Approximating the Predictive Distribution via Adversarially-Trained Hypernetworks

Christian Henning [*,†] henningc@ethz.ch
Johannes von Oswald [*,†] oswald@ini.ethz.ch
João Sacramento [†] sacramento@ini.ethz.ch
Simone Carlo Surace [‡] surace@ini.uzh.ch
Jean-Pascal Pfister [†,‡] jpfister@ini.uzh.ch
Benjamin F. Grewe [†] bgrewe@ethz.ch

Abstract

Being able to model uncertainty is a vital property for any intelligent agent. In an environment in which the domain of input stimuli is fully controlled neglecting uncertainty may work, but this usually does not hold true for any real-world scenario. This highlights the necessity for learning algorithms that robustly detect noisy and out-of-distribution examples. Here we propose a novel approach for uncertainty estimation based on adversarially trained hypernetworks. We define a weight posterior to uniformly allow weight realizations of a neural network that meet a chosen fidelity constraint. This setting gives rise to a posterior predictive distribution that allows inference on unseen data samples. In this work, we train a combination of hypernetwork and main network via the GAN framework by sampling from this posterior predictive distribution. Due to the indirect training of the hypernetwork our method does not suffer from complicated loss formulations over weight configurations. We report empirical results that show that our method is able to capture uncertainty over outputs and exhibits performance that is on par with previous work. Furthermore, the use of hypernetworks allows producing arbitrarily complex, multi-modal weight posteriors.

1 Introduction

Deep neural networks are flexible models that have succeeded in a great number of challenging problems [23]. However, in their standard form they do not provide any means of quantifying uncertainty. For example, when classifying unseen data far from training examples, deep networks often *overconfidently* attribute high probability to one of the classes, even though in reality they should express high uncertainty in their output. This can be devastating in situations in which a neural network bares responsibilities that might influence the integrity of valuable machinery/property or even the safety of human lives. In this work, we focus on weight uncertainty, although modelling other sources of uncertainty, for instance, from noisy inputs can be advantageous as well [15].

During learning, weight uncertainty naturally arises due to the finite nature of the available data. As a result, multiple model configurations are usually consistent with any given set of examples. One way to handle this source of uncertainty is to view learning as a problem of Bayesian inference [25]. In feedforward Bayesian neural networks (BNNs) the network output $y = f(x, W)$, parameterized by weights $W \in \mathbb{R}^K$, is given a statistical interpretation that is captured by the likelihood $p(y \mid x, W)$. Considering a probabilistic network $f(x, W)$, the likelihood $p(y \mid x, W)$ is a categorical distribution

[*]These authors contributed equally to this work.
[†]Institute of Neuroinformatics, UZH / ETH Zurich
[‡]Department of Physiology, University of Bern

Third workshop on Bayesian Deep Learning (NeurIPS 2018), Montréal, Canada.

in classification tasks and typically modelled via a softmax function, while for regression tasks the simplest choice is to assume that Gaussian noise corrupts the output. After observing a set of data points $D = \{(x_n, y_n)\}_{n=1}^{N}$, the prior parameter distribution $p(W)$, which incorporates all initial beliefs over plausible network configurations, can then be updated through Bayes' theorem. This yields the posterior parameter distribution $p(W \mid D) = \frac{p(Y|X,W)p(W)}{p(Y|X)}$.

For new unseen data points, predictions can then be made via the posterior predictive distribution by integrating over the posterior $p(W \mid D)$

$$p(y \mid x, D) = \int_W p(W \mid D)\, p(y \mid x, W)\, dW \qquad . \tag{1}$$

However, the analytical computation of $p(W \mid D)$ is intractable for neural networks, so in practice, an approximate inference scheme must be used. One possibility is to use Laplace's method and to approximate the posterior distribution by a multivariate Gaussian with the precision set to the Hessian of the log-likelihood [25]. Although computing the Hessian is not always feasible, this approximation is simple and works well in some cases. Alternatively, Markov chain Monte Carlo methods can be used to obtain samples from the posterior distribution [3, 30, 35]. However, these methods can be time-consuming and they often suffer from convergence issues. Another possibility is to consider *variational inference* (VI) [2, 4, 9, 12, 17]. In VI, an auxiliary distribution $q(W)$ is constructed to match the posterior, as measured by the Kullback-Leibler divergence. However, due to feasibility reasons, the expressiveness of the approximate posterior $q(W)$ or the actual posterior $p(W|D)$ (due to a poor choice of prior) is often severely deteriorated.

Note, the aforementioned methods are working with $p(W \mid D)$ as a means to approach inference via eq. (1). We attempt to bypass the problem of approximating or sampling from the posterior parameter distribution $p(W \mid D)$ by choosing it to model a class of *high-fidelity* weight realizations. These are parameter configurations of $f(x, W)$ that meet a certain (*high-fidelity*) performance on D (e.g., accuracy in classification tasks). This fully determines the posterior predictive distribution $p(y \mid x, D)$ given a certain choice of *main network* $f(x, W)$. Our goal is to approximate $p(y \mid x, D)$ via $q(y \mid x)$ as described in section 3. The approximation is learned via an adversarial scheme that contrasts samples from $p(y \mid x, D)$ with those retrieved from our combination of *main network* and *hypernetwork*. Empirical results are reported in section 4.

