $\{\mathbf{W}_l, \mathbf{a}_l, \mathbf{b}_l\}$[10] defining the distribution over $\mathbf{W}_l$. Suppose $\mathbf{W}_l \in \mathbb{R}^{r\times c}$, then the corresponding MVN distribution has $rc + r + c$ parameters, which is more efficient compared to $rc + rc$ parameters when modeling $\mathbf{W}_l$ as fully-factorized Gaussian random variable. To further reduce the amount of parameters we use a vector-scaling parametrization similar to the one used by [3] and [4] for the mean matrices $\{\mathbf{M}_l\}_{l=1}^{L}$. That is, we make a trade-off between the expressiveness of the hypernetwork g_l and the number of parameters by replacing $\mathbf{M}_l$ with a fixed parameter matrix $\mathbf{V}_l$ of the same size (with c-dimensional row vectors $\mathbf{v}_{li}$, $i = 1, \ldots, r$) and a scaling vector $\mathbf{d}_l \in \mathbb{R}^r$. Then g_l maps $\mathbf{h}_{l-1}$ to $\{\mathbf{d}_l, \mathbf{a}_l, \mathbf{b}_l\}$ (instead of $\{\mathbf{M}_l, \mathbf{a}_l, \mathbf{b}_l\}$) and $\mathbf{M}_l$ is defined by

$$\mathbf{M}_l \approx \begin{bmatrix} d_{l1}\mathbf{v}_{l1} \\ d_{l2}\mathbf{v}_{l2} \\ \cdots \\ d_{lr}\mathbf{v}_{lr} \end{bmatrix} \;.$$

That is, each element of $\mathbf{d}_l$ is being used to scale the corresponding row of $\mathbf{V}_l$. Thus, the number of parameter of g_l is reduced to $2r + c$, which is more manageable and implementable for large weight matrices.

To be able to learn the parameters of the mixing distribution by back-propagation, the reparametrization trick [19] is employed. That is, we first sample $\mathcal{E} \in \mathbb{R}^{r\times c}$ from $\mathcal{MN}(\mathbf{0}, \mathbf{I}_r, \mathbf{I}_c)$ with $\mathbf{0} \in \mathbb{R}^{r\times c}$ zero matrix, $\mathbf{I}_r \in \mathbb{R}^{r\times r}$ and $\mathbf{I}_c \in \mathbb{R}^{c\times c}$ being the identity matrices, by separately sampling each entry, $\epsilon_{ij} \sim \mathcal{N}(0, 1)\ \forall i = 1, \ldots, r\ \forall j = 1, \ldots, c$, and than estimate $\mathbf{W}_l$ based on the output of the hypernetwork as

$$\mathbf{W}_l = \mathbf{M}_l + \mathrm{diag}(\mathbf{a}_l)^{\frac{1}{2}} \mathcal{E}\, \mathrm{diag}(\mathbf{b}_l)^{\frac{1}{2}} \;.$$

For the regularization during likelihood maximization (i.e. for the second term in eq. (6)), we define the prior over $\boldsymbol{\theta}$ to be $p(\boldsymbol{\theta}) := \prod_{l=1}^{L} \mathcal{MN}(\mathbf{W}_l; \mathbf{0}, \mathbf{I}_r, \mathbf{I}_c)$.

The D_{KL}-Term can then be calculated deterministically during training, since the D_{KL}-divergence between $\mathcal{MN}(\mathbf{W}_l; M_l, \mathbf{a}, \mathbf{b})$ and $\mathcal{MN}(\mathbf{W}_l; \mathbf{0}, \mathbf{I}_r, \mathbf{I}_c)$ following [5] is given by

$$D_{\mathrm{KL}}[\,p(\mathbf{W}_l) \| \mathcal{MN}(\mathbf{0}, \mathbf{I}_r, \mathbf{I}_c)] = \frac{1}{2}\Bigg(\sum_{i=1}^{r} a_{li} \sum_{j=1}^{c} b_{lj} + \|\mathbf{M}_l\|_F^2 - rc - c\sum_{i=1}^{r} \log a_{li} - r \sum_{j=1}^{c} \log b_{lj} \Bigg).$$

[10]Assume $\mathbf{W}_l \in \mathbb{R}^{r\times c}$ then $\mathbf{M}_l \in \mathbb{R}^{r\times c}$, $\mathbf{A} \in \mathbb{R}^{r\times r}$ and $\mathbf{B} \in \mathbb{R}^{c\times c}$, where $\mathbf{a}_l$ and $\mathbf{b}_l$ are the diagonal values.

