Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning

Yarin Gal **Zoubin Ghahramani**

University of Cambridge
{yg279,zg201}@cam.ac.uk

Abstract

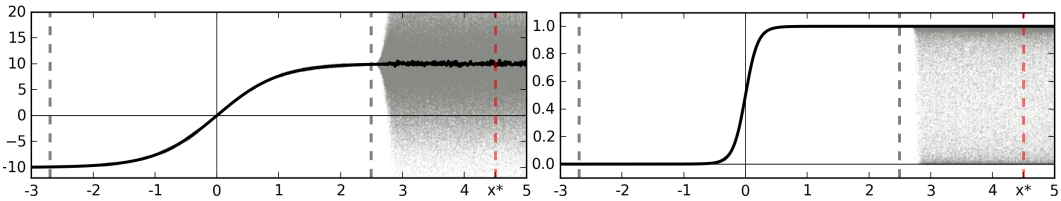
Deep learning tools have gained tremendous attention in applied machine learning. However such tools for regression and classification do not capture model uncertainty. In comparison, Bayesian models offer a mathematically grounded framework to reason about model uncertainty, but usually come with a prohibitive computational cost. In this paper we develop a new theoretical framework casting dropout training in deep neural networks (NNs) as approximate Bayesian inference in deep Gaussian processes. A direct result of this theory gives us tools to model uncertainty with dropout NNs – extracting information from existing models that has been thrown away so far. This mitigates the problem of representing uncertainty in deep learning without sacrificing either computational complexity or test accuracy. We perform an extensive study of the properties of dropout's uncertainty. Various network architectures and non-linearities are assessed on tasks of regression and classification, using MNIST as an example. We show a considerable improvement in predictive log-likelihood and RMSE compared to existing state-of-the-art methods, and finish by using dropout's uncertainty in deep reinforcement learning.

1 Introduction

Deep learning has attracted tremendous attention from researchers in fields such as physics, biology, and manufacturing, to name a few [1, 2, 3]. Tools such as the neural network (NN), dropout, convolutional neural networks (convnets), and others are used extensively. However, these are fields in which representing model uncertainty is of crucial importance [4, 5]. With the recent shift in many of these fields towards the use of Bayesian uncertainty [6, 7, 8] new needs arise from deep learning tools.

Standard deep learning tools for regression and classification do not capture model uncertainty. In classification, predictive probabilities obtained at the end of the pipeline (the softmax output) are often erroneously interpreted as model confidence. A model can be uncertain in its predictions even with a high softmax output (fig. 1). Passing a point estimate of a function (solid line 1a) through a softmax (solid line 1b) results in extrapolations with unjustified high confidence for points far from the training data. x^* for example would be classified as class 1 with probability 1. However, passing the distribution (shaded area 1a) through a softmax (shaded area 1b) better reflects classification uncertainty far from the training data.

Model uncertainty is indispensable for the deep learning practitioner as well. With model confidence at hand we can treat uncertain inputs and special cases explicitly. For example, in the case of classification, a model might return a result with high uncertainty. In this case we might decide to pass the input to a human for classification. This can happen in a post office, sorting letters according to their zip code, or in a nuclear power plant with a system responsible for critical infrastructure [9]. Uncertainty is important in reinforcement learning (RL) as well [10]. With uncertainty information an agent can decide when to exploit and when to explore its environment. Recent advances in RL



(a) Softmax *input* as a function of data $\mathbf{x}$: $f(\mathbf{x})$ (b) Softmax *output* as a function of data $\mathbf{x}$: $\sigma(f(\mathbf{x}))$

Figure 1: **A sketch of softmax input and output for an idealised binary classification problem.** Training data is given between the dashed grey lines. Function point estimate is shown with a solid line. Function uncertainty is shown with a shaded area. Marked with a dashed red line is a point x^* far from the training data. Ignoring function uncertainty, point x^* is classified as class 1 with probability 1.

have made use of NNs for Q-value function approximation. These are functions that estimate the quality of different actions an agent can make. Epsilon greedy search is often used where the agent selects its best action with some probability and explores otherwise. With uncertainty estimates over the agent's Q-value function, techniques such as Thompson sampling [11] can be used to learn much faster.

Bayesian probability theory offers us mathematically grounded tools to reason about model uncertainty, but these usually come with a prohibitive computational cost. It is perhaps surprising then that it is possible to cast recent deep learning tools as Bayesian models – without changing either the models or the optimisation. We show that the use of dropout (and its variants) in NNs can be interpreted as a Bayesian approximation of a well known probabilistic model: the Gaussian process (GP) [12]. Dropout is used in many models in deep learning as a way to avoid over-fitting [13], and our interpretation suggests that dropout approximately integrates over the models' weights. We develop tools for representing model uncertainty of existing dropout NNs – extracting information that has been thrown away so far. This mitigates the problem of representing model uncertainty in deep learning without sacrificing either computational complexity or test accuracy.

In this paper we give a complete theoretical treatment of the link between Gaussian processes and dropout, and develop the tools necessary to represent uncertainty in deep learning. We perform an extensive exploratory assessment of the properties of the uncertainty obtained from dropout NNs and convnets on the tasks of regression and classification. We compare the uncertainty obtained from different model architectures and non-linearities in regression, and show that model uncertainty is indispensable for classification tasks, using MNIST as a concrete example. We then show a considerable improvement in predictive log-likelihood and RMSE compared to existing state-of-the-art methods. Lastly we give a quantitative assessment of model uncertainty in the setting of reinforcement learning, on a practical task similar to that used in deep reinforcement learning [14].