2 Related Work

Generative Adversarial Networks. *Generative Adversarial Networks* (GANs) [7] perform distribution-matching using two neural networks, a *generator* and a *discriminator*. The *generator* is a generative model that translates latent embeddings into data samples such as images. Thus, it defines an implicit distribution (i.e., we can use it to easily draw samples from an *unknown* distribution and compute gradients with respect to those samples). The *discriminator* judges the likelihood of these samples compared to a reference distribution. Both networks are trained in a two-player game, in which the *discriminator* tries to discriminate real (from the reference distribution) from fake (produced by the *generator*) samples and the *generator* tries to fool the *discriminator*. The optimum of this game is shown to minimize some divergence between these distributions, but the exact form of divergence (or even distance) depends on the loss function [1, 31]. In our work, we will use this framework to find an approximation of the posterior predictive distribution.

Uncertainty Estimation. A promising direction to capture parameter uncertainty in neural networks is guided by variational inference, that allows learning an approximate posterior. This can be achieved by maximizing the so-called *evidence lower bound* (ELBO), which consists of a data log-likelihood term and a prior-matching term $-\mathrm{KL}(q(W) \,||\, p(W))$. This last term is usually hard to optimize, thus necessitating simplifications by restricting the richness of the family of approximations $q(W)$ or by severely restricting the actual posterior $p(W|D)$ due to a poor choice of prior. This is evident in many of the related studies on *hypernetworks*.

A *hypernetwork* $h_\theta(z)$ is a neural network with weights θ that maps a latent input $z \sim p_z(z)$ into the weights W of a *main network* $f(x, W)$. Hypernetworks were originally introduced by Ha et al. [10] to reduce the number of trainable weights while keeping a rich architecture. They were repurposed and

reintroduced multiple times in different forms to approximate the weight posterior while employing different strategies to estimate the so-called prior-matching term [14, 20, 24, 32, 33]. However, all of these methods have drawbacks, in that either scalability is limited or the expressiveness in terms of the posteriors that can be modelled is reduced. Louizos and Welling [24] are using *Normalizing Flows* to estimate the uncertainty of a multiplicative influence on the means of a Gaussian weight posterior. *Normalizing Flows* [33, 18] are invertible neural networks, that make use of the *change of variables* formula to compute the density of network outputs given the probability density function (pdf) of their inputs. Such networks need to be carefully designed to allow an efficient computation of the determinant of its Jacobian. As such, *Normalizing Flows* allow a Monte Carlo estimate of the prior-matching term as the pdf of the network outputs can be computed, which makes VI feasible. This method is also utilized by Krueger et al. [20] in a similar way by training a *hypernetwork* that outputs a single scaling factor per unit for all incoming weights. Pawlowski et al. [32] applied VI to a *hypernetwork* that was not restricted in learning an arbitrary complex weight posterior. They use an approximation of the KL-divergence to optimize the prior-matching term, which they apply to all weights independently. An interesting method has been recently proposed by Karaletsos et al. [14] by shifting the problem of performing VI to the input space of the *hypernetwork*. The presented *hypernetwork* receives a combination of two unit-specific *Meta Embeddings* to generate the weight connecting these units. These embeddings are sampled from a learned data-dependent posterior over the latent space which they termed *MetaPrior*.

VI has also been applied directly to the weights of a neural network (without the need of an additional *hypernetwork*), for instance, as employed by *Bayes by Backprop* (BbB) [2]. In BbB each weight is replaced by a Gaussian distribution whose mean and variance are learned via VI using unbiased Monte-Carlo gradients. One approach that only requires a minimal set of assumptions (i.e., choosing a prior) to learn arbitrary complex posteriors via VI is *adversarial variational Bayes* [28]. This method relies on *Generative Adversarial Networks* (GANs) to estimate the density ratio appearing within the prior-matching term.

Also using GANs, another group (Wang et al. [34]) trained a *hypernetwork* to directly match the weight posterior $p(W \mid D)$. Samples from the weight posterior are acquired via approximate MCMC samples (using stochastic gradient Langevin dynamics; SGLD). The advantage of this method is that the approximate posterior is ready to use during inference to retrieve an estimate of eq. (1) without the need of storing samples from the true posterior.

A scalable method for uncertainty estimation was presented in Lakshminarayanan et al. [21]. They train an ensemble of networks by using different random weight initializations and random shuffling of the training data. To smooth the predictive distribution they propose to train by incorporating adversarial examples.

The method proposed by Korattikara et al. [19] is similar to ours in that it aims to estimate the posterior predictive distribution $p(y \mid x, D)$. However, Korattikara et al. [19] use a single neural network and try to find W such that $p(y \mid x, W) \approx p(y \mid x, D)$ and use SGLD to find a teacher network (or a set of teacher networks) to retrieve samples from $p(y \mid x, D)$.