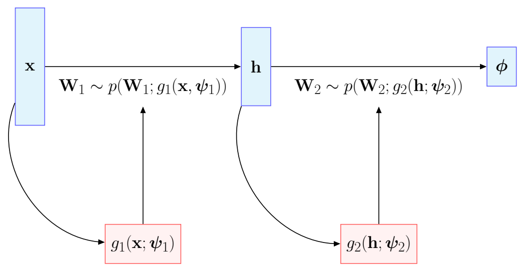

$\mathbf{x}$
$\mathbf{W}_1 \sim p(\mathbf{W}_1; g_1(\mathbf{x}, \boldsymbol{\psi}_1))$
$\mathbf{h}$
$\mathbf{W}_2 \sim p(\mathbf{W}_2; g_2(\mathbf{h}; \boldsymbol{\psi}_2))$
$\boldsymbol{\phi}$
$g_1(\mathbf{x}; \boldsymbol{\psi}_1)$
$g_2(\mathbf{h}; \boldsymbol{\psi}_2)$

Figure 2: An example of a probabilistic hypernetwork applied to a MLP with one hidden layer.

B Pseudocode for training CDNs

Algorithm 1 The training procedure of CDNs with $\mathcal{L}_{\text{ML}}$.

```
Require:
   Mini-batch size $M$, number of samples $S$ of $\boldsymbol{\theta}$, regularization strength $\lambda$, and learning rate $\alpha$.
1: while the stopping criterion is not satisfied do
2:     $\{\mathbf{x}_m, \mathbf{y}_m\}_{m=1}^M \sim \mathcal{D}$
3:     for $m = 1, \ldots, M$; $s = 1, \ldots, S$ do
4:         $\boldsymbol{\theta}_{ms} \sim p(\boldsymbol{\theta}; g(\mathbf{x}_m; \boldsymbol{\psi}))$
5:         $\boldsymbol{\phi}_s(\mathbf{x}_m) = f(\mathbf{x}_m; \boldsymbol{\theta}_{ms})$
6:     end for
7:     $\mathcal{L}(\boldsymbol{\psi}) = \sum_{m=1}^M \log \frac{1}{S} \sum_{s=1}^S p(\mathbf{y}_m; \boldsymbol{\phi}_s(\mathbf{x}_m)) - \lambda \sum_{m=1}^M D_{\text{KL}}[p(\boldsymbol{\theta}; g(\mathbf{x}_m; \boldsymbol{\psi})) \| p(\boldsymbol{\theta}))]$
8:     $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} + \alpha \nabla \mathcal{L}(\boldsymbol{\psi})$
9: end while
```

C Related work

We briefly describe the models we compare our algorithm with, for a more detailed description the reader is referred to the original papers:

Variational Matrix Gaussian (VMG) proposed by [5] are Bayesian neural networks which are trained with variational inference (VI). They are trained with a MVN distribution as the approximate posterior of each weight matrix.

Multiplicative Normalizing Flow (MNF) [8] models the approximate posterior as a compound distribution, where the mixing density is given by a normalizing flow.

Kronecker-factored Approximate Curvature (K-FAC) [10] use an MVN approximate posterior and apply approximate natural gradient [20] based maximization on the VI objective.

Variational Information Bottleneck (VIB) [2] assumes $\boldsymbol{\theta}$ to be the hidden units of a certain layer instead of the parameters of an NN and trains the model with an objective derived from the information bottleneck method [21].