2 Related Research

It has long been known that infinitely wide (single hidden layer) NNs with distributions placed over their weights converge to Gaussian processes [15, 16]. This known relation is through a limit argument that does not allow us to translate properties from the Gaussian process to finite NNs easily. Finite NNs with distributions placed over the weights have been studied extensively as *Bayesian neural networks* [15, 17]. These offer robustness to over-fitting as well, but with challenging inference and additional computational costs. Variational inference has been applied to these models, but with limited success [18, 19, 20]. Recent advances in variational inference introduced new techniques into the field such as *sampling-based* variational inference and *stochastic* variational inference [21, 22, 23, 24, 25]. These have been used to obtain new approximations for Bayesian neural networks that perform as well as dropout [26]. However these models come with a prohibitive computational cost. To represent uncertainty, the number of parameters in these models is doubled for the same network size. Further, they require more time to converge and do not improve on existing techniques. Given that good uncertainty estimates can be cheaply obtained from common dropout models, this results in unnecessary additional computation. An alternative approach to variational inference makes use of expectation propagation [27] and has improved considerably in RMSE and uncertainty estimation on VI approaches such as [20]. In the results section we compare dropout to these approaches and show a significant improvement in both RMSE and uncertainty estimation.

3 Dropout as a Bayesian Approximation

We show that a neural network with arbitrary depth and non-linearities, with dropout applied before every weight layer, is mathematically equivalent to an approximation to the probabilistic deep Gaussian process [28]. We would like to stress that no simplifying assumptions are made on the use of dropout in the literature, and that the results derived are applicable to any network architecture that makes use of dropout exactly as it appears in practical applications. Furthermore, our results carry to other variants of dropout as well (such as drop-connect [29], multiplicative Gaussian noise [13], hashed neural networks [30], etc.). We show that the dropout objective, in effect, minimises the Kullback–Leibler divergence between an approximate model and the deep Gaussian process. Due to space constraints we refer the reader to the appendix for an in depth review of dropout, Gaussian processes, and variational inference (section 2), as well as the main derivation for dropout and its variations (section 3). The results are summarised here and in the next section we obtain uncertainty estimates for dropout NNs.

Let $\widehat{\mathbf{y}}$ be the output of a NN model with L layers and a loss function $E(\cdot, \cdot)$ such as the softmax loss or the Euclidean loss (square loss). We denote by $\mathbf{W}_i$ the NN's weight matrices of dimensions $K_i \times K_{i-1}$, and by $\mathbf{b}_i$ the bias vectors of dimensions K_i for each layer $i = 1, ..., L$. We denote by $\mathbf{y}_i$ the observed output corresponding to input $\mathbf{x}_i$ for $1 \leq i \leq N$ data points, and the input and output sets as $\mathbf{X}, \mathbf{Y}$. During NN optimisation a regularisation term is often added. We often use L_2 regularisation weighted by some weight decay λ, resulting in a minimisation objective (often referred to as cost),

$$\mathcal{L}_{\text{dropout}} := \frac{1}{N} \sum_{i=1}^{N} E(\mathbf{y}_i, \widehat{\mathbf{y}}_i) + \lambda \sum_{i=1}^{L} \big(||\mathbf{W}_i||_2^2 + ||\mathbf{b}_i||_2^2 \big). \tag{1}$$

With dropout, we sample binary variables for every input point and for every network unit in each layer (apart from the last one). Each binary variable takes value 1 with probability p_i for layer i. A unit is dropped (i.e. its value is set to zero) for a given input if its corresponding binary variable takes value 0. We use the same values in the backward pass propagating the derivatives to the parameters.

In comparison to the non-probabilistic NN, the deep Gaussian process [28] is a powerful tool in statistics that allows us to model distributions over functions. Assume we are given a covariance function of the form $\mathbf{K}(\mathbf{x}, \mathbf{y}) = \int p(\mathbf{w}) p(b) \sigma(\mathbf{w}^T \mathbf{x} + b) \sigma(\mathbf{w}^T \mathbf{y} + b) \mathrm{d}\mathbf{w} \mathrm{d}b$ with some element-wise non-linearity $\sigma(\cdot)$ and distributions $p(\mathbf{w}), p(b)$. In sections 3 and 4 in the appendix we show that a deep Gaussian process with L layers and covariance function $\mathbf{K}(\mathbf{x}, \mathbf{y})$ can be approximated by placing a variational distribution over each component of a spectral decomposition of the GPs' covariance functions. This spectral decomposition maps each layer of the deep GP to a layer of explicitly represented hidden units, as will be briefly explained next.

Let $\widehat{\mathbf{M}}_i$ be a random matrix of dimensions $K_i \times K_{i-1}$ for each layer i, and write $\boldsymbol{\omega} = \{\widehat{\mathbf{M}}_i\}_{i=1}^L$. A-priori, we let each row of $\widehat{\mathbf{M}}_i$ distribute according to the $p(\mathbf{w})$ above. In addition, assume vectors $\mathbf{m}_i$ of dimensions K_i for each GP layer. The predictive probability of the deep GP model (with a finite rank covariance function and some precision parameter $\tau > 0$) can be re-parametrised as

$$p(\mathbf{y}|\mathbf{x}, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) \mathrm{d}\boldsymbol{\omega} \tag{2}$$

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega}) = \mathcal{N}\big(\mathbf{y}; \widehat{\mathbf{y}}(\mathbf{x}, \boldsymbol{\omega}), \tau^{-1} \mathbf{I}_D\big) \tag{3}$$

$$\widehat{\mathbf{y}}\big(\mathbf{x}, \boldsymbol{\omega} = \{\widehat{\mathbf{M}}_1, ..., \widehat{\mathbf{M}}_L\}\big) = \sqrt{\frac{1}{K_L}} \widehat{\mathbf{M}}_L \sigma\bigg(... \sqrt{\frac{1}{K_1}} \widehat{\mathbf{M}}_2 \sigma\big(\widehat{\mathbf{M}}_1 \mathbf{x} + \mathbf{m}_1\big) ... \bigg) \tag{4}$$