3 Method

The *hypernetwork* $h_\theta(z)$ maps samples $z \sim p_z(z)$ from a chosen latent distribution $p_z(z)$ into weight realizations $W \sim q(W)$. Here, $q(W)$ denotes the unknown but (in the non-parametric limit) arbitrary density of $p_z(z)$ transformed through $h_\theta(z)$. The goal of our new approach is to find a transformation $h_\theta(z)$ of the latent distribution $p_z(z)$ into an arbitrarily complex distribution $q(W)$, such that the posterior predictive distribution evoked by $q(W)$ matches the one generated by a "high performance" weight posterior given a choice of *main network* $f(x, W)$. To train the hypernetwork, we use an additional discriminator network d_ψ with weights ψ.

In the most straightforward scenario, one could utilize the distribution matching capabilities of GANs by using $h_\theta(z)$ as *generator* and a network $\hat{d}_{\psi'}(W)$ to discriminate whether its input comes from the hypernet or the real posterior $p(W|D)$. This is similar to what is proposed by Wang et al. [34], but the scalability of such an approach might be limited as the dimensionality of W rapidly increases and a huge amount of samples from $p(W|D)$ would be required to capture the actual shape of this distribution with the *hypernetwork*.

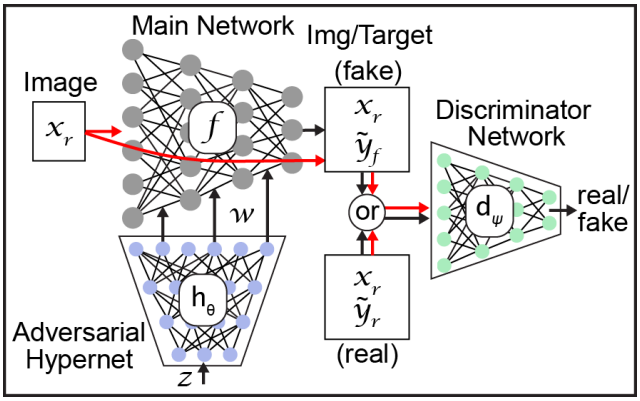

Figure 1: This figure illustrates our *Adversarial Hypernetwork*. The weights W of the main network f are generated by an auxiliary network h_θ. The discriminator d_ψ is used for training to ensure that h_θ generates weights that are appropriate for a desired output inference scheme $p(y \mid x, D)$.

Thus, we choose an indirect approach, in which the discriminator d_ψ observes inputs x and outputs [4] $\tilde{y}$ from the *main network*, where $\tilde{y}$ is sampled from a posterior predictive distribution $p(\tilde{y} \mid x, D)$. Our setup is sketched in Figure 1.

Note, that in general for two joint distributions $p(x, y)$ and $q(x, y)$ with a common marginal $p(x)$ it holds that

$$p(x, y) = q(x, y) \iff p(y \mid x) = q(y \mid x) \qquad \text{iff } p(x) > 0 \qquad . \tag{2}$$

Hence, we are considering the case of matching conditionals in the following even though we present pairs sampled from the joint distribution to the discriminator .

Fake samples are implicitly defined through the combination of *hyper-* and *main network*, i.e., by drawing an x from the real data distribution $p_{\text{data}}(x)$ and a corresponding $\tilde{y}$ from

$$q(\tilde{y} \mid x) = \int_W p(\tilde{y} \mid x, W)\, q(W)\, dW = \int_z \int_W p(\tilde{y} \mid x, W) \underbrace{q(W \mid z)}_{\delta\left(h_\theta(z) - W\right)} p_z(z)\, dW\, dz \qquad , \tag{3}$$

where $\delta(\cdot)$ denotes a Dirac-delta function since $h_\theta(z)$ is deterministic.

There are multiple options to retrieve samples from a desired posterior predictive distribution $p(\tilde{y} \mid x, W)$. One possibility is to generate samples from a Bayesian parameter posterior $p(W \mid D)$ (e.g., via SGLD as in [19, 34]). We choose to explore an alternative by defining a data-dependent parameter distribution $\tilde{p}(W \mid D)$.

We first introduce a fidelity function $\tau_f(W, D)$ that measures the *performance* of the main network f

$$\tau_f(W, D) = \frac{1}{N} \sum_{n=1}^{N} \tau_s\big(y_n, f(x_n, W)\big) \qquad , \tag{4}$$

where $\tau_s(\cdot, \cdot)$ denotes a per sample fidelity. For example, in a classification task[5], a natural choice for τ_f would be the mean accuracy over the dataset, i.e., $\tau_s(y, \tilde{y}) = \mathbb{1}_{y = \arg\max(\tilde{y})}$, where the function $\mathbb{1}_\beta$ is 1 only if the Boolean expression β evaluates to true and 0 otherwise. Using τ_f, we introduce the distribution of *high fidelity* solutions $\tilde{p}(W \mid D)$:

$$\tilde{p}(W \mid D) = \frac{1}{Z_W}\, p_u(W)\, \Theta\left[\tau_f(W, D) - \tau^*\right] \qquad , \tag{5}$$

[4] Note, we distinguish in our notation between the outputs $\tilde{y}$ of the deterministic *main network* $f(x, W)$ and the labels y as observed in the dataset D. This distinction becomes important if we are considering a probabilistic *main network* in which case $\tilde{y}$ might be the parameters of a distributions from which a label y can be sampled.