Dirichlet Prior Networks (DPNs) [9] can be described by $p(\mathbf{y}|\mathbf{x}; \psi) = \int p(\mathbf{y}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathbf{x}; \boldsymbol{\psi})\, \mathrm{d}\boldsymbol{\theta}$ and aim at modeling data uncertainty with the component distribution and what they call "distributional uncertainty" (i.e. uncertainty due to mismatch between the distributions of test and training data) with the mixture distribution. Specifically, they propose DPNs for uncertainty quantification in classification tasks, where $p(\boldsymbol{\theta}|\mathbf{x}; \psi)$ is assumed to follow a Dirichlet distribution. In contrast to CDNs, DPNs use only a single NN to parametrize the model and an objective that augments likelihood maximization/ D_{KL} minimization by a term explicitly making use of out-of-distribution samples.

Deep Ensembles (DEs) [12] are ensembles of NNs trained with a proper scoring rule and adversarial training to quantify the prediction uncertainty of deep NNs. A DE provides a non-Bayesian approach for quantifying prediction uncertainty, and is in this sense related to the approaches of [22] and [23].

Monte Carlo Dropout (MCD) [11] is based on a theoretical framework that relates dropout training in NNs to approximate Bayesian inference and, as a result, allows to approximate the predictive distribution by an average over the different networks resulting from independently sampled dropout-masks, a technique which they applied to estimate the prediction uncertainty in NNs.

C.1 Hyperparameter

For the baseline models in our experiments, we use the hyperparameter values that are suggested in the respective publications and summarized in the following:

- **MNF:** The D_{KL}-term is weighted by $1/B$, where B is the number of mini-batches used during optimization (see [24] for a justification of this). Moreover it is annealed with a hyperparameter initialized to 0 and increasing to 1 during training. We found that this yields better results than using no re-weighting of the D_{KL}-term and is necessary to achieve the results reported by [8].
- **noisy K-FAC:** D_{KL}-term weight is set to $\lambda = 1$ to reflect the Bayesian objective and the prior variance is set to $\eta = 0.1$ as suggested by the authors.
- **DPN:** the OOD datasets used are Fashion-MNIST and MNIST, respectively for the MNIST and Fashion-MNIST training set.
- **Deep Ensemble:** The number of mixture components is 5, the adversarial perturbation strength is set to 1% of the input range, and the weight decay is 0.0001.
- **MC-dropout:** The dropout probability is set to 0.5 and the weight decay parameter to 0.0001.
- **VIB:** λ is set to $1e-5$ which maximizes the validation accuracy compared to other choices of $\lambda \in \{1e-4, 1e-5, 1e-6, 1e-7, 1e-8\}$.

D Further experiments

D.1 Comparison of different objectives objectives

We experimentally compare the effects of training the proposed model by optimizing the likelihood based objective $\mathcal{L}_{\mathrm{ML}}$ versus the VIB approach, which corresponds to optimizing the following objective

$$\mathcal{L}_{\mathrm{VIB}}(\boldsymbol{\psi}) = \sum_{n=1}^{N} \mathbb{E}_{p(\boldsymbol{\theta}; g(\mathbf{x}_n;\boldsymbol{\psi}))}[\log p(\mathbf{y}_n; f(\mathbf{x}_n;\boldsymbol{\theta})] - \lambda \sum_{n=1}^{N} D_{\mathrm{KL}}[p(\boldsymbol{\theta}; g(\mathbf{x}_n;\boldsymbol{\psi}))\|p(\boldsymbol{\theta})] \ . \tag{6}$$

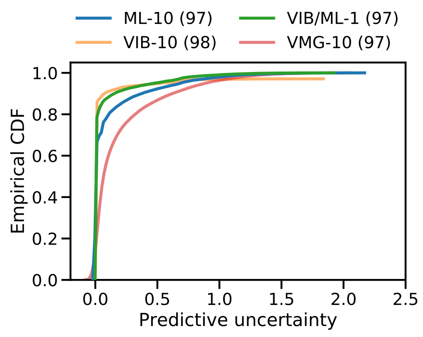
As stated in the main text Sec.1, both objectives become equivalent when approximated by a single sample of $\boldsymbol{\theta}$. Therefore, to analyze the differences we approximate them based on 10 samples of $\boldsymbol{\theta}$. As a special case we investigated $\mathcal{L}_{\mathrm{VIB}}$ with $\lambda = 1$ which resembles the variational inference (VI) objective of BNNs with an amortized prior but did not obtained reasonable accuracy in this setting. For comparison, we analysed the performance of an VMG trained by approximating the ELBO with 10 samples because it is a closely related BNN.