The posterior distribution $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$ in eq. (2) is intractable. We use $q(\boldsymbol{\omega})$, a distribution over matrices whose columns are randomly set to zero, to approximate the intractable posterior. We define $q(\boldsymbol{\omega})$ as:

$$\boldsymbol{\omega} = \{\widehat{\mathbf{M}}_i\}_{i=1}^L \tag{5}$$

$$\widehat{\mathbf{M}}_i = \mathbf{M}_i \cdot \text{diag}([\mathbf{z}_{i,j}]_{j=1}^{K_i}) \tag{6}$$

$$\mathbf{z}_{i,j} \sim \text{Bernoulli}(p_i) \text{ for } i = 1, ..., L,\ j = 1, ..., K_{i-1} \tag{7}$$

given some probabilities p_i and matrices $\mathbf{M}_i$ as variational parameters. The binary variable $\mathbf{z}_{i,j} = 0$ corresponds then to unit j in layer $i - 1$ being dropped out as an input to layer i. The variational

distribution $q(\boldsymbol{\omega})$ is highly multi-modal, inducing strong joint correlations over the rows of the matrices $\widehat{\mathbf{M}}_i$ (which correspond to the frequencies in the sparse spectrum GP approximation).

We minimise the KL divergence between the approximate posterior $q(\boldsymbol{\omega})$ above and the posterior of the full deep GP, $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$. We can rewrite this KL to obtain a minimisation objective

$$-\int q(\boldsymbol{\omega}) \log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} + \mathrm{KL}(q(\boldsymbol{\omega})||p(\boldsymbol{\omega})). \tag{8}$$

We rewrite the first term as a sum

$$-\sum_{n=1}^{N} \int q(\boldsymbol{\omega}) \log p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}$$

and approximate each term in the sum by Monte Carlo integration with a single sample $\widehat{\boldsymbol{\omega}}_n \sim q(\boldsymbol{\omega})$ (section 3.4 in the appendix) to get $-\log p(\mathbf{y}_n|\mathbf{x}_n, \widehat{\boldsymbol{\omega}}_n)$. We further approximate the second term in eq. (8) and obtain $\sum_{i=1}^{L} \left(\frac{p_i l^2}{2}||\mathbf{M}_i||_2^2 + \frac{l^2}{2}||\mathbf{m}_i||_2^2\right)$ with prior length-scale l (see section 4.2 in the appendix). Given model precision τ we scale the result by the constant $1/\tau N$ to obtain the objective:

$$\mathcal{L}_{\text{GP-MC}} \propto \frac{1}{N} \sum_{n=1}^{N} \frac{-\log p(\mathbf{y}_n|\mathbf{x}_n, \widehat{\boldsymbol{\omega}}_n)}{\tau} + \sum_{i=1}^{L} \left(\frac{p_i l^2}{2\tau N}||\mathbf{M}_i||_2^2 + \frac{l^2}{2\tau N}||\mathbf{m}_i||_2^2 \right). \tag{9}$$

Setting

$$E(\mathbf{y}_n, \widehat{\mathbf{y}}(\mathbf{x}_n, \widehat{\boldsymbol{\omega}}_n)) = -\log p(\mathbf{y}_n|\mathbf{x}_n, \widehat{\boldsymbol{\omega}}_n)/\tau$$

we recover eq. (1) for an appropriate setting of the precision hyper-parameter τ and length-scale l. The sampled $\widehat{\boldsymbol{\omega}}_n$ result in realisations from the Bernoulli distribution $\mathbf{z}_{i,j}^n$ equivalent to the binary variables in the dropout case[1].

4 Obtaining Model Uncertainty

We next derive results extending on the above showing that model uncertainty can be obtained from dropout NN models.

Following section 2.3 in the appendix, our approximate predictive distribution is given by

$$q(\mathbf{y}^*|\mathbf{x}^*) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) q(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} \tag{10}$$

where $\boldsymbol{\omega} = \{\widehat{\mathbf{M}}_i\}_{i=1}^{L}$ is our set of random variables for a model with L layers.

We will perform moment-matching and estimate the first two moments of the predictive distribution empirically. More specifically, we sample T sets of vectors of realisations from the Bernoulli distribution $\{\mathbf{z}_1^t, ..., \mathbf{z}_L^t\}_{t=1}^{T}$ with probabilities $\{p_1, ..., p_L\}$ and estimate

$$\mathbb{E}_{q(\mathbf{y}^*|\mathbf{x}^*)}(\mathbf{y}^*) \approx \frac{1}{T} \sum_{t=1}^{T} \widehat{\mathbf{y}}^*(\mathbf{x}^*, \widehat{\mathbf{M}}_1^t, ..., \widehat{\mathbf{M}}_L^t) \tag{11}$$

following proposition C in the appendix. We refer to this Monte Carlo estimate as *MC dropout*. In practice this is equivalent to performing T stochastic forward passes through the network and averaging the results.

This result has been presented in the literature before as model averaging. We have given a new derivation for this result which allows us to derive mathematically grounded uncertainty estimates as well. Srivastava et al. [13, section 7.5] have reasoned empirically that this quantity can be approximated by averaging the weights of the network (multiplying each $\mathbf{W}_i$ by p_i at test time, referred to as *standard dropout*).