[5] Assuming the output of the *main network* $f(x, W)$ resembles class probabilities.

where $\Theta[\cdot]$ denotes the Heaviside step function, $p_u(W)$ is a uniform prior over the weights and $Z_W = \int dW\, p_u(W)\, \Theta\left[\tau_f(W, D) - \tau^*\right]$ ensures that the distribution is properly normalized. Therefore, we need to make sure that Z_W is non-zero and finite. This is either naturally the case given the performance criterion $\tau_f(W, D) \geq \tau^*$ or enforced by restricting W to a finite but arbitrary domain. In eq. (5) we constrain the solution space by imposing a high fidelity criterion τ^*[6].

In a regression task, in which we want to ensure a mean-squared error smaller than ϵ, an obvious choice for the per sample fidelity is $\tau_s(y, \tilde{y}) = -\|y - \tilde{y}\|_2^2$, where the fidelity criterion is set to $\tau^* = -\epsilon$.

From $\tilde{p}(W \mid D)$ we construct via marginalization the *target* predictive distribution:

$$\tilde{p}(\tilde{y} \mid x, D) = \int_W p(\tilde{y} \mid x, W)\, \tilde{p}(W \mid D)\, dW \qquad . \tag{6}$$

As in eq. (3), the likelihood $p(\tilde{y} \mid x, W)$ is evaluated by using the *main network* f. We use $\tilde{p}(\tilde{y} \mid x, D)$ to create a dataset of *real* samples as described in section 3.1 to train the *hypernetwork* via $d_\psi(x, \tilde{y})$.

We utilize the GAN framework to find a configuration of the *hypernetwork* weights θ, such that $\tilde{p}(\tilde{y} \mid x, D) = q(\tilde{y} \mid x)$[7]. Note, we do not need to compute the densities $\tilde{p}(\tilde{y} \mid x, D)$, $q(\tilde{y} \mid x)$ nor $p_{\text{data}}(x)$ as long as we can sample from these distributions. Algorithm 1 summarizes our approach.

Algorithm 1 Training of an Adversarial Hypernetwork

$n_d \leftarrow$ # iterations to train discriminator before updating the generator
$G(\cdot), D(\cdot, \cdot) \leftarrow$ generator and discriminator loss
$m \leftarrow$ mini-batch size
while not convergered **do**
    // Train discriminator to optimality.
    for $t \leftarrow 1$ to n_d **do**
        Sample $z^{(1)}, \ldots, z^{(m)}$ from $p_z(z)$
        Sample $x^{(1)}, \ldots, x^{(m)}$ from $p_{\text{data}}(x)$
        Sample $\tilde{y}_r^{(i)}$ from $\tilde{p}(\tilde{y} \mid x^{(i)})$ for $i \in \{1, \ldots, m\}$
        Sample $\tilde{y}_f^{(i)}$ from $\tilde{p}\big(\tilde{y} \mid x^{(i)}, h_\phi(z^{(i)})\big)$ for $i \in \{1, \ldots, m\}$
        // Compute discriminator gradient g_ψ and perform weight update.
        $g_\psi \leftarrow \frac{1}{m} \sum_{k=1}^{m} \nabla D(d_\psi(x^{(k)}, \tilde{y}_f^{(k)}), d_\psi(x^{(k)}, \tilde{y}_r^{(k)}))$
        $\psi \leftarrow$ Adam-Optimizer(ψ, g_ψ)
    // Update the generator.
    Sample $z^{(1)}, \ldots, z^{(m)}$ from $p_z(z)$
    Sample $x^{(1)}, \ldots, x^{(m)}$ from $p_{\text{data}}(x)$
    Sample $\tilde{y}_f^{(i)}$ from $\tilde{p}\big(\tilde{y} \mid x^{(i)}, h_\phi(z^{(i)})\big)$ for $i \in \{1, \ldots, m\}$
    // Compute generator gradient g_θ and perform weight update.
    $g_\theta \leftarrow \frac{1}{m} \sum_{k=1}^{m} \nabla G(d_\psi(x^{(k)}, \tilde{y}_f^{(k)}))$
    $\theta \leftarrow$ Adam-Optimizer(θ, g_θ)