We present the results for OOD classification on MNIST in Figure 3 (a) and (b). $\mathcal{L}_{\mathrm{ML}}$ and $\mathcal{L}_{\mathrm{VIB}}$ both work well, the former showing slightly better results. Using 10 instead of one sample leads to higher uncertainty on OOD data for the ML objective but also increases uncertainty on the in-distribution test set a tiny bit.

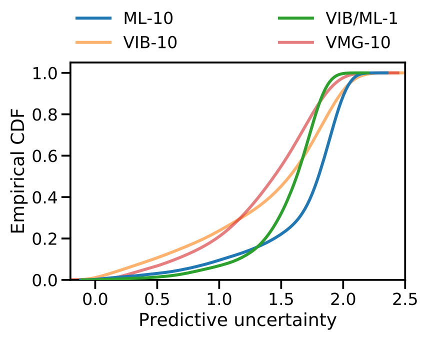
Results for the robustness to adversarial examples created with FGSM are shown in Figure 3 (c) and (d). We observe that optimizing based on the likelihood objective approximated with 10 samples of $\boldsymbol{\theta}$ gives the best results. While the increased sample size improved the performance of models trained with the VIB as well as with the ML objective, the model trained with $\mathcal{L}_{\mathrm{ML}}$ and 10 samples clearly outperforms the others, reaching a surprisingly high accuracy about 0.8 even under strong perturbations.

Generally, the VMG showed worse results, which suggests that the input dependency of the distribution over $\boldsymbol{\theta}$ plays a crucial role for the increased performance observed compared to baseline models.

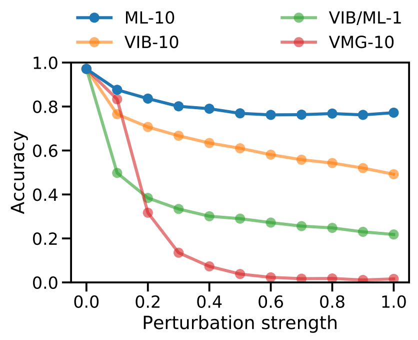
In summary, the VI objective for a BNN with an amortized prior could not be employed for training our model successfully, and based on our observations ML seems to be superior to the VIB objective, when approximated by several samples.



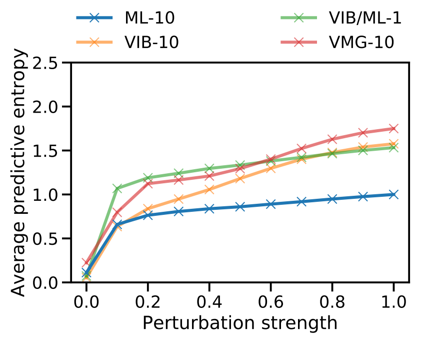
(a) MNIST uncertainty



(b) notMNIST uncertainty



(c) MNIST adversarial accuracy



(d) MNIST adversarial entropy

Figure 3: Comparison of the effects of training the proposed model with the VIB or the CDN objective and VMG on MNIST. "Objective-S" denotes that the objective was approximated based on S samples of $\boldsymbol{\theta}$ during training. Note that the entropy is calculated over all examples and not only the missclassified ones.

D.2 Stronger adversarial attacks

One might argue, that taking one sample for adversarial attacks in probabilistic nets is not sufficient. Therefore, we varied the strength of our adversarial attack by using 10 and 20 samples for the attack, instead of 1 as in the baseline in the main paper. The results are shown in Figure 4. Using stronger attacks does decrease the accuracy and entropy of the CDN to some degree. However, the accuracy is still superior to that of the other methods when they are attacked with only a one sample attack, cf. Figure 1(c). Furthermore, there is almost no difference between using 10 and 20 samples when attacking the CDN.