We estimate the second raw moment in the same way:

$$\mathbb{E}_{q(\mathbf{y}^*|\mathbf{x}^*)}\big((\mathbf{y}^*)^T(\mathbf{y}^*)\big) \approx \tau^{-1}\mathbf{I}_D + \frac{1}{T} \sum_{t=1}^{T} \widehat{\mathbf{y}}^*(\mathbf{x}^*, \widehat{\mathbf{M}}_1^t, ..., \widehat{\mathbf{M}}_L^t)^T \widehat{\mathbf{y}}^*(\mathbf{x}^*, \widehat{\mathbf{M}}_1^t, ..., \widehat{\mathbf{M}}_L^t)$$

[1] In the appendix (section 4.1) we extend this derivation to classification. $E(\cdot, \cdot)$ is defined as the softmax loss and τ is set to 1.

following proposition D in the appendix. To obtain the model's predictive variance we have:
$\text{Var}_{q(\mathbf{y}^*|\mathbf{x}^*)}(\mathbf{y}^*) \approx$

$$\tau^{-1}\mathbf{I}_D + \frac{1}{T}\sum_{t=1}^{T}\widehat{\mathbf{y}}^*(\mathbf{x}^*, \widehat{\mathbf{M}}_1^t, ..., \widehat{\mathbf{M}}_L^t)^T\widehat{\mathbf{y}}^*(\mathbf{x}^*, \widehat{\mathbf{M}}_1^t, ..., \widehat{\mathbf{M}}_L^t) - \mathbb{E}_{q(\mathbf{y}^*|\mathbf{x}^*)}(\mathbf{y}^*)^T\mathbb{E}_{q(\mathbf{y}^*|\mathbf{x}^*)}(\mathbf{y}^*)$$

which equals the sample variance of T stochastic forward passes through the NN plus the inverse model precision. Note that $\mathbf{y}^*$ is a row vector thus the sum is over the *outer-products*. Given the weight-decay λ (and our prior length-scale l) we can find the model precision from the identity

$$\tau = \frac{pl^2}{2N\lambda}. \tag{12}$$

We can estimate our predictive log-likelihood by Monte Carlo integration of eq. (2). This is an estimate of how well the model fits the mean and uncertainty (see section 4.4 in the appendix). For regression this is given by:

$$\log p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) \approx \text{logsumexp}\left(-\frac{1}{2}\tau||\mathbf{y} - \widehat{\mathbf{y}}_t||^2\right) - \log T - \frac{1}{2}\log 2\pi - \frac{1}{2}\log\tau^{-1} \tag{13}$$

with a sum of T terms and $\widehat{\mathbf{y}}_t$ stochastic forward passes through the network.

Our predictive distribution $q(\mathbf{y}^*|\mathbf{x}^*)$ is expected to be highly multi-modal, and the above approximations only give a glimpse into its properties. This is because the approximating variational distribution placed on each weight matrix column is bi-modal, and as a result the joint distribution over each layer's weights is multi-modal (section 3.2 in the appendix).

Note that the dropout NN model itself is not changed. To estimate the predictive mean and predictive uncertainty we simply collect the results of stochastic forward passes through the model. As a result, this information can be used with existing NN models trained with dropout. Furthermore, the forward passes can be done concurrently, resulting in constant time complexity identical to that of standard dropout.

5 Experiments

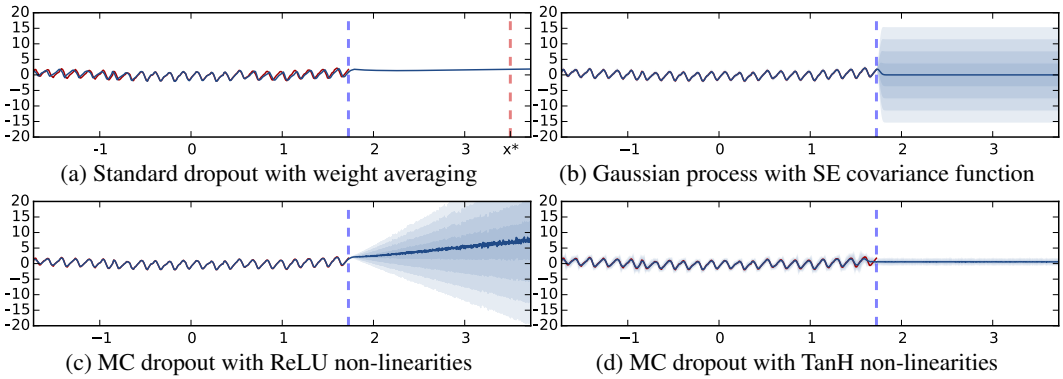
We next perform an extensive assessment of the properties of the uncertainty estimates obtained from dropout NNs and convnets on the tasks of regression and classification. We compare the uncertainty obtained from different model architectures and non-linearities, both on tasks of extrapolation, and show that model uncertainty is important for classification tasks using MNIST [31] as an example. We then show that using dropout's uncertainty we can obtain a considerable improvement in predictive log-likelihood and RMSE compared to existing state-of-the-art methods. We finish with an example use of the model's uncertainty in a Bayesian pipeline. We give a quantitative assessment of the model's performance in the setting of reinforcement learning on a task similar to that used in deep reinforcement learning [14].

Using the results from the previous section, we begin by qualitatively evaluating the dropout NN uncertainty on two regression tasks. We use two regression datasets and model scalar functions which are easy to visualise. These are tasks one would often come across in real-world data analysis. We use a subset of the atmospheric CO_2 concentrations dataset derived from in situ air samples collected at Mauna Loa Observatory, Hawaii [32] (referred to as *CO_2*) to evaluate model extrapolation. In the appendix (section D.1) we give further results on a second dataset, the reconstructed solar irradiance dataset [33], to assess model interpolation. The datasets are fairly small, with each dataset consisting of about 200 data points. We centred and normalised both datasets.