3.1 Generating Samples from the Target Predictive Distribution

In this work, we are focusing on using a deterministic *main network* $f(x, W)$. In a classification task, we assume that the output $\tilde{y}$ of $f(x, W)$ are the parameters $\pi \in \mathbb{R}^C$ of a categorical distribution with C classes. Samples from $\tilde{p}(\tilde{y} \mid x, D)$ can be retrieved by training an ensemble of M classifiers satisfying a given fidelity τ^*: $\tilde{W}_1, \ldots, \tilde{W}_M$. These ensemble weights are subsequently used to generate a training dataset $\tilde{D} = \{\big(x_i, f(x_i, \tilde{W}_j)\big) \mid j = \text{assign}(i) \text{ for all } (x_i, \cdot) \in D\}$, where the function "assign" distributes data indices uniformly onto ensembles. Note, the ensemble is only needed for the generation of the training set. Specifically, an ensemble weight realization $\tilde{W}_j$ can

[6] We assume, that the set of admissible weights is not empty, i.e., the chosen *main network* is expressive enough to fulfill the τ^* high fidelity criterion.

[7] Assuming the hypernetwork as a universal function approximator.

be deleted once it has been used to generate samples of the predictive distribution. This is in strong contrast to the ensemble method in Lakshminarayanan et al. [21], for which the ensemble needs to be kept during inference, causing a storage and runtime overhead.

In a regression task, an unknown function $g : \mathcal{X} \to \mathcal{Y}$ should be learned via observing (possibly noisy) samples $\big(x, y \approx g(x)\big) \in D$. The output of the *main network* are samples from the same space $\mathcal{Y}$ that hopefully capture the trend of the training data. A possible way to retrieve a training set $\tilde{D}$ for our approach is to train an ensemble of *main networks* that satisfy, for instance, a maximum mean-squared error. However, we choose a simpler alternative approach, in which we consider the dataset D as a set of samples that satisfy a desired but implicit (as unknown to us) fidelity on the true data population defined by g. Hence, we consider $\tilde{D} \approx D$, where each sample in $(x, y) \in D$ is generated by a weight realization $\tilde{W}_j$ from an unknown set of ensemble weights, such that $f\big(x, \tilde{W}_j\big) = y$ and $\tau_f(\tilde{W}_j, D)$ greater or equal some fidelity. This approach of choosing $\tilde{D}$ relies on the assumption that the *main network* f is expressive enough to explain the data observed in D. Comparing this heuristic approach with a $\tilde{D}$ constructed from a trained ensemble is the aim of our future work.

Another interesting direction that can be explored in future work is the use of a probabilistic *main network*. For instance, $f(x, W)$ could output the parameters $\tilde{y}$ of a distribution from which an actual output y would be sampled via $p(y|\tilde{y})$. A natural choice for a regression task would be a Gaussian $p(y|\tilde{y})$, where y is presented to the discriminator and $f(x, W)$ is trained using the reparametrization trick [16]. Such a probabilistic *main network* is especially interesting once a classification task is considered. As it is in general not possible to learn a mapping from a continuous to a discrete variable via backpropagation, a continuous relaxation of the categorical output has to be considered. This could be the Gumbel-softmax or Concrete distribution that have been simultaneously developed by Jang et al. [13] and Maddison et al. [26]. Using this method, the output presented to the discriminator would be prototypical 1-hot encodings. Now, using the same line of thought as applied to a regression task with a deterministic *main network*, the heuristic $\tilde{D} \approx D$ can be explored.

4 Experiments

All of our experiments have been performed using the Least-Squares GAN loss from Mao et al. [27] to train both, discriminator and hypernetwork. Code by Pawlowski et al. [32] was used for comparison with other methods.

4.1 Toy Regression

As introduced by Hernández-Lobato and Adams [11], we approximate a cubic function via noisy observations: $g(x, \epsilon) = x^3 + \epsilon$, $\epsilon \sim \mathcal{N}(0, 9)$. This benchmark is intended as a *human-comprehensible* sanity check of uncertainty modelling, as it is tested within and outside the domain of the training data and the output uncertainty in these domains can be visually assessed.

Implementation Details. The training data consists of 20 samples $\big(x, g(x, \epsilon)\big)$ with $x \in [-4, 4]$ and $\epsilon \sim \mathcal{N}(0, 9)$. Thus, the *discriminator* gets real samples $\big(x, g(x, \epsilon)\big)$ and fake samples $\big(x, f(x, h_\theta(z))\big)$ with $z \sim \mathcal{N}(0, I)$ as input. For this task, all networks consist of fully-connected layers only. The *main network* has a single 100-units hidden layer. The *hypernetwork* has 3 hidden layers consisting of 16, 32 and 64 units and receives an 8-dimensional latent input z. The *discriminator* was chosen to have 3 hidden layers with 64, 32 and 16 units. To ensure consistency for all methods, we always used the same setup for the *main network*. We train both networks, *hypernetwork* $h_\theta(z)$ and *discriminator* $d_\psi(x, y)$, using the Adam optimizer with the PyTorch default parameters.

Results. The results of this experiment are depicted in Figure 2. Compared to other methods, our approach shows a smooth approximation in areas that have a high density of training samples, nicely capturing their variance, while the uncertainty drastically increases in a non-linear fashion when training data points are sparse or absent. Here, our approach accounts for the fact that it hasn't seen enough evidence to confidently predict in these areas.