D.3 Toy regression

Following [25], we generate the first toy regression dataset as follows: We sample 20 input points $x \sim \mathcal{U}[-4, 4]$ and their target values $y = x^3 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 3^2)$, i.e. the data noise is homoscedastic. We aim at analyzing how well the target function is modeled over the larger interval $[-6, 6]$. Having only few data points, it is a desirable property of a model to express high (epistemic) uncertainty in regions with no or only few samples, e.g. between -6 and -4 or 4 and 6. The second toy regression dataset is constructed by sampling 100 data points as above, this time with different scale of noise in different intervals: $\epsilon \sim \mathcal{N}(0, 3^2)$, if $x \geq 0$. and $\epsilon \sim \mathcal{N}(0, 15^2)$, otherwise. This dataset is designated for testing whether a model can capture heteroscedastic aleatoric uncertainty.

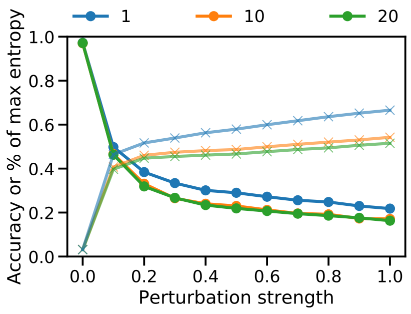


Figure 4: Prediction accuracy (circles) and average entropy (crosses) of CDNs for stronger FGSM based adversarial examples, constructed by averaging over multiple forward-backward passes.

In these experiments, we use a MLP with one layer with 100 hidden units as the predictive network, while the hypernetworks of the CDNs are modeled with two-layer MLPs with 10 hidden units each. Three samples of $\boldsymbol{\theta}$ are used to approximate the objectives during training of CDNs and BNNs.[11] If not stated otherwise models are optimized over 10000 iterations. A regularization hyperparameter of $\lambda = 10^{-3}$ is used for training CDNs.

The results for the first dataset (shown in the first row of Figure 5) demonstrate that all models were able to capture the epistemic uncertainty. For enabling the CDN to do so, it was however necessary to stop training earlier (results shown for stopping after 1700 iterations), and we observed that the uncertainty decreased over training. A detailed analysis of this behavior and its dependency on initialization is left for future work. Results for the second dataset show that the mixture models, i.e. the CDN and the DE, are the only ones able to capture the aleatoric uncertainty on the second dataset, as shown in the second row of Figure 5. This can be explained by the ability of CDNs and DEs to model input-dependent variance.

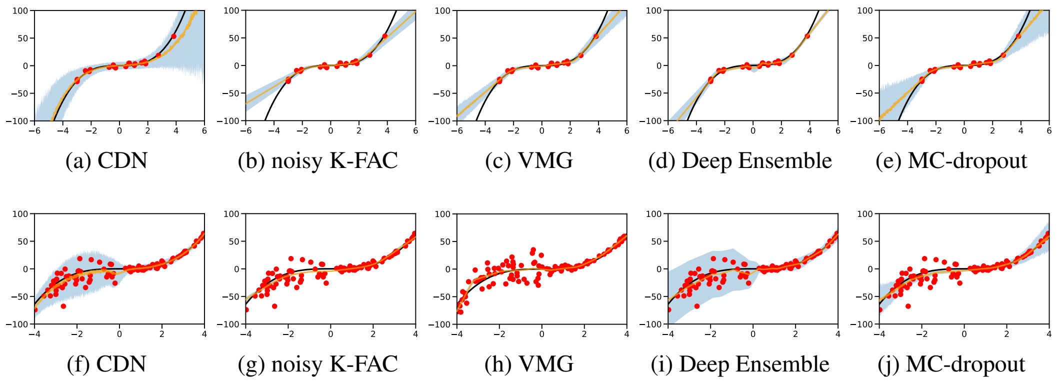


Figure 5: Comparison of the predictive distributions given by the CDN and the baselines on toy datasets with homoscedastic noise and few samples (first row) and heteroscedastic noise and more samples (second row). Black lines correspond to the true noiseless function, red dots correspond to samples, orange lines and shaded regions correspond to the empirical mean and the ± 3 standard deviation of the predictive distribution, respectively.