5.1 Model Uncertainty in Regression Tasks

We trained several models on the CO_2 dataset. We use NNs with either 4 or 5 hidden layers and 1024 hidden units. We use either ReLU non-linearities or TanH non-linearities in each network, and use dropout probabilities of either 0.1 or 0.2. Exact experiment set-up is given in section E.1 in the appendix.

Extrapolation results are shown figure 2. The model is trained on the training data (left of the dashed blue line), and tested on the entire dataset. Fig. 2a shows the results for standard dropout (i.e. with weight averaging and without assessing model uncertainty) for the 5 layer ReLU model. Fig. 2b



(a) Standard dropout with weight averaging

(b) Gaussian process with SE covariance function

(c) MC dropout with ReLU non-linearities

(d) MC dropout with TanH non-linearities

Figure 2: **Predictive mean and uncertainties on the Mauna Loa CO_2 concentrations dataset, for various models.** In red is the observed function (left of the dashed blue line); in blue is the predictive mean plus/minus two standard deviations (8 for fig. 2d). Different shades of blue represent half a standard deviation. Marked with a dashed red line is a point far away from the data: standard dropout confidently predicts an insensible value for the point; the other models predict insensible values as well but with the additional information that the models are uncertain about their predictions.

shows the results obtained from a Gaussian process with a squared exponential covariance function for comparison. Fig. 2c shows the results of the same network as in fig. 2a, but with MC dropout used to evaluate the predictive mean and uncertainty for the training and test sets. Lastly, fig. 2d shows the same using the TanH network with 5 layers (plotted with 8 times the standard deviation for visualisation purposes). The shades of blue represent model uncertainty: each colour gradient represents half a standard deviation (in total, predictive mean plus/minus 2 standard deviations are shown, representing 95% confidence). Not plotted are the models with 4 layers as these converge to the same results.

Extrapolating the observed data, none of the models can capture the periodicity (although with a suitable covariance function the GP will capture it well). The standard dropout NN model (fig. 2a) predicts value 0 for point x^* (marked with a dashed red line) with high confidence, even though it is clearly not a sensible prediction. The GP model represents this by increasing its predictive uncertainty – in effect declaring that the predictive value might be 0 but the model is uncertain. This behaviour is captured in MC dropout as well. Even though the models in figures 2 have an incorrect predictive mean, the increased standard deviation expresses the models' uncertainty about the point.

Note that the uncertainty is increasing far from the data for the ReLU model, whereas for the TanH model it stays bounded. This is not surprising, as dropout's uncertainty draws its properties from the GP in which different covariance functions correspond to different uncertainty estimates. ReLU and TanH approximate different GP covariance functions (section 3.1 in the appendix) and TanH saturates whereas ReLU does not. For the TanH model we assessed the uncertainty using both dropout probability 0.1 and dropout probability 0.2. Models initialised with dropout probability 0.1 initially exhibit smaller uncertainty than the ones initialised with dropout probability 0.2, but towards the end of the optimisation when the model has converged the uncertainty is almost undistinguishable. This is because the moments of the dropout models converge to the moments of the GP – its mean and uncertainty. It is worth mentioning that we attempted to fit the data with models with a smaller number of layers unsuccessfully.

The number of forward iterations used to estimate the uncertainty (T) was 1000 for drawing purposes. A much smaller numbers can be used to get a reasonable estimation to the predictive mean and uncertainty (see fig. 3 for example with $T = 10$).

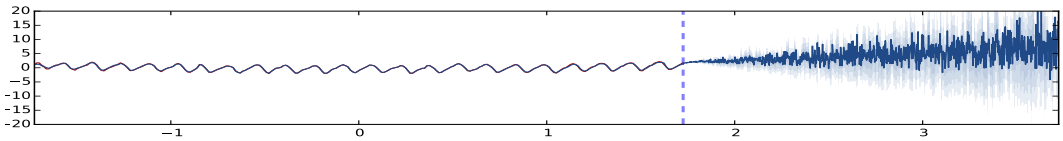
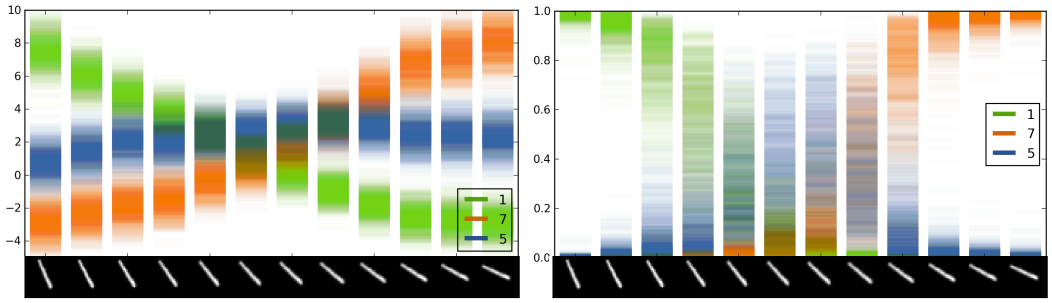


Figure 3: Predictive mean and uncertainties on the Mauna Loa CO_2 concentrations dataset for the MC dropout model with ReLU non-linearities, approximated with 10 samples.



(a) Softmax *input* scatter

(b) Softmax *output* scatter

Figure 4: **A scatter of 100 forward passes of the softmax input and output for dropout LeNet.** On the X axis is a rotated image of the digit 1. The input is classified as digit 5 for images 6-7, even though model uncertainty is extremly large (best viewed in colour).

5.2 Model Uncertainty in Classification Tasks

To assess model classification confidence in a real world example we test a convolutional neural network trained on the full MNIST dataset [31]. We trained the LeNet convolutional neural network model [34] with dropout applied before the last fully connected inner-product layer (the usual way dropout is used in convnets). We used dropout probability of 0.5. We trained the model for $1e6$ iterations with the same learning rate policy as before with $\gamma = 0.0001$ and $p = 0.75$. We used Caffe [35] reference implementation for this experiment.