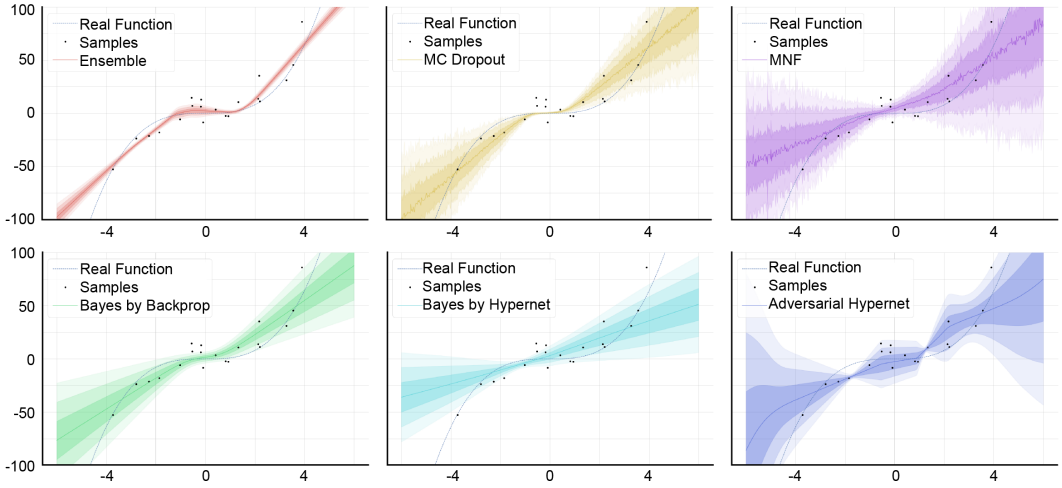

Figure 2: Comparison of our *Adversarial Hypernetwork* approach (bottom right) to other state-of-the-art methods on the toy regression task from section 4.1. Plots show the following methods (from top left to bottom right): Ensemble [21], MC Dropout [5], MNF [24], BbB [2], BbH [32] and the *Adversarial Hypernetwork* (ours).

4.2 MNIST

This experiment is performed on the MNIST dataset [22], which consists of 60.000 training and 10.000 test samples of labeled hand-written digits. To measure uncertainty, we consider two kinds of out-of-distribution samples: Turning MNIST [6] and notMNIST[8]. Turning MNIST is a qualitative experiment in which a single MNIST digit is rotated while its change in the predictive distribution is measured in terms of entropy. For a quantitative analysis of uncertainty we resort to the widely used notMNIST benchmark [21, 24, 32], which consists of letters rather than digits. Thus, data that the model has never observed during training. Therefore, the uncertainty on output predictions on these samples is ideally very high.

Implementation Details. The *main network* uses a Convolutional Neural Network architecture with $K = 21.840$ weights. It consists of two convolutional layers using max-pooling (with stride 2) followed by two fully-connected hidden layers of size 320 and 50, respectively. The network uses ReLUs as non-linearities for all hidden layers and a softmax output. The *hypernetwork* consists of one fully-connected hidden layer with 100 units using a ReLU non-linearity. Multiple fully-connected layers then project to the individual weight tensors of the *main network*. The *discriminator* consists of 2 strided-convolutional and 2 non-strided-convolutional layers with a final fully-connected layer to process input images x. The input $\tilde{y}$ is send through a fully-connected layer and then added as a dynamic bias to the output filters of the second convolutional layer. All layers, except the output, are using LeakyReLU non-linearities. To prevent mode collapse, we allow the discriminator to compare samples within a mini-batch (that corresponds to one weight draw W) and computed simple batch statistics $v = \frac{1}{m}\sum_{i=1}^{m} \tilde{y}_i$, where m denotes the batch size. These are concatenated to the input $\tilde{y}$ before send into the *discriminator*. We use Spectral Normalization [29] to stabilize GAN training.

The dimensionality of the latent space is 8 with $z \sim \mathcal{N}(0, I)$. As for the toy example we use the Adam optimizer for the hypernetwork and main network with the learning rate set to 0.00005 and default beta values.

In order to train our networks, we need to first generate a training dataset $\tilde{D}$ from D as described in section 3.1. First, a training set with 60.000 and test set with 10.000 data points $(x, \tilde{y})$ was constructed with $\tilde{y}$ the output of either 1 of 10 directly trained *main networks* (using cross-entropy loss; no *hypernetwork*) with different random initialization. We uniformly distributed MNIST training and test images across these networks to produce corresponding $\tilde{y}$ samples. We choose the training accuracy as fidelity measure with τ^* corresponding to 95%. Rather than exploring the hyperparameter