D.4 Experiments on Fashion-MNIST

For experiments on the Fashion-MNIST dataset we used the same network architecture as for MNIST cf. Section 2. The CDN achieves results competitive to the baselines for OOD classification (Figure 6).

[11]On these toy datasets, we found that using more than one sample is crucial for the results of the CDNs (i.e. results for using just one sample look similar to that of the VMG and Noisy K-FAC), while it does not significantly change the behaviour of the BNNs.

For adversarial attacks, the CDN achieves the highest accuracy by increased perturbation strength while having a lower entropy than baseline models. We note that strangely, the DPN's uncertainty estimate is decreasing with increasing perturbation strength.

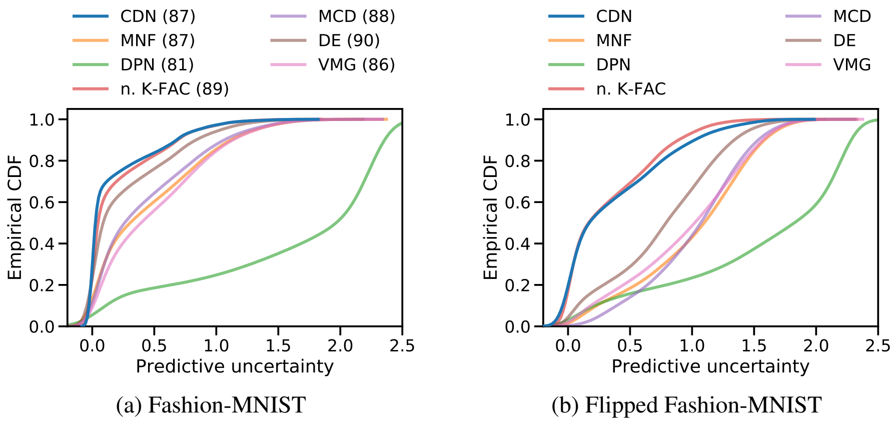


Figure 6: The first two pictures show the CDFs of the empirical entropy of the predictive distribution of the models, trained on Fashion-MNIST. The caption of each figure indicates the test set used, the y-axis denotes the fraction of predictions having entropy less than the corresponding value on the x-axis.

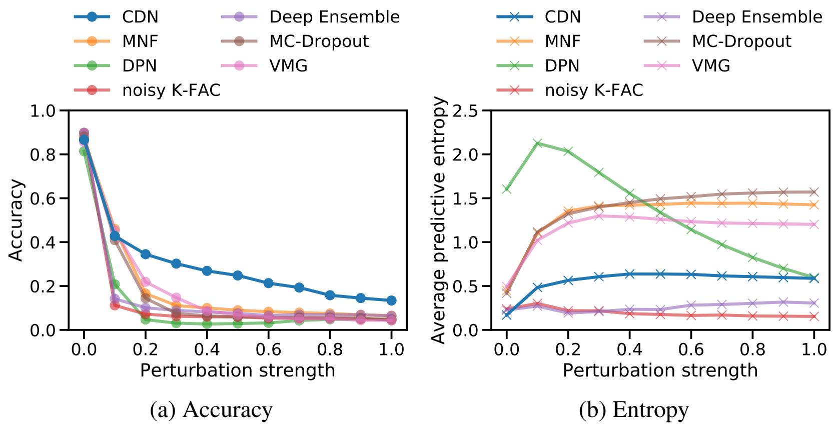


Figure 7: Prediction accuracy and average entropy of models trained on Fashion-MNIST when attacked by FGSM-based adversarial examples with varying perturbation strength.