We evaluated the trained model on a continuously rotated image of the digit 1 (shown on the X axis of fig. 4). We scatter 100 stochastic forward passes of the softmax input (the output from the last fully connected layer, fig. 4a), as well as of the softmax output for each of the top classes (fig. 4b). For the 12 images, the model predicts classes [1 1 1 1 1 5 5 7 7 7 7 7].

The plots show the softmax input value and softmax output value for the 3 digits with the largest values for each corresponding input. When the softmax input for a class is larger than that of all other classes (class 1 for the first 5 images, class 5 for the next 2 images, and class 7 for the rest in fig 4a), the model predicts the corresponding class. Looking at the softmax input values, if the uncertainty envelope of a class is far from that of other classes' (for example the left most image) then the input is classified with high confidence. On the other hand, if the uncertainty envelope intersects that of other classes (such as in the case of the middle input image), then even though the softmax output can be arbitrarily high (as far as 1 if the mean is far from the means of the other classes), the softmax output uncertainty can be as large as the entire space. This signifies the model's uncertainty in its softmax output value – i.e. in the prediction. In this scenario it is not reasonable to use probit to return class 5 for the middle image when its uncertainty is so high. One would expect the model to ask an external annotator for a label for this input.

Compared to the regression case where predictive uncertainty can be captured with output standard deviation, in classification tasks we often use variation ratios instead. In practice this means that we would sample from the softmax probabilities at the end of each stochastic forward pass to get a label for the input. Collecting a set of labels from multiple stochastic forward passes on the same input we can find the mode of the distribution. The variation ratio is then a measure of dispersion – how "spread" the distribution is around the mode. If most labels agree with the mode, the model is certain about the output. This happens when both the mean of the softmax output is high, and the envelope around it is small. Large dispersion indicates model uncertainty. This happens when either the softmax mean is low (e.g. two classes with probabilities 0.5), or when the envelope is large (so different classes will have high probability at different stochastic passes through the network).

5.3 Predictive Performance

Predictive log-likelihood captures how well a model fits the data, with larger values indicating better model fit. Uncertainty quality can be determined from this quantity as well (see section 4.4 in the appendix). We replicate the experiment set-up in Hernández-Lobato and Adams [27] and compare the RMSE and predictive log-likelihood of dropout (referred to as "Dropout" in the experiments)

	Avg. Test RMSE and Std. Errors			Avg. Test LL and Std. Errors		
Dataset	**VI**	**PBP**	**Dropout**	**VI**	**PBP**	**Dropout**
Boston Housing	4.32 ±0.29	3.01 ±0.18	**2.97 ±0.85**	-2.90 ±0.07	-2.57 ±0.09	**-2.46 ±0.25**
Concrete Strength	7.19 ±0.12	5.67 ±0.09	**5.23 ±0.53**	-3.39 ±0.02	-3.16 ±0.02	**-3.04 ±0.09**
Energy Efficiency	2.65 ±0.08	1.80 ±0.05	**1.66 ±0.19**	-2.39 ±0.03	-2.04 ±0.02	**-1.99 ±0.09**
Kin8nm	**0.10 ±0.00**	**0.10 ±0.00**	**0.10 ±0.00**	0.90 ±0.01	0.90 ±0.01	**0.95 ±0.03**
Naval Propulsion	**0.01 ±0.00**	**0.01 ±0.00**	**0.01 ±0.00**	3.73 ±0.12	3.73 ±0.01	**3.80 ±0.05**
Power Plant	4.33 ±0.04	4.12 ±0.03	**4.02 ±0.18**	-2.89 ±0.01	-2.84 ±0.01	**-2.80 ±0.05**
Protein Structure	4.84 ±0.03	4.73 ±0.01	**4.36 ±0.04**	-2.99 ±0.01	-2.97 ±0.00	**-2.89 ±0.01**
Wine Quality Red	0.65 ±0.01	0.64 ±0.01	**0.62 ±0.04**	-0.98 ±0.01	-0.97 ±0.01	**-0.93 ±0.06**
Yacht Hydrodynamics	6.89 ±0.67	**1.02 ±0.05**	1.11 ±0.38	-3.43 ±0.16	-1.63 ±0.02	**-1.55 ±0.12**
Year Prediction MSD	9.034 ±NA	8.879 ±NA	**8.849 ±NA**	-3.622 ±NA	-3.603 ±NA	**-3.588 ±NA**

Table 1: **Average test performance in RMSE and predictive log likelihood** for a popular variational inference method (VI, Graves [20]), Probabilstic back-propagation (PBP, Hernández-Lobato and Adams [27]), and dropout uncertainty (Dropout).

to that of Probabilistic Back-propagation (referred to as "PBP", [27]) and to a popular variational inference technique in Bayesian NNs (referred to as "VI", [20]). The aim of this experiment is to compare the uncertainty quality obtained from standard dropout networks (in the same way they are used in current research) to that of specialised models developed to capture uncertainty.

Following our Bayesian interpretation of dropout (eq. (9)) we need to define a prior length-scale, and find an optimal model precision parameter τ which will allow us to evaluate the predictive log-likelihood (eq. (13)). Similarly to [27] we use Bayesian optimisation (BO, [36, 37]) to find optimal τ, and set the prior length-scale to 10^{-2} for most datasets based on the range of the data. Note that this is a standard dropout NN, where the prior length-scale l and model precision τ are simply used to define the model's weight decay through eq. (12). We used dropout with probabilities 0.05 and 0.005 since the network size is very small (with 50 units) and the datasets are fairly small as well. We used 10x more iterations with the optimal parameter values as dropout takes longer to converge, but for BO we used the original number of 40 iterations. Even though the model doesn't converge, it gives BO a good indication of whether the parameter is good or not. Finally, we used mini-batches of size 32 and the Adam optimiser [38]. Further details about the various datasets are given in [27].