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

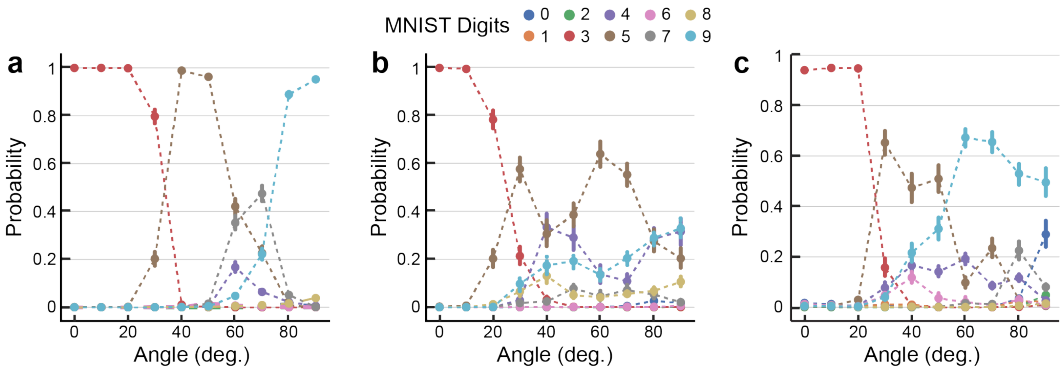

Figure 3: Turning MNIST example with an image of the digit 3. Method comparison (a) BbB [2], (b) MNF [24] and *Adversarial Hypernetwork* (c). For our method, variance on predictions for examples far away from the training sample (corresponding to high angle) is high while the confidence is low.

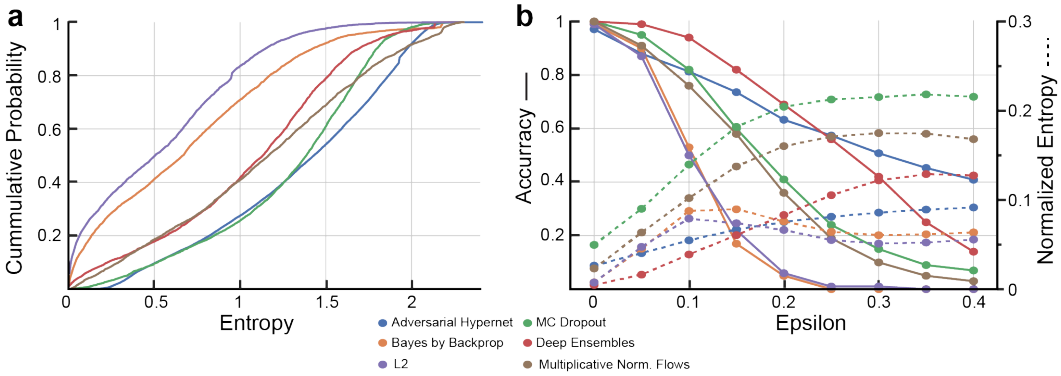

Figure 4: (a) Cumulative density function (CDF) of the entropies evaluated on the parameters of the predictive distributions observed for notMNIST samples. (b) Evolution of accuracy (bold lines) and observed uncertainty (dashed lines) with increasing strength of an adversarial attack on the inputs.

τ^* we decided to explore the cheap option of considering noisy perturbations of 1-hot encodings to produce a dataset similar to $\tilde{D}$. Therefore, we took all samples (x, y) from the MNIST dataset and perturbed the 1-hot encodings corresponding to y by Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.0075)$ such that zero values were positively perturbed and one values negatively. After renormalization we retrieved a dataset $\tilde{D}_{\text{noisy}}$ that showed similar performance to $\tilde{D}$. We used $\tilde{D}_{\text{noisy}}$ for all our experiments. A thorough investigation on how to generate appropriate datasets $\tilde{D}$ is left for future work to explore.

Results. We successfully trained the *main network* with our approach on MNIST with an accuracy of 98.3%. We closely follow Louizos and Welling [24] as standard experiments for measuring the quality of uncertainty estimation. First, we demonstrate the effectiveness of our method on Turning MNIST images, Figure 3. The predicted confidence values of a rotated image showing a 3 of different angles is measured over 100 weight samples for all compared methods. Those softmax outputs averaged across weight draws are plotted with their variance indicated through error bars. Our method successfully identifies images far from training data through predictions with low probabilities and high variance comparable to the results achieved by MNF [24].

We also estimate the entropy of the predictive distributions on notMNIST as in Lakshminarayanan et al. [21]. The results of this experiment are depicted in Figure 4a. In these experiments, the softmax outputs of a 100 weight draws are averaged and the entropy of this averaged *ensemble* categorical distribution is computed for the first 1000 images from the notMNIST dataset. These entropies are used to generate the depicted cumulative density plot. Note, that high uncertainty corresponds to high output entropy, i.e., it is desirable to push the mass as far as possible to the right for out-of-distribution samples in this CDF plot. Our method successfully identifies this dataset as unknown through

overall high prediction entropy and clearly performs on par with the state-of-the-art methods on this uncertainty measure.