The results are shown in table 1. Dropout significantly outperforms all other models both in terms of RMSE as well as test log-likelihood on all datasets apart from Yacht, for which PBP obtains better RMSE. All experiments were averaged on 20 random splits of the data (apart from Protein for which only 5 splits were used and Year for which one split was used). Some of the dropout results have a rather large standard deviation because of single outliers: the median for most datasets gives much better performance than the mean. For example, on the Boston Housing dataset dropout achieves median RMSE of 2.68 with an IQR interval of [2.45, 3.35] and predictive log-likelihood median of -2.34 with IQR [-2.54, -2.29]. In the Concrete Strength dataset dropout achieves median RMSE of 5.15.

To implement the model we used Keras [39], an open source deep learning package based on Theano [40]. In [27] BO for VI seems to require a considerable amount of additional time compared to PBP. However our model's running time (including BO) is comparable to PBP's Theano implementation. On Naval Propulsion for example our model takes 276 seconds on average per split (start-to-finish, divided by the number of splits). With the optimal parameters BO found, model training took 95 seconds. This is in comparison to PBP's 220 seconds. For Kin8nm our model requires 188 seconds on average including BO, 65 seconds without, compared to PBP's 156 seconds.

Dropout's RMSE in table 1 is given by averaging stochastic forward passes through the network following eq. (11) (MC dropout). We observed an improvement using this estimate compared to the standard dropout weight averaging, and also compared to much smaller dropout probabilities (near zero). For the Boston Housing dataset for example, repeating the same experiment with dropout probability 0 results in RMSE of 3.07 and predictive log-likelihood of -2.59. This demonstrates that dropout significantly affects the predictive log-likelihood and RMSE, even though the dropout probability is fairly small.

5.4 Model Uncertainty in Reinforcement Learning

In reinforcement learning an agent receives various rewards from different states, and its aim is to maximise its expected reward over time. The agent tries to learn to avoid transitioning into states

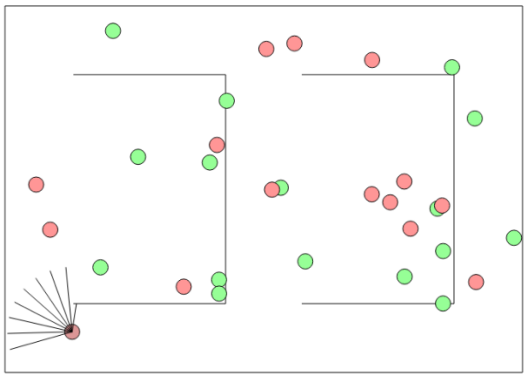


Figure 5: Depiction of the reinforcement learning problem used in the experiments. The agent is in the lower left part of the maze, facing north-west.

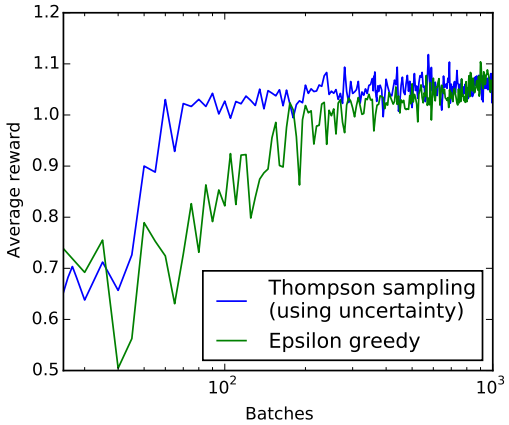


Figure 6: **Log plot of average reward** obtained by both epsilon greedy (in green) and our approach (in blue), as a function of the number of batches.

with low rewards, and to pick actions that lead to better states instead. Uncertainty is of great importance in this task – with uncertainty information an agent can decide when to exploit rewards it knows of, and when to explore its environment.

Recent advances in RL have made use of NNs to estimate agents' Q-value functions (referred to as Q-networks), a function that estimates the quality of different actions the agent can make at different states. This has led to impressive results on Atari game simulations, where agents superseded human performance on a variety of games [14]. Epsilon greedy search was used in this setting, where the agent selects the best action following its current Q-function estimation with some probability, and explores otherwise. With our uncertainty estimates given by a dropout Q-network we can use techniques such as Thompson sampling [11] to converge faster than epsilon greedy while avoiding over-fitting.

We use code by [41] that replicated the results by [14] with a simpler 2D setting. We simulate an agent in a 2D world with 9 eyes pointing in different angles ahead (depicted in fig. 5). Each eye can sense a single pixel intensity of 3 colours. The agent navigates by using one of 5 actions controlling two motors at its base. An action turns the motors at different angles and different speeds. The environment consists of red circles which give the agent a positive reward for reaching, and green circles which result in a negative reward. The agent is further rewarded for not looking at (white) walls, and for walking in a straight line.

We trained the original model and an additional model with dropout with probability 0.1 applied before the every weight layer. To make use of the dropout Q-network's uncertainty estimates, we use Thompson sampling instead of epsilon greedy. In effect this means that we perform a single stochastic forward pass through the network every time we need to make an action. In replay, we perform a single stochastic forward pass and then back-propagate with the sampled Bernoulli random variables. Exact experiment set-up is given in section E.2 in the appendix.

In fig. 6 we show a *log plot* of the average reward obtained by both the original implementation (in green) and our approach (in blue), as a function of the number of batches. Not plotted is the burn-in intervals of 25 batches (random moves). Thompson sampling gets reward larger than 1 within 25 batches from burn-in. Epsilon greedy takes 175 batches to achieve the same performance. It is interesting to note that our approach seems to get worse results after 1K batches. This is because we are still sampling random moves, whereas epsilon greedy only exploits at this stage.

6 Conclusions and Future Research

We have built a probabilistic interpretation of dropout which allowed us to obtain model uncertainty out of existing deep learning models. We have studied the properties of this uncertainty in detail, and demonstrated possible applications, interleaving Bayesian models and deep learning models together. This extends on initial research studying dropout from the Bayesian perspective [42, 43].

Bernoulli dropout is only one example of a regularisation technique corresponding to an approximate variational distribution which results in uncertainty estimates. Other variants of dropout follow our interpretation as well and correspond to alternative approximating distributions. These would result in different uncertainty estimates, trading-off uncertainty quality with computational complexity. We explore these in follow-up work.

Furthermore, each GP covariance function has a one-to-one correspondence with the combination of both NN non-linearities and weight regularisation. This suggests techniques to select appropriate NN structure and regularisation based on our a-priori assumptions about the data. For example, if one expects the function to be smooth and the uncertainty to increase far from the data, cosine non-linearities and L_2 regularisation might be appropriate. The study of non-linearity–regularisation combinations and the corresponding predictive mean and variance are subject of current research.

Acknowledgements

The authors would like to thank Dr Yutian Chen, Mr Christof Angermueller, Mr Roger Frigola, Mr Rowan McAllister, Dr Gabriel Synnaeve, Mr Mark van der Wilk, Mr Yan Wu, and many other reviewers for their helpful comments. Yarin Gal is supported by the Google European Fellowship in Machine Learning.

References

[1] P Baldi, P Sadowski, and D Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5, 2014.

[2] O Anjos, C Iglesias, F Peres, J Martínez, Á García, and J Taboada. Neural networks applied to discriminate botanical origin of honeys. *Food chemistry*, 175:128–136, 2015.

[3] S Bergmann, S Stelzer, and S Strassburger. On the use of artificial neural networks in simulation-based manufacturing control. *Journal of Simulation*, 8(1):76–90, 2014.

[4] M Krzywinski and N Altman. Points of significance: Importance of being uncertain. *Nature methods*, 10 (9), 2013.

[5] Z Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553), 2015.

[6] S Herzog and D Ostwald. Experimental biology: Sometimes Bayesian statistics are better. *Nature*, 494, 2013.

[7] D Trafimow and M Marks. Editorial. *Basic and Applied Social Psychology*, 37(1), 2015.

[8] Regina Nuzzo. Statistical errors. *Nature*, 506(13):150–152, 2014.

[9] O Linda, T Vollmer, and M Manic. Neural network based intrusion detection system for critical infrastructures. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*. IEEE, 2009.

[10] C Szepesvári. Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1), 2010.

[11] W R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933.

[12] C E Rasmussen and C K I Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006.

[13] N Srivastava, G Hinton, A Krizhevsky, I Sutskever, and R Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 2014.

[14] V Mnih, K Kavukcuoglu, D Silver, A A Rusu, J Veness, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[15] R M Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

[16] C K I Williams. Computing with infinite networks. *NIPS*, 1997.

[17] D J C MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4 (3), 1992.

[18] G E Hinton and D 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*, 1993.

[19] D Barber and C M Bishop. Ensemble learning in Bayesian neural networks. *NATO ASI SERIES F COMPUTER AND SYSTEMS SCIENCES*, 168:215–238, 1998.

[20] A Graves. Practical variational inference for neural networks. In *NIPS*, 2011.

[21] D M Blei, M I Jordan, and J W Paisley. Variational Bayesian inference with stochastic search. In *ICML*, 2012.

[22] D P Kingma and M Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[23] D J Rezende, S Mohamed, and D Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

[24] M Titsias and M Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *ICML*, 2014.

[25] M D Hoffman, D M Blei, C Wang, and J Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[26] C Blundell, J Cornebise, K Kavukcuoglu, and D Wierstra. Weight uncertainty in neural networks. *ICML*, 2015.

[27] J M Hernández-Lobato and R P Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *ICML-15*, 2015.

[28] A Damianou and N Lawrence. Deep Gaussian processes. In *AISTATS*, 2013.

[29] L Wan, M Zeiler, S Zhang, Y LeCun, and R Fergus. Regularization of neural networks using dropconnect. In *ICML-13*, 2013.

[30] W Chen, J T Wilson, S Tyree, K Q Weinberger, and Y Chen. Compressing neural networks with the hashing trick. In *ICML-15*, 2015.

[31] Y LeCun and C Cortes. The mnist database of handwritten digits, 1998.

[32] C D Keeling, T P Whorf, and the Carbon Dioxide Research Group. Atmospheric CO2 concentrations (ppmv) derived from in situ air samples collected at Mauna Loa Observatory, Hawaii, 2004.

[33] J Lean. Solar irradiance reconstruction. NOAA/NGDC Paleoclimatology Program, USA, 2004.

[34] Y LeCun, L Bottou, Y Bengio, and P Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[35] Y Jia, E Shelhamer, J Donahue, S Karayev, J Long, R Girshick, S Guadarrama, and T Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[36] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[37] Jasper Snoek and authors. Spearmint. `https://github.com/JasperSnoek/spearmint`, 2015.

[38] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[39] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[40] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[41] A Karpathy and authors. A Javascript implementation of neural networks. `https://github.com/karpathy/convnetjs`, 2014–2015.

[42] S Wang and C Manning. Fast dropout training. *ICML*, 2013.

[43] S Maeda. A Bayesian encourages dropout. *arXiv preprint arXiv:1412.7003*, 2014.