We observe a very strong robustness of our method against the fast gradient sign adversarial attack from Goodfellow et al. [8], which is shown in Figure 4b. This adversarial attack adds a perturbation vector to the original MNIST images, which is computed as the sign of the gradient of a Cross-Entropy loss with respect to the inputs applying the ground-truth label. This perturbation will increase the loss, while its influence is controlled by a scalar ϵ. The Figure 4b depicts the development of the test accuracy (as bold lines) with increasing strength of the adversarial attack. Ideally, a drop in accuracy is accompanied by a rise in entropy of the predictive distribution (dotted lines). The entropy is normalized by its maximum value. Note, that the methods have different initial entropy values (at $\epsilon = 0$), which is most likely influencing the notMNIST experiments (as methods with a higher baseline entropy may have a higher entropy in general), showing the weakness of the current benchmarks in this field. Our method retains a high accuracy for moderate values of ϵ, which is a possible explanation for the modest increase in entropy. It should be noted that the Deep Ensembles method [21] is trained using adversarial examples to obtain a smoother approximation of the predictive distribution.

We emphasize the need for experiments that do take the performance accuracy and the mentioned *baseline entropy* on training data into account – for example, poor performance on the task at hand often leads to high entropy values on in- and out-of-distribution samples, which might result in wrong interpretations on the quality of uncertainty measurements.

Although we only report results on MNIST with a small architecture, preliminary results indicate the scalability of our method to larger architectures. Here, we tested the possibility of using a *conditional hypernetwork*, that receives learned embeddings as additional input to produce only a portion of the weights at a time as already suggested in Ha et al. [10].

5 Conclusions

Using two auxiliary networks, we demonstrate a novel training method for neural networks to mimic a posterior predictive distribution that incorporates uncertainty over the weights by learning from observed samples. We acknowledge the fact that multiple weight posteriors might result in the same posterior predictive distribution, which is why we conjecture that the solution space of the employed *hypernetwork* has a simpler error landscape and optima are easier to find. *Adversarial Hypernetworks* enable fast sampling of weight realizations and allow to perform approximate inference via $\tilde{q}(\tilde{y} \mid x)$ (see eq. 3), that empirically captures uncertainty over the outputs comparable to other state-of the-art methods. In this work, we explored the option of matching a predictive distribution defined through a *high-fidelity* weight distribution as defined by eq. (5). However, the proposed method is more general and might be applied to any kind of predictive distribution as long as one can sample from the underlying weight posterior.

Acknowledgments

This work was supported by the Swiss National Science Foundation (B.F.G. CRSII5-173721) and funding from the Swiss Data Science Center (B.F.G, C17-18). Jean-Pascal Pfister was supported by the Swiss National Science Foundation grant (PP00P3_179060). We want to thank all members of the Grewe-Lab and the group of Jean-Pascal Pfister for helpful discussions and feedback. Special thanks to Stefan Braun for constructive counselling on efficient implementations of our ideas.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1613–1622, 2015.

[3] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*, pages 1683–1691, January 2014.

[4] 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.

[5] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. In *4th International Conference on Learning Representations (ICLR) workshop track*, 2016.

[6] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the International Conference on Machine Learning*, pages 1050–1059, 2016.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[8] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of the 3th International Conference on Learning Representations (ICLR 2015)*. 2015.

[9] Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems 24*, pages 2348–2356. 2011.

[10] David Ha, Andrew Dai, and Quoc Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[11] José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proceedings of the 32nd International Conference on Machine Learning - Volume 37*, ICML'15, pages 1861–1869. JMLR.org, 2015.

[12] Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, COLT '93, pages 5–13, New York, NY, USA, 1993. ACM.

[13] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[14] Theofanis Karaletsos, Peter Dayan, and Zoubin Ghahramani. Probabilistic meta-representations of neural networks. *arXiv preprint arXiv:1810.00555*, 2018.

[15] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems 30*, pages 5574–5584. 2017.

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

[17] Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems 28*, pages 2575–2583. 2015.

[18] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems 29*, pages 4743–4751. Curran Associates, Inc., 2016.

[19] Anoop Korattikara, Vivek Rathod, Kevin Murphy, and Max Welling. Bayesian dark knowledge. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 3438–3446, Cambridge, MA, USA, 2015. MIT Press.

[20] David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. 2017.

[21] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems 30*, pages 6402–6413. Curran Associates, Inc., 2017.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[24] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational Bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2218–2227, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[25] David J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, May 1992.

[26] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[27] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2813–2821. IEEE, 2017.

[28] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2391–2400, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[29] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[30] Radford M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer New York, New York, NY, 1996.

[31] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems 29*, pages 271–279. Curran Associates, Inc., 2016.

[32] Nick Pawlowski, Martin Rajchl, and Ben Glocker. Implicit weight uncertainty in neural networks. *arXiv preprint arXiv:1711.01297*, 2017.

[33] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1530–1538, Lille, France, 2015.

[34] Kuan-Chieh Wang, Paul Vicol, James Lucas, Li Gu, Roger Grosse, and Richard Zemel. Adversarial distillation of bayesian neural network posteriors. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[35] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 681–688, USA, 2011. Omnipress.